Mendelova univerzita v Brně
Provozně ekonomická fakulta

Možnosti využití Microsoft Power BI v prostředí malých a středních firem

Diplomová práce

Vedoucí práce:
Ing. Jan Přichystal, Ph. D

Bc. Ondřej Plánička

Brno 2015

Chtěl bych poděkovat panu Ing. Janu Přichystalovi, Ph. D. za vedení této diplomové práce a také panu Radimu Hampelovi ze společnosti Intelligent Technologies s. r. o. za podporu a přístup k potřebným technologiím.

Čestné prohlášení

Prohlašuji, že jsem tuto práci: **Možnosti využití Microsoft Power BI v prostředí malých a středních firem**
vypracoval samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*
Jsem si vědom, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona.
Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

V Brně dne 4. ledna 2015 .....

Abstract

Plánička, O. The possibilities of using Microsoft Power BI in small and medium business enviroment. Master's thesis. Brno, 2015.

This thesis describes modern approaches to Business intelligence. It also explains in detail platform Power BI and Busineess intelligence add-ins for Microsoft Excel 2013. This thesis also shows how to create an application that communicates with free billing service iDoklad. The resulting application is evaluated from an economic and implementation standpoint.

Abstrakt

Plánička, O. Možnosti využití Microsoft Power BI v prostředí malých a středních firem. Diplomová práce. Brno, 2015.

V této práci jsou popsány moderní přístupy k Business intelligence. Dále je detailně rozebrána a vysvětlena platforma Power BI a Business intelligence doplňky do aplikace Microsoft Excel 2013. V práci je také popsán postup vytváření aplikace, která komunikuje s cloudovou fakturační službou iDoklad. Výsledná aplikace je zhodnocena z ekonomického hlediska a z hlediska implementace.

Obsah

1 Úvod a cíl práce

1.1 Úvod do problematiky

Business intelligence (BI) je pojem, který je v poslední době v informačních technologiích často skloňován. Důvodem je zejména poptávka na trhu od firem, které chtějí do svých systémů implementovat podporu rozhodování. Podle (Kimball a Ross, 2002) je Business intelligence: „obecný termín, který popisuje využití interních a externích informací pro podporu lepšího rozhodování v podnikání“. BI technologie jsou schopné zpracovávat velké objemy nestrukturovaných dat a pomáhají identifikovat, určovat i rozvíjet nové podnikové příležitosti. Právě identifikace nových příležitostí a následné použití efektivních strategií může podnikům přinést konkurenční výhodu a větší dlouhodobou stabilitu. BI technologie poskytují jak historické, tak i aktuální a prediktivní přehledy. Mohou být použity k podpoře široké škály podnikových rozhodnutí od operativních až po strategické. Největší efektivnosti s BI lze dosáhnout kombinací dat z prostředí vnějšího (data z trhu, na kterém podnik působí) a vnitřního (finanční data, data o podnikových operacích). V takovém případě vzniká komplexní pohled na situaci podniku.

Tradiční způsob vytváření BI řešení, který je kombinací datových skladů, OLAP kostek a reportů nad nimi, je v dnešní době čím dál více doplňován a dokonce nahrazován dalšími komponenty. Tyto komponenty mohou zjednodušit práci s daty a poskytnout tak možnost vytvářet BI řešení koncovým uživatelem. Tento trend vznikl díky rozšíření cloud a in-memory computingu, který z pohledu BI přináší hned několik výhod.

Prostředí malých a středních podniků jsou v právu EU definovány jako podniky, které mají od 10 do 250 zaměstnanců a obrat se pohybuje od 10 do 50 milionů EUR.[1]

1.2 Cíl práce

Cílem práce je prozkoumání možností využití platformy Microsoft Power BI v prostředí malých a středních firem.

Prozkoumání těchto možností budu realizovat na příkladu konkrétní aplikace. Tato BI aplikace bude využívat doplňky Power Query, Power Pivot, Power View a Power Map, které ve spojení s cloudovou službou iDoklad, budou zajišťovat podporu rozhodování pro její uživatele. V aplikaci iDoklad je minimální podpora Business intelligence a proto může být toto BI řešení vhodným doplňkem.

Aplikace se bude skládat ze série dashboardů při čemž každý z nich bude zaměřen na jiný pohled na data. Celá aplikace bude podporovat přihlášení různých uživatelům na základě účtu na iDokladu a hesla. Vytvořená aplikace také bude nahrána do cloudové služby Power BI, kde bude zhodnoceno její fungování a využití.

[1] Zdroj: http://ec.europa.eu/enterprise/policies/sme/facts-figures-analysis/sme-definition/ index_cs.htm

2 Metodika

Před samotnou prací na aplikaci je nutné prozkoumat platformu Power BI, kterou bude aplikace využívat. V první řadě bude platforma představena společně s dalšími dvěma konkurenty, kteří se pohybují na stejném trhu.

Poté budou detailně prozkoumány jednotlivé doplňky, které budou při práci použity. Dále bude vysvětlena jejich funkce, jejich umístění v celé platformě Power BI a také jejich technické možnosti. Společně s těmito doplňky bude představeno cloudové účetnictví iDoklad a jeho funkce. V této části bude také vysvětleno teoretické fungování všech částí aplikace a bude možné přejít k samotné vlastní práci.

Ze služby iDoklad budou data stahována přes doplněk Power Query, kde bude prováděno čištění dat a následně jejich uložení do datového modelu. Především se zaměříme na data o zákaznících a vydaných fakturách, ze kterých získáme jednotlivé položky. Poté si vytvoříme v Power Query datumovou dimenzi jen za pomoci jazyka M. Dále také vytvoříme jednoduchou Power Query funkci, která bude podle poštovního směrovacího čísla určovat, z jakého kraje České republiky zákazník je.

Po načtení dat do datového modelu bude nutné vytvořit metriky, které budou uživatelé sledovat. Budou to metriky sledující tržby, kumulované tržby, průměrné tržby, DPH, počty faktur, faktury nezaplacené a zákazníky. Celému datovému modelu bude nastaveno formátování jednotlivých sloupců, aby se správně zobrazovaly v doplňku Power View.

Jakmile bude datový model připraven, začnou se vytvářet jednotlivé Power View listy. Celá aplikace bude mít několik částí, které budou na faktury pohlížet z různých úhlů. Bude vytvořen hlavní přehled, ve kterém bude k dispozici měsíční porovnání tržeb s předchozími lety. Budou zde také uvedeny informace o aktuálním měsíci a jeho výsledcích. Na dalším listu budeme sledovat vydané faktury z pohledu zákazníků, kde budeme sledovat nejvíce nakupující zákazníky, přehled faktur jednotlivých zákazníků a poměr mezi novými a vracejícími se zákazníky. Samostatně se budou analyzovat faktury, které budou rozděleny podle roků, kvartálů a měsíců. Zde budeme sledovat tržby, kumulované tržby, ceny bez DPH a další. Celý výstup bude možné filtrovat podle jednotlivých krajů. Pro lepší názornost budou na tomto listu i grafy s počtem vydaných faktur v jednotlivých měsících a přehled, v jakých dnech v týdnu se nejvíce fakturovalo. Pro produkty bude vytvořen vlastní list, kde budou k dispozici informace o tom, kolik se prodalo kterého produktu a za jakou cenu. Bude možné vidět žebříček nejprodávanějších produktů. Vše bude možné filtrovat pomocí roků a jednotlivých měsíců v roce. Jako doplněk k BI funkcím bude aplikace obsahovat také možnost plánování tržeb. Zde si uživatel bude moci zadat své plánované tržby na jednotlivé měsíce v roce. Po uložení do datového modelu se zobrazí grafické porovnání zadaného plánu a dat z iDokladu. Poslední funkcí celé aplikace bude možnost sledování DPH. Na tomto listu bude uveden přehled vyfakturovaného DPH za měsíc, kvartál a rok. Aplikace také bude sledovat obrat za posledních 12 měsíců a uživatel uvidí, zda se blíží k milionové hranici, za kterou už se stává plátcem DPH.

Posledním krokem v aplikaci bude vytvoření návodu na zprovoznění, ve kterém budou popsány kroky vedoucí ke správnému fungování, a kde bude dostupný odkaz na stažení doplňku Power Query, který je k používání aplikace potřeba.

Následně se celá vytvořená aplikace vloží do cloudové služby Power BI a bude povolena možnost vyhledávání v datech pomocí funkce *Otázky a odpovědi*.

V závěru bude celá implementace zhodnocena. Budou shrnuty dosažené výsledky a dále bude doporučeno, na co si dát při vypracování aplikace pozor.

3 Moderní přístupy k Business intelligence

Díky novým možnostem, které přinesl cloud a in-memory computing, se na trhu v posledních letech objevily nové techniky pro vytváření Business inteligence řešení. Společnosti se stále více snaží zjednodušit a urychlit práci s daty jednotlivých systémů. Stále více se skloňuje sousloví „self-service BI“, které se dá přeložit jako samoobslužné BI. Cílem tohoto přístupu je poskytnout co nejvíce možností práce s daty koncovým uživatelům. Díky tomu nebude zbytečně vznikat komunikační prodleva mezi uživatelem a IT oddělením, které má na starosti vytváření BI řešení. Pokud budou mít uživatelé k dispozici nástroje, za jejichž pomoci budou schopni vytvářet vlastní BI řešení, bude to výhodné pro obě strany.

Dalším trendem posledních let jsou cloudová řešení. I v oblasti BI se na tuto problematiku společnosti v posledních letech více zaměřily a snaží se přinést co nejlepší řešení cloudového BI. Myšlenka vytvoření BI řešení, které není závislé na žádném vlastním hardwaru a může být vytvářeno jen s pomocí prohlížeče, je hlavním úspěchem české firmy GoodData. Díky cloudu je možné BI řešení používat bez nutnosti instalace softwaru a je možné reporty procházet jen za pomoci prohlížeče.

3.1 Tableau

Tableau má zajímavou historii, která byla odstartována lidmi s velkým zájmem především o vizualizaci dat. Od začátku bylo Tableau vytvářeno jako nástroj, který by umožnil rychlé vizuální prezentace dat. Tableau samo sebe označuje jako „ohromující alternativa k tradičnímu Business intelligence“ a snaží se získat část trhu, pro který jsou například Cognos nebo Microsoft Reporting Services příliš těžkopádné nástroje.

Jednou z více oblastí, ve které Tableau překonává svou konkurenci, je rychlost. „Zpracovávejte data ve vysoce výkoném enginu Tableau s ohromnou rychlostí a vše bez náročného programování. Tableau přemění miliony řádků dat na odpovědi rychlostí myšlenky“ píše se na stránce softwaru. (Beck, 2014)

Celá platforma Tableau se skládá z několika částí, které se navzájem doplňují a také rozšiřují své funkce.

Základním stavebním kamenem celé platformy je *Tableau Desktop*, skrz který se vytváří veškerý obsah, který je na platformě konzumován. k dispozici jsou dvě verze, *Personal* a Professional, které se liší hlavně cenou a svými možnostmi. *Tableau Desktop Personal* stojí 999 dolarů na jednoho uživatele a podporuje načítání dat jen ze souborů typu Excel a Access, textových souborů, oData, Tableau Data Extract a Azure Marketplace. Chybí mu také podpora pro sdílení přes *Tableau Server*. Vyšší verze *Professional* se prodává za 1 999 dolarů a značně rozšiřuje podporované zdroje dat. V této verzi je podporováno přibližně 30 různých datových zdrojů, přes všechny typy relačních databází, Hadoop, SAP HANA, až po Google Analytics.

K prohlížení konkrétních řešení slouží program *Tableau Reader*. Tento program slouží výhradně ke spouštění řešení, vytvořených v Tableau Desktop.

Tableau Server a *Tableau Online* jsou systémy pro sdílení vytvořených řešení, díky kterým je možné pracovat s daty v prohlížeči. Nejvýraznějším rozdílem je jen to, že *Tableau Server* je on-premise[2] instalace narozdíl od *Tableau Online*, který je hostován v cloudu. Oba tyto systémy podporují mobilní přístup, rozdělení přístupových práv k jednotlivým řešením a zabezpečení dat. Cena *Tableau Server* je individuální pro různé firmy, ale pohybuje se v rozmezí 800–1500 dolarů, podle specifikace serveru. *Tableau Online* se prodává za 500 dolarů za uživatele a rok.

Tableau je nástroj, který umožňuje nepřekonatelné vizualizace dat. Data je možné analyzovat díky technologii VizQL, která je založena na výzkumu Standfordské univerzity. VizQL překládá klikání myši do databázových dotazů a zobrazuje grafické výstupy dat (Tableau Software, 2014). pro vytvoření BI řešení není potřeba napsat jediný řádek kódu a přesto je možné vytvořit obsáhlé řešení, které bude přehledné a rychlé.

Výhodou používání platformy Tableau jsou obrovské možnosti vizualizace dat a jednoduchý proces vytváření jednotlivých reportů. V neprospěch hraje cena, která je oproti konkurenci vysoká. Jen cena licenčních poplatků při vytvoření takového BI řešení se počítá v řádech tisíců dolarů. Dále zde chybí propracovanější podpora ETL[3], která by mohla lépe data připravit pro zpracování.

3.2 GoodData

GoodData je česká společnost, která byla založena v roce 2007 a vytváří svou vlastní platformu pro Business intelligence a analýzu Big Data[4]. GoodData vytváří SaaS[5] Business intelligence řešení tak, že se snaží všechny části hostovat na svých serverech. Díky tomu zákazník nemá další náklady spojené s nákupem hardwaru a speciálního software. Servery, na kterých běží celá platforma, jsou hostovány firmou Amazon.

GoodData, stejně jako Tableau, mají svou vlastní platformu, kterou vyvíjí. Avšak na rozdíl od Tableau provádí i konečnou implementaci BI projektů. To znamená, že zákazník nepotřebuje žádné IT oddělení pro vytváření svých řešení. Na jednu stranu je to výhoda, která firmám zjednoduší přístup k BI. Na stranu druhou je to i nevýhoda, protože je díky tomu služba dražší.

I když je možné použít jakýkoliv ETL nástroj, GoodData doporučuje používat *CloudConnect Designer*, což je desktopová aplikace speciálně navržená pro budování datových integrací pro platformu GoodData. Díky snadno použitelnému gra-

[2]On-premise je označení místa, kde je určitý software uložen. On-premise software je uložen na serveru zákazníka, který jej vlastní a sám spravuje. Zdroj: http://www.techopedia.com/definition/26714/on-premises-software

[3]ETL označuje právě mechanizmus získávání dat z provozních systémů podniku, jejich následné zpracování a poskytnutí aplikacím pro podporu rozhodování. Zdroj: http://www.systemonline.cz/clanky/co-se-skryva-pod-zkratkou-etl.htm

[4]Big Data jsou velice objemná, rozdílná a rychle se měnící data, která vyžadují nákladově efektivní a inovativní formy zpracování informací pro lepší pochopení a rozhodování. Zdroj: http://www.gartner.com/it-glossary/big-data/

[5]Software as a Service

fickému rozhraní a velké knihovně předpřipravených komponent, můžeme rychle sestavit ETL. Integrované testovací nástroje nám umožní provádět ladění ETL na místní úrovni předtím, než bude publikováno. Jakmile je vývoj ETL kompletní, je možné publikovat data do svých projektů jediným kliknutím, po kterém může být naplánované automatické nahrávání nových dat.

Po úspěšném nahrání dat do cloudu je možné vytvářet metriky, které budou používány v různých reportech. Metriky mohou být vytvořeny podle jednoduchého průvodce nebo pro složitější kalkulace v pokročilém editoru. Tento editor umožňuje definovat dotazy v jazyku MAQL[6], který si pro svou potřebu vytvořili v GoodData.

Jakmile je metrika vytvořena, může se používat v jakémkoliv reportu v příslušném projektu. Metriky mohou být dále používány a přidávány do reportů pomoci *Report editoru*. Tento editor nabízí grafické rozhraní pro přidání metrik do reportů, nebo vytváření metrik přímo při vytváření reportu. Jakmile byla do reportu přidána jedna nebo více metrik, mohou se k nim připojit *atributy* – kvantitavní popisy dat, jako například regiony, produkty a oddělení. Nakonec je možné přidat filtry dat, které budou omezovat zobrazená data.

Data zobrazená v reportu je možné vizualizovat pomocí grafů a číselných vyjádření. Pro jednotlivé reporty je možné nastavit různá přístupová a editační práva. (GoodData Developers, 2014)

Protože GoodData není jen výrobcem BI platformy, ale také rovnou řešení implementuje, nejsou nikde uvedeny přesné ceny platformy. Cena se skládá z ceny za implementaci projektu a poté měsíčních poplatků za využívání platformy GoodData. Podle starších cen stál pronájem 500 USB za měsíc pro neomezený počet uživatelů a pouze jeden report [7].

Platformu GoodData využijí převážně firmy, které chtějí využívat možnosti BI, ale nemají k dipozici zkušené IT oddělení. Tyto firmy dostanou hotové řešení, které bude přístupné odkudkoliv a bude obsahovat stále aktuální data. Nevýhodou může být malá kontrola nad daty, sdílení svých dat s jinou firmou a vyšší cena, která je placena formou měsíčního předplatného.

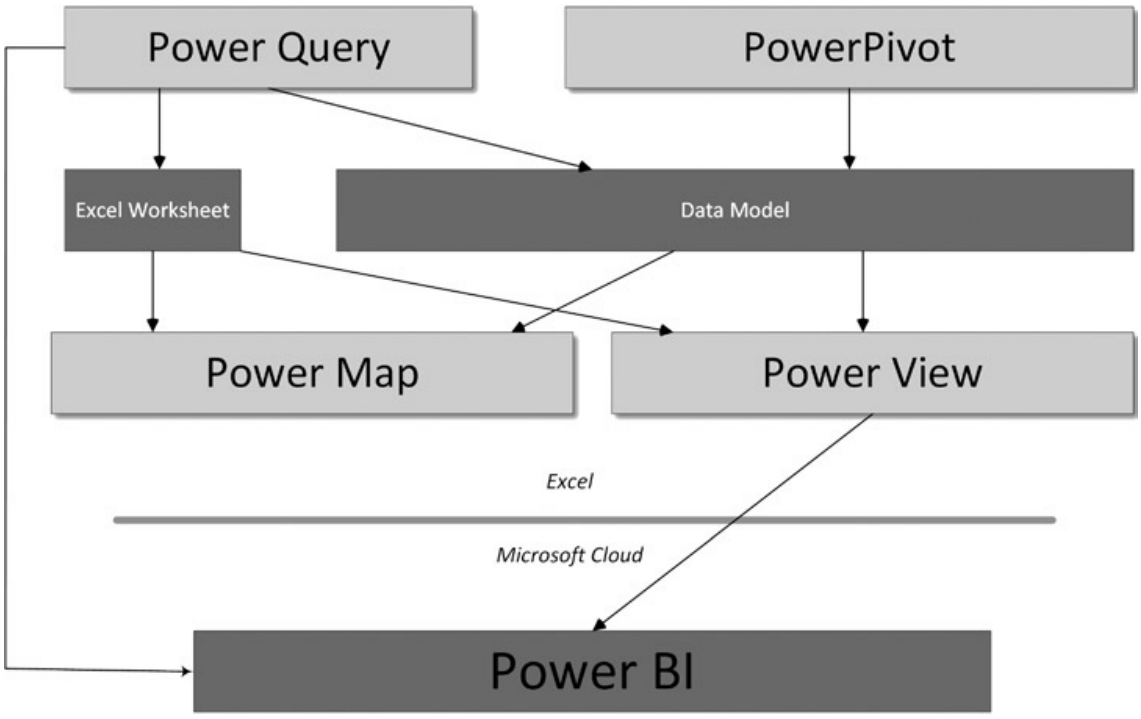
3.3 Power BI

Power BI je platforma pro „self-service BI“ od společnosti Microsoft. Jedná se o sadu nástrojů, která rozšiřuje práci s programem Microsft Excel. Pod názvem Power BI můžeme nalézt dvě oddělené části, které se starají o Business intelligence v podniku (obrázek 1).

První část vytváří BI řešení za pomocí Microsft Excel a jeho doplňků. Tato část je samostatná a pokud firma nemá speciální nároky na fungování BI řešení ve firmě, je plně dostačující. Skládá se z několika doplňků do aplikace Microsoft Excel, takzvaných Power Tools, které se starají o načtení, transformaci, uložení a následnou prezentaci dat. Všechny tyto doplňky začínají slovem „Power“, tak aby

[6]Multi-Dimensional Analytical Query Language

[7]Zdroj: http://web.archive.org/web/20100811023135/http://www.gooddata.com/pricing/



Obrázek 1: Grafické znázornění Power BI platformy. (Aspin, 2014)

zapadly do jednotného konceptu Power BI. Jedná se o doplňky *Power Query*, *Power Pivot*, *Power Power View* a *Power Map*.

Power Query je zdarma stažitelný doplněk do aplikace Excel, který se stará o import dat. V tomto doplňku je možné importovat data z mnoha externích zdrojů (např. relační databáze, oData, sešity aplikace Excel, soubory csv, Facebook a další), poté tato data transformovat a takto transformovaná uložit do datového modelu. Tento proces vytvoří *dotaz*, který je možné sdílet napříč firmou pomocí služby Power BI. Tyto dotazy jsou aktualizovatelné, takže není potřeba provádět stejné operace stále znovu.

Power Pivot se stará o datový model na pozadí aplikace Excel. V dřívějších verzích aplikace Excel, stejně jako nyní Power Query, byl volně stažitelný doplněk. Od verze 2013 je pevnou součástí aplikace Excel a není potřeba ho doinstalovávat. Doplněk je dostupný ve verzích Microsoft Office Proffesional Plus, Office 365 pro Plus a v krabicové verzi Microsoft Excel 2013. Nižší verze balíku Microsft Office tento doplněk neobsahují. Power Pivot značně zrychluje a usnadňuje práci s velkými objemy dat díky použití xVelocity enginu a desktopové verzi Analysis Servises. V datovém modelu je možné definovat vztahy mezi jednotlivými tabulkami a vytvářet počítaná pole a metriky. Veškeré kalkulace a práce s daty probíhají právě v tomto doplňku.

Power View si je možné představit jako plátno, na které se zobrazují data z datového modelu. Tento doplněk slouží výhradně pro prezentaci dat za pomocí interaktivních grafů, průřezů a tabulek. Vyznačuje se jednoduchým, přehledným

ovládáním a vysokou čitelností důležitých dat. Vytváření reportu za pomocí Power View je jednoduché a je uzpůsobeno tak, aby i netechnicky založení uživatelé byli schopni vytvářet přínosné výstupy.

Power Map je další prezentační nástroj, který se zaměřuje hlavně na geografická data. Pomocí tohoto doplňku je možné vytvářet videa, která zkoumají změny v čase na jednotlivých částech mapy. Tento doplněk je spíše zaměřen na prezentování výsledků v Power Point prezentacích než na procházení dat a hledání souvislostí.

Druhou částí je cloudová služba *Power BI*. Tato služba má za úkol rozšiřovat možnosti sdílení a práce s reporty na webu. Jakmile uživatel vytvoří report v aplikaci Microsoft Excel má možnost tento report nahrát na stránku Power BI, kde je možné pomocí prohlížeče reporty procházet. k takto nahraným reportům je možné přistupovat pomocí speciální aplikace pro tablety. Díky podpoře HTML5 je možné si stránku Power BI otevřít v jakémkoliv prohlížeči. Další funkcí jsou takzvané *Otázky a odpovědi*, ve kterých se uživatel může dotazovat na data uložená v reportech pomocí přirozeného jazyka. Funguje to jednoduše tak, že uživatel napíše otázku a systém zobrazí odpověď ve formátu, který bude nejčitelnější. Například pokud se uživatel ptá na data vztažená k určité lokalitě, bude se mu zobrazovat mapa. Cena služby Power BI je 40 dolarů za měsíc a uživatele, pokud už máte k dispozici Microsoft Excel 2013 a jeho doplňky. Pokud zaplatíte 52 dolarů za měsíc a uživatele, dostanete navíc ke cloudové službě Power BI také balík Office 365 pro Plus.

Celá platforma Power BI je postavena na rozšířenosti a oblíbenosti aplikace Microsoft Excel. Myšlenka toho, že si uživatel může vytvořit reporty jen pomocí tohoto programu, je velice lákavá. Díky tomu má Microsoft neskutečnou výhodu oproti ostatním řešením, protože aplikace Microsoft Excel je nainstalována ve většině podnikových počítačů. Cloudová služba Power BI je jen doplněk, který využijí větší firmy, které kladou důraz na nastavení přístupu k jednotlivým reportům, jednoduchou práci odkudkoliv a rozšířenější formu sdílení reportů. Nevýhodou tohoto řešení může být mírná složitost nastavení jednotlivých doplňků tak, aby spolu správně spolupracovali. Dále také menší míra uživatelského nastavení vzhledu reportů v doplňku Power View.

4 Použité technologie

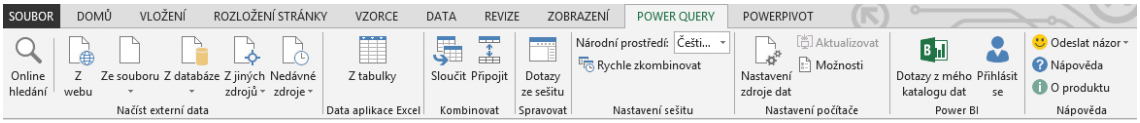
Pro vytvoření BI aplikace, která bude zpracovávat data z cloudové služby iDoklad, byla použita platforma Microsoft Excel s Power BI nástroji. Jedná se o volně dostupné doplňky, které Excel rozšiřují o další možnosti práce s daty, jejich zpracování a prezentaci. Koncepce Power BI byla poprvé představena na konci roku 2013 a dostupná pro veřejnost je od začátku roku 2014. Jejím hlavním cílem je více přiblížit Business intelligence koncovým uživatelům tak, aby byli schopni vytvářet své vlastní reporty a dashboardy na základě jejich preferencí.

4.1 Power Query

Power Query je doplněk programu Excel, vytvořený firmou Microsoft, jehož úkolem je usnadnit načítání dat z externích zdrojů. Je součástí Power BI sady nástrojů, která byly vyvinuta pro vytváření self-service Business intelligence řešení. (Webb, 2014)

Tento doplněk slouží k ETL procesu v programu Excel. To znamená, že se stará o všechny jeho kroky od načtení dat z různých externích zdrojů přes transformaci, kombinaci a úpravu dat, až po uložení dat v Excel sešitu nebo v datovém modelu.

První fází práce s doplňkem je připojení k externímu zdroji, ze kterého se data budou načítat. Jakmile budou zadány všechny potřebné informace, údaje ze zdroje se načtou do samostatného okna doplňku Power Query, ve kterém se s nimi může dále pracovat. Všechny úpravy, které se při transformaci dat provádí, se ukládají jako jednotlivé kroky, které se opakují vždy, když je tento dotaz znovu použit. Jakmile jsou transformace dat hotovy, výsledek se uloží jako dotaz v sešitu Excel. Tyto dotazy jsou vidět v přehledu doplňku Power Query. Pokud bude potřeba dotaz aktualizovat, stiskneme tlačítko, které provede znovu všechny kroky uložené v definici dotazu.



Obrázek 2: Karta Power Query v Excelu 2013

V levé části na obr.2 lze vidět možnosti, které doplněk podporuje pro import z externích zdrojů. Následně jsou zde možnosti kombinování a spojování jednotlivých dotazů mezi sebou a možnost volby nastavení sešitu, kam uživatel zadává jazykové prostředí zdrojů, se kterými aplikace pracuje. Další možností je volba jednoduššího kombinování zdrojů. *Nastavení počítače* obsahuje volby pro správu hesel a nastavení připojení pro jednotlivé dotazy. Skupina *Power BI* se stará o sdílení jednotlivých dotazů napříč firmou díky cloudové službě Power BI. Poslední část je zaměřena na sekci nápovědy a sekci informace o produktu.

Zdroje, ze kterých je možno načítat, jsou rozděleny do několika skupin:

- **Z webu** – jedna z nejzajímavějších součástí Power Query je možnost načítání strukturovaných dat z jakékoliv stránky na internetu. Stačí zadat url odkaz webu, na kterém se data nachází, a doplněk sám vyhledá data a importuje je. Tato funkce vyhledá jakoukoliv HTML anotaci pro tabulku v textu a tuto tabulku nabídne jako zdroj dat ze zadané webové adresy.
- **Ze souboru** – v této skupině najdeme nástroje pro import dat ze souborů různých typů. Možnosti importu jsou ze souborů Excel, souborů s příponou .csv, ze souboru XML, obyčejného textového souboru nebo z obsahu vybrané složky.
- **Z databáze** – import dat z databáze je jedním z nejdůležitějších nástrojů Power Query, data můžeme jednoduše získat z těchto databází: SQL Server, Azure SQL, Access, SQL Server Analysis Services, Oracle, IBM DB2, MySQL, PostgreSQL, Teradata a Sybase.
- **Z jiných zdrojů** – Tato skupina zdrojů se zaměřuje na nedatabázové systémy a datové zdroje, například: Facebook, Active Directory, Azure Blob Storage, Odata a Salesforce.

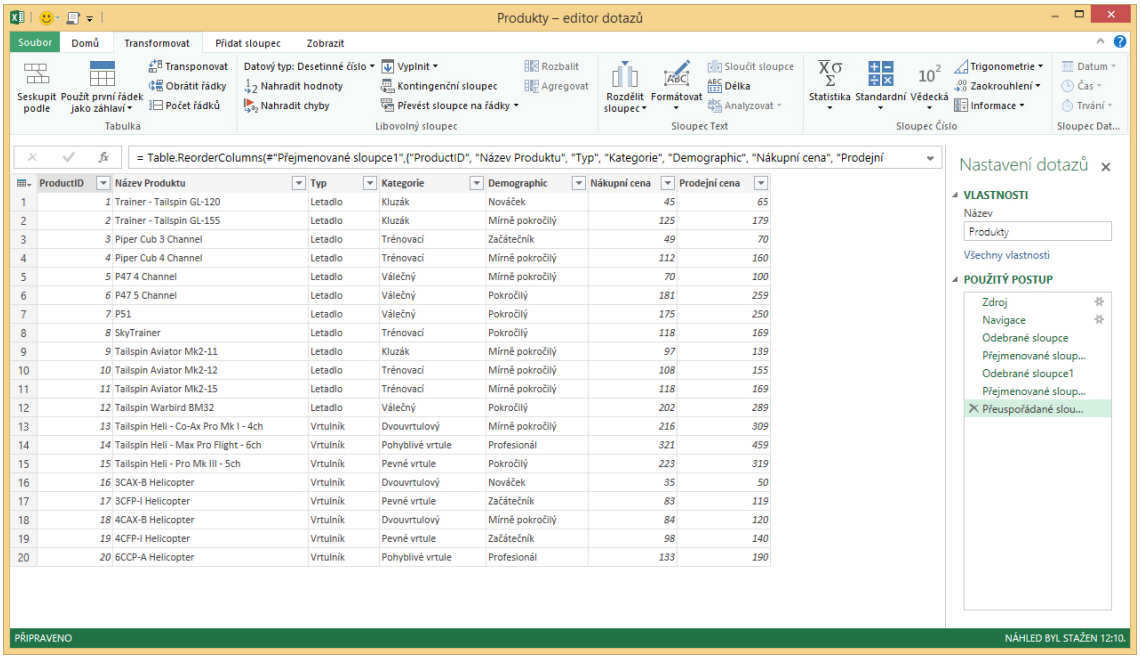
Samostatnou funkcí je zde *online hledání*, které umožňuje vyhledávání v podnikových a veřejných zdrojích. Při spuštění této funkce Excel otevře vyhledávací okno, kde je možné zadat hledaný výraz. Prohledávána jsou volně dostupná data veřejné správy a data ze serveru wikipedia. Bohužel v České republice není eGoverment[8] na tak vysoké úrovni jako v USA, takže pomocí této funkce nejsme schopni vyhledat žádná data veřejné správy. Navíc jsme odkázání pouze na vyhledávání v angličtině, protože českou lokalizací tato funkce nedisponuje.

Nový rozměr dostává tato funkce ve spojení s cloudovou službou Power BI. Jakmile je uživatel přihlášen pod svým účtem do Power BI, může vyhledávat datové zdroje a dotazy, které vytvořili jiní uživatelé a může je začít používat ve svém řešení. Uživatel má možnost filtrovat tyto nabízené zdroje pomocí parametrů pro efektivnější nalezení správného dotazu. Je možné filtrovat data podle toho, kdo dotaz vytvořil, kdy byl vytvořen, podle názvu, podle jeho popisu, nebo podle datového zdroje, který používá. Pokud si uživatel není jistý ani jedním z předchozích parametrů, může filtrovat přímo podle názvu sloupců, které dotaz vrací. Tato funkce velice usnadňuje a zrychluje práci při vytváření datového modelu a dává uživatelům možnost využívat předem vytvořených dotazů, které nemusí znovu vytvářet.

Jakmile uživatel zvolí datový zdroj, ze kterého chce data načíst a zadá potřebné údaje, tak se mu otevře samostatné okno doplňku, ve kterém provádí transformaci importovaných dat.

Toto okno (obr. 3) se dělí na tři základní části: nastavení dotazu, karta transformací a nastavení a náhled dat.

[8] eGovernment využívá informační a komunikační technologie k posílení a zlepšení kvality a efektivity veřejné správy. Zdroj: http://www.digitales.oesterreich.gv.at/site/6506/default.aspx



	ProductID	Název Produktu	Typ	Kategorie	Demographic	Nákupní cena	Prodejní cena
1	1	Trainer - Tailspin GL-120	Letadlo	Kluzák	Nováček	45	65
2	2	Trainer - Tailspin GL-155	Letadlo	Kluzák	Mírně pokročilý	125	179
3	3	Piper Cub 3 Channel	Letadlo	Trénovací	Začátečník	49	70
4	4	Piper Cub 4 Channel	Letadlo	Trénovací	Mírně pokročilý	112	160
5	5	P47 4 Channel	Letadlo	Válečný	Mírně pokročilý	70	100
6	6	P47 5 Channel	Letadlo	Válečný	Pokročilý	181	259
7	7	P51	Letadlo	Válečný	Pokročilý	175	250
8	8	SkyTrainer	Letadlo	Trénovací	Pokročilý	118	169
9	9	Tailspin Aviator Mk2-11	Letadlo	Kluzák	Mírně pokročilý	97	139
10	10	Tailspin Aviator Mk2-12	Letadlo	Trénovací	Mírně pokročilý	108	155
11	11	Tailspin Aviator Mk2-15	Letadlo	Trénovací	Mírně pokročilý	118	169
12	12	Tailspin Warbird BM32	Letadlo	Válečný	Pokročilý	202	289
13	13	Tailspin Heli - Co-Ax Pro Mk I - 4ch	Vrtulník	Dvouvrtulový	Mírně pokročilý	216	309
14	14	Tailspin Heli - Max Pro Flight - 6ch	Vrtulník	Pohyblivé vrtule	Profesionál	321	459
15	15	Tailspin Heli - Pro Mk III - 5ch	Vrtulník	Pevné vrtule	Pokročilý	223	319
16	16	3CAX-B Helicopter	Vrtulník	Dvouvrtulový	Nováček	35	50
17	17	3CFP-I Helicopter	Vrtulník	Pevné vrtule	Začátečník	83	119
18	18	4CAX-B Helicopter	Vrtulník	Dvouvrtulový	Mírně pokročilý	84	120
19	19	4CFP-I Helicopter	Vrtulník	Pevné vrtule	Začátečník	98	140
20	20	6CCP-A Helicopter	Vrtulník	Pohyblivé vrtule	Profesionál	133	190

Obrázek 3: Okno doplňku Power Query

Horní karta obsahuje několik záložek pro práci s daty. Záložka *Domů* se stará o základní práci s dotazem, tzn. nastavení, kam se budou data ukládat, aktualizace náhledu, změna datových typů, odebrání, přejmenování sloupců a řazení. V záložce *transformace* je možné provádět veškeré změny dat ze zdroje. Pokud se například načítají data ze souboru csv, zde zvolíme, aby se první řádek použil jako názvy sloupců. Možností transformací je mnoho, od textových přes číselné až po možnosti práce s chybami, prázdnými a duplicitními daty.

V hlavním okně se zobrazuje náhled dat, se kterými se zrovna pracuje. Jakmile se provede konkrétní transformace, data se ihned změní a je možné vidět výsledek. Hlavní okno navíc slouží pro zobrazení chybových oznámení. Do Power Query se nenačítá celá databáze dat, ale z důvodu efektivity se načítá pouze náhled o velikosti 100 řádků.

Na pravé straně je panel *nastavení dotazu*, kde je v části *vlastnosti* možné zadat název a popis dotazu, podle kterého bude tento dotaz vyhledán. Název dotazu bude také použit jako název tabulky v datovém modelu. Pod názvem dotazu je část *použitý postup*, ve které se ukládají všechny změny, které byly s daty provedeny. Jednotlivé položky seznamu zobrazují právě jednu operaci s daty, od načtení ze zdroje až po poslední úpravu. Jednotlivé kroky se mohou také v průběhu práce mazat a nahrazovat.

Chcete-li plně využít potenciál Power Query, musíte se naučit M. M je neoficiální název pro Power Query formula language. Power Query generuje M kód pro každý krok vašeho dotazu. Uživatel rozhraní Power Query také umožňuje napsat vlastní M výrazy v situacích, kdy budete potřebovat větší flexibilitu a kontrolu nad tím, co

váš dotaz dělá. (Webb, 2014)

Díky M kódu se uživatelům dostává do ruky mocný nástroj, kterým mohou v Power Query vytvářet složité struktury a funkce.

Obrázek 4: Editor jazyka M

Možnost vytvořit z dotazu funkci je jednou z největších předností tohoto jazyka. Například pokud je zapotřebí spojit větší množství stejně strukturovaných csv souborů pro následnou analýzu, stačí vytvořit dotaz, který bude zpracovávat pouze jeden tento soubor a z něj vytvořit funkci, jejímž parametrem bude cesta k souboru. Takže vytvoříme jeden dotaz, který znovu použijeme pro stejnou opakující se činnost. A to je celá myšlenka Power Query. Snažit se stejné opakující se procesy zautomatizovat natolik, aby nad nimi uživatelé netrávili zbytečný čas.

Něco, co není na první pohled zřejmé o Power Query, je kde probíhá všechna ta tvrdá práce. Když se dotaz připojí ke zdroji dat a začne přidávat sérii transformací k datům, kde je to možné, Power Query se bude snažit co nejvíc práce protlačit zpět ke zdroji dat, kde (tak se to alespoň předpokládá) se může provádět efektivněji. Toto chování se nazývá *query folding*. Jestli je to doplněk Power Query schopen udělat nebo ne, závisí na typu zdroje dat a typu transformací v dotazu. S relační databází jako SQL Server se Power Query pokusí přeložit všechnu logiku v dotazu na příkazy SQL SELECT. Se zdrojem dat OData se pokusí převést všechny logiky do jednoho OData URL. Nicméně, pro datové zdroje jako jsou například textové soubory, Power Query nemá jinou možnost, než načíst všechna data do vlastního enginu a provádět všechny transformace v dotazu interně. (Webb, 2014)

Power Query nabízí dvě možnosti importu zpracovaných dat: přímo do listu v Excelu nebo do datového modelu Power Pivot. Obě varianty se liší jen tím, jak se bude s daty dále pracovat. Při importu do listu můžeme s daty ihned pracovat, vytvořit si jednoduché vzorce a data filtrovat a procházet. Při importu do Power Pivot se vytvoří spojení k externímu zdroji a data se ukládají na pozadí. Všechna další práce s daty probíhá v doplňku Power Pivot. Po úspěšném vytvoření je dotaz uložen uvnitř sešitu a je možné ho později aktualizovat a pracovat se stále aktuálními daty bez potřeby opakovat stále stejné kroky znovu.

Power Query je, podle mého názoru, jeden z nejlepších nástrojů pro ETL proces na trhu. Je jednoduchý, rychlý a neustále se vyvíjí. Jeho aktualizace Microsoft

vydává každý měsíc a s každou novou verzí přichází více možností a funkcí. Na druhou stranu je v něm ale spoustu věcí, které lze zlepšit. Jeho největší problém vidím v rozšířeném editoru jazyka M, který je pouze textový. Uvítal bych zde IntelliSense[9] funkce a zvýrazňování syntaxe známé například z programu Visual Studio. Dalším zlepšením by mohla být lepší komunikace mezi doplňky Power Query a Power Pivot, kde při vytváření datového modelu dochází ke spoustě problémů.

4.2 Power Pivot

Power Pivot je doplněk zaměřený na poskytnutí self-service Business intelligence, což je skutečná revoluce uvnitř světa analýzy dat, protože dává koncovému uživateli všechny potřebné nástroje k provedení komplexní analýzy dat bez nutnosti zásahu BI techniků. PowerPivot je doplněk aplikace Excel, který implementuje rychlou a silnou, in-memory databázi, která může být použita k organizaci dat, odhalování zajímavých vztahů a poskytování nejrychlejšího způsobu procházení informací. (Ferrari a Russo, 2013)

Aplikace Excel je ve skutečnosti databázový stroj v pozadí Power Pivot doplňku, poprvé uvedený pro Excel 2010. V aplikaci Excel 2010, byl Power Pivot samostatná jednotka. Doplněk se skládal z uživatelského rozhraní, stejně jako databázový engine. V aplikaci Excel 2013, došlo k rozdělení na dvě části a databázový stroj byl vložen přímo do aplikace Excel. Doplněk Power Pivot stále zůstává, ale už pouze jako uživatelské rozhraní. Je nutné použít Power Pivot doplněk, aby bylo možné využít některé složitější funkce. (Webb, 2014)

Některé z nejzajímavějších funkcí Power Pivotu:

- Neomezená množství uložených dat.
- Vytváření vztahů, hierarchií a KPI[10].
- Vysoká rychlost kalkulací.
- Jazyk DAX[11].

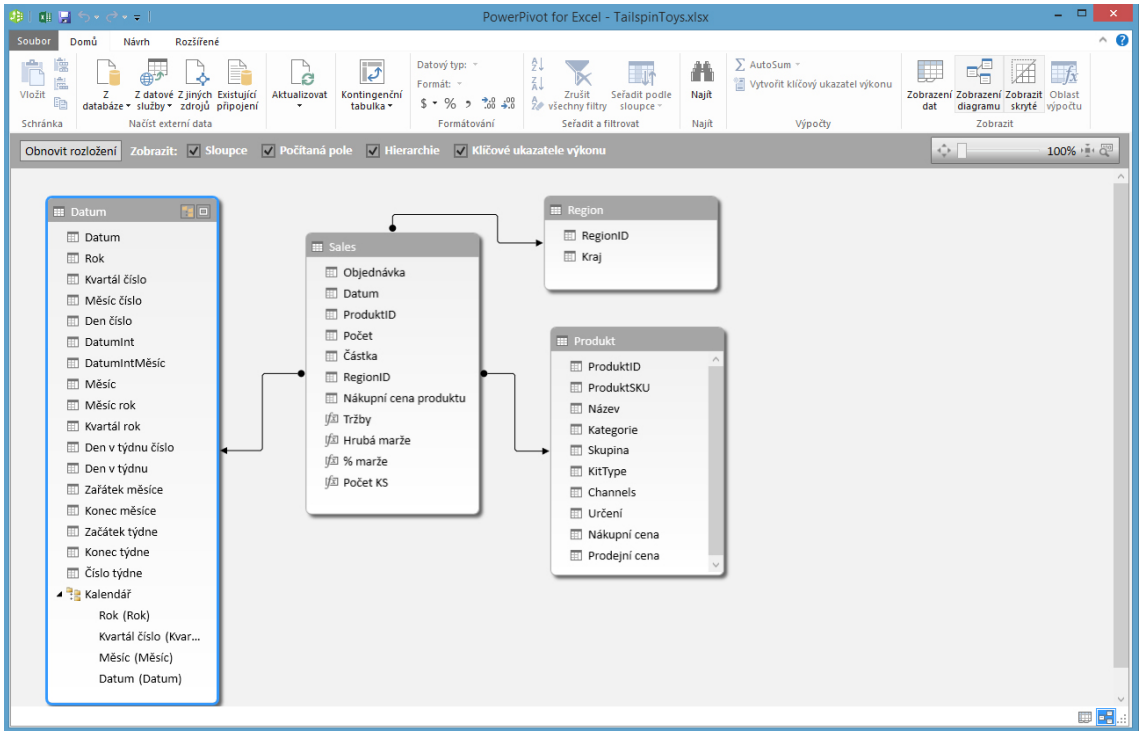
Množství uložených dat v datovém modelu je nesrovnatelně větší, než kdyby byla data uložena v listech Excel. Do tabulky v Excel 2013 je možné uložit maximálně 1 048 576 záznamů, zatímco datový model je omezen pouze dostupnou pamětí. Dále také díky xVelocity enginu je na data aplikována komprese v poměru přibližně 10:1, proto jsou sešity s datovým modelem velikostně mnohem menší a efektivněji pracují s daty.

Na obr.5 je znázorněno okno doplňku Power Pivot, ve kterém je definován vztah mezi jednotlivými tabulkami. V tomto diagramu je možné spojovat jednot-

[9]IntelliSense je obecný termín pro několik funkcí: seznamy členů, informace o parametrech, rychlé informace a dokončování slov. Zdroj: http://msdn.microsoft.com/cs-cz

[10]Key performance value

[11]Data Analysis Expressions

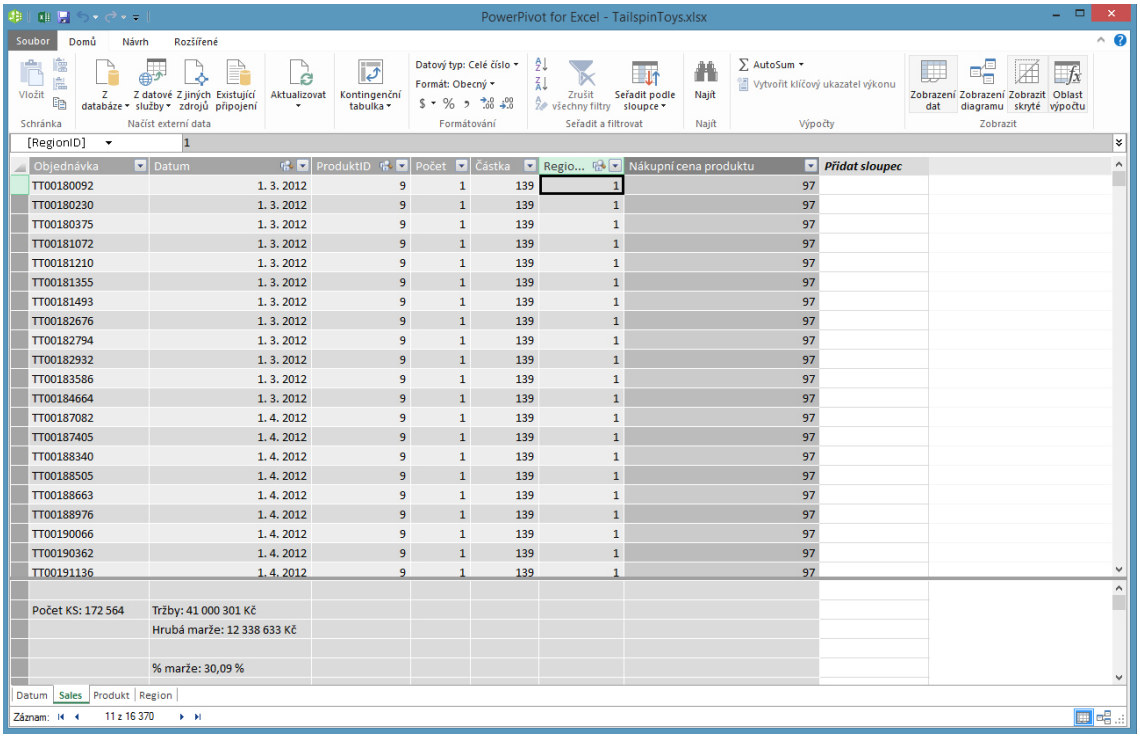


Obrázek 5: Zobrazení diagramu v Power Pivotu

livé tabulky pomocí atributů, vytvářet hierarchie a přejmenovávat sloupce tabulek. Jakmile jsou vztahy mezi tabulkami nastaveny, je možné využít Power Pivot pro výpočet složitějších kalkulací s datovým modelem.

Kalkulace v Power Pivot se dělí na dvě skupiny: kalkulované sloupce a kalkulovaná pole. Na obr. 6 jsou vidět v pravé části kalkulované sloupce a pod tabulkou část, kde se vypisují kalkulovaná pole. Rozdíl mezi nimi je především v kontextu, ve kterém se s nimi pracuje. Každý řádek v kalkulovaném sloupci má stejný vzorec. Oproti tabulce v Excel nemůžeme pro různé řádky použít různé vzorce, ale vzorec pouze jeden pro celý sloupec. (Collie, 2012) Naopak kalkulované pole se definuje pro celou tabulku jako agregační funkce. Největší síla kalkulovaných polí je v jejich možnostech filtrování a výpočtu odvozených kalkulací. Jednoduše je tak možné vypočítat kumulované tržby do určitého data, tržby pro stejné období minulého roku nebo procentní vyjádření tržeb v kategoriích a subkategoriích. Všechny tyto kalkulace se zapisují pomocí jazyka DAX.

Syntaxe DAX vzorců je velmi podobná vzorcům v aplikaci Excel, protože používá kombinaci funkcí, operátorů a hodnot. Kde se ale DAX vzorce liší od vzorců v aplikaci Excel je fakt, že funkce jazyka DAX umějí pracovat s tabulkami a sloupci, ne buňkami jako v klasickém excelu a nechají vás vytvořit složité vyhledávání na hodnoty ze souvisejících tabulek. Díky DAX vzorcům, můžete vytvořit agregace, které by běžně vyžadovaly hluboké znalosti o relačních databázových schématech nebo konceptu OLAP kostek. Navíc se na výpočty DAX vzorců využívá vysoce op-



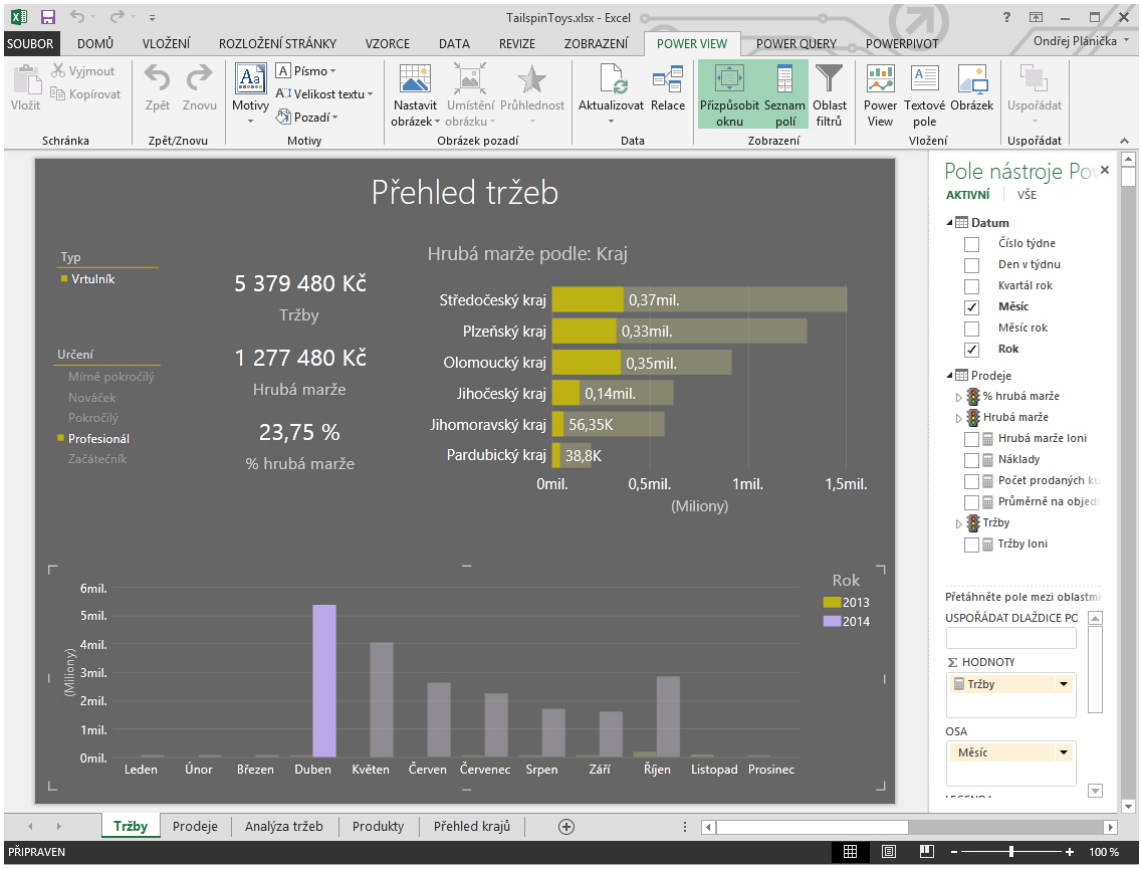
Obrázek 6: Zobrazení dat s kalkulovanými sloupci a poli

timalizovaný in-memory engine, díky kterému můžete rychle vyhledat a vypočítat hodnot napříč rozsáhlému počtu sloupců nebo tabulek. (TechNet Microsoft, 2014)

4.3 Power View

Power View report je speciální list v Excelu, sloužící k vizualizaci dat. Ve většině případů se jako zdroj pro Power View reporty využívá datový model v Power Pivotu, ale je také možné vytvořit report jen s dat tabulkách aplikace Excel. Nicméně datový model se dá mnohem lépe vyladit pro použití s Power View reporty než tabulky v Excelu. (Aspin, 2014)

Power View report si můžeme představit jako plátno, na kterém se zobrazují data z datového modelu. Pro zobrazení dat je možné využívat interaktivní grafy, tabulky a průřezy. Hlavní výhodou tohoto doplňku je jednoduchost vytváření jednotlivých reportů. Uživateli se v pravé straně zobrazí tabulky, se kterými může pracovat a pomocí přetažení může tato data vkládat do reportu. Jakmile má vytvořený pohled na data (například tržby z pohledu času) může si je rychle zobrazit pomocí různých grafů. Pro snadnější čitelnost dat a jejich procházení jsou všechny části grafu interaktivní. Pokud označíme část jednoho z grafů, ostatní také zobrazí jen hodnoty pro svou vybranou část, jak je možné vidět na obrázku 7. Za jímavou možností je také zobrazení geografických dat v mapě. Tato služba pracuje s Bing mapami, takže je nutné mít pro její správné fungování připojení k internetu.



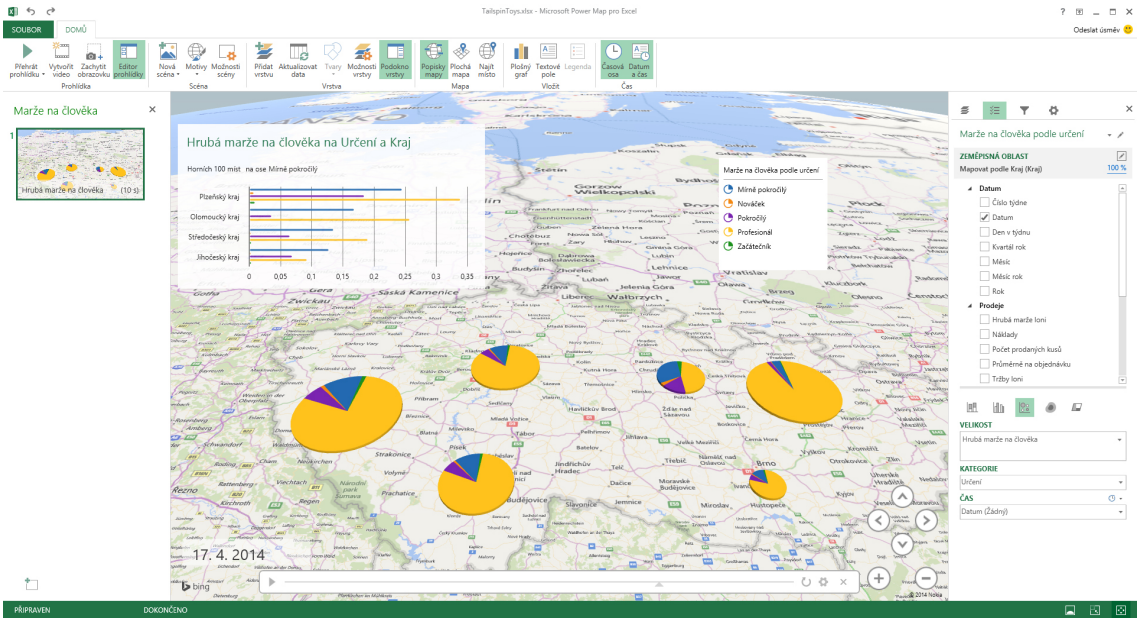
Obrázek 7: Power View report zobrazený v Excelu

Menší nevýhodou Power View je horší personalizace jednotlivých reportů. V reportech je nemožné přejmenovávat názvy metrik a sloupců, tudíž se vše musí řešit na úrovni datového modelu. Zvláště nepříjemné je to ve spojení s českým jazykem, protože Power Pivot má problémy se správností výpočtů u sloupců obsahujících diakritiku.

4.4 Power Map

Power Map slouží jako prezentační vrstva primárně pro data z datového modelu. Jak už jeho název napovídá, zaměřuje se na prezentování geografických dat. Rozdíl oproti Power View mapě je ten, že tato mapa je součástí programu Excel jako jakýkoliv jiný report a je možné využívat všechny možnosti filtrování a průřezu daty. Naopak v doplňku Power Map se vytváří exportovatelná videa, která mají více možností vizualizace a škálovatelnosti. (Aspin, 2014)

Důležitá je pro tento doplněk příprava datového modelu. Aby bylo možné správně zobrazit data na mapě, musí být v datovém modelu určeno, která data se týkají geoinformace. Pro data můžeme například určit, zda se jedná o město, stát, kraj nebo ulici. Podle zvoleného typu se budou informace zobrazovat na mapě.



Obrázek 8: Prohlídka vytvořená pomocí doplňku Power Map

Pokud ke každému zákazníkovi budeme mít přesnou adresu, můžeme si zvolit granularitu podle potřeby analýzy.

Report vytvořený pomocí Power Map se nazývá *prohlídka*. Každá prohlídka se skládá ze sérií *scén*, které mohou zobrazovat naprosto odlišné pohledy na data. V každé scéně může být několik vrstev dat, které se mohou navzájem doplňovat. Je tedy možné mít v mapě koláčové grafy znázorňující hrubou marži v jednotlivých městech a v druhé vrstvě graf, který bude zobrazovat tržby pro jednotlivé kraje (viz obrázek 8). Výhodou oproti mapách v Power View je možnost vytvoření časové osy, kterou je možné spustit a tím nechat data ožít. Jak se čas na ose posunuje, tak se mění i data zobrazená na mapě. Díky tomu je možné vidět, jak přesně se data v čase měnila, jak rostla či klesala. Pokud se tento efekt spojí s plynulým přechodem mezi jednotlivými scénami, které se odehrávají v jiné části mapy, výsledek je opravdu úchvatný. Za jímavou možností je vytváření mapy z vlastních podkladů. Datům jen stačí zadat souřadnice a může být vytvořena prohlídka. Použití je možné například pro sledování vytíženosti wifi hotspotů nebo tiskáren v podlaží určitých oddělení.

Tento doplněk může být využit pouze pro vytvoření videa, které je k další analýze bohužel nevyužitelné. Jediné využití tak nachází v prezentaci dat zákazníkům nebo svým nadřízeným.

4.5 Power BI

Cloudová služba Power BI, společně s aplikací Excel, poskytuje kompletní analytické self-service řešení. Za použití aplikací Microsoft Excel pro vytváření reportů a Power

BI pro Office 365 pro jejich sdílení, mají všichni lidé ve Vaší organizaci nový mocný nástroj pro práci s daty. (Microsoft, 2014)

Power BI, což je technicky vzato SharePoint knihovna, umožňuje načíst sešity do cloudu a podělit se o ně s vybranou skupinou spolupracovníků. A nejen to, s reporty je možné pracovat a používat filtry a průřezy. Power BI také umožňuje informačním pracovníkům sdílet dotazy a případně komplexní připojení na datové zdroje, které již mají vytvořené pomocí Power Query. Tímto způsobem může organizace zabránit zdvojování souborů s reporty, které může nastat, pokud si zaměstnanci reporty navzájem posílají přes email. Power BI může také zajistit, aby sešity aplikace Excel, které byly nasdíleny, byly aktualizovány automaticky a pravidelně tak, aby uživatelé pracovali vždy s nejnovějšími daty.

Podle (Aspin, 2014) jsou hlavní výhody použití Power BI tyto:

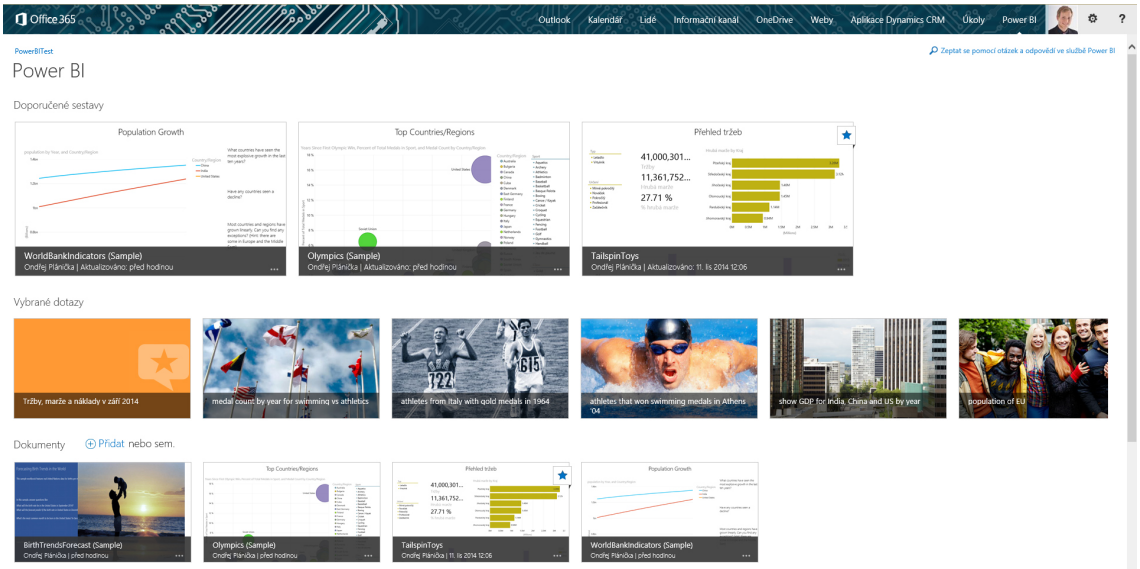
- Sdílení Business intelligence reportů v sešitech Excel v cloudu a umožnění kolegům procházet vytvořené reporty jen pomocí prohlížeče.
- Přístup k reportům pomocí nativní Power BI aplikace, která je vytvořená pro interaktivní práci s reporty na tabletu.
- Nastavení vloženým sešitům v Power BI stránce připojení k on-premise datovým zdrojům, tak aby bylo možné tyto sešity automaticky aktualizovat. Tímto je možné zaručit, že kolegové budou neustále používat aktuální data.
- Sdílení složitých dotazy na data.
- Přístup k on-premise zdrojům pomocí Power Query.
- Umožňuje uživatelům zeptat se na data, která jsou obsažena v jednotlivých sešitech pomocí přirozeného jazyka, bez potřeby znalosti učit se speciální dotazovací jazyk.

Cloudová služba Power BI se dá rozdělit na dvě části: část uživatelskou a část administrátorskou.

Uživatel může do společné knihovny sešitů nahrávat své vytvořené reporty. Jakmile jsou reporty nahrány, je možné s nimi dále pracovat. Reporty se mohou pro přehlednost seskupovat ve složkách a podknihovnách reportů. Power BI stránka může obsahovat desítky až stovky reportů, proto je zde možnost zařadit report do *doporučených sestav*, které se zobrazují ve vrchní části stránky s většími náhledy. Další možností odlišení je označení reportu jako *oblíbený*. Na rozdíl od zařazení do doporučených sestav, jde jen o zviditelnění v rámci svých reportů. Toto zvýraznění se například projeví v Power BI aplikaci na tabletech a na stránce *Moje Power BI*.

Celý smysl Power BI stránky je sdílení informací obsažených v sešitech programu Excel, které jsou na stránce umístěny. Podle toho, jak chceme s sešity pracovat, má tři stupně interakce:

- Zobrazit sešit ve službě Excel web app, kde můžeme pracovat s Power View reporty a filtrovat, procházet a hledat v datech.



Obrázek 9: Power BI stránka

- Upravit sešity v službě Excel online. Tento způsob je spíše pro tradiční data, která nejsou uložena v Power View a Power Pivot datovém modelu.
- Upravit sešity přímo v aplikaci Excel za podmínky, že máte na svém počítači nainstalovanou aplikaci Excel.

To jsou možnosti, které může uživatel využít v prohlížeči. V aplikaci pro tablety je pouze možné procházet, filtrovat v oblíbených reportech, popřípadě poslat snímek obrazovky kolegovi. V této době je aplikace dostupná pouze na platformu Windows 8.1, ale v nejbližší době by měla být rozšířena i na zařízení, která běží na iOS a Androidu.

Data Management Gateway

K aktualizaci reportů nahraných na Power BI stránce je zapotřebí nainstalovat Data Management Gateway na PC, který je v podnikové síti. Technická definice Data Management Gateway je „klientský agent, který poskytuje přístup k on-premise datovým zdrojům ve vaší organizaci.“ Tato „*brána*“ může být v síti nastavena více než jedna. Každá takto nastavená „*brána*“ může mít jiné nastavení a může zobrazovat jinak omezená data. (Aspin, 2014)

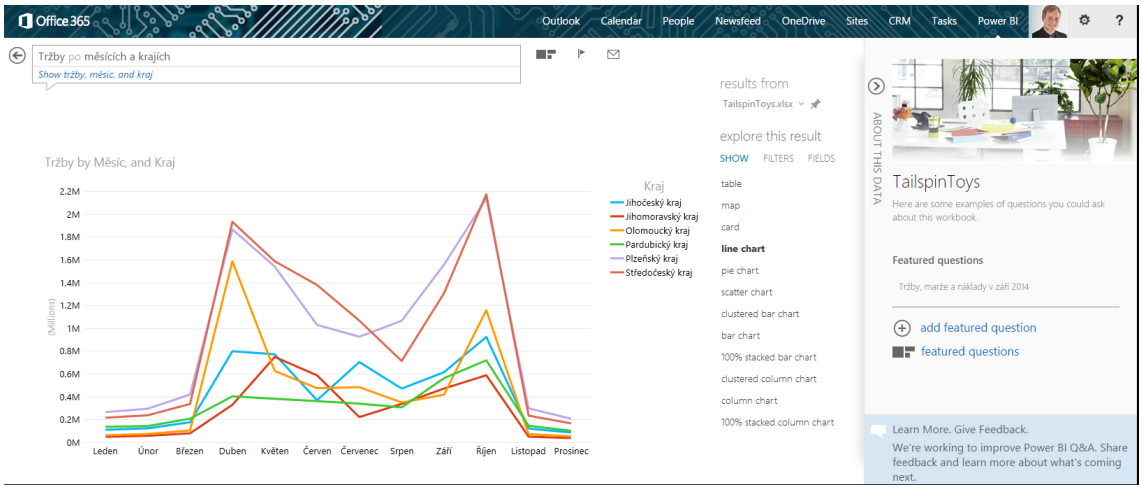
Jakmile je „*brána*“ nastavena, je nutné nastavit datový zdroj přímo ve službě Power BI. Tyto kroky se provádí v administrátorské části. Vytvoření datového zdroje se provádí v různých krocích, ve kterých se páruje „*brána*“ s určitým datovým zdrojem. Je zadána metoda ověření a uživatelé nebo skupiny, kteří mohou tento zdroj použít. Po úspěšném vytvoření datového zdroje se v nastavení reportu určí, v jakých intervalech a přes který datový zdroj se bude report aktualizovat. Pak už vše probíhá automaticky a uživatel má k dispozici stále aktuální data. Pokud v databázi

uživateli omezíme přístup jen k některým datům, tyto omezení se promítnou i do reportů.

V době psaní této práce byly přes Data Management Gateway přístupny pouze zdroje z SQL Serveru a Oracle databáze. Což znamená, že jakákoliv data z webu, ze souborů, z Facebooku a dalších není možné automaticky aktualizovat ve službě Power BI.

Otázky a odpovědi

Jednou z nejvíce impozantních částí Power BI je možnost dotazovat se na data pomocí českých vět. Tato technika, v originále *natural language querying*, může být použita pro jakýkoliv report, který byl nahrán do Power BI. (Aspin, 2014)



Obrázek 10: Q&A – Otázky a odpovědi

Dotazování funguje překvapivě spolehlivě, ale jsou zde samozřejmě různá jazyková omezení. Jedním z nich je optimalizace vyhledávače v pozadí na angličtinu. Proto není schopen rozeznat skloňování, jiné tvary slov a další specifické znaky českého jazyka. Čím lépe je vytvořený datový model, tím lépe funguje vyhledávání. Vyhledávač primárně sleduje názvy sloupců a kalkulovaných polí, podle kterých se snaží uživateli poskytnout co nejlepší výsledky. Ten, kdo report vytváří má možnost pro jednotlivé názvy zadat jejich synonyma. Díky tomu není uživatel odkázán jen na názvy sloupců, ale může vyhledávat podle více známějších a používanějších slov. Například pro slovo *tržby* můžeme vytvořit synonyma *prodáno* nebo *prodeje* a zeptat se na data pomocí těchto synonym.

Pokud víme, že nějaký dotaz se bude často používat a chceme ho uložit, je zde možnost vytvořit takzvanou *vybranou otázku*. Tato otázka se zařadí na hlavní Power BI stránku, kde bude doplňovat jednotlivé doporučené reporty.

Administrátorská část

Self-service bohužel neznamená „samospravované“, tudíž je potřeba konkrétní věci nastavit ručně. Power BI je vytvořen tak, aby bylo možné ho bez jakékoliv další úpravy používat. Pokud z něj však chceme dostat maximum, musíme použít jeho administraci.

Správce Power BI stránky má možnost přiřazovat uživatelům určité role v systému, aby vše nemusel řešit jeden člověk. Pokud bude potřeba vytvořit nový datový zdroj nebo nainstalovat data management gateway, může tuto činnost vykonat jiný uživatel, který bude mít příslušná práva.

Zajímavou skupinou uživatelů je skupina takzvaných *správců hlavních zdrojů*. Tato skupina má velice důležitou roli ve společnosti, protože se zaručuje za správnost datových zdrojů. Při sdílení dotazů je možné nastavit příznak, že tento dotaz byl schválen od správce hlavních zdrojů. Pouze a jedině uživatelé z této skupiny jsou schopni tyto příznaky měnit, ani administrátoři nemají takové právo.

Moje dotazy

Mimo společnou Power BI stránku je zde také osobní prostor, který se nazývá *Moje Power BI*. Na této stránce se jednoduše dostanete ke svým oblíbeným reportů a návodům, jak s Power BI pracovat. Další zajímavou možností je sledování svých vytvořených dotazů z Power Query. Je zde uveden přehled dotazů, které jste vytvořili a jejich stav. Stejné možnosti jsou pro datové zdroje, které jste nadefinovali. V neposlední řadě je zde možnost sledovat kolikrát byl váš dotaz vyhledán ve firmě pomocí doplňku Power Query a kolikrát byl aktuálně použit v různých reportech.

Zhodnocení

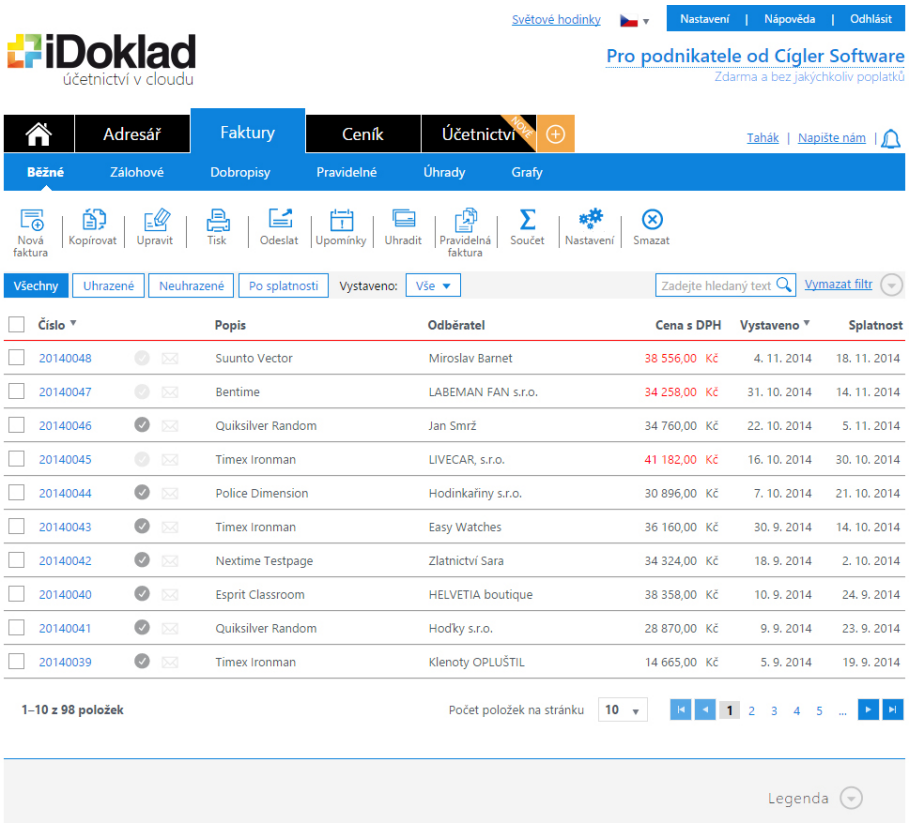
Cloudová služba Power BI je výborný doplněk pro již tak dobré BI nástroje v Excelu. Otázkou je, zda je v prostředí malých a středních firem nutná. Primárně je zaměřena na velké společnosti, ve kterých může dojít k určité desinformaci a nepřehlednosti ve sdílených složkách a v reportech posílaných v emailu. V malých a středních firmách s reporty pracuje jen hrstka zaměstnanců, kteří by natolik nevyužili potenciál, který jim tato služba nabízí.

4.6 iDoklad

Společnost Cígler Software, která stojí za úspěšným účetním systémem Money, vytvořila cloudovou aplikaci pro fakturaci s názvem iDoklad. Tato aplikace je kompletně zdarma a jejím úkolem je pomoci malým a středním firmám a podnikatelům se správou faktur.

První verze iDokladu byla spuštěna v březnu 2011 a nyní ji používá přes 50 000 podnikatelů v České republice a na Slovensku. Aplikace je pod neustálým vývojem a přibližně každé tři měsíce se přidávají nové funkce a možnosti.

Celá aplikace běží na platformě Microsoft Azure, což je cloudová služba pro hostování a běh aplikací v datacentrech Microsoftu. Díky tomu je práce s aplikací vždy svižná, protože výpočetní výkon v Microsoft Azure se automaticky zvyšuje, aby běh aplikace zůstal co nejplynulejší. Frontend aplikace stojí na technologii HTML5 a tím zaručuje bezproblémové zobrazení obsahu na jakémkoliv zařízení. Navíc je veškerá komunikace zabezpečena protokolem SSL, aby nedošlo k úniku dat uživatelů. (Cígler Software, 2014)



Obrázek 11: Prostředí aplikace iDoklad

Aplikace iDoklad slouží pouze k evidenci vydaných faktur, takže zde nenajdeme žádnou možnost pracovat se sklady, přijatými fakturami nebo docházkou. Rozhraní aplikace je rozděleno na čtyři základní nabídky: adresář, faktury, ceník a účetnictví.

V sekci *adresář* se ukládají kontakty na zákazníky, kterým se posléze vytvářejí faktury. Při zadání nového zákazníka je možné data vyhledat ze systému ARES[12] pomocí jeho identifikačního čísla a automaticky vyplnit v aplikaci. Systém také podporuje nahrání kontaktů z dalších systémů jako je Gmail nebo Microsoft Exchange.

Stěžejní částí celé aplikace jsou *faktury* (viz obrázek 11). Zde najdeme přehled všech faktur, které firma vydala. Přehledy jsou uvedeny pro běžné faktury, zálohové

[12]Administrativní registr ekonomických subjektů je informační systém, který umožňuje vyhledávání nad ekonomickými subjekty registrovanými v České republice. Zdroj: http://wwwinfo.mfcr.cz/ares/ares.html.cz

faktury, pravidelné faktury a dobropisy. Běžná faktura je daňový doklad a plátce DPH[13] ho vystavuje pro daňové účely. Fakturu daňový plátce musí vystavit nejdříve v den zdanitelného plnění (datum prodeje zboží, poskytnutí služeb atd.), nejpozději však do 15 dnů od uskutečnění zdanitelného plnění. Podnikatel, který fakturu připravuje, je zodpovědný nejen za termín vystavení, ale také za správnost uvedených údajů v souladu se zákonem o dani z přidané hodnoty. Zálohová faktura (proforma faktura) na rozdíl od faktury není daňovým dokladem. Časově zálohová faktura předchází před vystavenou fakturou. Používá se především pro uplatnění peněžitého nároku dodavatele na zálohu vůči odběrateli. Může také sloužit jako cenová nabídka. Zálohová faktura obsahuje standardní údaje jako faktura, jen nemusí mít rozepsáno DPH. Po uhrazení zálohové faktury odběratelem, má dodavatel povinnost vystavit daňový doklad - fakturu (Fakturman, 2012). Dobropis slouží k opravě nebo stornování běžné faktury. z běžné faktury je uživatel schopen vytvořit fakturu pravidelnou, která se bude generovat automaticky po zvoleném období. Dalšími podnabídkami na kartě *faktury* jsou úhrady a grafy. V úhradách je přehled o všech platbách za faktury, které proběhly. V detailu úhrady je možné sledovat, kdy platba za fakturu přišla. V nabídce grafů je vytvořený grafický přehled vydaných faktur, třech největších dlužníků a odběratelů.

Při zadání nové faktury uživatel zadá odběratele, kterému bude faktura vystavena. Následně vybere formu úhrady, jakou bude faktura zaplacena a datum její splatnosti. Následně může vybrat z ceníku zboží, které bude fakturovat a zadá požadované množství. Pokud produkt, který chce prodat není v ceníku, může do políčka pro položky na faktuře vepsat informace o novém produktu. Tento produkt se však následně do ceníku neuloží. Na konci faktury je možné zvolit slevu, která se na fakturu bude vztahovat.

V části *ceník*, je možné si předdefinovat položky, které se mohou na fakturách objevovat. Jakmile je produkt v ceníku, tak jeho přidání je mnohem rychlejší a pohodlnější. Pro produkty je možné volit druh sazby DPH, aby na faktuře byl uveden správný odvod DPH.

Nejnovější nabídkou je *účetnictví*, které se má primárně starat o platby za zboží v hotovosti. Je možné ukládat přijaté a vydané pokladní doklady a také vytisknout příjmový doklad při platbě ihned na místě. Další možnost je spárovat iDoklad se svým bankovním účtem, který může posílat přijaté platby na váš email. Aplikace tento email analyzuje a pokud najde shodu ve variabilním symbolu, čísle účtu a částce, tak připíše automaticky úhradu k této faktuře. Tuto funkci mohou prozatím využívat pouze zákazníci ČSOB, FIO, Raiffeisenbank a Komerční banky.

Služba iDoklad vystavuje data pomocí RestAPI, které komunikuje pomocí http protokolu. Uživatel posílá takzvaný *http request* na službu a ta mu odpovídá strukturovanou JSON[14] odpověď v podobě *http respond*. Autorizace dotazů na RestAPI iDokladu probíhá v hlavičce podle jedinečného zabezpečovacího tokenu. Token se

[13]Daň z přidané hodnoty

[14]Javascript object notation je vysoce strukturovaný formát pro výměnu dat. Zdroj: http://www.json.org/

získá z kombinace přihlašovacího emailu, znaku „|“ a hesla. Tato kombinace se ještě zašifruje pomocí hashovací funkce SHA256[15], stejnou funkci používají například datové schránky.

Pro kontrolu výstupu z RestAPI je doporučen program POSTman, který strukturovaně zobrazuje data které API vrací. Přímo na vývojářské stránce iDokladu je předdefinovaná kolekce požadavků, kterými můžeme s iDokladem komunikovat. Definovány jsou čtyři typy požadavků: *GET, POST, PUT, DELETE*. Požadavek *GET* se používá v případech, kdy chceme číst data z API. Pokud chceme vytvořit nové položky v iDokladu použijeme požadavek typu *POST*. Pomocí požadavků *PUT* můžeme hodnoty uložené v databázi aktualizovat, a mazat je požadavkem *DELETE*. Na základě těchto čtyř požadavků je možné vytvářet aplikace, které jsou schopny plně měnit obsah databáze iDokladu.

Ve své aplikaci používám pouze požadavky typu *GET*, protože mi stačí data pouze číst a není zapotřebí je měnit. RestAPI iDokladu obsahuje 13 tříd, které mají implementované metody díky kterým je možné s třídou pracovat. Příkladem třídy je například *Contacs, IssuedInvoices, Banks*. Adresa pro přístupu k API je *https://app.idoklad.cz/developer/api/*. Za tuto adresu se vždy doplní požadovaná metoda, která by se měla provést. Pro výpis všech zákazníků uložených v databázi by adresa vypadala takto *https://app.idoklad.cz/developer/api/Contacts*. API automaticky stránkuje výstupy po 20 záznamech, takže pokud má odpověď více než 20 záznamů bude mít více stránek. Pomocí parametru *PageSize* můžeme nastavit počet záznamů na stránce a to až do výše 200. Stránky se dají procházet pomocí parametru *Page*.

Každá odpověď z API má stejnou strukturu a obsahuje atributy *Data, TotalItems, TotalPages* a *Links*. V atributu *Data* jsou uloženy ty záznamy, které jsme požadovaly. Atribut *TotalItems* indikuje kolik je záznamů v databázi pro daný požadavek. *TotalPages* nám říká na kolik stránek se záznamy rozdělí (odvozeno podle parametru *PageSize*). Poslední atribut *Links* slouží k procházení stránek záznamů.

Aplikace iDoklad je velice jednoduchým a přehledným řešením pro fakturaci podnikatelů a menších firem, které nepotřebují žádné další moduly jako například sklady nebo přijaté faktury. Výhodou je jeho jednoduchost, rychlost a možnost fakturovat odkudkoliv bez potřeby mít nainstalovaný speciální software na počítači. Ale podle mého názoru je největší výhoda tohoto řešení to, že je pro všechny kompletně zdarma.

[15]Secure Hash Algorithm

5 Vlastní práce

V této kapitole budu popisovat postup vytvoření aplikace komunikující se službou iDoklad. Aplikaci jsem nazval *iSmart*, kvůli podobnosti s názvem iDokladu.

5.1 Vytvoření databáze v iDokladu

Základem práce bude fiktivní firma s názvem *Světové hodinky*, která se zabývá prodejem hodinek na internetu. Společnost podniká na trhu od ledna 2013 a do prosince 2014 má 33 zákazníků a prodává po celé České republice. O veškerou fakturaci se stará aplikace iDoklad, ze které se budou data importovat do BI řešení v aplikaci Microsoft Excel.

Nejdříve jsem musel vytvořit veškerou fakturaci této firmy, aby bylo možné aplikaci testovat a analyzovat výsledky. Veškeré zadávání údajů probíhalo na webu iDokladu. Pro testovací účely jsem vytvořil 33 produktů různých cenových kategorií. Poté bylo nutné vytvořit seznam odběratelů a zákazníků v sekci *Kontakty*.

Do databáze jsem uložil faktury za dva roky fungování firmy, ve kterých se společnost pohybuje na českém trhu. Firma primárně prodává do kamenných obchodů s hodinkami nebo šperky. Ve výjimečných případech prodává i jednotlivých osobám.

5.2 Ověření verze Excel a přítomnost Power Query

Jakmile byla připravena databáze, začal jsem vytvářet samotnou aplikaci iSmart. Jako první věc, kterou jsem vytvořil byly funkce, které ověřovaly zda má uživatel správnou verzi programu Excel a kontrolovaly, jestli má nainstalován doplněk Power Query. Ihned po otevření sešitu je zkontrolováno, zda se jedná o verzi aplikace Microsoft Excel 2013 nebo vyšší. Pokud je verze nižší, uživatele upozorní okno na to, že aplikace nemusí fungovat se starší verzí programu Excel. Dále se ověřuje pomocí funkcí *IsPowerQueryAvailable()* a *IsPowerQueryConnected()* (viz Příloha A.1), zda je nainstalován a povolen doplněk Power Query, který je nutný pro správné fungování celé aplikace. Pokud je vše v pořádku, uživatel může pokračovat.

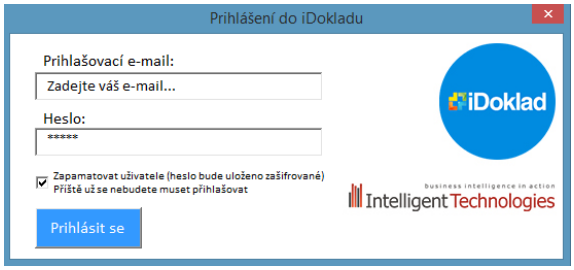
5.3 Autentizace uživatele

Pomocí zabezpečovacího tokenu, který jsem zmiňoval v předchozí kapitole, je systém schopný odlišit jednotlivé uživatele a je možné použít stejné dotazy z doplňku Power Query pro různé uživatele, kteří zadají své přihlašovací jméno a heslo. Aby bylo možné jednoduché přihlašování různých uživatelů, kteří mají účet v iDokladu, vytvořil jsem přihlašovací okno. V tomto okně se jednotlivý uživatelé přihlašují a pomocí makra se ověřuje, zda jsou zadané údaje správné.

Pro správu přihlašování jsem vytvořil list s názvem *Přihlášení k iDokladu*, kde je možné se k iDokladu přihlásit nebo data aktualizovat. Pokud je uživatel

již přihlášený a má uložené heslo, stačí mu kliknout na tlačítko *Aktualizovat data* a všechny informace se automaticky aktualizují.

Ještě než se uživatel začne přihlašovat je vhodné, aby na kartě doplňku Power Query zaškrtl možnost „Rychle zkombinovat“. Tato možnost zjednoduší komunikaci se serverem a nebude neustále uživatele vyzývat k nastavení úrovně soukromí, které by bylo nutné potvrdit pro všechny dotazy v sešitu.



Obrázek 12: Přihlašovací okno do iDokladu

V přihlašovacím okně (obrázek 12) uživatel zadá svůj email a heslo k iDokladu. Může také zvolit možnost uložení hesla v šifrované podobě pro další použití aplikací. Jakmile uživatel klikne na tlačítko *Přihlásit se* spustí se makro, které je svázáno s kliknutím na tlačítko. Toto makro (*PrihlasitSe_Click()* viz Příloha A.2) zkontroluje, zda uživatel vyplnil všechny potřebné údaje. Pokud byly údaje správně vyplněny uloží se email a heslo do skrytého listu *login*, ze kterého si tyto údaje bude načítat Power Query dotaz pro autentizaci. Poté se spustí metoda *GetSecureToken*, která spustí dotaz vytvořený v Power Query s názvem *GetSecureToken*.

V tomto dotazu (viz Příloha B.1) se čtou data ze skrytého listu *login*, kde je uložen email a heslo uživatele. Tyto údaje se vloží do požadavku a odešlou se pro ověření na API iDokladu. Pokud je kombinace emailu a hesla správná, tak API vrátí zabezpečovací token. Pokud se autentizace nezdaří, API vrátí chybu. V každém případě je výsledek uložen do skrytého listu *secureToken*. Zde se vytvoří tabulka s výsledkem dotazu. Vytvořil jsem si pojmenované pole do kterého kopíruji hodnoty z tabulky, abych k ní mohl přistupovat přes VBA kód.

Jakmile předchozí Power Query dotaz doběhne, pokračuje opět makro, které aktualizaci vyvolalo. z listu *login* vymaže uživatelské jméno a heslo, aby ho nemohl poté vyčíst někdo, kdo k němu nemá přístup. Poté na základě stejné hash funkce (SHA256 viz kapitola iDoklad), která se používá na straně API k vytvoření tokenu, vygeneruje zabezpečující token z uvedeného jména a hesla. Aby se zjistilo, zda uživatel zadal platné přihlašovací údaje, porovnají se hodnoty tokenů. Na jedné straně je token, který aplikace vytvořila sama a na druhé straně je ten, který jsme obdrželi z API. Pokud se tyto hodnoty rovnají, můžeme říct, že uživatel zadal správné údaje.

Následně zjistíme zda chtěl uživatel heslo uložit a na úvodní obrazovku vypíšeme informaci o tom, jestli je heslo uložené v sešitu nebo bylo vymazáno. Následně zobrazíme okno s informací, že přihlášení proběhlo úspěšně. Dále si také udržujeme informaci ve skrytém listu, zda ověření proběhlo v pořádku.

5.4 Spouštění aktualizace dotazů

Jakmile proběhne úspěšné přihlášení nebo uživatel klikne na tlačítko *aktualizovat data*, tak se spustí příslušná metoda (viz Příloha A.3). V této metodě se ověří, zda je nainstalován a povolen doplněk Power Query (pomocí funkcí, které jsme definovaly výše). Pokud je vše v pořádku a na skrytém listu *SecureToken* je uvedeno, že ověření proběhlo úspěšně, tak se zobrazí okno aktualizace a v pozadí se spouští jednotlivé Power Query dotazy, které se starají o načítání dat do datového modelu. Jen co všechny dotazy úspěšně skončí, objeví se uživateli informace, že jsou data aktualizována. Nyní uživatel načetl do aplikace aktuální data z iDokladu a může s nimi začít pracovat.

5.5 Power Query dotazy

Power Query dotazy které jsou součástí aplikace slouží převážně k importu dat z API do datového modelu Power Pivot. Sešit obsahuje 8 dotazů pro načítání dat a 4 funkce, které pomáhají při načítání dotazů po stranách.

Odpovědi z API iDokladu je nutné procházet po stranách, protože limit je pouze 200 záznamů na stranu. Proto jsem vytvořil funkce, které načítají data po stranách z API. Tyto funkce přijímají číslo strany jako parametr a vrací tabulku se záznamy. Jedná se o funkce *fnKontakty, fnVydaneFaktury* a *fnStaty*. *fnKontakty* se používají pro načtení dat zákazníků. *fnVydaneFaktury* pro načítání vydaných faktur, položek faktur a produktech. *fnStaty* stahuje informace o státech.

Abych spojil všechny strany do jednoho dotazu, vytvořil jsem primární dotazy pro jednotlivé třídy. Na začátku těchto dotazů je stejný požadavek jako v jeho funkci. z tohoto požadavku získám atribut *TotalPages*, který říká na kolik stran je požadavek rozdělen. Poté vytvořím list hodnot od jedné po počet stran, které jsem získal. Jakmile mám vytvořený tento list, mohu díky funkci Power Query přidat nový sloupec. Tento sloupec bude volat funkci pro konkrétní třídu (například *Kontakty*) a jako parametr bude posílat číslo stránky, které získá z listu. Díky tomu dostanu jednotlivé stránky záznamů pod sebou v jednom dotazu. Tento princip funguje stejně ve všech dotazech.

Další vytvořenou funkci jsem nazval *fnKraje* a slouží k přiřazení jména kraje podle PSČ. V této funkci (viz Příloha B.4) jsem vytvořil podle databáze České pošty proměnnou, která obsahuje dvojice prvních dvou číslic PSČ a názvu kraje. Tato funkce podle zadaného PSČ vyhledá ze seznamu a vrátí zpět název kraje, ve kterém se PSČ nachází. Funkce se používá při zpracování třídy *Kontakt*.

Dotaz Kontakt

Abych dostal seznam zákazníků, pro které jsem fakturoval, musím načíst data z třídy *Contact* v API iDokladu. Seznam načítám pomocí funkce *fnKontakt*, jejíž princip je popsán výše (viz Příloha B.2). Potom všem sloupcům nadefinuji datové typy, aby s nimi pak byla jednodušší práce v datovém modelu. Jakmile mám úplný seznam

zákazníků, přidám nový sloupec, do kterého funkcí *fnKraje* přidám název kraje, ze kterého zákazník je.

Dotaz FakturyVydane

Pro faktury vydané používám třídu *IssuedInvoices*, která obsahuje data o vydaných fakturách včetně jejich položek (viz Příloha B.3). Tudíž každý záznam pro fakturu může obsahovat ještě několik podzáznamů pro jednotlivé položky faktury. Po načtení všech dat pomocí funkce *fnVydaneFaktury* jsem musel vytvořit nové sloupce pro datumové atributy, protože API vrací datum ve formátu, které Power Query neumí identifikovat. Bylo tedy nutné vzít jen prvních 10 znaků z data, které bylo vráceno API a tuto posloupnost již bylo možné převést na datum. Následně jsem přejmenoval a vymazal nepotřebné sloupce a nadefinoval datové typy.

Dotaz Cenik

Při vytváření tabulky produktů jsem našel chybu, kvůli které jsem musel vymyslet úplně nové řešení. Nejprve jsem se snažil stáhnout data z API stejně jako u ostatních dotazů. Našel jsem si třídu *PriceListItems* a začal jsem stahovat data. Všechno vypadalo v pořádku až na některé položky, které se mi v seznamu neukazovaly. Nakonec jsem zjistil, že neplatí má domněnka, že všechny položky na faktuře jsou automaticky i v ceníku. Uživatel v iDokladu může na fakturu přidat buď položku z ceníku nebo může vepsat novou položku. Pokud tuto položku vepíše, tak se ale tato položka do ceníku neuloží. Tudíž jsem tento přístup nemohl použít.

Nakonec jsem vyšel ze stejného základu jako předchozí dotaz (viz Příloha B.4). Rozdíl je v tom, že z vrácených sloupců tabulky jsem nechal pouze sloupec s názvem *IssuedInvoiceItems*, který obsahuje další tabulku. V této tabulce jsou uloženy všechny informace o položkách, které se na faktuře objevily. z této tabulky jsem si vybral jen sloupce s názvem produktu a jeho jednotkou. Poté jsem na sloupec s názvem aplikoval transformaci, která vyřadila duplicity. Tím jsem získal jedinečný seznam všech produktů, které se kdy objevily na faktuře. Posledním krokem bylo přidání indexového sloupce, abych mohl tuto tabulku použít pro určení produktů z položek faktur.

Dotaz PolozkyFaktur

Pro položky faktur použiji stejný postup, jako v minulém dotazy (viz Příloha B.3). z tabulky *IssuedInvoiceItems* si vyberu sloupce, které budu moci použít pro analýzu jednotlivých položek. Upravím názvy jednotlivých sloupců a uspořádám je. Dále tento dotaz sloučím s předchozím dotazem *Ceník*. Tento proces je možné si představit jako příkaz *JOIN* v relační databázi. Tyto dva dotazy sloučím přes název produktu, který je v obou tabulkách. Po spojení dotazů odstraním nepotřebné sloupce a zanechám z dotazu *Ceník* pouze identifikátor. Následně už jen určím datové typy a dotaz uložím.

Dotaz Datum

Aby bylo možné data procházet v čase je nutné mít vytvořenou časovou dimenzi. Tu jsem vytvořil jen za pomoci Power Query. Díky možnosti vytvoření prázdného dotazu, do kterého je možné psát kód jazyka M, jsem mohl jednoduše vytvořit datumovou tabulku (viz Příloha B.5). Nejprve je nutné určit rozsah datumové tabulky a počet dní mezi tímto rozsahem. Jakmile mám počet dní, tak můžu využít funkce *List.Dates*, která od zvoleného data vytvoří seznam hodnot posunutý o určený interval. V tomto případě je interval jeden den. Jakmile mám seznam dat, mohu jednoduše pomocí kontextové nabídky ve vrchní části okna přidat další sloupce jako jsou rok, měsíc, den apod. Do této datumové tabulky ještě navíc přidám několik svých sloupců pro srozumitelnější a lepší přehlednost. Vytvořím sloupce, které obsahují název měsíce, měsíc s rokem, kvartál s rokem. Aby bylo možné v datovém modelu správně seřadit atribut měsíc s rokem, vytvořil jsem sloupec *MesicInt*, díky kterému je možné v datovém modelu tento sloupec řadit.

Dotaz MesicniPlan

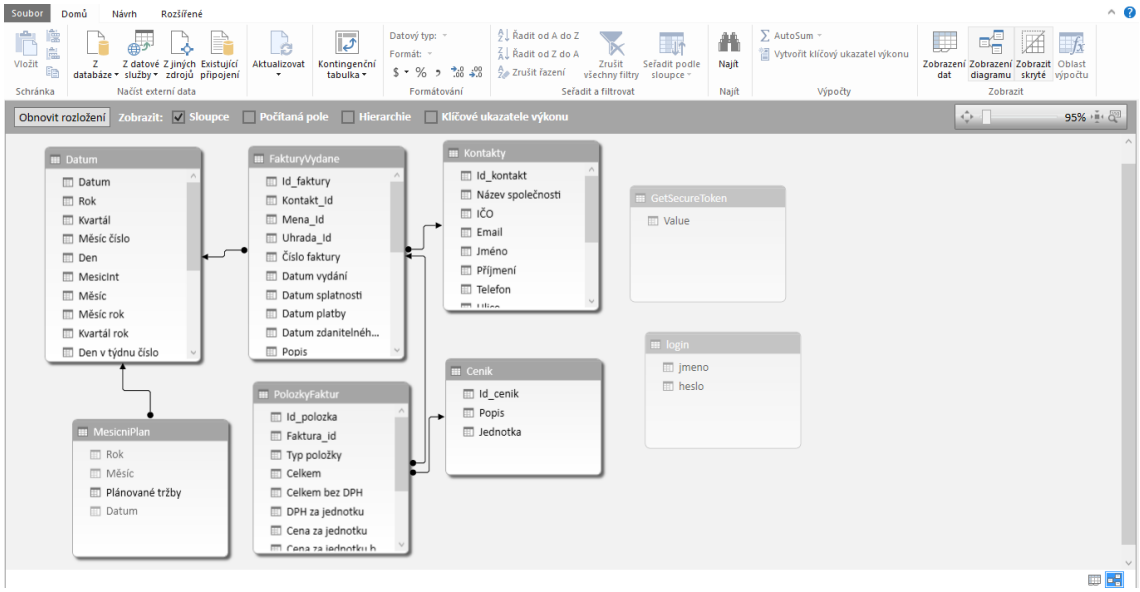
Tento dotaz se používá pro načítání dat z tabulky, která je součástí listu *Plánování tržeb*. Funkce tohoto listu bude vysvětlena níže při představování jednotlivých reportů.

5.6 Datový model

Po vytvoření všech dotazů v Power Query je možné začít vytvářet datový model v doplňku Power Pivot. Všechny dotazy, které byly nadefinovány v doplňku Power Query, se zobrazují v datovém modelu jako jednotlivé tabulky. Než začneme s daty pracovat a vytvářet kalkulace, musíme nejdříve mezi tabulkami vytvořit vazby. Přepneme se do zobrazení, kde je možné vidět diagram. V tomto diagramu jednoduchým přetažením z jednoho atributu na druhý vytvoříme vazbu, která data spojí. Po spojení všech tabulek získáme ucelený datový model, nad kterým můžeme začít vytvářet kalkulace pro naše reporty (viz obrázek 13).

U doplňku Power Pivot je potřeba dát si velký pozor, abychom neměnili názvy sloupců v tabulkách. Pokud bychom název změnili, nastane chyba při načítání nových dat z Power Query. Tato chyba může být velice nepříjemná a zničí celý datový model. Všechny vazby, které byly s tabulkou vytvořeny budou ztraceny a všechny kalkulace budou smazány. Tato špatná komunikace mezi těmito doplňky je velice nepříjemná a na základě toho často dochází k různým problémům s datovým modelem.

Abychom tedy mohli pokračovat ve vytváření uceleného datového modelu pro report, je zapotřebí data formátovat. V doplňku Power View je možná jen minimální forma editace formátu a popisu jednotlivých sloupců. Proto je nutné všechny tyto věci řešit již na úrovni datového modelu nebo přímo při načítání dat.



Obrázek 13: Datový model aplikace iSmart

Pro všechny vyjádření peněz jsem použil desetinné číslo. Všechna ostatní čísla se zobrazují jako celočíselné hodnoty.

V tabulce *Kontakty* je možné pro sloupce *ulice, město, PSČ* a *kraj* zadat kategorii dat. Pokud správně pro tyto sloupce zadáme kategorie (to znamená že určíme, že se jedná o lokality), je možné tyto sloupce použít při zobrazování údajů na mapě. Při vytváření reportu pak doplněk Power View v Bing mapách vyhledá hodnoty v těchto sloupcích a zobrazí je na mapě.

Dalším krokem v práci s datovým modelem bude označení tabulky kalendářních dat. To se provede jednoduše v tabulce *Datum*, kde v nabídce *Návrh* zvolíme možnost *Označit jako tabulku kalendářních dat*. Díky tomu budeme moci filtrovat data z této tabulky s ohledem na časovou posloupnost. Například se zpřístupní volby filtrů včera, dnes, minulý rok apod. V této tabulce jsem vytvořil několik kalkulovaných polí a přidal jsem kalkulované sloupce. V kalkulovaných polích získávám první a poslední rok, který se objevuje na fakturách a také první den v roce a konce aktuálního měsíce. Tyto hodnoty používám dále v kalkulovaných sloupcích. V těchto sloupcích si jen udržuji logickou hodnotu, zda datum spadá do definovaného intervalu. Hlídám například zda datum je z tohoto roku, tohoto měsíce, ze všech fakturovaných let nebo jen posledních 12 měsíců. Všechny tyto sloupce jsou pomocné a slouží výhradně pro zlepšení zobrazení kalendářních dat v doplňku Power View. Konkrétní příklady uvedu přímo při vytváření jednotlivých reportů.

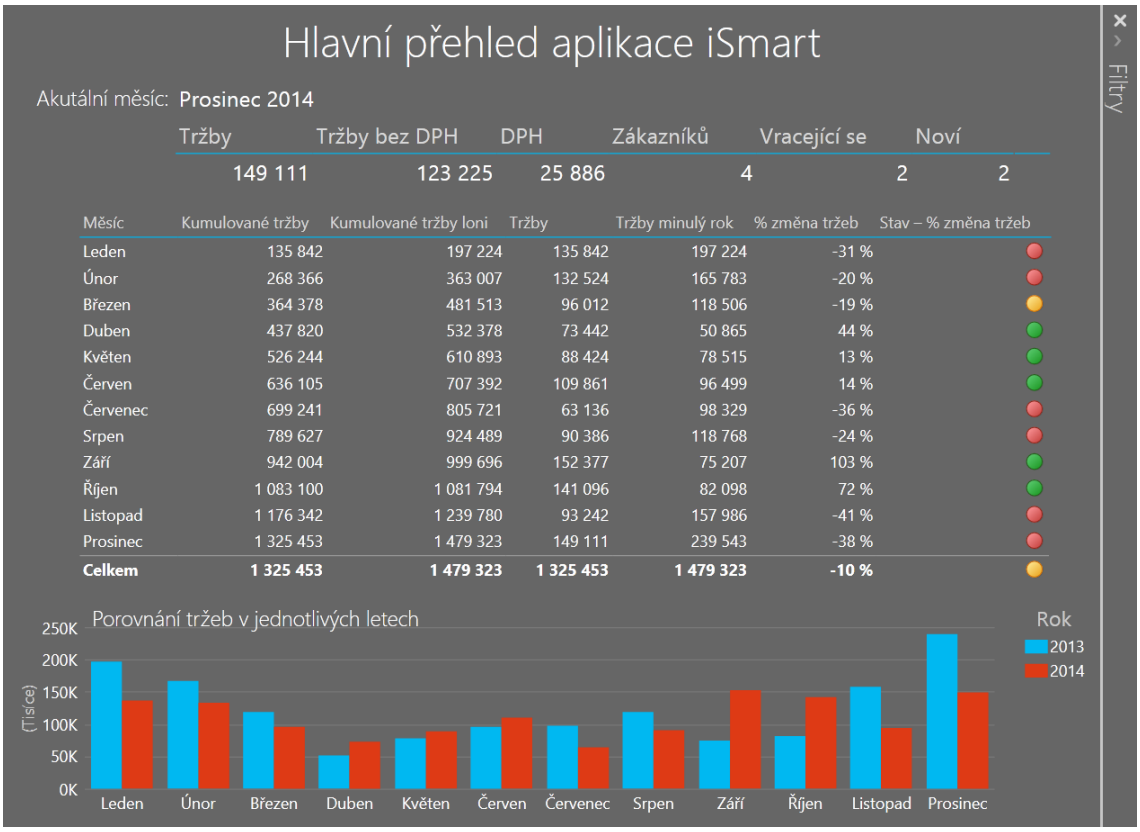
5.7 Vytváření reportů

Pokud je dobře vytvořený datový model Power Pivot, tak je vytváření reportů jen otázkou citu a zkušeností. Reporty jsem vytvářel primárně v doplňku Power View,

abych mohl ukázat jeho možnosti. Poslední dva reporty jsem vytvořil jen za pomoci kontingenčních grafů a tabulek. Reportů je v aplikaci dohromady pět a každý je zaměřen na jiný úhel pohledu na data. Pokud jsou grafy vytvořeny správně, informace je vnímána jednoduše a velmi efektivně. (Few, 2012)

Podle (Few, 2009) správné Business intelligence by se mělo snažit co nejvíce využívat přehledné vizualizace, kterými můžeme jednoduše rozšiřovat naše poznání.

Hlavní přehled



Obrázek 14: Hlavní přehled aplikace iSmart

Tento report slouží k rychlému přehledu o tržbách firmy a výsledkům v aktuálním měsíci. Je to základní srovnání mezi tržbami v jednotlivých měsících. V tomto reportu (obrázek 14) jsem použil několik kalkulovaných polí, které jsem si vytvořil v datovém modelu.

V horní liště používám kalkulovaná pole *Tržby, Tržby bez DPH, DPH, Zákazníci, Vracející se a Noví*. Pole *Tržby, Tržby bez DPH a DPH* získám pouhým součtem všech hodnot ve sloupci (viz Příloha C.1). Počet zákazníků získám funkcí *DISTINCTCOUNT*, která mi vrátí jedinečné výskyty identifikátorů tabulky Kontakt. Vracející se zákazníci jsou ti, kteří ve firmě již dříve nakoupili a nyní kupují znovu. Noví zákazníci jsou ti, kteří ještě nic nekoupili. Díky sloupci *Tento měsíc*,

který jsem definoval výše v tabulce Datum a který použiji jako filtr, mohu omezit data jen na aktuální měsíc.

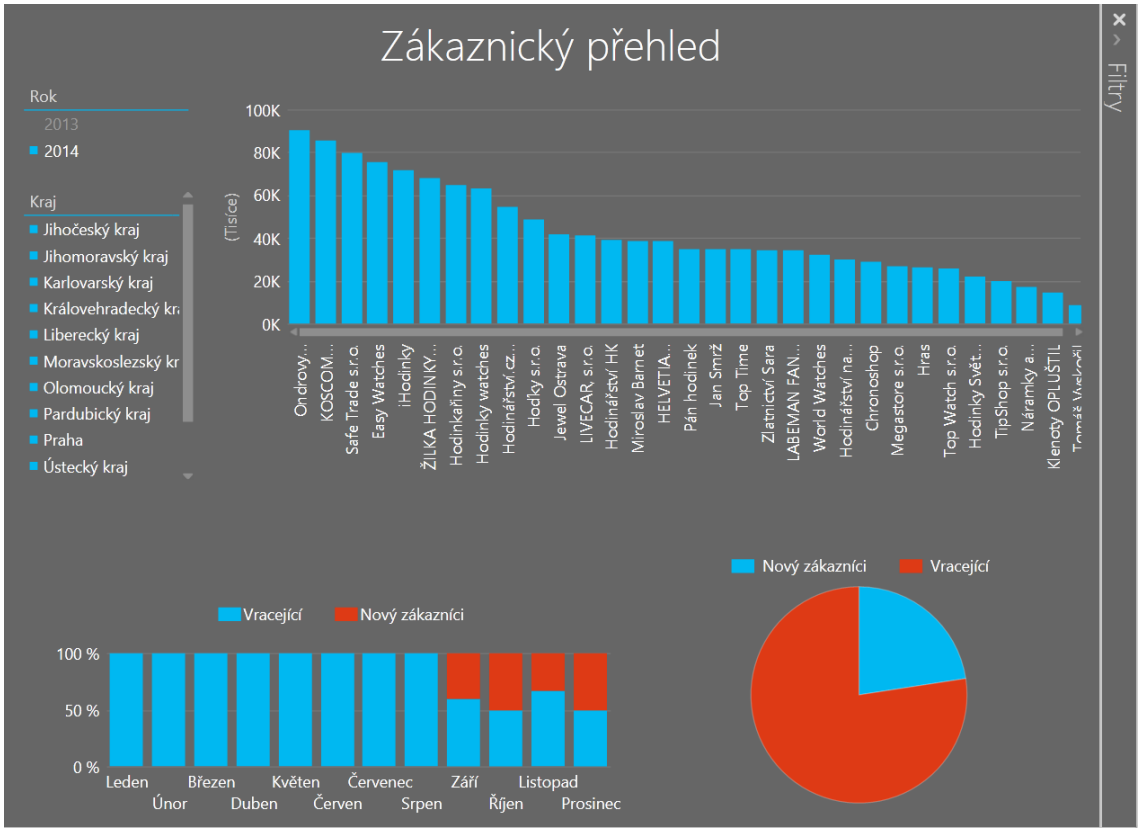
V tabulce uprostřed reportu porovnávám jednotlivé měsíce aktuálního roku s hodnotami z let minulých. Zde je velmi důležité, aby do pole filtrů této tabulky byl přidán atribut *Tento rok* z tabulky Datum. Tomuto atributu nastavíme hodnotu *True*, díky které se zobrazí jen data z tohoto roku. Důležité je to z důvodu porovnávání mezi letošním a minulým rokem. Pokud v systému nebude pevně určen jeden rok, tak datový model nebude moci vypočítat hodnoty pro minulé období. Pole *Kumulované tržby* v tabulce, je součet všech tržeb od začátku roku po zvolený měsíc. Hned vedle je možné porovnat tuto hodnotu z loňským číslem, které jsme jednoduše vytvořili pomocí jazyka DAX. To stejné platí pro sloupec *Tržby a Tržby loni*. Ve sloupci procentní změna tržeb počítáme zvýšení nebo snížení tržeb oproti minulému roku. Pro tento sloupec jsem vytvořil klíčový ukazatel výkony, který barevně znázorňuje stav oproti minulému roku. Pokud jsou tržby o 20 a více procent horší, tak je kolečko červené, od $-20\,\%$ do $2,5\,\%$ je kolečko oranžové a vše co meziročně stouplo o více než $2,5\,\%$ tak je zelené. Hranici $2,5\,\%$ jsem zvolil záměrně, aby se zelené kolečko zobrazilo jen pro měsíce, kdy došlo k růstu tržeb. Celá tabulka je ještě omezena filtrem *Rozsah měsíců*, který se stará o to, aby se nezobrazovala data pro minulé měsíce i když tento měsíc ještě nenastal.

Spodní graf zobrazuje porovnání tržeb mezi jednotlivými roky ve kterých firma fakturovala.

Zákazníci

Díky tomuto reportu má uživatel přehled o svých zákaznících. Může si zobrazovat své nejlepší zákazníky v jednotlivých krajích a má přehled kolik má nových a vracejících se zákazníků (obrázek 15).

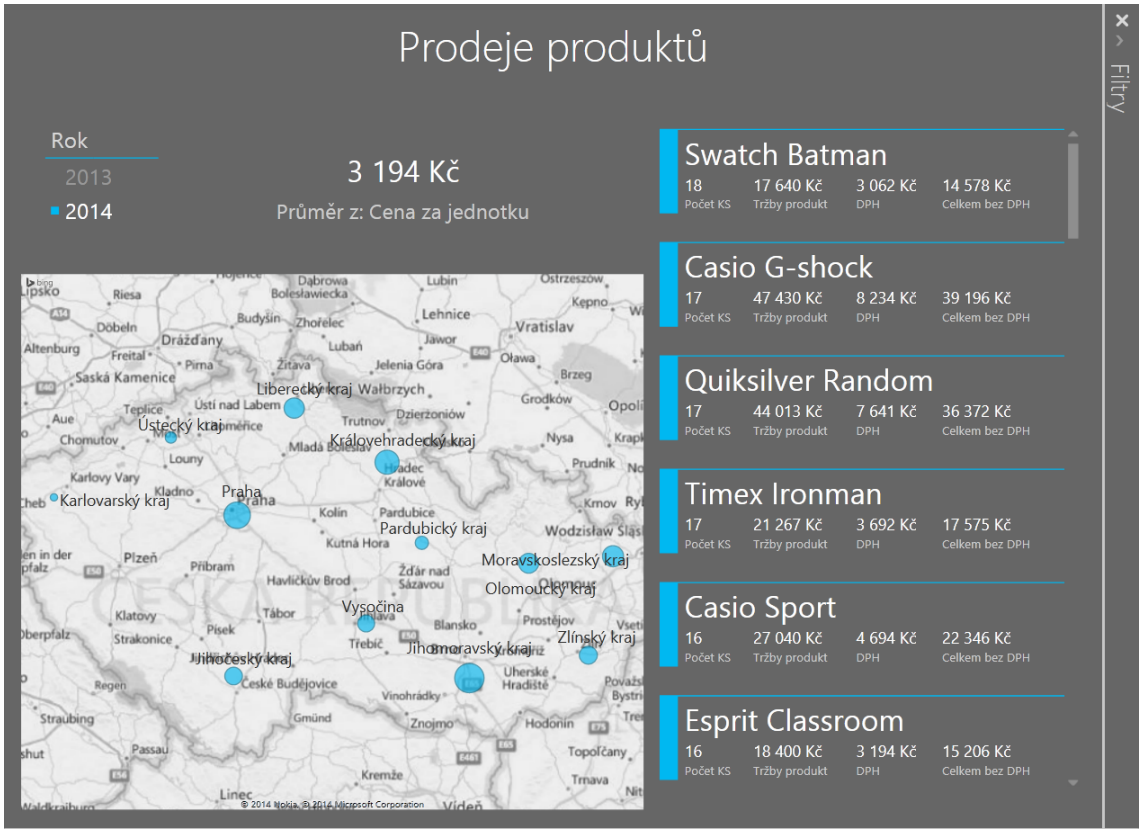
Jako první z celého reportu jsem si vytvořil hodnoty, kterými budu zobrazovaná data filtrovat. Aby bylo možné data filtrovat pomocí hodnot zobrazených v Power View okně, je potřeba vytvořit takzvaný *Průřez*. Tento průřez se vytvoří jednoduchým přetažení sloupce, podle kterého chceme filtrovat, do okna reportu. V tomto případě jsme přesunuli sloupec *Rok* a na kartě doplňku Power View jsme zvolili možnost *Průřez*. Díky tomu je možné zvolit jednu z hodnot a tím se celý report vyfiltruje jen na záznamy, které vyhovují podmínce. Ještě přidáme další filtr pro kraje stejným způsobem. Pro zobrazení tržeb jednotlivých zákazníků využijeme sloupcový graf, kde na osách budou názvy zákazníků a tržby. Celý graf si seřadíme od největších tržeb po nejmenší. Ve spodní části vytvoříme grafy pro analýzu vracejících se a nových zákazníků. První graf bude sloupcový skládaný, a bude sestávat z hodnot nových a vracejících se zákazníků po jednotlivých měsících. Díky tomu bude jasné vidět, jaký byl poměr zákazníků v jednotlivých měsících. Druhý graf bude koláčový, který bude zobrazovat celkový poměr nových a vracejících se zákazníků.



Obrázek 15: Zákaznický přehled

Produkty

Při analyzování prodeje produktů jsem využil další možnost Power View doplňku a tou je zobrazení dat do mapy (obrázek 16). Nejprve jsem si vytvořil filtr podle let prodeje produktů, stejně jako v minulém reportu. Následně jsem si do reportu vložit pouze jedno číslo a to hodnotu sloupce *Cena za jednotku*, ze které jsem si vytvořil průměr. Tímto průměrem budu moci porovnávat jednotlivé kraje z hlediska průměrné ceny na jeden produkt. Pak jsem vložil do okna Power View mapu. Můžeme vidět, že u sloupce *Kraj* je ikonka zeměkoule. To znamená, že tento sloupec je možné použít jako umístění na mapě. Pro umístění na mapě použijeme tedy sloupec *Kraj* a jako ukazatel velikosti bude sloužit počet prodaných kusů. Nyní je možné na mapě vidět různě velké koláčové grafy v jednotlivých krajích. Pokud klikneme na jeden z grafů, změní se i průměr, který jsme si definovali dříve. Posledním krokem bude přidání seznamu produktů, které se prodaly. k tomu stáčí jen přetáhnout sloupce, které chceme sledovat do reportu a vytvořit seznam pomocí vizualizace *Karta*. Tak nám vznikne přehledný seznam, který se filtruje podle zvoleného kraje a roku.



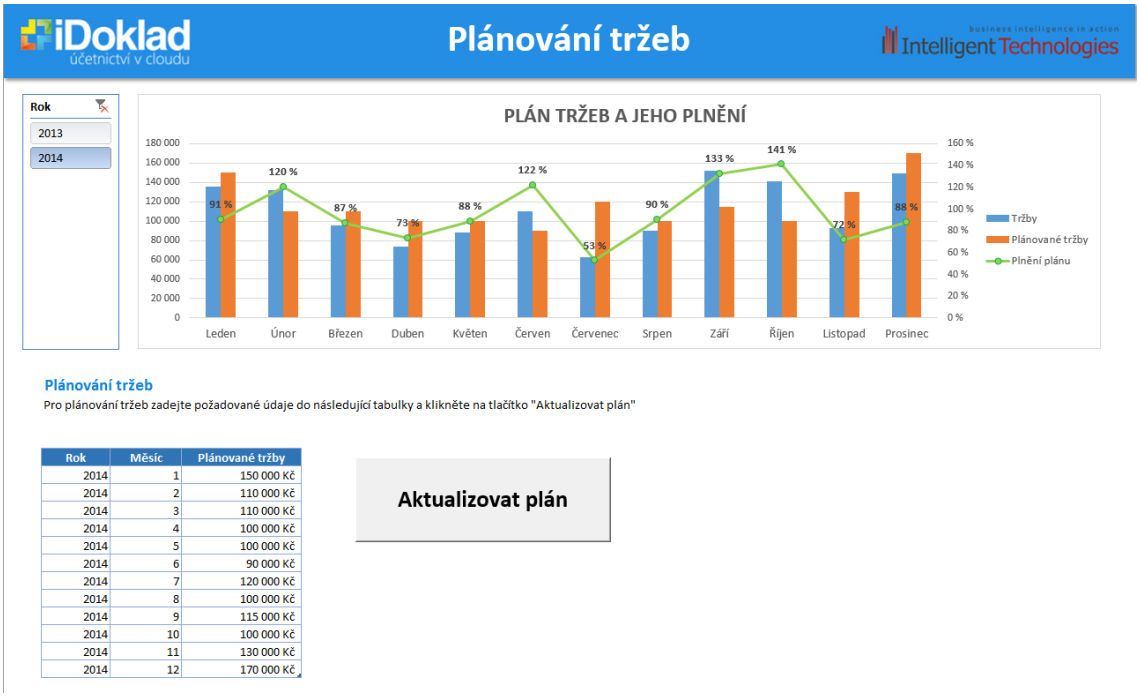
Obrázek 16: Přehled prodeje produktů

Plánování tržeb

Aby nebyla aplikace iSmart jen o prozkoumávání dat a hledání souvislostí, implementoval jsem možnost si dopředu zvolit plány tržeb (viz obrázek 17). Uživatel má možnost naplánovat si objem tržby, které chce v dalším období provést. Do tabulky vloží hodnoty pro jednotlivé měsíce a klikne na tlačítko *Aktualizovat Plán*. Po stisknutí tlačítka se spustí dotaz Power Query, který načte tuto tabulku do datového modelu. Zde se provede výpočet plnění plánu (viz Příloha C.2), který porovnává tržby s jejich plánem a vrátí procentuální vyjádření tohoto plnění. Společně s aktualizací dat v datovém modelu se také aktualizuje graf v reportu, tím pádem uživatel vidí ihned změnu. V jednom sloupci jsou vyjádřeny aktuální tržby, v druhém sloupci jsou plánované tržby a spojnicový graf zobrazuje procento plnění plánu.

Přehled DPH

Posledním z reportů je přehled o DPH. Tento report má za úkol pomoct při sledování výše odvedeného DPH pro plátce DPH. Pro neplátce DPH zase sleduje objem tržeb za posledních 12 uplynulých měsíců a kontroluje zda neplátce nepřekročil zákonem



Rok	Měsíc	Plánované tržby
2014	1	150 000 Kč
2014	2	110 000 Kč
2014	3	110 000 Kč
2014	4	100 000 Kč
2014	5	100 000 Kč
2014	6	90 000 Kč
2014	7	120 000 Kč
2014	8	100 000 Kč
2014	9	115 000 Kč
2014	10	100 000 Kč
2014	11	130 000 Kč
2014	12	170 000 Kč

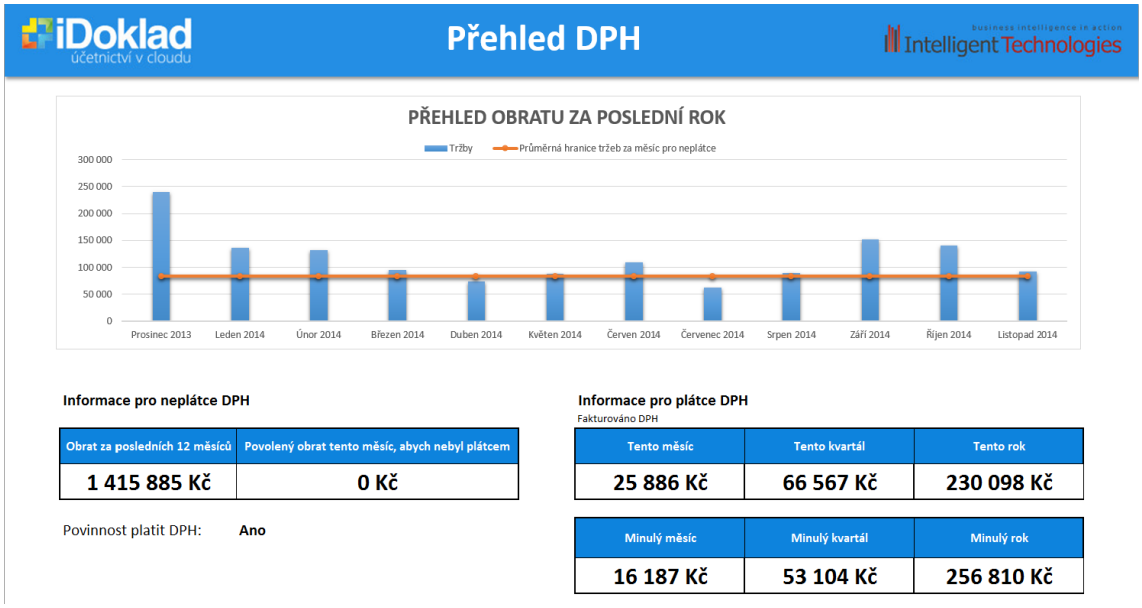
Obrázek 17: Plánování tržeb

stanovenou hranici 1 000 000 Kč. Pokud by tuto hranici překonal stal by se automaticky plátcem DPH (obrázek 18).

Aby bylo možné report udělat přehledný a upravený, musel jsem vytvořit skrytý list do kterého jsem načetl data jako kontingenční tabulku. Buňky, které se zobrazují v reportu se na tyto data pouze odkazují. Tímto jsem mohl report vytvořit do podoby, jakou jsem sám chtěl a nemusel jsem se omezovat možnostmi kontingenční tabulky.

Tržby za poslední rok dostanu výpočtem tržeb od stejného měsíce minulého roku až po předcházející měsíc (viz Příloha C.1). *Povolený obrat* uživatele informuje kolik tento měsíc ještě může utržit, než se stane plátcem DPH. Abychom získali tuto částku, tak musíme od čísla 1 000 000 odečíst tržby za poslední rok a přidat částku, která představuje tržby za první měsíc posledního roku. Je to proto, že jakmile se dostaneme do dalšího měsíce, tak se posuneme o měsíc v kalendáři a tyto tržby se už do výpočtu nebudou započítávat. Graf znázorňuje tržba firmy a linka je pomyslná průměrná měsíční tržba, přes kterou bychom se neměli dostat, abychom se nestali plátci DPH.

Druhá část reportu je zaměřená na uživatele, kteří již jsou plátci DPH. Jsou zde přehledné informace o zaplaceném DPH a porovnání s minulými obdobími.



Obrázek 18: Přehled DPH

5.8 Power Map a Power BI

Ještě předtím než celý hotový report nahrajeme do cloudové služby Power BI, vytvoříme si ještě prezentační video pomocí doplňku Power Map. Abychom video vytvořili stačí otevřít doplněk v záložce *Vložení* a zahájit novou prohlídku. Díky tomu, že jsme si připravili datový model pro práci s geografikými daty, bude vytváření videa velice jednoduché. Do pravé spodní části vložíme úroveň, ve které chceme data zobrazovat. Já jsem zvolil *Kraj*, jelikož pro něj budeme mít nejvíce dat. Poté už stačí jen vybrat podle kterého atributu se bude určovat výška sloupce na mapě a můžeme vytvářet. Vytvoříme dvě scény, kde každá bude sledovat jinou část republiky. První scéně se nastaví doba trvání jen 5 sekund a bude zobrazovat data jen od 1. ledna 2013 do 31. prosince 2013. Druhá scéna bude zaměřena na jinou část republiky, aby při přechodu se musela kamera přesunout. Pro tuto scénu nastavíme data trvání od 1. ledna 2014 do 31. prosince 2014. Nyní již stačí jen vytvořit požadované video, zvolit kvalitu výstupu a uložit na disk.

Nyní můžeme přistoupit k nahrání reportu do cloudové služby Power BI. Otevřeme si v prohlížeči Power BI stránku a pokusíme se vložit report do knihovny dokumentů. Zde dochází k prvnímu problému, Power BI neumožňuje nahrát soubory Microsoft Excel s podporou maker. Budeme tedy muset výslednou aplikaci uložit jako klasický soubor aplikace Excel. Tímto krokem ztratíme veškerý kód, který jsme v pozadí vytvořili. Tudíž nebude fungovat přihlašování, aktualizace dat a plánování, ostatní budou zachovány. Po nahrání reportu na Power BI stránku můžeme report otevřít v prohlížeči a pracovat daty jako na počítači, zde se nic nemění.

Nyní budeme chtít data procházet pomocí funkce *Otázky a odpovědi*. Abychom toho docílili, stačí v nastavení reportu zvolit „přidat do otázek a odpovědí“.

Po zadání jednoduchých dotazů na tržby nebo zákazníky, jsem nebyl schopen ze systému dostat smysluplná data. Je potřeba přidat k jednotlivým názvům také synonyma, aby systém lépe s datovým modelem pracoval. Takže je nutné se opět vrátit zpět k datovému modelu v Power Pivot a v záložce *Rozšířené* otevřít nabídku synonyma. Zde je možné k názvu každého sloupce a kalkulovaného pole v datovém modelu, přidat podobné slovo, které bude také vyhledatelné v otázkách a odpovědích. Vytvořím synonyma pro tržby, zákazníky a datumovou tabulku a report nahraji zpět do Power BI. Bohužel se práce s otázkami a odpovědmi vůbec nezlepšila a systém se chová velice nestandardně. Na vinně je především to, že systém neumí pracovat s českými názvy sloupců.

Posledním věcí, kterou Power BI nabízí, je automatická aktualizace reportu z cloudu. Chtěl jsem tedy nastavit, aby se moje vytvořená aplikace, sama aktualizovala po hodinových intervalech. To ale naneštěstí také není možné, protože Power BI povoluje automatickou aktualizaci zatím jen z on-premise datových zdrojů. Tím, že aplikace načítá data z API iDokladu, ji není možné aktualizovat přes Power BI. Takže jediná možnost jak aktualizovat data v cloudu, je aktualizovat reporty na desktopu a nahrávat nové verze do Power BI.

6 Zhodnocení a diskuze

Celá platforma Power BI se stále rychlým tempem vyvíjí a Microsoft se snaží přinášet aktualizace doplňků v co nejmenších časových intervalech. Například Power Query je standardně aktualizováno jeden krát za měsíc a s každou novou verzí přibývá více funkcí a práce je stabilnější. Jen po dobu vytváření aplikace iSmart vyšly tři nové verze, které značně zlepšili práci na aplikaci.

6.1 Zhodnocení implementace

Implementace aplikace iSmart neprobíhala tak jednoduše, jak Microsoft prezentuje ve svých propagačních videích.

Ze začátku byly velké problémy přímo se spojením s iDokladem, protože Power Query neuměl posílat požadavky na server, které by obsahovaly hlavičku. Tím pádem nebylo možné autentizovat uživatele a aplikace by byla velmi špatně ovladatelná.

Transformace dat v Power Query za pomocí kontextových nabídek a funkcí fungovala naprosto jedinečně. Je to velice silný nástroj a dokážu si představit i netechnického člověka vytvářet jednoduché dotazy na data. Problém přišel ve chvíli, kdy člověk musel zasáhnout do kódu jazyka M. Editor tohoto jazyka je jedna z nejhorších částí celé platformy. Práce v tomto editoru je nepřehledná, zdlouhavá a často se stává že se kód musel vytvářet celý znovu od začátku.

Nevyřešeným problémem je také komunikace mezi doplňky Power Query a Power Pivot. Pokud jsem změnil jen některou část tabulky importované z Power Query v datovém modelu, tak nastala chyba. Power Pivot neumí předat informace do Power Query o tom, že se struktura tabulky změnila. To platí také o určení datových typů sloupců, všechny tyto věci je nutné řešit už v Power Query.

Jazyk DAX je velice mocný nástroj pro zkušeného analytika. Vytváření složitých kalkulací je naprosto nepředstavitelné v prostředí klasického sešitu Excel. Ale rychlost a komplexnost s jakou to zvládá datový model je přímo vynikající.

Vytváření reportů pomocí doplňku Power View je jednoduché a rychlé. U tohoto doplňku si dokážu představit, že nezkušený manažer firmy je schopen vytvořit reporty, které budou dávat nějaký smysl. Interaktivita reportů je jedna z nejlepších, které jsou na trhu k vidění. Možnost prozkoumávat data kliknutím (nebo dotykem) na nějaký graf, dává lidem nové příležitosti najít v datech nový význam. Bohužel možnosti personalizace reportů jsou na bídné úrovni. Uživatel nemá možnost měnit názvy sloupců, barvy jednotlivých grafů ani popisků dat. Pokud chce uživatel změnit název sloupce v Power View, musí jít až do dotazu Power Query, kde název sloupce může změnit. Pokud však datový model, používal tento název pro výpočet nějakého kalkulovaného pole, bude se muset tato kalkulace přepsat. Je to velice kostrbaté a zbytečně to zhoršuje práci s jinak výborným doplňkem.

Pro výstup z Power Map jsem po dobu vytváření celé aplikace nemohl nalézt rozumné použití. Tento doplněk je z mého pohledu zbytečný a těžko bych pro něj hledal uplatnění.

Problémy s cloudovou službou jsem popsal již v předchozí kapitole, ale jedná se o problémy jen s touto konkrétní aplikací, která načítá data z webového API. Cloudová služba Power BI je spíše doplněk k již zavedenému podnikovému BI. Možnosti, které přináší se spíše hodí pro firmy, kde je potřeba udržovat v reportech přehled a mít je neustále k dispozici a aktuální.

6.2 Ekonomické zhodnocení

Z ekonomického hlediska je toto řešení Business intelligence ve firmě jedno z nejlepších. Microsoft výborně využívá toho, že má většinový podíl na trhu s kancelářskými nástroji a dělá vše proto, aby to tak zůstalo. Představa toho, že firma může mít kvalitní Business intelligence řešení prakticky zadarmo (spousta firem již má Excel zakoupený), je velice lákavá.

Představme si střední firmu s přibližně 100 zaměstnanci. 15 lidí z této firmy bude potřebovat přístup k Business intelligence a tým který vytváří reporty se skládá ze tří lidí. Firma by chtěla vytvořit sérii reportů pro svá čtyři oddělení. Pokud by firma neměla žádný software, tak při aktuální ceně 1 200 Kč/měsíc/uživatel (Power BI s Office 365 Pro Plus), byl by její roční náklad na BI 259 200 Kč.

Při využití softwaru Tableau by firma musela zaplatit tři licence za cenu přibližně 46 137 Kč pro vývojáře a navíc ještě 962 Kč/měsíc/uživatele za cloudové sdílení. Při ročním zúčtování by se náklady na toto řešení vyšplhaly na cenu 346 203 Kč[16]. Cloudové služby Tableau Software vychází levněji než u Power BI, ale zase je nutné započítat dražší licence pro vývojáře řešení, což jeho cenu výrazně zvyšuje.

Cena řešení na platformě GoodData se odvíjí od více faktorů než jen počet uživatelů. Licence se dělí do různých kategorií, podle počtu projektů a podle počtu dashboardů. Nejzákladnější licence *GoodData Base* je omezena na dva projekty, pouze jeden dashboard a stojí přibližně 11 540 Kč/měsíc. Pokročilejší verze *GoodData Plus* je omezena pěti projekty, ale už není omezena počtem dashboardů. Tato verze stojí 23 080 Kč/měsíc. Nejlepší edice *GoodData Premium* je omezena pouze na deset projektů a rozšiřuje možnosti GoodData o další funkce. Tato edice stojí 115 400 Kč/měsíc[17]. Žádná verze není omezena počtem uživatelů. Pro vzorovou firmu by tedy cena softwaru vyšla na 276 960 Kč za rok. Ale není zde započítána ještě cena vytvoření všech reportů, které si GoodData účtuje.

Při porovnání všech tří řešení vychází, že cena platformy Power BI je nejnižší. Díky tomu má Power BI platforma nespornou výhodu oproti konkurenci. Je levnější, uživatelé jsou zvyklí používat každodenně Excel. Vše se dá vyřešit pomocí klasického

[16]Zdroj: https://tableau.secure.force.com/webstore

[17]Zdroj: http://web.archive.org/web/20100811023135/http://www.gooddata.com/pricing/

desktopu nebo notebooku. Cena licencí může být také nižší díky tomu, že velká část firem má již nakoupený Excel.

Vytvoření aplikace iSmart zabralo přibližně 100 hodin, tudíž při mzdě 250 Kč/hodinu náklady na jeho vytvoření byly 25 000 Kč.

6.3 Diskuze

Aplikace iSmart byla vytvářena na zakázku pro firmu Intelligent Technologies, která chtěla prozkoumat možnosti platformy Power BI ve spojení se službou iDoklad. Po úspěšné implementaci aplikace jsme kontaktovali firmu Cígler Software, a. s. S návrhem o propagaci aplikace jako primární BI aplikace pro iDoklad. Po měsíci, kdy byla aplikace testována firmou Cígler Software, jsme se s firmou domluvili, že se aplikace iSmart stane oficiálním Business intelligence řešením pro iDoklad.

Platforma Power BI se osvědčila jako nejlepší pro možnosti šíření této aplikace, protože není zapotřebí žádný jiný software. Díky tomu je použití aplikace více uživateli jednodušší. Aplikaci bylo možné poskytnout ke stažení na webových stránkách, kde si ji uživatelé iDokladu mohou stahovat a zadarmo používat. Výsledný soubor je možné otevřít a používat přímo v aplikaci Excel 2013.

Pokud by měla být aplikace vytvářena na jiné platformě, nebylo by tak jednoduché ji rozšířit mezi větší počet uživatelů. V GoodData by nebylo možné řešení poskytnout veřejnosti vůbec, protože se jednotlivé projekty omezují pouze na jednotlivé uživatele. To znamená, že by každý uživatel, který by chtěl tuto aplikaci využít, musel platit licenční poplatky. A to by pro zákazníky bezplatné fakturace nebylo přijatelné. Podobný případ je se softwarem Tableau, u kterého by bylo nutné zakoupit vývojářskou licenci v ceně 46 137 Kč. Jednotlivé reporty by bylo nutné procházet jen s pomocí speciálního prohlížeče. Největším úskalím by u těchto dvou řešení byla nemožnost přihlášení různých uživatelů.

Tento problém se dá jednoduše vyřešit pomocí VBA kódu a prostředí aplikace Excel. Díky tomu je možné s daty více pracovat a upravit aplikaci přesně podle svých preferencí. Proto je podle mého názoru Microsoft Excel s doplňky Power Query, Power Pivot a Power View nejlepším nástrojem pro vytvoření této aplikace.

6.4 Možnosti rozšíření aplikace

Logickým krokem k dalšímu rozšíření aplikace by bylo napojení dalších systému do datového modelu, aby bylo možné získat komplexnější pohled na fungování firmy. Například by se mohlo jednat o data z vydaných faktur, díky kterým by bylo možné vypočítat další ekonomické ukazatele firmy jako marži nebo výši DPH, která se musí odvést státu. Tato data by se mohla čerpat z konkrétního softwaru nebo ze souborů aplikace Excel.

API iDokladu také umožňuje posílání požadavků typu *POST*, proto by také bylo možné aplikaci iSmart rozšířit o vkládání záznamů přímo do iDokladu. Aplikace

Excel je pro zadávání dat výborný nástroj a společně s Power Query by bylo možné administraci iDokladu přesunout do aplikace Excel.

Další možností by například mohlo být zkombinování dat s daty z českého statistického úřadu, kde jsou uložena statistická data pro celou Českou republiku. Díky tomu by bylo možné zjistit statistickou demografii jednotlivých regionů a podle toho bychom mohli odvodit, průměrný věk svých zákazníků. Získáním tohoto údaje bychom mohli využít při vytváření reklamních kampaních nebo při rozhodování se, jak své zákazníky oslovit. Na stránkách českého statistického úřadu jsou také vystaveny údaje o růstu nebo poklesu tržeb v jednotlivých odvětvích ekonomiky. Bylo by tedy možné porovnat vývoj firmy s celorepublikovými hodnotami.

7 Závěr

Cílem této práce bylo prozkoumat možnosti použití platformy Power BI v prostředí malých a středních firem, které bylo demonstrováno vytvářením aplikace komunikující s bezplatnou fakturační cloudovou službou iDoklad.

V první části jsem porovnával tři platformy pro vytváření Business intelligence řešení. Každá z nich má svá specifika a hodí se pro jiná řešení a společnosti. Všechny spojuje nějaká forma spolupráce více uživatelů v cloudovém prostředí a jednoduchost vytváření Business intelligence řešení.

Před samotným vytvářením aplikace jsem popsal jednotlivé doplňky aplikace Microsoft Excel. Doplněk Power Query, který se stará o načtení dat z externích zdrojů do aplikace Excel. Power Pivot, který se stará o uložení dat v datovém modelu a o kalkulace v jazyku DAX. Power View, jenž je prezentační nástroj pro zobrazení dat z datového modelu. Power Map sloužící k vytváření vizualizací dat na mapových podkladech. A nakonec cloudovou službu Power BI, která slouží ke sdílení a práci s reporty v cloudu.

Společně s doplňky jsem také popsal samotnou službu iDoklad, která slouží k bezplatné fakturaci v cloudu. Popsal jsem hlavní funkce služby a také její RestAPI, které se používá pro stahování dat ve vytvořené aplikaci.

Dalším krokem bylo vytvořený aplikace iSmart, která využívala všech výše popsaných doplňků. Ve spojení se službou iDoklad vytvořila kompletní Business intelligence řešení pro jeho uživatele. Během implementace aplikace došlo k několika problémům, které však byly vyřešeny a následně popsány v příslušných kapitolách. Největším problémem aplikace byla nemožnost použití vytvořeného řešení v cloudové službě Power BI.

Posledním krokem bylo zhodnocení celého řešení z hlediska implementace a ekonomické náročnosti řešení v porovnání s konkurenty. Také se celé řešení diskutovalo a navrhli se budoucí možnosti rozšíření aplikace iSmart.

Přesto, že je služba Power BI zatím nejvíce rozšířena v Americe, v České republice se už o toto řešení začíná zajímat čím dál více lidí. Společně s panem Hampelem z Intelligent Technologies jsme v budově Microsoftu představovali toto řešením členům asociace malých a středních podniků. Ohlas na tuto přednášku byl ohromný, lidem se myšl enka vytvoření BI řešení jen za pomoci aplikace Excel velice líbila.

Platforma Power BI se rychle rozrůstá a rozšiřuje své funkce. Microsoft má s touto platformou velké plány a snaží se ji propagovat na velkých vývojářských akcích. V blízké době se chystá nová verze Power BI, která bude umožňovat vytváření reportů jen za pomoci prohlížeče v cloudu. Pro své partnery chystá možnost vytvářet ucelená řešení, která budou nabízena jako doplněk do cloudové služby Power BI. Jestli něco bude v příštích letech hýbat světem Business intelligence, tak to zcela určitě bude Power BI.

8 Reference

Aspin, A. *High Impact Data Visualization with Power View, Power Map, and Power BI.* Apress, 2014.
ISBN 978-1-4302-6616-7.

Beck, P. *A Quick Review of Tableau.* BI Professional [online]. 2014 [cit. 2014-11-16]. Dostupné z: http://www.biprofessional.com/2012/04/a-quick-look-at-tableau/.

Cígler Software *IDoklad: on-line fakturace zdarma.* Účetní program Money S3, ERP systém a informační systémy S4 & S5 - CÍGLER SOFTWARE [online]. 2014 [cit. 2014-11-13]. Dostupné z: http://www.money.cz/idoklad/.

Collie, R. *DAX formulas for powerpivot: the excel pro's guide to mastering dax.* Uniontown: Holy Macro! Books, 2012.
ISBN 978-1-615-47015-0.

Fakturman *Jaký je rozdíl mezi fakturou a zálohovou fakturou?* Fakturman.cz [online]. 2012 [cit. 2014-11-16]. Dostupné z: http://www.fakturman.cz/informace/jaky-je-rozdil-mezi-fakturou-a-zalohovou-fakturou.

Ferrari, A., Russo, M. *Microsoft Excel 2013: building data models with PowerPivot.* Sebastopol: O'Reilly, 2013
ISBN 978-0-73-567634-3.

Few, S. *Show Me the Numbers: Designing Tables and Graphs to Enlighten.* Analytics Press, 2012.
ISBN 978-0-97-060197-1.

Few, S. *Now You See It: Simple Visualization Techniques for Quantitative Analysis.* Oakland, CA: Analytics Press, 2009.
ISBN 978-0-970-60198-8.

GoodData GoodData Developers [online]. 2014 [cit. 2014-11-16]. Dostupné z: https://developer.gooddata.com/.

Kimball, R.,Ross, M. *The data warehouse toolkit: the complete guide to dimensional modeling.* 2nd ed. New York: Wiley, 2002, 436 p.
ISBN 04-712-0024-7.

Larson, B., Davis, M., English, D. A Purington, P. *Visualizing data with Microsoft Power View.* McGraw-Hill Osborne Media, 2012.
ISBN 00-717-8082-3.

Microsoft *Microsoft Power BI* [online]. 2014 [cit. 2014-11-20]. Dostupné z: http://www.microsoft.com/en-us/powerbi/home/power-bi.aspx.

Tableau Software Tableau Software [online]. 2014 [cit. 2014-11-16]. Dostupné z: http:// www.tableausoftware.com/.

TechNet Microsoft *Data Analysis Expressions (DAX) Language.* [online]. 2014 [cit. 2014-11-08]. Dostupné z: http://technet.microsoft.com/library /ee835613.aspx.

Webb, Ch. *Power Query for Power BI and Excel.* Apress, 2014. ISBN 978-1-4302-6691-4.

Přílohy

A Makra definované v aplikaci iSmart

A.1 Funkce ověřování přítomnosti Power Query

```
Function IsPowerQueryAvailable() As Boolean
    On Error GoTo ChybaHandle
    Dim bAvailable As Boolean
    Dim Dostupny As Boolean

    bAvailable = True
    Dostupny = Application.COMAddIns("Microsoft.Mashup.Client.Excel").Connect
    IsPowerQueryAvailable = bAvailable
Exit Function

ChybaHandle:
    bAvailable = False
    Resume Next
End Function


Function IsPowerQueryConnected() As Boolean
On Error GoTo ChybaHandle
    Dim bAvailable      As Boolean
    Dim oPQ             As COMAddIn
    On Error Resume Next
    Set oPQ = Application.COMAddIns("Microsoft.Mashup.Client.Excel")
    If Not oPQ Is Nothing Then
        If Not oPQ.Connect Then oPQ.Connect = True
        bAvailable = oPQ.Connect
    End If
    IsPowerQueryConnected = bAvailable
Exit Function

ChybaHandle:
    MsgBox ("V aplikaci iSmart se vyskytla chyba, omlouváme se. Zkuste to znovu.")
End Function
```

A.2 Metoda při přihlášení uživatele

```
Private Sub PrihlasitSe\_Click()

On Error GoTo ChybaHandle

Dim username As String
Dim password As String
Dim secureTokenString As String
Dim generatedSecureToken As String
Dim hash As String

Set LoginTable = Sheets("login").ListObjects("login")
If ((TextBoxJmeno.Text = "Přihlašovací jméno")
Or (TextBoxJmeno.Text = "")
Or (TextBoxPassword.Text = "")) Then
MsgBox ("Vyplňte prosím všechny údaje.")
Else
    username = TextBoxJmeno.Text
    password = TextBoxPassword.Text
    zapamatovat = ZapamatovatButton.Value

LoginTable.ListColumns("jmeno").DataBodyRange = username
    LoginTable.ListColumns("heslo").DataBodyRange = password

    hash = username \& "|" \& password
    GetSecureToken
```

```
    LoginTable.ListColumns("jmeno").DataBodyRange = ""
    LoginTable.ListColumns("heslo").DataBodyRange = ""

    Login.Hide

TextBoxJmeno.Text = "Zadejte váš e-mail..."
    TextBoxPassword.Text = "heslo"

    generatedSecureToken = SHADD256(hash)
    secureTokenString = Sheets("secureToken").Range("secureTokenValue").Value

    If generatedSecureToken <> secureTokenString Then
Sheets("secureToken").Range("uspechValue").Value = 0
Sheets("Pripojeni k iDokladu").Range("prihlasovaciJmeno").Value = username
Sheets("Pripojeni k iDokladu").Range("stav").Value =
"Chyba! Neexistující uživatel nebo chybné heslo"
LoginNeuspech.Show
    Else
      Sheets("secureToken").Range("uspechValue").Value = 1
      Sheets("Pripojeni k iDokladu").Range("prihlasovaciJmeno").Value = username

If getZapamatovat = True Then
Sheets("Pripojeni k iDokladu").Range("stav").Value = "Prihlášen, stací data jen aktualizovat"
      Else
Sheets("Pripojeni k iDokladu").Range("stav").Value = "Heslo není uloženo"
      End If

LoginUspech.Show

End If

End If

Exit Sub

ChybaHandle:
    MsgBox ("V aplikaci iSmart se vyskytla chyba, omlouváme se. Zkuste to znovu.")
End Sub
```

A.3 Aktualizace dotazů

```
Sub refresh()
On Error GoTo ChybaHandle
If IsPowerQueryAvailable Then
If IsPowerQueryConnected() = False Then
PQNepovolen.Show
    Else
If Sheets("SecureToken").Range("uspechValue").Value = 0 Then
NespravneUdaje.Show
        Else
Aktualizace.Show
            ActiveWorkbook.Connections("LinkedTable_GetSecureToken").refresh
            ActiveWorkbook.Connections("LinkedTable_login").refresh
            ActiveWorkbook.Connections("Power Query - Cenik").refresh
            ActiveWorkbook.Connections("Power Query - Kontakty").refresh
            ActiveWorkbook.Connections("Power Query - Datum").refresh
            ActiveWorkbook.Connections("Power Query - FakturyVydane").refresh
            ActiveWorkbook.Connections("Power Query - PolozkyFaktur").refresh
            ActiveWorkbook.Connections("ThisWorkbookDataModel").refresh
            Aktualizace.Hide
            AktualizaceUspech.Show
         End If
     End If
```

```
     Else
     PQNenainstalovan.Show
End If
Exit Sub

ChybaHandle:
    MsgBox ("V aplikaci iSmart se vyskytla chyba, omlouváme se. Zkuste to znovu.")
End Sub
```

B Power Query dotazy

B.1 Dotaz pro získání tokenu

```
let
    Excel = Excel.CurrentWorkbook(){[Name="login"]}[Content],
    username = Excel{0}[jmeno],
    password = Excel{0}[heslo],
    SecureToken = try Json.Document(
Web.Contents("https://app.idoklad.cz/developer/api/
Agendas/GetSecureToken?username="\&username\&"\&password="\&Text.From(password)))
in
if SecureToken[HasError] then "Chyba! Neexistující uživatel nebo chybné heslo"
else SecureToken[Value]
```

B.2 Načtení tabulky kontaktů

Dotaz Kontakty

```
let
    Excel = Excel.CurrentWorkbook(){[Name="GetSecureToken"]}[Content],
    secureToken = Excel{0}[Value],

Zdroj = Web.Contents(
"https://app.idoklad.cz/developer/api/Contacts?PageSize=200",
[Headers=[#"SecureToken" = secureToken]]),

ImportedJSON = Json.Document(Zdroj),
    ConvertedToTable = Record.ToTable(ImportedJSON),
    Value1 = ConvertedToTable{3}[Value],
    Kontakty = Table.FromValue({1..Value1}),

InsertedCustom = Table.AddColumn(Kontakty, "Kontakty", each fnKontakt([Value])),
    #"Rozbalit Kontakty" = Table.ExpandTableColumn(InsertedCustom, "Kontakty",
{"Id_kontakt", "Název společnosti", "Město", "Email", "Jméno", "IČO", "Telefon", "PSČ", "Ulice",
"Příjmení", "DIČ", "DIČ_SK","Stat_id"}, {"Id_kontakt", "Název společnosti", "Město", "Email",
"Jméno", "IČO", "Telefon", "PSČ", "Ulice", "Příjmení", "DIČ", "DIČ_SK","Stat_id"}),

RemovedColumns = Table.RemoveColumns(#"Rozbalit Kontakty",{"Value"}),
    ChangedType = Table.TransformColumnTypes(RemovedColumns,{{"Id_kontakt", Int64.Type},
{"Stat_id", Int64.Type}}),

#"Vložené: Vlastní" = Table.AddColumn(ChangedType, "Kraj", each fnKraje([PSČ])),

#"Přeuspořádané sloupce" = Table.ReorderColumns(#"Vložené: Vlastní",{"Id_kontakt", "Stat_id",
"Název společnosti", "IČO", "Email", "Jméno", "Příjmení", "Telefon", "Ulice", "Město", "PSČ",
"Kraj", "DIČ", "DIČ_SK"})
in
    #"Přeuspořádané sloupce"
```

Dotaz fnKontakt

```
let
    GetPage = (page as number) =>
let
    Excel = Excel.CurrentWorkbook(){[Name="GetSecureToken"]}[Content],
    secureToken = Excel{0}[Value],

Zdroj = Web.Contents(
"https://app.idoklad.cz/developer/api/Contacts?PageSize=200&Page="&Number.ToText(page),
[Headers=[#"SecureToken" = secureToken]]),

ImportedJSON = Json.Document(Zdroj),
    Data = ImportedJSON[Data],
```

```
TableFromList = Table.FromList(Data, Splitter.SplitByNothing(), null, null, ExtraValues.Error),
    #"Rozbalit Column1" = Table.ExpandRecordColumn(TableFromList, "Column1", {"Id", "CompanyName"
, "City", "Email", "Firstname", "IdentificationNumber", "Phone", "PostalCode", "Street"
, "Surname", "VatIdentificationNumber", "VatIdentificationNumberSk", "CountryId"},
{"Column1.Id", "Column1.CompanyName", "Column1.City", "Column1.Email", "Column1.Firstname",
"Column1.IdentificationNumber", "Column1.Phone", "Column1.PostalCode", "Column1.Street",
"Column1.Surname",
"Column1.VatIdentificationNumber", "Column1.VatIdentificationNumberSk", "Column1.CountryId"}),

RenamedColumns = Table.RenameColumns(#"Rozbalit Column1",{{"Column1.Id", "Id_kontakt"},
{"Column1.CompanyName", "Název společnosti"}, {"Column1.City", "Město"},
{"Column1.Email", "Email"}, {"Column1.Firstname", "Jméno"},
{"Column1.Phone", "Telefon"}, {"Column1.PostalCode", "PSČ"}, {"Column1.Street", "Ulice"},
{"Column1.Surname", "Příjmení"}, {"Column1.VatIdentificationNumber", "DIČ"},
{"Column1.VatIdentificationNumberSk", "DIČ_SK"}, {"Column1.IdentificationNumber", "IČO"},
{"Column1.CountryId","Stat_id"}})
in
   RenamedColumns
in
   GetPage
```

Dotaz fnKraje

```
let
   GetPSC = (psc as text) =>
let
    CaseValues = {
                    {"00", "Slovensko"},
                    {"01", "Slovensko"},
                    {"02", "Slovensko"},
                    {"03", "Slovensko"},
                    {"04", "Slovensko"},
                  {"05", "Slovensko"},
                    {"06", "Slovensko"},
                    {"07", "Slovensko"},
                    {"08", "Slovensko"},
                    {"09", "Slovensko"},
                    {"10", "Praha"},
                    {"11", "Praha"},
                    {"12", "Praha"},
                    {"13", "Praha"},
                    {"14", "Praha"},
                  {"15", "Praha"},
                    {"16", "Praha"},
                    {"17", "Praha"},
                    {"18", "Praha"},
                    {"19", "Praha"},
                    {"20", "Středočeský kraj"},
                    {"21", "Středočeský kraj"},
                    {"22", "Středočeský kraj"},
                    {"23", "Středočeský kraj"},
                    {"24", "Středočeský kraj"},
                  {"25", "Středočeský kraj"},
                    {"26", "Středočeský kraj"},
                    {"27", "Středočeský kraj"},
                    {"28", "Středočeský kraj"},
                    {"29", "Středočeský kraj"},
                    {"30", "Plzeňský kraj"},
                    {"31", "Plzeňský kraj"},
                    {"32", "Plzeňský kraj"},
                    {"33", "Plzeňský kraj"},
                    {"34", "Plzeňský kraj"},
                  {"35", "Karlovarský kraj"},
                    {"36", "Karlovarský kraj"},
```

```
                    {"37", "Jihočeský kraj"},
                    {"38", "Jihočeský kraj"},
                    {"39", "Jihočeský kraj"},
                    {"40", "Liberecký kraj"},
                    {"41", "Ústecký kraj"},
                    {"42", "Ústecký kraj"},
                    {"43", "Ústecký kraj"},
                    {"44", "Ústecký kraj"},
                  {"45", "Liberecký kraj"},
                    {"46", "Liberecký kraj"},
                    {"47", "Liberecký kraj"},
                    {"48", "Liberecký kraj"},
                    {"49", "Liberecký kraj"},
                    {"50", "Královehradecký kraj"},
                    {"51", "Královehradecký kraj"},
                    {"52", "Královehradecký kraj"},
                    {"53", "Pardubický kraj"},
                    {"54", "Královehradecký kraj"},
                  {"55", "Královehradecký kraj"},
                    {"56", "Pardubický kraj"},
                    {"57", "Pardubický kraj"},
                    {"58", "Vysočina"},
                    {"59", "Vysočina"},
                    {"60", "Jihomoravský kraj"},
                    {"61", "Jihomoravský kraj"},
                    {"62", "Jihomoravský kraj"},
                    {"63", "Jihomoravský kraj"},
                    {"64", "Jihomoravský kraj"},
                  {"65", "Jihomoravský kraj"},
                    {"66", "Jihomoravský kraj"},
                    {"67", "Jihomoravský kraj"},
                    {"68", "Zlínský kraj"},
                    {"69", "Jihomoravský kraj"},
                    {"70", "Moravskoslezský kraj"},
                    {"71", "Moravskoslezský kraj"},
                    {"72", "Moravskoslezský kraj"},
                    {"73", "Moravskoslezský kraj"},
                    {"74", "Moravskoslezský kraj"},
                  {"75", "Zlínský kraj"},
                    {"76", "Zlínský kraj"},
                    {"77", "Olomoucký kraj"},
                    {"78", "Olomoucký kraj"},
                    {"79", "Moravskoslezský kraj"},
                    {"82", "Slovensko"},
                    {"83", "Slovensko"},
                    {"84", "Slovensko"},
                  {"85", "Slovensko"},
                    {"90", "Slovensko"},
                    {"91", "Slovensko"},
                    {"92", "Slovensko"},
                    {"93", "Slovensko"},
                    {"94", "Slovensko"},
                  {"95", "Slovensko"},
                    {"96", "Slovensko"},
                    {"97", "Slovensko"},
                    {"98", "Slovensko"},
                    {"99", "Slovensko"},
                    {ZacPSC, "Ostatní"}
                },
    TrimmedText = Text.Replace(psc," ",""),
    ZacPSC = if Text.Length(TrimmedText)=5 then Text.Start(TrimmedText,2) else "Není PSČ",
    SimpleCase = List.First(List.Select(CaseValues, each _{0}=ZacPSC)){1}
in
    SimpleCase
in
```

```
    GetPSC
```

B.3 Načtení tabulky vydaných faktur

Dotaz FakturyVydane

```
let
    Excel = Excel.CurrentWorkbook(){[Name="GetSecureToken"]}[Content],
    secureToken = Excel{0}[Value],

Zdroj = Web.Contents("https://app.idoklad.cz/developer/api/IssuedInvoices?PageSize=200",
[Headers=[#"SecureToken" = secureToken]]),

ImportedJSON = Json.Document(Zdroj),

ConvertedToTable = Record.ToTable(ImportedJSON),

Value1 = ConvertedToTable{3}[Value],
    Faktury = Table.FromValue({1..Value1}),
    InsertedCustom = Table.AddColumn(Faktury, "Faktury", each fnFakturyVydane([Value])),

#"Rozbalit Faktury" = Table.ExpandTableColumn(InsertedCustom, "Faktury",{"Column1.BaseTaxBasicRate",
"Column1.BaseTaxBasicRateHc", "Column1.BaseTaxReducedRate1", "Column1.BaseTaxReducedRate1Hc",
"Column1.BaseTaxReducedRate2", "Column1.BaseTaxReducedRate2Hc","Column1.BaseTaxZeroRate",
"Column1.BaseTaxZeroRateHc","Column1.CashVoucherNumber", "Column1.CashVoucherSerialNumber",
"Column1.ConstantSymbolId", "Column1.CurrencyId", "Column1.DateLastChange",
"Column1.DateOfAccountingEvent","Column1.DateOfIssue", "Column1.DateOfLastReminder",
"Column1.DateOfMaturity","Column1.DateOfPayment","Column1.DateOfTaxing", "Column1.Description",
"Column1.DocumentNumber", "Column1.DocumentSerialNumber","Column1.ExchangeRate",
"Column1.ExchangeRateAmount","Column1.Exported","Column1.IsProformaTaxed",
"Column1.IsSentToAccountant","Column1.IsSentToPurchaser", "Column1.IssuedInvoiceItems",
"Column1.ItemsTextPrefix","Column1.ItemsTextSuffix", "Column1.LanguageId", "Column1.Maturity",
"Column1.MyCompanyDocumentAdrressId", "Column1.Note", "Column1.OrderNumber",
"Column1.PaymentOptionId", "Column1.PaymentStatus", "Column1.PurchaserDocumentAddressId",
"Column1.PurchaserId", "Column1.RemindersCount", "Column1.ReportColorValue", "Column1.ReportId",
"Column1.RoundingDifference","Column1.TaxBasicRate","Column1.TaxBasicRateHc",
"Column1.TaxReducedRate1",
"Column1.TaxReducedRate1Hc", "Column1.TaxReducedRate2","Column1.TaxReducedRate2Hc",
"Column1.TotalBasicRate", "Column1.TotalBasicRateHc", "Column1.TotalReducedRate1",
"Column1.TotalReducedRate1Hc", "Column1.TotalReducedRate2", "Column1.TotalReducedRate2Hc",
"Column1.TotalVat","Column1.TotalVatHc", "Column1.TotalWithVat", "Column1.TotalWithVatHc",
"Column1.TotalWithoutVat","Column1.TotalWithoutVatHc","Column1.VariableSymbol",
"Column1.VatRateBasic",
"Column1.VatRateReduced1", "Column1.VatRateReduced2", "Column1.Id", "Column1.Links"},
    {"Faktury.Column1.BaseTaxBasicRate", "Faktury.Column1.BaseTaxBasicRateHc",
"Faktury.Column1.BaseTaxReducedRate1","Faktury.Column1.BaseTaxReducedRate1Hc",
"Faktury.Column1.BaseTaxReducedRate2", "Faktury.Column1.BaseTaxReducedRate2Hc",
"Faktury.Column1.BaseTaxZeroRate", "Faktury.Column1.BaseTaxZeroRateHc",
"Faktury.Column1.CashVoucherNumber","Faktury.Column1.CashVoucherSerialNumber",
"Faktury.Column1.ConstantSymbolId", "Faktury.Column1.CurrencyId",
"Faktury.Column1.DateLastChange","Faktury.Column1.DateOfAccountingEvent",
"Faktury.Column1.DateOfIssue",
"Faktury.Column1.DateOfLastReminder", "Faktury.Column1.DateOfMaturity",
"Faktury.Column1.DateOfPayment",
"Faktury.Column1.DateOfTaxing", "Faktury.Column1.Description",
"Faktury.Column1.DocumentNumber",
"Faktury.Column1.DocumentSerialNumber", "Faktury.Column1.ExchangeRate",
"Faktury.Column1.ExchangeRateAmount","Faktury.Column1.Exported",
"Faktury.Column1.IsProformaTaxed",
"Faktury.Column1.IsSentToAccountant","Faktury.Column1.IsSentToPurchaser",
"Faktury.Column1.IssuedInvoiceItems", "Faktury.Column1.ItemsTextPrefix",
"Faktury.Column1.ItemsTextSuffix", "Faktury.Column1.LanguageId", "Faktury.Column1.Maturity",
"Faktury.Column1.MyCompanyDocumentAdrressId", "Faktury.Column1.Note",
"Faktury.Column1.OrderNumber",
"Faktury.Column1.PaymentOptionId", "Faktury.Column1.PaymentStatus",
```

```
"Faktury.Column1.PurchaserDocumentAddressId", "Faktury.Column1.PurchaserId",
"Faktury.Column1.RemindersCount", "Faktury.Column1.ReportColorValue","Faktury.Column1.ReportId",
"Faktury.Column1.RoundingDifference", "Faktury.Column1.TaxBasicRate",
"Faktury.Column1.TaxBasicRateHc","Faktury.Column1.TaxReducedRate1",
"Faktury.Column1.TaxReducedRate1Hc",
"Faktury.Column1.TaxReducedRate2","Faktury.Column1.TaxReducedRate2Hc",
"Faktury.Column1.TotalBasicRate",
"Faktury.Column1.TotalBasicRateHc", "Faktury.Column1.TotalReducedRate1",
"Faktury.Column1.TotalReducedRate1Hc", "Faktury.Column1.TotalReducedRate2",
"Faktury.Column1.TotalReducedRate2Hc", "Faktury.Column1.TotalVat",
"Faktury.Column1.TotalVatHc", "Faktury.Column1.TotalWithVat",
"Faktury.Column1.TotalWithVatHc",
"Faktury.Column1.TotalWithoutVat","Faktury.Column1.TotalWithoutVatHc",
"Faktury.Column1.VariableSymbol",
"Faktury.Column1.VatRateBasic", "Faktury.Column1.VatRateReduced1",
"Faktury.Column1.VatRateReduced2",
"Faktury.Column1.Id", "Faktury.Column1.Links"}),

InsertedDatumVydani = Table.AddColumn(#"Rozbalit Faktury","Datum vydání",
each Date.FromText(Text.Start([Faktury.Column1.DateOfIssue],10)), type datetime),

InsertedDatumSplatnosti = Table.AddColumn(InsertedDatumVydani, "Datum splatnosti",
each Date.FromText(Text.Start([Faktury.Column1.DateOfMaturity],10)), type datetime),

InsertedDatumPlatby = Table.AddColumn(InsertedDatumSplatnosti, "Datum platby",
each Date.FromText(Text.Start([Faktury.Column1.DateOfPayment],10)), type datetime),

InsertedDatumPlneni = Table.AddColumn(InsertedDatumPlatby, "Datum zdanitelného plnění",
each Date.FromText(Text.Start([Faktury.Column1.DateOfTaxing],10)), type datetime),

RenamedColumns = Table.RenameColumns(InsertedDatumPlneni,{{"Faktury.Column1.Description", "Popis"},
{"Faktury.Column1.ExchangeRate", "Kurz"}, {"Faktury.Column1.ExchangeRateAmount", "Výše kurzu"},
{"Faktury.Column1.DocumentNumber", "Číslo faktury"}, {"Faktury.Column1.Note", "Poznámka"},
{"Faktury.Column1.OrderNumber", "Číslo objednávky"}, {"Faktury.Column1.PaymentOptionId",
"Uhrada_Id"},
{"Faktury.Column1.PurchaserId", "Kontakt_Id"}, {"Faktury.Column1.RoundingDifference",
"Zaokrouhlení"},
{"Faktury.Column1.Maturity", "Splatnost ve dnech"}, {"Faktury.Column1.CurrencyId", "Mena_Id"},
{"Faktury.Column1.TotalVatHc", "DPH"}, {"Faktury.Column1.TotalWithVatHc", "Celkem"},
{"Faktury.Column1.TotalWithoutVatHc", "Celkem bez DPH"},{"Faktury.Column1.VariableSymbol",
"Variabilní symbol"},
{"Faktury.Column1.VatRateBasic", "Sazba DPH"}, {"Faktury.Column1.VatRateReduced1",
"Snížená sazba DPH"},
{"Faktury.Column1.Id", "Id_faktury"}}),

RemovedColumns = Table.RemoveColumns(RenamedColumns,{"Value", "Faktury.Column1.BaseTaxBasicRate",
"Faktury.Column1.BaseTaxBasicRateHc", "Faktury.Column1.BaseTaxReducedRate1",
"Faktury.Column1.BaseTaxReducedRate1Hc",
"Faktury.Column1.BaseTaxReducedRate2", "Faktury.Column1.BaseTaxReducedRate2Hc",
"Faktury.Column1.BaseTaxZeroRate",
"Faktury.Column1.BaseTaxZeroRateHc", "Faktury.Column1.CashVoucherNumber",
"Faktury.Column1.CashVoucherSerialNumber",
"Faktury.Column1.ConstantSymbolId", "Faktury.Column1.DateLastChange",
"Faktury.Column1.DateOfAccountingEvent",
"Faktury.Column1.DateOfIssue", "Faktury.Column1.DateOfLastReminder",
"Faktury.Column1.DateOfMaturity",
"Faktury.Column1.DateOfPayment", "Faktury.Column1.DateOfTaxing",
"Faktury.Column1.DocumentSerialNumber",
"Faktury.Column1.Exported", "Faktury.Column1.IsProformaTaxed",
"Faktury.Column1.IsSentToAccountant",
"Faktury.Column1.IsSentToPurchaser", "Faktury.Column1.IssuedInvoiceItems",
"Faktury.Column1.ItemsTextPrefix",
"Faktury.Column1.ItemsTextSuffix", "Faktury.Column1.LanguageId",
"Faktury.Column1.PaymentStatus",
"Faktury.Column1.PurchaserDocumentAddressId", "Faktury.Column1.RemindersCount",
```

```
"Faktury.Column1.ReportColorValue",
"Faktury.Column1.ReportId", "Faktury.Column1.TaxBasicRate",
"Faktury.Column1.TaxBasicRateHc",
"Faktury.Column1.TaxReducedRate1", "Faktury.Column1.TaxReducedRate1Hc",
"Faktury.Column1.TaxReducedRate2",
"Faktury.Column1.TaxReducedRate2Hc", "Faktury.Column1.TotalBasicRate",
"Faktury.Column1.TotalBasicRateHc",
"Faktury.Column1.TotalReducedRate1", "Faktury.Column1.TotalReducedRate1Hc",
"Faktury.Column1.TotalReducedRate2",
"Faktury.Column1.TotalReducedRate2Hc", "Faktury.Column1.TotalVat",
"Faktury.Column1.TotalWithVat",
"Faktury.Column1.TotalWithoutVat", "Faktury.Column1.VatRateReduced2",
"Faktury.Column1.Links",
"Faktury.Column1.MyCompanyDocumentAdrressId"}),

ReorderedColumns = Table.ReorderColumns(RemovedColumns,{"Id_faktury", "Kontakt_Id", "Mena_Id",
"Uhrada_Id",
"Číslo faktury", "Datum vydání", "Datum splatnosti", "Datum platby", "Datum zdanitelného plnění",
"Popis", "Kurz",
"Výše kurzu", "Splatnost ve dnech", "Poznámka", "Číslo objednávky", "Zaokrouhlení", "DPH", "Celkem",
"Celkem bez DPH",
"Variabilní symbol",
"Sazba DPH", "Snížená sazba DPH"}),

TrimmedText = Table.TransformColumns(ReorderedColumns,{{"Popis", Text.Trim}}),

ChangedType = Table.TransformColumnTypes(TrimmedText,{{"Id_faktury", Int64.Type},
{"Kontakt_Id", Int64.Type},
{"Mena_Id", Int64.Type}, {"Uhrada_Id", Int64.Type}, {"Datum vydání", type date},
{"Datum splatnosti", type date},
{"Datum platby", type date}, {"Datum zdanitelného plnění", type date}, {"Kurz", type number},
{"Výše kurzu", type number},
{"Zaokrouhlení", type number}, {"DPH", type number}, {"Celkem", type number},
{"Celkem bez DPH", type number}})
in
    ChangedType
```

Dotaz fnFakturyVydane

```
let
    GetPage = (page as number) =>
let
    Excel = Excel.CurrentWorkbook(){[Name="GetSecureToken"]}[Content],
    secureToken = Excel{0}[Value],
    Zdroj = Web.Contents("https://app.idoklad.cz/developer/api/IssuedInvoices?
PageSize=200&Page="&Number.ToText(page),[Headers=[#"SecureToken" = secureToken]]),

ImportedJSON = Json.Document(Zdroj),
    Data = ImportedJSON[Data],
    TableFromList = Table.FromList(Data, Splitter.SplitByNothing(), null, null, ExtraValues.Error),

#"Rozbalit Column1" = Table.ExpandRecordColumn(TableFromList, "Column1", {"BaseTaxBasicRate",
"BaseTaxBasicRateHc", "BaseTaxReducedRate1", "BaseTaxReducedRate1Hc", "BaseTaxReducedRate2",
"BaseTaxReducedRate2Hc", "BaseTaxZeroRate", "BaseTaxZeroRateHc", "CashVoucherNumber",
"CashVoucherSerialNumber", "ConstantSymbolId", "CurrencyId", "DateLastChange",
"DateOfAccountingEvent", "DateOfIssue", "DateOfLastReminder", "DateOfMaturity", "DateOfPayment",
"DateOfTaxing", "Description", "DocumentNumber", "DocumentSerialNumber", "ExchangeRate",
"ExchangeRateAmount", "Exported", "IsProformaTaxed", "IsSentToAccountant", "IsSentToPurchaser",
"IssuedInvoiceItems", "ItemsTextPrefix", "ItemsTextSuffix", "LanguageId", "Maturity",
"MyCompanyDocumentAdrressId", "Note", "OrderNumber", "PaymentOptionId", "PaymentStatus",
"PurchaserDocumentAddressId", "PurchaserId", "RemindersCount", "ReportColorValue", "ReportId",
"RoundingDifference", "TaxBasicRate", "TaxBasicRateHc", "TaxReducedRate1", "TaxReducedRate1Hc",
"TaxReducedRate2", "TaxReducedRate2Hc", "TotalBasicRate", "TotalBasicRateHc", "TotalReducedRate1",
"TotalReducedRate1Hc", "TotalReducedRate2", "TotalReducedRate2Hc", "TotalVat", "TotalVatHc",
"TotalWithVat", "TotalWithVatHc", "TotalWithoutVat", "TotalWithoutVatHc", "VariableSymbol",
```

```
"VatRateBasic", "VatRateReduced1", "VatRateReduced2", "Id", "Links"}, {"Column1.BaseTaxBasicRate",
"Column1.BaseTaxBasicRateHc", "Column1.BaseTaxReducedRate1", "Column1.BaseTaxReducedRate1Hc",
"Column1.BaseTaxReducedRate2", "Column1.BaseTaxReducedRate2Hc", "Column1.BaseTaxZeroRate",
"Column1.BaseTaxZeroRateHc", "Column1.CashVoucherNumber", "Column1.CashVoucherSerialNumber",
"Column1.ConstantSymbolId", "Column1.CurrencyId", "Column1.DateLastChange",
"Column1.DateOfAccountingEvent", "Column1.DateOfIssue", "Column1.DateOfLastReminder",
"Column1.DateOfMaturity", "Column1.DateOfPayment", "Column1.DateOfTaxing", "Column1.Description",
"Column1.DocumentNumber", "Column1.DocumentSerialNumber", "Column1.ExchangeRate",
"Column1.ExchangeRateAmount", "Column1.Exported", "Column1.IsProformaTaxed",
"Column1.IsSentToAccountant", "Column1.IsSentToPurchaser", "Column1.IssuedInvoiceItems",
"Column1.ItemsTextPrefix", "Column1.ItemsTextSuffix", "Column1.LanguageId", "Column1.Maturity",
"Column1.MyCompanyDocumentAdrressId", "Column1.Note", "Column1.OrderNumber",
"Column1.PaymentOptionId", "Column1.PaymentStatus", "Column1.PurchaserDocumentAddressId",
"Column1.PurchaserId","Column1.RemindersCount", "Column1.ReportColorValue", "Column1.ReportId",
"Column1.RoundingDifference","Column1.TaxBasicRate", "Column1.TaxBasicRateHc",
"Column1.TaxReducedRate1", "Column1.TaxReducedRate1Hc", "Column1.TaxReducedRate2",
"Column1.TaxReducedRate2Hc", "Column1.TotalBasicRate", "Column1.TotalBasicRateHc",
"Column1.TotalReducedRate1", "Column1.TotalReducedRate1Hc", "Column1.TotalReducedRate2",
"Column1.TotalReducedRate2Hc", "Column1.TotalVat", "Column1.TotalVatHc", "Column1.TotalWithVat",
"Column1.TotalWithVatHc", "Column1.TotalWithoutVat", "Column1.TotalWithoutVatHc",
"Column1.VariableSymbol", "Column1.VatRateBasic", "Column1.VatRateReduced1",
"Column1.VatRateReduced2", "Column1.Id", "Column1.Links"})
in
    #"Rozbalit Column1"
in
    GetPage
```

B.4 Načtení tabulky položek faktur

Dotaz PolozkyFaktur

```
let
    Excel = Excel.CurrentWorkbook(){[Name="GetSecureToken"]}[Content],
    secureToken = Excel{0}[Value],

Zdroj = Web.Contents("https://app.idoklad.cz/developer/api/IssuedInvoices?PageSize=200",
[Headers=[#"SecureToken" = secureToken]]),

ImportedJSON = Json.Document(Zdroj),
    ConvertedToTable = Record.ToTable(ImportedJSON),
    Value1 = ConvertedToTable{3}[Value],
    Faktury = Table.FromValue({1..Value1}),
    InsertedCustom = Table.AddColumn(Faktury, "Faktury", each fnFakturyVydane([Value])),

#"Rozbalit Faktury" = Table.ExpandTableColumn(InsertedCustom, "Faktury",
{"Column1.IssuedInvoiceItems"}, {"Faktury.Column1.IssuedInvoiceItems"}),

  #"Rozbalit Faktury.Column1.IssuedInvoiceItems" = Table.ExpandListColumn(#"Rozbalit Faktury",
"Faktury.Column1.IssuedInvoiceItems"),

#"Rozbalit Faktury.Column1.IssuedInvoiceItems1" =
Table.ExpandRecordColumn(#"Rozbalit Faktury.Column1.IssuedInvoiceItems",
"Faktury.Column1.IssuedInvoiceItems", {"Code", "DateLastChange", "InvoiceId", "IsRoundedItem",
"IsTaxMovement", "ItemType", "Price", "PriceTotalWithVat", "PriceTotalWithVatHc",
"PriceTotalWithoutVat", "PriceTotalWithoutVatHc", "PriceUnitVat", "PriceUnitVatHc",
"PriceUnitWithVat", "PriceUnitWithVatHc", "PriceUnitWithoutVat", "PriceUnitWithoutVatHc",
"TotalPrice", "VatRate", "VatTotal", "VatTotalHc", "Amount", "Name", "PriceType", "Unit",
"UnitPrice", "VatRateType", "Id", "Links"}, {"Faktury.Column1.IssuedInvoiceItems.Code",
"Faktury.Column1.IssuedInvoiceItems.DateLastChange", "Faktury.Column1.IssuedInvoiceItems.InvoiceId",
"Faktury.Column1.IssuedInvoiceItems.IsRoundedItem",
"Faktury.Column1.IssuedInvoiceItems.IsTaxMovement", "Faktury.Column1.IssuedInvoiceItems.ItemType",
"Faktury.Column1.IssuedInvoiceItems.Price", "Faktury.Column1.IssuedInvoiceItems.PriceTotalWithVat",
"Faktury.Column1.IssuedInvoiceItems.PriceTotalWithVatHc",
"Faktury.Column1.IssuedInvoiceItems.PriceTotalWithoutVat",
"Faktury.Column1.IssuedInvoiceItems.PriceTotalWithoutVatHc",
```

```
"Faktury.Column1.IssuedInvoiceItems.PriceUnitVat",
"Faktury.Column1.IssuedInvoiceItems.PriceUnitVatHc",
"Faktury.Column1.IssuedInvoiceItems.PriceUnitWithVat",
"Faktury.Column1.IssuedInvoiceItems.PriceUnitWithVatHc",
"Faktury.Column1.IssuedInvoiceItems.PriceUnitWithoutVat",
"Faktury.Column1.IssuedInvoiceItems.PriceUnitWithoutVatHc",
"Faktury.Column1.IssuedInvoiceItems.TotalPrice", "Faktury.Column1.IssuedInvoiceItems.VatRate",
"Faktury.Column1.IssuedInvoiceItems.VatTotal", "Faktury.Column1.IssuedInvoiceItems.VatTotalHc",
"Faktury.Column1.IssuedInvoiceItems.Amount", "Faktury.Column1.IssuedInvoiceItems.Name",
"Faktury.Column1.IssuedInvoiceItems.PriceType", "Faktury.Column1.IssuedInvoiceItems.Unit",
"Faktury.Column1.IssuedInvoiceItems.UnitPrice", "Faktury.Column1.IssuedInvoiceItems.VatRateType",
"Faktury.Column1.IssuedInvoiceItems.Id", "Faktury.Column1.IssuedInvoiceItems.Links"}),

RenamedColumns = Table.RenameColumns(#"Rozbalit Faktury.Column1.IssuedInvoiceItems1",
{{"Faktury.Column1.IssuedInvoiceItems.InvoiceId", "Faktura_id"},
{"Faktury.Column1.IssuedInvoiceItems.ItemType", "Typ položky"},
{"Faktury.Column1.IssuedInvoiceItems.PriceTotalWithVatHc", "Celkem"},
{"Faktury.Column1.IssuedInvoiceItems.PriceTotalWithoutVatHc", "Celkem bez DPH"},
{"Faktury.Column1.IssuedInvoiceItems.PriceUnitVatHc", "DPH za jednotku"},
{"Faktury.Column1.IssuedInvoiceItems.PriceUnitWithVatHc", "Cena za jednotku"},
{"Faktury.Column1.IssuedInvoiceItems.PriceUnitWithoutVatHc", "Cena za jednotku bez DPH"},
{"Faktury.Column1.IssuedInvoiceItems.VatRate", "Sazba DPH"},
{"Faktury.Column1.IssuedInvoiceItems.VatTotalHc", "DPH"},
{"Faktury.Column1.IssuedInvoiceItems.Amount", "Počet KS"},
{"Faktury.Column1.IssuedInvoiceItems.Name", "Popis"},
{"Faktury.Column1.IssuedInvoiceItems.Unit", "Jednotka"},
{"Faktury.Column1.IssuedInvoiceItems.Id", "Id_polozka"}}),

RemovedColumns = Table.RemoveColumns(RenamedColumns,
{"Value", "Faktury.Column1.IssuedInvoiceItems.Code",
"Faktury.Column1.IssuedInvoiceItems.DateLastChange",
"Faktury.Column1.IssuedInvoiceItems.IsRoundedItem",
"Faktury.Column1.IssuedInvoiceItems.IsTaxMovement", "Faktury.Column1.IssuedInvoiceItems.Price",
"Faktury.Column1.IssuedInvoiceItems.PriceTotalWithVat",
"Faktury.Column1.IssuedInvoiceItems.PriceTotalWithoutVat",
"Faktury.Column1.IssuedInvoiceItems.PriceUnitVat",
"Faktury.Column1.IssuedInvoiceItems.PriceUnitWithVat",
"Faktury.Column1.IssuedInvoiceItems.PriceUnitWithoutVat",
"Faktury.Column1.IssuedInvoiceItems.TotalPrice", "Faktury.Column1.IssuedInvoiceItems.VatTotal",
"Faktury.Column1.IssuedInvoiceItems.PriceType", "Faktury.Column1.IssuedInvoiceItems.UnitPrice",
"Faktury.Column1.IssuedInvoiceItems.VatRateType", "Faktury.Column1.IssuedInvoiceItems.Links"}),

ReorderedColumns = Table.ReorderColumns(RemovedColumns,{"Id_polozka", "Faktura_id", "Typ položky",
"Celkem", "Celkem bez DPH", "DPH za jednotku", "Cena za jednotku", "Cena za jednotku bez DPH",
"Sazba DPH", "DPH", "Počet KS", "Popis", "Jednotka"}),

TrimmedText = Table.TransformColumns(ReorderedColumns,{{"Popis", Text.Trim}}),

Sloučit = Table.NestedJoin(TrimmedText,{"Popis"},Cenik,{"Popis"},"NewColumn"),
    #"Rozbalit NewColumn" = Table.ExpandTableColumn(Sloučit, "NewColumn", {"Id_cenik"},
{"NewColumn.Id_cenik"}),

RenamedColumns1 = Table.RenameColumns(#"Rozbalit NewColumn",{{"NewColumn.Id_cenik", "Cenik_id"}}),
    RemovedColumns1 = Table.RemoveColumns(RenamedColumns1,{"Popis", "Jednotka"}),

ChangedType = Table.TransformColumnTypes(RemovedColumns1,{{"Id_polozka", Int64.Type},
{"Faktura_id", Int64.Type}, {"Typ položky", Int64.Type}, {"Sazba DPH", Int64.Type},
{"Počet KS", Int64.Type}, {"Cenik_id", Int64.Type}, {"Celkem", type number},
{"Celkem bez DPH", type number}, {"DPH za jednotku", type number},{"Cena za jednotku", type number},
{"Cena za jednotku bez DPH", type number}, {"DPH", type number}})
in
    ChangedType
```

B.5 Načtení tabulky produktů

Dotaz Cenik

```
let
    Excel = Excel.CurrentWorkbook(){[Name="GetSecureToken"]}[Content],
    secureToken = Excel{0}[Value],
    Zdroj = Web.Contents("https://app.idoklad.cz/developer/api/IssuedInvoices?PageSize=200",
[Headers=[#"SecureToken" = secureToken]]),

ImportedJSON = Json.Document(Zdroj),
    ConvertedToTable = Record.ToTable(ImportedJSON),
    Value1 = ConvertedToTable{3}[Value],
    Faktury = Table.FromValue({1..Value1}),
    InsertedCustom = Table.AddColumn(Faktury, "Faktury", each fnFakturyVydane([Value])),

#"Rozbalit Faktury" = Table.ExpandTableColumn(InsertedCustom, "Faktury",
{"Column1.IssuedInvoiceItems"}, {"Faktury.Column1.IssuedInvoiceItems"}),

#"Rozbalit Faktury.Column1.IssuedInvoiceItems" = Table.ExpandListColumn(#"Rozbalit Faktury",
"Faktury.Column1.IssuedInvoiceItems"),

#"Rozbalit Faktury.Column1.IssuedInvoiceItems1" =
Table.ExpandRecordColumn(#"Rozbalit Faktury.Column1.IssuedInvoiceItems",
"Faktury.Column1.IssuedInvoiceItems", {"Name", "Unit"}, {"Faktury.Column1.IssuedInvoiceItems.Name",
"Faktury.Column1.IssuedInvoiceItems.Unit"}),

RemovedColumns = Table.RemoveColumns(#"Rozbalit Faktury.Column1.IssuedInvoiceItems1",{"Value"}),
    DuplicatesRemoved = Table.Distinct(RemovedColumns, {"Faktury.Column1.IssuedInvoiceItems.Name"}),
    TrimmedText = Table.TransformColumns(DuplicatesRemoved,
{{"Faktury.Column1.IssuedInvoiceItems.Name",Text.Trim}}),

InsertedIndex = Table.AddIndexColumn(TrimmedText,"Index",1),
    ReorderedColumns = Table.ReorderColumns(InsertedIndex,{"Index",
"Faktury.Column1.IssuedInvoiceItems.Name", "Faktury.Column1.IssuedInvoiceItems.Unit"}),

RenamedColumns = Table.RenameColumns(ReorderedColumns,{{"Index", "Id_cenik"},
{"Faktury.Column1.IssuedInvoiceItems.Name", "Popis"},
{"Faktury.Column1.IssuedInvoiceItems.Unit", "Jednotka"}}),

ChangedType = Table.TransformColumnTypes(RenamedColumns,{{"Id_cenik", Int64.Type}})
in
    ChangedType
```

B.6 Načtení tabulky Datum

```
let
    PocetDni = Duration.Days(Duration.From(#date(2018, 1, 1) - #date(2012, 1, 1))),
    Zdroj = List.Dates(#date(2012, 1, 1),PocetDni,#duration(1,0,0,0)),
    TableFromList = Table.FromList(Zdroj, Splitter.SplitByNothing()),
    ZmenenyTyp = Table.TransformColumnTypes(TableFromList,{{"Column1", type date}}),
    PrejmenovaneSloupce = Table.RenameColumns(ZmenenyTyp,{{"Column1", "Datum"}}),

#"Vložené: Rok" = Table.AddColumn(PrejmenovaneSloupce, "Year", each Date.Year([Datum]),
type number),

    #"Vložené: Čtvrtletí" = Table.AddColumn(#"Vložené: Rok", "Quarter",
each Date.QuarterOfYear([Datum]), type number),

    #"Vložené: Měsíc" = Table.AddColumn(#"Vložené: Čtvrtletí", "Month",
each Date.Month([Datum]), type number),

    #"Vložené: Den" =Table.AddColumn(#"Vložené: Měsíc", "Day",each Date.Day([Datum]),type number),
    VlozitMesicInt = Table.AddColumn(#"Vložené: Den", "MesicInt", each [Year] * 100 + [Month]),
    VlozitMonthName = Table.AddColumn(VlozitMesicInt, "Měsíc",
```

```
each Date.ToText([Datum], "MMMM", "cs-CZ"), type text),

    VlozitCalendarMonth = Table.AddColumn(VlozitMonthName, "Měsíc rok",
each (Text.Proper([Měsíc])) & " " & Number.ToText([Year])),

    VlozitCalendarQtr = Table.AddColumn(VlozitCalendarMonth, "Kvartál rok",
each "Q" & Number.ToText([Quarter]) & " " & Number.ToText([Year])),

    #"Vložené: Den týdne" = Table.AddColumn(VlozitCalendarQtr, "DayOfWeek",
each Date.DayOfWeek([Datum],Day.Monday)+1, type number),

    VlozitDayName = Table.AddColumn(#"Vložené: Den týdne", "Den v týdnu",
each Date.ToText([Datum], "dddd", "cs-CZ"), type text),

    #"Vložené: Začátek měsíce" = Table.AddColumn(VlozitDayName, "StartOfMonth",
each Date.StartOfMonth([Datum]), type date),

    #"Vložené: Konec měsíce" = Table.AddColumn(#"Vložené: Začátek měsíce", "EndOfMonth",
each Date.EndOfMonth([Datum]), type date),

    #"Přejmenované sloupce" = Table.RenameColumns(#"Vložené: Konec měsíce",{{"Year", "Rok"},
{"Quarter", "Kvartál"}, {"Month", "Měsíc číslo"}, {"Day", "Den"},
{"DayOfWeek", "Den v týdnu číslo"}, {"StartOfMonth", "Měsíc začátek"},
{"EndOfMonth", "Měsíc konec"}}),

    #"Změněný typ" = Table.TransformColumnTypes(#"Přejmenované sloupce",{{"Datum", type date},
{"Rok", Int64.Type}, {"Kvartál", Int64.Type}, {"Měsíc číslo", Int64.Type}, {"Den", Int64.Type},
{"MesicInt", Int64.Type}, {"Měsíc rok", type text}, {"Kvartál rok", type text},
{"Den v týdnu číslo", Int64.Type}, {"Den v týdnu", type text}, {"Měsíc začátek", type date},
{"Měsíc konec", type date}}),

    #"Velké první písmeno každého slova" = Table.TransformColumns(#"Změněný typ",
{{"Měsíc", Text.Proper}, {"Den v týdnu", Text.Proper}})
in
    #"Velké první písmeno každého slova"
```

B.7 Načtení plánovací tabulky

```
let
    Zdroj = Excel.CurrentWorkbook(){[Name="MesicniPlan"]}[Content],
    ChangedType = Table.TransformColumnTypes(Zdroj,{{"Plánované tržby", Int64.Type}}),
    ReplacedValue = Table.ReplaceValue(ChangedType,null,0,Replacer.ReplaceValue,
{"Plánované tržby"}),
    ChangedType1 = Table.TransformColumnTypes(ReplacedValue,{{"Plánované tržby", Int64.Type}})
in
    ChangedType1
```

C Kalkulace v Power Pivot datovém modelu

C.1 Kalkulovaná pole z tabulky FakturyVydane

```
Tržby:=SUM([Celkem])

Tržby minulý rok:=CALCULATE([Tržby];SAMEPERIODLASTYEAR(Datum[Datum]))

Kumulované tržby:=TOTALYTD([Tržby];Datum[Datum])

Kumulované tržby loni:=CALCULATE([Kumulované tržby];SAMEPERIODLASTYEAR(Datum[Datum]))

\% změna tržeb:=IF(([Tržby minulý rok]=0))
                    ;BLANK()
                    ;([Tržby]-[Tržby minulý rok])/[Tržby minulý rok])

Tržby bez DPH:=SUM([Celkem bez DPH])

Tržby za poslední rok:=CALCULATE([Tržby]; DATESBETWEEN(Datum[Datum];
DATE(YEAR(EOMONTH(TODAY();-12));MONTH(EOMONTH(TODAY();-12));1);
EOMONTH(TODAY();-1)))

Plátce DPH:=IF([Tržby za poslední rok]>1000000;"Ano";"Ne")

Průměrné tržby za měsíc pro neplátce:=1000000/12

Tržby první měsíc poslední rok:=CALCULATE(SUM([Celkem]); DATESBETWEEN(Datum[Datum];
    DATE(YEAR(EOMONTH(TODAY();-12));MONTH(EOMONTH(TODAY();-12));1);
    EOMONTH(TODAY();-12)))

Povolený obrat:=IF((1000000-[Tržby za poslední rok]+[Tržby první měsíc poslední rok])<0;0;
(1000000-[Tržby za poslední rok]+[Tržby první měsíc poslední rok]))

DPH tento měsíc:=TOTALMTD(SUM([DPH]);Datum[Datum])

DPH tento kvartál:=TOTALQTD(SUM([DPH]);Datum[Datum])

DPH tento rok:=TOTALYTD(SUM([DPH]);Datum[Datum])

DPH minulý měsíc:=CALCULATE(SUM([DPH]);DATESBETWEEN(Datum[Datum];
STARTOFMONTH(DATEADD(Datum[Datum];-1;MONTH));ENDOFMONTH(DATEADD(Datum[Datum];-1;MONTH))))

DPH minulý kvartál:=CALCULATE(SUM([DPH]); DATESBETWEEN(Datum[Datum];
STARTOFQUARTER(DATEADD(Datum[Datum];-1;QUARTER));ENDOFQUARTER(DATEADD(Datum[Datum];-1;QUARTER))))

Zákazníků:=DISTINCTCOUNT([Kontakt_id])

Vracející:=CALCULATE(COUNTROWS(DISTINCT(FakturyVydane[Kontakt_id]))
                    ; DISTINCT(FakturyVydane[Kontakt_id])
                    ; DATESBETWEEN(Datum[Datum]
                                  ; BLANK()
                                  ; DATEADD(FIRSTDATE(Datum[Datum]); -1; DAY)
                                 )
                   )

Vracející se:=IF(ISBLANK([Vracející]);0;[Vracející])

Nový zákazníci:=IF(([Zákazníků]-[Vracející se])=0;BLANK();[Zákazníků]-[Vracející se])

Noví:=IF(ISBLANK([Nový zákazníci]);0;[Nový zákazníci])

Průměrně na zákazníka:=[Tržby]/[Zákazníků]

Počet faktur:=DISTINCTCOUNT([Id_faktury])
```

```
Tržby na 1 fakturu:=[Tržby]/[Počet faktur]

Nezaplacených faktur:=IF(ISBLANK(COUNTROWS(FILTER('FakturyVydane';[Datum platby]>NOW())))
;0
;COUNTROWS(FILTER ('FakturyVydane';[Datum platby]>NOW()))
)

Nezaplacená částka:=IF(ISBLANK(CALCULATE(SUM([Celkem]);FILTER('FakturyVydane';[Datum platby]>NOW())))
; 0
; CALCULATE(SUM([Celkem]);FILTER('FakturyVydane';[Datum platby]>NOW()))
)
```

C.2 Kalkulovaná pole z tabulky MesicniPlan

```
Plnění plánu:=IF(sum([Plánované tržby])>0;[Tržby]/sum([Plánované tržby]);BLANK())
```