FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Modul pro zpracování datových toků v OBIEE od Oracle v projektu Manta
Student:	Yauheniy Buldyk
Vedoucí:	Ing. Michal Valenta, Ph.D.
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2019/20

Pokyny pro vypracování

Cílem práce je navrhnout a implementovat funkční prototyp modulu, který provede syntaktickou a sémantickou analýzu úloh v nástroji OBIEE od Oracle a následně její výsledek využije pro analýzu datových toků. Modul bude zařazen do projektu Manta.

1. Seznamte se s projektem Manta a s nástrojem OBIEE.
2. Navrhněte modul v projektu Manta pro zpracování úloh v OBIEE. Využijte existující infrastrukturu projektu.
3. Implementujte prototyp, řádně ho zdokumentujte a otestujte.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 30. října 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Modul pro zpracování datových toků v OBIEE od Oracle v projektu Manta

Yauheniy Buldyk

Katedra softwarového inženýrství
Vedoucí práce: Ing. Michal Valenta, Ph.D.

15. května 2019

Poděkování

Tímto bych rád poděkoval vedoucímu své práce Ing. Michalu Valentovi, Ph.D. za jeho cenné rady a připomínky. Dále bych chtěl poděkovat členům společnosti Manta, kteří mi pomáhali při návrhu a tvorbě implementační části této práce. Jmenovitě to jsou Lukáš Hermann, Jaroslav Kotrč a Petr Košvanec. V neposlední řadě chci také poděkovat svému nejlepšímu kamarádovi a své přítelkyni za jejich neuvěřitelnou podporu během celého procesu tvorby.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů (dále jen „autorský zákon"), především § 35 a § 60 autorského zákona upravující školní dílo.

V případě počítačových programů, jež jsou součástí mojí práce či její přílohou, a veškeré související dokumentace k počítačovým programům (dále jen „software"), uděluji v souladu s ust. § 2373 zákona 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, nevýhradní a neodvolatelné oprávnění (licenci) k užití software, a to všem osobám, které si přejí software užít. Tyto osoby jsou oprávněny software užít jakýmkoli způsobem a za jakýmkoli účelem v neomezeném rozsahu (včetně užití k výdělečným účelům), vč. možnosti software upravit či měnit, spojit jej s jiným dílem a/nebo zařadit jej do díla souborného. Toto oprávnění je časově, teritoriálně i množstevně neomezené a uděluji jej bezúplatně.

V Praze dne 15. května 2019 ……

Odkaz na tuto práci

Buldyk, Yauheniy. *Modul pro zpracování datových toků v OBIEE od Oracle v projektu Manta*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019.

Abstrakt

Cílem této práce je návrh a implementace funkčního prototypu modulu pro software Manta Flow, který dokáže zanalyzovat dokumenty reportovacího nástroje OBIEE (Oracle Business Intelligence Enterprise Edition) a na základě výsledků analýzy vytvořit graf popisující jejich datové toky. Výsledný graf dále slouží k vizualizaci toků dat směrem od datových zdrojů k jednotlivým položkám dokumentů.

Obsahem práce je analýza reportovacího nástroje OBIEE a jeho objektů, návrh možného řešení problému, následná implementace prototypu a provedení testování jeho jednotlivých částí. Vytvořený prototyp bude zařazen do existujícího produktu Manta Flow, čímž rozšíří množinu podporovaných nástrojů softwaru.

Klíčová slova datové toky, data lineage, business intelligence, reportovací nástroj, report, OBIEE, Manta

Abstract

The purpose of this thesis is to design and implement a functional prototype module for the Manta Flow software, which will be able to analyze OBIEE (Oracle Business Intelligence Enterprise Edition) reporting tool documents and create a graph describing their data flows. The resulting graph is used to visualize data flows from data sources to single document items.

The basis of this thesis is an analysis of OBIEE reporting tool and its objects, design of a possible solution to the problem, following implementation of the prototype and testing its different parts. The created prototype will be deployed into the existing Manta Flow product and expands the set of software supported tools.

Keywords data flow, data lineage, business intelligence, report, reporting tool, OBIEE, Manta

Obsah

Seznam obrázků

Úvod

V současné době jsou data nezbytná pro řízení velkých i menších firem. Na základě různých dat může firma snadno zjistit důležité informace, např. o finančním stavu, pozici na trhu nebo i některé interní statistiky ohledně zaměstnanců nebo produktů. Málo kdo ale může rychle a kvalitně provést analýzu „syrových“ dat, proto je nejdřív potřeba tato data převést do přehledné formy grafů, diagramů, tabulek apod. Pravě k tomuto účelu slouží tzv. reportovací nástroje, které dokážou shromáždit data a vizualizovat je ve formě specifického dokumentu (Report, Dashboard, Analysis atd.), což dovolí analytikovi soustředit se na přijetí důležitých rozhodnutí a nestarat se o data v pozadí. Jedním z takových nástrojů je OBIEE (Oracle Business Intelligence Enterprise Edition) od Oracle, který je předmětem této práce.

OBIEE a další podobné nástroje usnadňují práci s daty, ale nemají za úkol kontrolovat kvalitu nebo správnost těch dat a často ani neposkytují koncovému uživateli přesnou informaci o jejich zdrojích. Pri řešení právě těchto a mnohých dalších problémů může pomoct znalost datových toků, které poskytují informaci nejenom o zdrojích použitých dat, ale i o jejich transformacích na cestě k výslednému dokumentu. Analýzou výsledků reportovacích nástrojů, zjištěním datových toků a jejich následnou vizualizací se právě zabývá software Manta Flow. Aplikace dokáže pracovat s nejpopulárnějšími reportovacími nástroji, ale aktuálně neexistuje podpora OBIEE, což je hlavním důvodem vzniku této práce. Výsledkem práce je funkční prototyp modulu pro projekt Manta, účelem kterého je zpracování specifických objektů nástroje OBIEE a převod těchto objektů do modelu, který lze dál využít pro vizualizaci datových toků.

Cíle práce

Hlavním cílem této práce je navrhnout a implementovat funkční prototyp modulu pro software Manta Flow. Úkolem prototypu je provést analýzu objektů nástroje OBIEE a na její základě vytvořit graf, který přesně popíše toky dat od

jejich zdrojů do jednotlivých objektů. Dílčími cíli jsou řádně zdokumentovat a otestovat vytvořený modul. V průběhu analýzy a návrhu se také seznámíme s nástrojem OBIEE a softwarem Manta Flow, do kterého bude zařazen výsledný prototyp.

Členění práce

Táto práce je rozdělena do následujících kapitol:

- **Základní pojmy** V první kapitole vysvětlíme hlavní pojmy, které jsou nezbytné pro celkové pochopení práce.
- **Použité technologie** Ve druhé kapitole řekneme o nejdůležitějších technologií a nástrojů využitých při implementaci prototypu.
- **Analýza** V kapitole 3 podrobně popíšeme provedenou analýzu nástroje OBIEE a jeho objektů.
- **Návrh a implementace** V další kapitole se budeme věnovat návrhu a implementaci samotného modulu. Popíšeme zvolený postup vytváření modulu a podrobněji se podíváme na každou z jeho částí.
- **Testování** V poslední kapitole řekneme o testech, vytvořených pro kontrolu funkčnosti jednotlivých částí modulu.

Kapitola 1

Základní pojmy

V této kapitole jsme se seznámíme s nejdůležitějšími pojmy nezbytnými pro pochopení teoretické a praktické části práce.

1.1 Business Intelligence

Business Intelligence (BI) je software obsahující v sobě různé procesy, architektury a technologie, které dokážou převést syrová data na smysluplný a přehledný format. Získané informace jsou využity při přijetí důležitých strategických a taktických obchodních rozhodnutí organizace, přičemž BI nástroje dávají přednost faktům založeným na historických datech před pocity a předpoklady. [1]

Výsledkem analýzy podobných nástrojů jsou specifické dokumenty, ve kterých se data reprezentují ve formě tabulek, diagramu, obrázku atd. (viz obrázek 1.1).

1.2 OBIEE

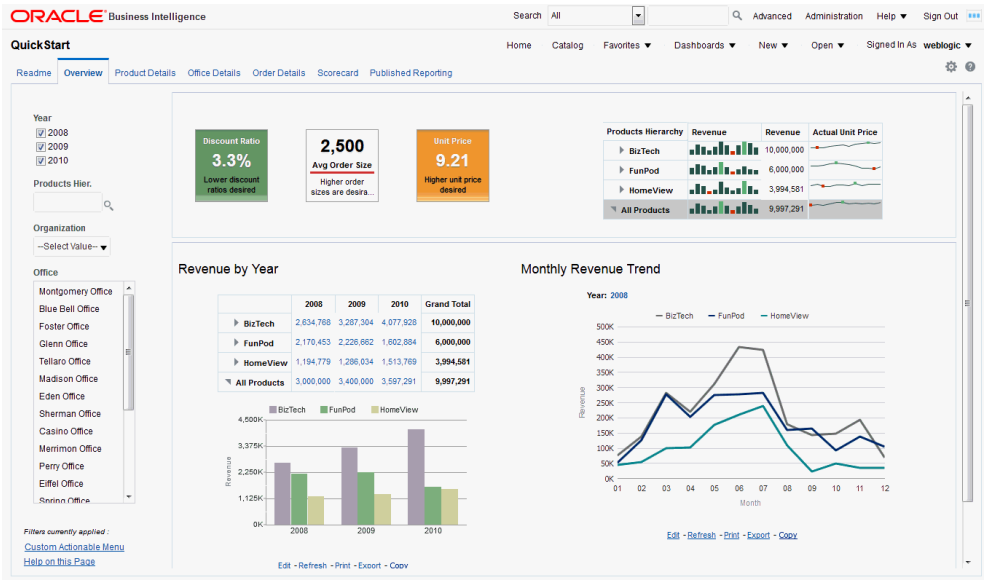
OBIEE (Oracle Business Intelligence Enterprise Edition) je výkonný BI nástroj pro analýzu a prezentování dat. OBIEE umožňuje shromažďovat aktuální data z organizace a prezentovat je ve snadno srozumitelných formátech (příklad výsledného dokumentu můžete vidět na obrázku 1.1). [2]

1.3 Manta a Manta Flow

Manta, celým názvem Manta Tools, s.r.o., je česká startupová společnost, která byla původně projektem firmy Profinit, s.r.o. a vyvíjela se ve spolupráci s FIT[1] ČVUT[2]. Hlavním produktem společnosti je software Manta Flow, který

[1] Fakulta informačních technologií

[2] České vysoké učení technické v Praze



Obrázek 1.1: Ukázka výsledného dokumentu nástroje OBIEE [2]

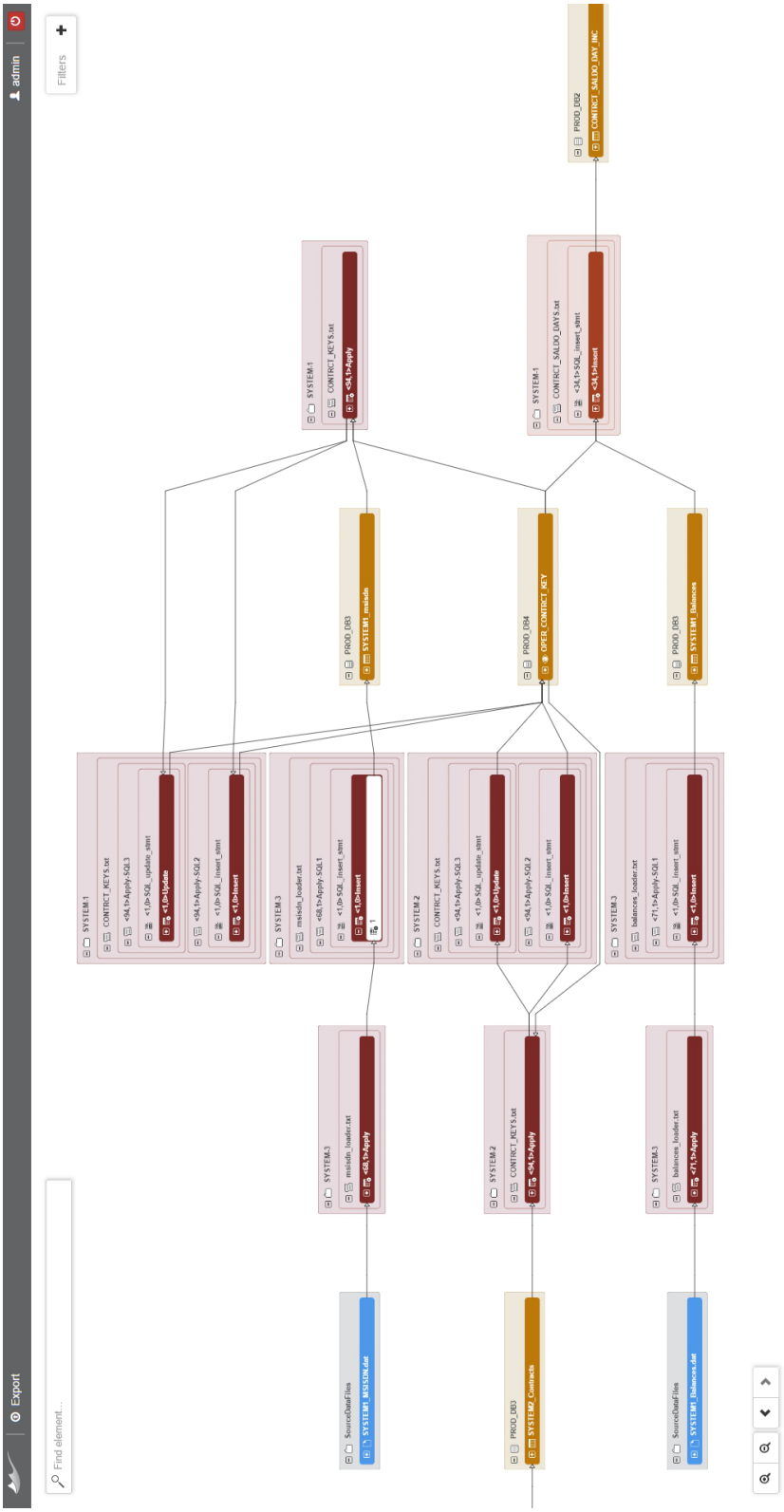
je dnes používán velkými organizacemi nejenom v Česku, ale i po celém světě.

Manta Flow je nástroj umožňující automatickou analýzu programovacího kódu (SQL[3], Java) nebo i BI nástrojů (IBM[4] Cognos, OBIEE, Microsoft SSRS[5] a jiné). Na základě provedené analýzy dokáže sestavit přehlednou mapu datových toků napříč BI prostředí, neboli Data Lineage. To se v praxi využívá převážně k optimalizaci datových skladů, snižování nákladů na vývoj softwaru a provádění dopadových analýz.[3]

1.4 Datové toky

Datové toky jsou jedním z druhů životního cyklu dat, který zahrnuje zdroj dat a informaci kam a kudy se tato data pohybují v čase. Datové toky také mohou popisovat, jak se data mění, když prochází různými procesy [4]. V softwaru Manta Flow se data vizualizují ve formě orientovaného grafu, ve kterém uzly reprezentují objekty nebo struktury obsahující data a hranami jsou datové toky mezi nimi (směr je vždycky od zdroje dat do výsledného objektu). Jednu z možných výsledných vizualizací můžete vidět na obrázku 1.2

[3] Structured Query Language
[4] International Business Machines
[5] SQL Server Reporting Services



Obrázek 1.2: Ukázka vizualizace datových toků v nástroji Manta Flow [5]

Kapitola 2

Použité technologie

2.1 Java

Java je objektově orientovaný programovací jazyk a v současné době je jedním z nejpoužívanějších ve světe. Java se používá pro vývoj obrovského množství aplikací (jak desktopových, tak i webových), software Manta Flow není výjimkou. Z toho důvodu je tento modul celkově napsán v jazyce Java, a předpokládáme, že čtenář je seznámen se základy tohoto jazyka.

2.2 JavaDoc

JavaDoc je speciální utilita pro automatické generování dokumentace programů. JavaDoc dokáže projít celý zdrojový kód, najít a zanalyzovat speciálně vytvořené komentáře a na jejich základě vygenerovat HTML[6] soubor s dokumentaci. Právě tenhle pomocný program využijeme v práci pro dokumentování prototypu.

2.3 Apache Maven

Maven je nástroj pro řízení a správu softwarových aplikací [6]. V rámci tohoto modulu se používá především pro sestavení aplikace, propojení různých částí modulu mezi sebou a stahování potřebných knihoven.

2.4 SOAP

SOAP[7] je protokol, umožňující výměnu zpráv ve formátu XML[8] mezi distribuovanými aplikacemi nebo jejich částmi přes síť. SOAP se často používá ve

[6] Hypertext Markup Language
[7] Simple Object Access Protocol
[8] Extensible Markup Language

webových aplikacích, které využívají WSDL[9] pro popis dostupných metod. V rámci této práce používáme WSDL pro generaci Java tříd, které umožňují snadnou komunikaci s webovými službami přímo v kódu. [7]

2.5 JUnit

JUnit je nejrozšířenější framework pro unit neboli jednotkové testy, které slouží k testování jednotlivých malých částí aplikace nezávisle na jiných částech a celku.

2.6 Spring

Spring je Java framework s širokým využitím, který usnadňuje proces vývoje JEE[10] aplikací. V této práce použijeme Spring Beans při implementaci třetí části modulu pro automatické vytváření a konfiguraci potřebných Java tříd.

2.7 Apache Subversion

Subversion (SVN) je systém pro správu a verzování zdrojových kódů, který využijeme během implementace prototypu pro usnadnění kontroly změn zdrojových kódů.

2.8 IntelliJ IDEA

Intellij IDEA je jedním z nejpoužívanějších vývojových prostředí od společnosti JetBrains, které podporuje práci se všemi technologiemi použitými při tvorbě této práce. Díky tomuto IDE[11] lze velice snadno a rychle najít překlepy a chyby v kódu, sestavit aplikaci a spouštět jednotlivé testy.

[9] Web Services Description Language
[10] Java Enterprise Edition
[11] Integrated Development Environment

Kapitola 3

Analýza

V této kapitole podrobně popíšeme provedenou analýzu nástroje OBIEE a jeho objektů, podíváme se na způsoby jejich stahování a zpracování a vysvětlíme důvody pro zvolené řešení.

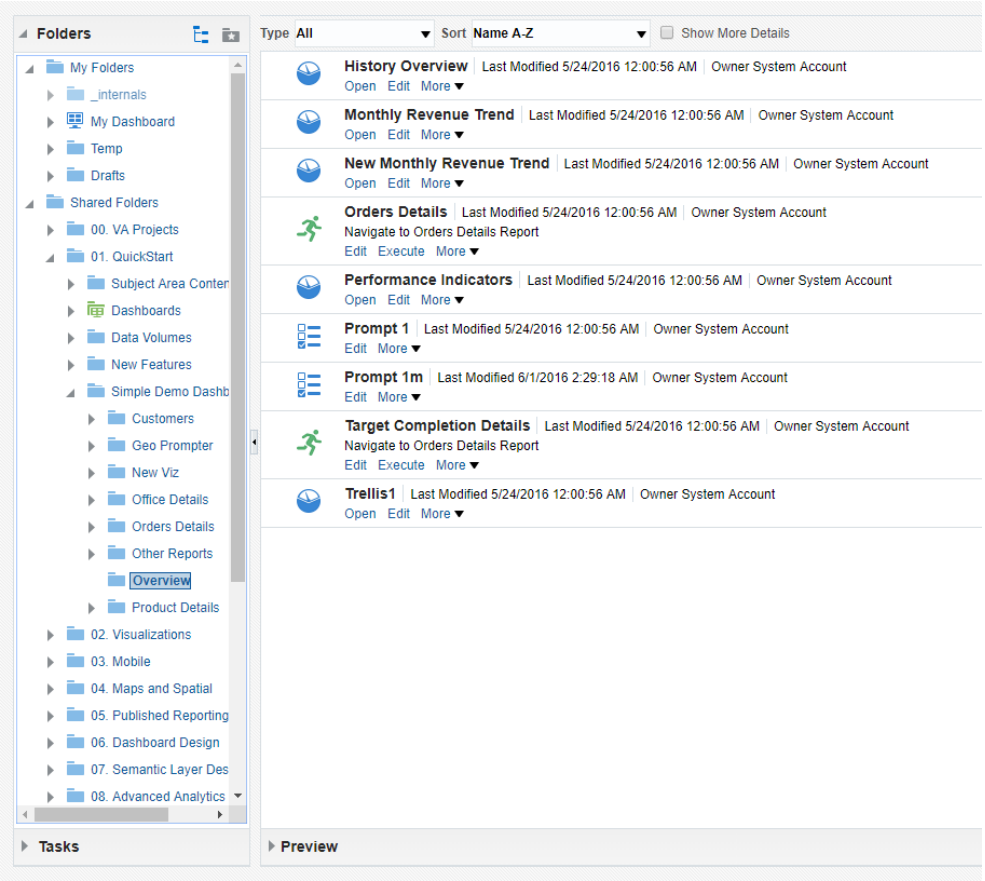
3.1 OBIEE

OBIEE je výkonný nástroj, který umožňuje analyzovat data a prezentovat je ve srozumitelných formátech (dokumentech). Nejpoužívanějšími typy výsledných dokumentů jsou Report, Analysis a Dashboard, které jsou velmi podobné mezi sebou ale mají své výhody a nevýhody v různých specifických situacích. Kvůli rozsáhlosti a složitosti každého z těchto objektů jsme se rozhodli zaměřit na zpracování a vizualizaci datových toků jenom pro Reporty.

3.1.1 Presentation Catalog

Všechny objekty, se kterými můžeme pracovat v rámci OBIEE, jsou dostupné v speciálním úložišti, takzvaném Presentation Catalog (dále Katalog). Katalog je dostupný přes webový prohlížeč a poskytuje uživatelům možnost provádět různé operace s OBIEE objekty (vytvářet, upravovat, exportovat atd.). [8]

Všechny objekty jsou uloženy v adresáři ve formě stromu, v kořenu kterého jsou dvě složky `My Folders` a `Shared Folders` (viz obrázek 3.1). V první složce (`My Folders`) jsou soukromé složky a soubory přihlášeného uživatele, které nejsou viditelné pro ostatní a nemají k sobě přístup zvenčí ani prostřednictvím webových služeb. Složka `Shared Folders` potom naopak obsahuje v sobě soubory a objekty, které jsou společné pro všechny uživatele. Přístup k takovým souborům lze nastavovat přímo v Katalogu pro jednotlivé uživatele nebo celé skupiny. Existuje možnost nejen částečně omezit používání (zakázat upravování, odstraňování, stahování atd.), ale také úplně zakázat přístup k souboru, čímž se stane neviditelný pro vybrané uživatele.



Obrázek 3.1: Ukázka adresářů Katalogu

3.1.2 Webové Služby

Existuje také alternativní způsob používání OBIEE a jeho Katalogu – pomocí webových služeb. Oracle poskytuje pro tyhle účely sadu webových služeb rozdělených do dvou skupin, v první jsou webové služby pro SOA[12] ve druhé jsou takzvané Session-Based[13] služby. Jelikož SOA služby neposkytují dostačující příležitosti ke zpracování a další práci s OBIEE objekty, podíváme se podrobněji na Session-Based webové služby.

[12] Service Oriented Architecture (česky architektura orientovaná na služby)

[13] Session-Based webové služby provádějí všechny operace v rámci jednotlivého spojení se serverem (tzv. session).

3.1.2.1 Session-Based

Při použití Session-Based služeb uživatel po přihlášení dostane unikátní klíč (tzv. session ID), který je platný po celou dobu přihlášení a musí se vyskytovat v každém dotazu ke službě. Díky podobnému přístupu lze se zbavit dlouhé operace přihlašování, která by měla proběhnout před zpracováním každého dotazu, a tím výrazně zrychlit celý proces komunikaci se serverem. Webové služby využijeme primárně ke stažení OBIEE objektů důležitých pro analýzu datových toků. Ve velkých organizacích počet takových objektů může být velmi rozsáhlým, proto využití Session-Based služeb přinese významnou výhodu a několikrát urychlí celkový proces stahování potřebných objektů.

Webové služby poskytované OBIEE jsou založené na protokolu SOAP, a veškerá komunikace s nimi se provádí prostřednictvím zpráv ve formátu XML. Pro zjednodušení integrace webových služeb do aplikace třetích stran Oracle nabízí WSDL soubory, pomocí kterých lze snadno vygenerovat Java třídy a zapojit je do modulu. Pro účely této práce využijeme následující dvě webové služby ze skupiny Session-Based:

`SAWSessionService`
: `SAWSessionService` se používá pro autentifikaci uživatelů (přihlašování a odhlašování) [9]. S její pomocí dostaneme unikátní klíč (session ID), který následně použijeme při komunikaci s ostatními službami.

`WebCatalogService`
: Služba `WebCatalogService` slouží k navigaci v Katalogu a k jeho řízení, umožňuje vytvářet, upravovat a získávat informaci o různých OBIEE objektech [10]. V rámci práce využijeme tuto službu pro nalezení a stahování důležitých pro datové toky objektů.

3.2 OBIEE objekty

V této sekci se podrobně podíváme na existující objekty v nástroji OBIEE, popíšeme jejich vnitřní strukturu, jak jsou provázány mezi sebou a jak je můžeme využít při vizualizaci datových toků.

OBIEE je velmi rozsáhlý nástroj a podporuje velké množství způsobů vizualizaci analyzovaných dat. Nejpoužívanější výsledné dokumenty jsou Report, Analysis a Dashboard, které mají stejný účel ale různou vnitřní strukturu. Právě kvůli rozsáhlosti každého z těchto objektů a jejich specifickým strukturám, v rámci této práce se zabýváme jenom typem Report a souvislými s nim objekty.

Celkově pro analýzu datových toků Reportu potřebujeme tři typy objektů (Report, Layout a Data Model) a o každém z nich podrobněji řekneme v následujících podsekcích.

3.2.1 Report

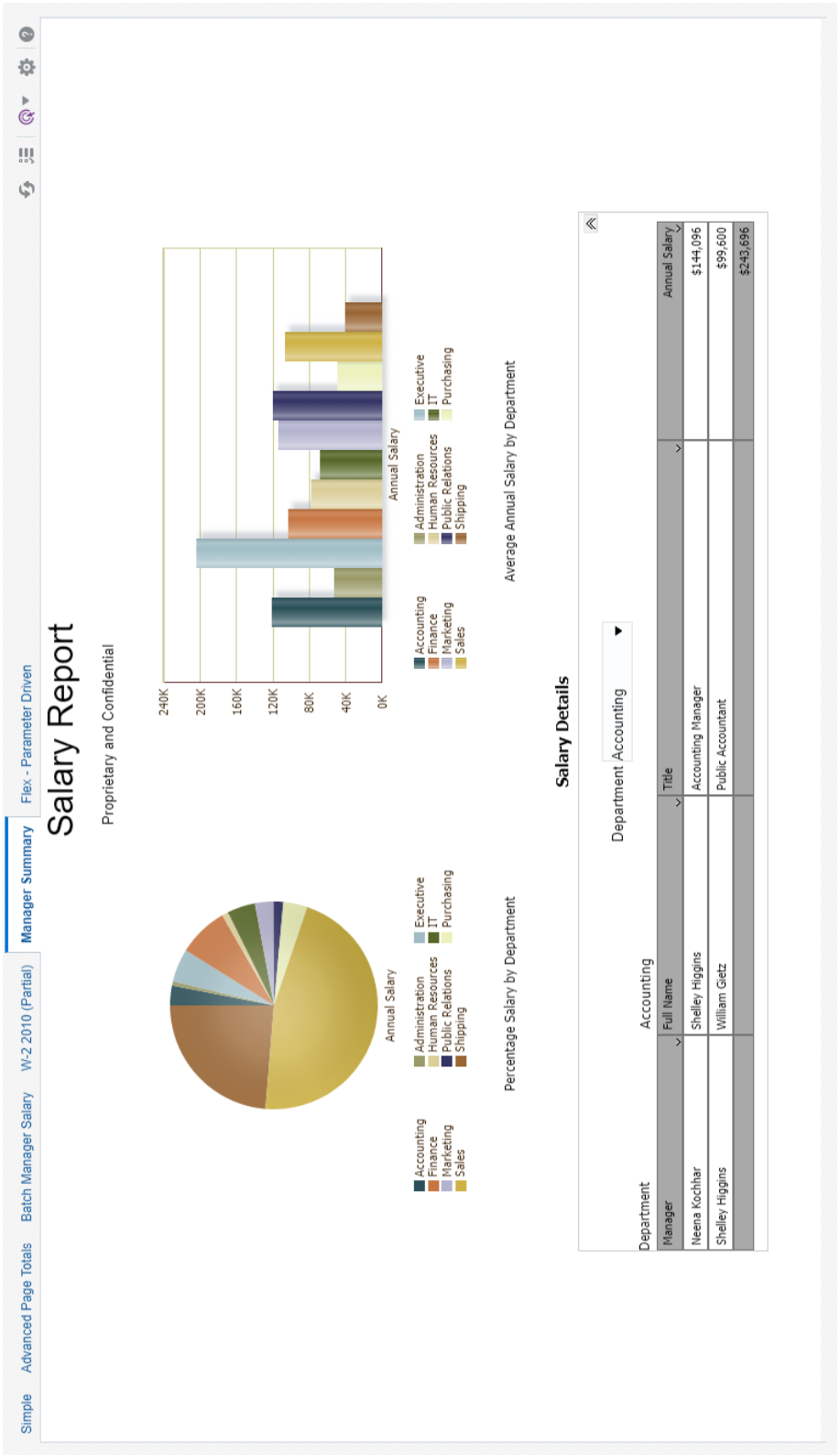
Report představuje sebou dokument, který obsahuje nejčastěji nějaká analytická data v přehledné formě (tabulky, diagramy, obrázky atd.). Report slouží k vizualizaci dat z různých zdrojů, což může značně zjednodušit proces prezentování výsledků založených na těchto datech nebo například jejich následující analýzu. V nástroji OBIEE je Report představen souborem jednostránkových šablon (tzv. Layout), ve kterých jsou jednotlivé objekty vizualizující data ze zdrojů (viz obrázek 3.2). [11]

3.2.1.1 Vnitřní struktura

V Katalogu se Report ukazuje jako soubor s příponou `xdo` ale ve skutečnosti v souborovém systému představuje složku. Tato složka obsahuje v sobě soubor `_report.xdo`, soubory reprezentující jednotlivé Layouty (o nich řekneme podrobněji v sekci Layout) a obrázky šablon (používají se jenom pro náhled v seznamu Layoutů a nenesou v sobě žádnou důležitou informaci). V rámci analýzy jednotlivého Reportu nás zajímá právě soubor `_report.xdo`, který obsahuje všechny potřebné informace pro zjištění datových toků. Soubor je uložen ve formátu XML a má následující strukturu[14] (uvedeny jsou pouze elementy důležité pro analýzu datových toků):

- *dataModel* – atribut ukazující, jestli Report používá jako zdroj dat Data Model (nabízí hodnoty `true` nebo `false`).
- **dataModel** – obsahuje v sobě informaci o použitém Data Modelu, každý Report se může odkazovat nejvíce na jeden Data Model.
  - *url* – cesta k odpovídajícímu Data Modelu v Katalogu.
- **description** – krátký popis Reportu.
- **templates** – seznam všech Layoutů sestavujících Report, každý Report musí mít nejméně jeden Layout.
  - **template** – element popisující jednotlivý Layout.
    - *label* – nadpis, který se ukazuje v horní části stránky v seznamu šablon pri vizualizaci Reportu
    - *url* – cesta k souboru, ve kterém je popsána samotná struktura Layoutu. Jedná se o relativní cestu (v tomto případě obsahuje pouze název souboru), jelikož jsou všechny Layouty patřící k Reportu uloženy v jeho složce.
    - *type* – typ šablony (napr. RTF[15], PDF[16], XLS[17] atd.).

[14] Pro lepší rozlišení **elementy** jsou zvýrazněny tučným písmem a jejich *atributy* kurzivou
[15] Rich Text Format
[16] Portable Document Format
[17] Soubor vytvořeny v aplikaci Microsoft Excel



Obrázek 3.2: Ukázka jedné z šablon Reportu

- *locale* – kód jazyku, ve kterém je uložen Layout (např. `en_US`, `cs_CZ` atd.).

Celkem Report nenese v sobě žádnou informaci potřebnou pro analýzu datových toků, ale obsahuje v sobě důležité odkazy na Data Model a Layouty, ze kterých dostaneme potřebné informace pro vytváření uzlů výsledného grafu. Také Report využijeme pro propojení jednotlivých Layoutů a Data Modelu, ve kterém se nachází informace o použitých datech.

3.2.2 Data Model

Data Model je jeden z typů objektů v Katalogu a používá se pro ukládání informaci o datových zdrojích a popisování struktury použitých dat [12]. Na každý Data Model se může odkazovat více nezávislých na sobě Reportů. Data Model potom obsahuje v sobě dva velmi užitečné pro tuto práci objekty – Data Set a Data Structure.

3.2.2.1 Data Set

Data Set slouží k ukládání informaci o použitých Data Modelem datových zdrojů. Data Model může čerpat data současně z několika Data Setů a samotné Data Sety potom mohou být různých typů. Jako datový zdroj pro Data Set lze použit databázi (SQL), specifický soubor (XLS, XML, CSV[18]) anebo i webovou službu. V rámci této práce se zamíříme na analýzu pouze nejpoužívanějšího a nejrozšířenějšího typu Data Setů – SQL. V tomhle typu je informace o datových zdrojích představena SQL dotazem, ze kterého lze snadno zjistit datové toky v rámci databáze. Takový dotaz vrací jednu databázovou tabulku s několika sloupci, které se potom společně s výsledky jiných Data Setů provážou s dalším důležitým objektem – Data Structure.

3.2.2.2 Data Structure

Data Structure představuje sebou objekt ve formě stromu, ve kterém jsou uloženy informace o všech Data Setech příslušného Data Modelu. V kořenu stromu je samotná struktura, jejími potomky jsou takzvané grupy a listy jsou představeny elementy. Grupy reprezentují jednotlivé tabulky získané z SQL dotazu a obsahují v sobě odkaz na odpovídající Data Set. Vztah mezi grupou a Data Setem je vždycky 1 : 1. Elementy potom popisují sloupce výsledných tabulek, na které se odkazují jednotlivé položky Reportu.

3.2.2.3 Vnitřní struktura

Data Model představuje sebou XML soubor s příponou `xdo` a má následující vnitřní strukturu:

[18] Comma Separated Values

- *defaultDataSourceRef* – odkaz na výchozí Data Source[19], který se použije, pokud v Data Setu nebude uveden žádný jiný.
- **description** – krátký popis Data Modelu.
- **dataSets** – seznam použitých Data Setů.
  - **dataSet** – jednotlivý Data Set.
    - *name* – název Data Setu, na který se potom odkazuje grupa v Data Structure.
    - **sql** – obsah Data Setu (může být různých typů). V případě SQL Data Setu představuje sebou SQL query (dotaz).
      - *dataSourceRef* – odkaz na zdroj dat použitých v Data Setu.
- **dataStructure** – Data Structure objekt popisující strukturu dat. Data Model může mít nejvíce jeden takový objekt, který potom obsahuje informaci o všech Data Setech dohromady.
  - *tagName* – název struktury použity v cestě pro nalezení jednotlivých elementů.
  - **group** – objekt popisující jednotlivou grupu (reprezentuje tabulku z SQL dotazu).
    - *name* – název grupy použity v cestě pro nalezení jednotlivých elementů.
    - *source* – odkaz na Data Set, který byl použit pro získání dat.
    - **element** – popisuje jednotlivé elementy, které jsou vždycky v listech stromu (reprezentuje sloupec tabulky z SQL dotazu).
      - *name* – název elementu.
      - *value* – přesný název sloupce, který element reprezentuje (slouží pro provázání jednotlivých elementů s Data Setem).

Informace, které můžeme dostat z Data Modelu, jsou velmi užitečná pro analýzu datových toků. Pomocí Data Setů zjistíme datové toky objektů v rámci databáze a Data Structure umožní propojit je s toky v jednotlivých Reportech.

3.2.3 Layout

Layout je objekt sloužící k určení obsahu Reportu a popisu rozložení jednotlivých položek (Report Item) na stránce. Layouty jsou neoddělitelnou částí Reportu, ukládají se ve stejné složce s Reportem a mají unikátní názvy v rámci této složky.

[19] Data Source je specificky OBIEE objekt popisující datový zdroj. Nejčastěji Data Source obsahuje v sobě informace pro připojení do databáze (connection string) nebo cestu k úložišti se zdrojovými soubory

3.2.3.1 Typy Layoutů

Typ Layoutu se určuje typem výsledného souboru, OBIEE podporuje následující sedm typů souborů [13]:

- **BI Publisher Layout (XPT[20])** – specifický OBIEE typ Layoutu
- **Rich Text Format (RTF)**
- **Portable Document Format (PDF)**
- **Microsoft Excel (XLS)**
- **XSL[21] Stylesheet**
- **eText** – specifický RTF soubor, který lze použit pro tvorbu textových výstupů pro EDI[22] nebo EFT[23]
- **Flash** – interaktivní Adobe Flash soubor

Všechny typy kromě XPT Layoutu představují sebou externí soubory a nevytvářejí se v rámci OBIEE. To znamená, že tyto typy Layoutů berou data zvenčí a nepoužívají Data Model. Z toho důvodu nejsme schopni získat potřebnou informaci o datových tocích takových Layoutů a nejsou pro účely této práce důležité.

Naopak BI Publisher Layout se vytváří rovnou v OBIEE pomocí nástroje BI Publisher's Layout Editor. Nástroj umožňuje uživatelům přidávat a měnit položky Layoutu, určovat jejich polohu na stránce a používat data přímo z příslušného Data Modelu. Layouty vytvořené takovým způsobem můžeme snadno analyzovat a využijeme je pro získání datových toků jednotlivých položek (tzv. Report Item).

3.2.4 BI Publisher (XPT) Layout

V souborovém systému je BI Publisher Layout představen archivním souborem s příponou `xpt`. Uvnitř archivu jsou tři soubory: `version.txt` (textový soubor obsahující verzi Layoutu), `businessView.xml` (soubor popisující strukturu použitých dat) a `layout.xml` (obsahuje popis položek Reportu). Soubor `businessView` je zjednodušená kopie objektu Data Structure z Data Modelu, z toho důvodu nebudeme ho potřebovat pro další analýzu. Jediným důležitým souborem je `layout.xml`, který obsahuje v sobě podrobný popis všech položek (Report Item) ve formátu XML.

[20] XPT je speciální formát, ve kterém se ukládají Layouty vytvořené pomocí nastroje BI Publisher's Layout Editor

[21] Extensible Stylesheet Language

[22] Electronic Data Interchange

[23] Electronic Funds Transfer

3.2.4.1 Vnitřní struktura

Layout ma uvnitř stromovou strukturu, v kořenu stromu je vždy jediná stránka (tzv. Page) toho Layoutu. Stránka se potom dělí na jednotlivé části:

- **Body** – nezbytná část stránky, obsahuje hlavní informaci a položky Reportu
- **Header** – záhlaví, které se opakuje na každé stránce[24]
- **Footer** – patička, která se opakuje na každé stránce
- **ReportHeader** – záhlaví celého Layoutu, které se vyskytuje jenom jednou na začátku
- **ReportFooter** – patička celého Layoutu, která se vyskytuje jenom jednou na konci

Z pohledu vnitřní struktury jsou všechny části stránky stejné a jsou pouze kontejnery pro jednotlivé položky Layoutu neboli Report Itemy.

3.2.4.2 Report Item

Report Item je položka Layoutu, která nese v sobě informaci z Data Modelu v přehledné formě tabulky, diagramu atd. Ve výsledné vizualizaci jsou Report Itemy koncovými body datových toků. Celkově OBIEE podporuje 9 různých typů položek:

- **Layout Grid** – mřížka, která v sobě nenese žádnou informaci a je pouze kontejnerem pro ostatní Report Itemy
- **Repeating Section** – speciální položka, která je také kontejnerem a navíc umožňuje opakování všech vnitřních objektů podle zvoleného parametru
- **Data Table** – obyčejná tabulka se sloupci
- **Pivot Table** – kontingenční tabulka neboli cross tabulation
- **Chart** – diagramy různých typů (sloupcové, kruhové, liniové atd.)
- **Gauge** – diagram ve formě měřidla
- **Data List** – seznam všech položek vybraného elementu
- **Text Box** – jednoduché textové pole

[24] Ve vnitřní struktuře je vždy přesně jedna stránka, ale ve výsledném dokumentu může být vizuálně rozdělena pomocí speciální položky PageBreak

- **Image** – obrázek (jako i Text Box nenese v sobě žádnou informaci o datových zdrojích)

Každý Report Item obsahuje v sobě takzvaný Field, který představuje sebou cestu k jednotlivému elementu v Data Structure v Data Modelu. Díky Fieldům jsme schopni propojit položky Layoutu s jejich datovými zdroji a tím způsobem dokončit popis datových toků.

Kapitola 4

Návrh a implementace

V této kapitole se budeme věnovat návrhu a implementaci samotného modulu, řekneme o zvoleném řešení a podrobně popíšeme jednotlivé části modulu.

Celkově pro vizualizaci datových toků jednoho OBIEE serveru musí modul postupně projít následujícími kroky:

1. Zanalyzovat celý Katalog a najít všechny objekty, které jsou důležité pro analýzu datových toků.

2. Stáhnout tyto objekty a uložit je na disk ve formě odpovídajících souborů.

3. Následně načíst soubory z disku a provést jejich analýzu.

4. Vytvořit model Java objektů, které budou reprezentovat a přesně popisovat analyzované soubory.

5. Na základě Java modelu vytvořit graf, který bude následně využit pro vizualizaci datových toků.

V softwaru Manta Flow již existuje několik podobných modulu, které ale slouží k analýze a vizualizaci datových toků jiných reportovacích nástrojů nebo jazyků. Řešení, které bylo použito při implementaci existujících modulů, zvládne provést všechny výše uvedené kroky, je vyzkoušené uživateli aplikace a ideálně vyhovuje i pro práci s OBIEE. Právě z těchto důvodů jsme rozhodli implementovat prototyp stejným způsobem a rozdělili jsme ho na tři menších modulu:

- **Extractor** – stahuje objekty ze serveru a ukládá je do souborů na disk.

- **Resolver** – načítá soubory z disku a vytváří model odpovídající jejich struktuře.

- **Generator** – na základě vytvořeného modelu generuje graf, který se následně používá pro vizualizaci datových toků.

Každý modul představuje sebou samostatnou funkční jednotku a může fungovat nezávisle na dvou ostatních. Takový přístup dovoluje implementovat jednotlivé moduly zvlášť a testovat je ještě před dokončením celkového prototypu. V následujících sekcích podrobně rozebereme každý z uvedených modulů.

4.1 Extractor

Hlavním cílem Extractoru je stáhnout všechny potřebné soubory z OBIEE serveru a uložit je na disk. Pro splnění toho cíle slouží třída `AbstractExtractor` a rozšiřující ji třída `SoapExtractor`.

`AbstractExtractor` je abstraktní třída, která je základem pro různé typy Extractorů (v našem případě tahle třída je trochu zbytečná, ale zvyšuje čitelnost kódu a přináší možnost jednoduchého rozšíření modulu o další typy Extractorů). Hlavní činnost třídy se provádí v metodě `extract`, která se volá po spouštění Extractoru a skládá se z dvou částí. V první částí se zavolá metoda `extractItems` (její činnost je popsána v rámci třídy `SoapExtractor`), která vrací seznam objektů vyhovujících ke stažení. Objekty jsou představeny speciální třídou `ObieeDAO`, která obsahuje informaci o cestě k objektu v Katalogu a jeho typ (jak jsme uváděli v analýze, zatím jsou to objekty tří typů – Report, Data Model a Layout). Ve druhé části se prochází vytvořený seznam a pomocí metody `extractItemContent` se zpracovává každý objekt a ukládá se do souboru na disk.

Třída `SoapExtractor` rozšiřuje předchozí třídu a přináší doplňky specifické pro SOAP Extractor. Tenhle Extractor komunikuje se serverem pomocí popsaných v analýze Session-Based webových služeb, které jsou založeny právě na SOAP protokolu. Tyto služby poskytují velkou množinu metod umožňujících skoro libovolnou práci s OBIEE nástrojem, ale pro účely Extractoru stačí následující tři:

- `logon` – použijeme pro přihlášení do serveru a zjištění unikátního session ID. Metoda se nachází ve třídě `SAWSessionServiceSoap` (tahle a všechny ostatní třídy webových služeb byly vygenerovány z WSDL souboru, který jsme zmínili v předchozí kapitole).

- `getSubItems` – využijeme pro vyhledávání potřebných objektů, patří do třídy `WebCatalogServiceSoap`. Tahle metoda dostává na vstup cestu ke složce a vrací seznam objektů, které se v ní nachází. Rekurzivní volaní funkce od kořenové složky Katalogu dovolí projít celý adresář a nalézt všechny potřebné objekty.

- `readObjects` – druhou metodu služby WebCatalogService použijeme pro získání samotných objektů. Metoda vrací řetězec popisující objekt v jazyce XML, který následně můžeme uložit do souboru na disk.

Dvě výše uvedené metody třídy `AbstractExtractor` (`extractItems` a `extractItemContent`) jsou přetíženy ve třídě `SoapExtractor`. V první metodě pomocí funkce `getSubItems` projdeme rekurzivně celý Katalog a přidáme do výsledného seznamu všechny objekty odpovídající podmínkám: typ objektu je podporován modulem (Report, Data Model, Layout) a samotný objekt vyhovuje vstupním parametrům `include` a `exclude`[25]. Druhá metoda dostane na vstup jeden `ObieeDAO` objekt, pomocí funkce `readObjects` získá XML řetězec popisující objekt z Katalogu, a uloží ho do souboru na disk.

Výsledkem Extractoru je tedy adresář přesně odpovídající struktuře Katalogu, ve kterém jsou soubory se všemi nalezenými podporovanými objekty.

4.2 Resolver a Model

Model reprezentuje sebou soubor Java rozhraní, které slouží k podrobnému popisu každého objektu a jejich částí potřebných pro vizualizaci datových toků. Resolver je potom soubor tříd, které implementují jednotlivá rozhraní Modelu a již obsahují v sobě i logiku vytváření Java objektů na základě vstupních XML řetězců. Takové rozdělení rozhraní a jejich implementací umožňuje následně v Generatoru používat pouze Model jako řídicí jednotku pro vytváření objektů, přičemž všechna logika se bude provádět v Resolveru, ke kterému Generator ani nemusí mít přístup. Hlavním cílem Resolveru je načíst soubory z disku, zpracovat je a vytvořit Java objekt reprezentující OBIEE server a obsahující všechny důležité objekty.

4.2.1 Načítání souborů

Pro načítání souborů z disku jsou vytvořeny dvě třídy `ObieeProjectReader` a `ObieeFileReader`. První třída postupně prochází všechny soubory ze složky, která je uvedena v parametru `inputFile`, a každý jednotlivý soubor předává na zpracovaní třídě `ObieeFileReader`. Tahle třída již dokáže na základě specifického názvu a přípony souboru rozpoznat jeho typ a zavolat odpovídající metodu pro získání potřebných informací. Během analýzy jsme zjistili, že všechny objekty jsou uloženy ve formátu XML, a proto pro jejich načtení využijeme již existující v projektu Manta `XmlStreamReader`. Tato třída dokáže zanalyzovat XML soubor a vytvořit Java objekt, který přesně odpovídá jeho struktuře. V případě Reportu a Data Modelu můžeme rovnou využít

[25] Parametry `include` a `exclude` se nastavují uživatelem v konfiguračním souboru modulu. Oba dva parametry představují sebou regulární výrazy obsahující absolutní cesty k objektům (`include` – objekty ke stažení, `exclude` – objekty, které se nesmí stahovat.)

`XmlStreamReader` a dostat objekty potřebné pro další analýzu. Ale pro zpracování Layoutu nejdřív potřebujeme rozbalit stažený soubor. K tomu účelu jsme vytvořili speciální metodu `decompressFile`, která dokáže rozbalit `xpt` soubor do paměti a dostat z něho potřebný `layout.xml` ve formě pole bytů. S takovým polem již dokáže pracovat `XmlStreamReader` a tím způsobem dostaneme všechny nezbytné informace i o Layoutu.

4.2.2 Struktura Modelu

V této sekci probereme strukturu Modelu a řekneme o jeho nejzajímavějších částí.

ObieeNode
: Jedním z nejdůležitějších rozhraní je `ObieeNode`, které je rodičem pro všechny ostatní rozhraní. Tohle obecné rozhraní umožňuje vytváření Modelu ve formě stromu, ve kterém každý objekt představuje jeden uzel. Z toho důvodu každý objekt implementující `ObieeNode` musí mít vlastní název a odkazovat se na svého rodiče ve stromové struktuře. Tímhle způsobem zajistíme unikátnost objektů ve výsledném grafu, který vytvoří Generator na základě tohoto Modelu.

ObieeServer
: Výsledek celé práci Resolveru potřebujeme uložit do jednoho objektu, který bychom mohli jednoduše předat Generatoru pro následné zpracovávání. Právě k tomu jsme vytvořili rozhraní `ObieeServer`, které představuje sebou kontejner pro podporované objekty (Report, Data Model a Layout) a ve výsledném grafu bude kořenem stromu a zároveň předkem pro všechny Reporty a Data Modely.

CatalogObject
: Pomocí rozhraní `CatalogObject` můžeme popsat všechny typy objektů, které jsou uloženy v OBIEE Katalogu (zatím jsou to Report, Data Model a Layout). Oproti obecnému rozhraní `ObieeNode`, `CatalogObject` obsahuje navíc cestu (tzv. path) k souboru v Katalogu. Díky této informaci můžeme rozlišovat objekty, které mají stejný název ale nacházejí se v různých složkách.

ReportItem
: Nejzajímavějším rozhraním v Modelu je `ReportItem`. Formálně mezi Report Itemy patří jenom 9 objektů, které jsme schopni přidat do Layoutu v editoru, ale v tomto Modelu rozhraní `ReportItem` popisuje každý objekt a každou jeho část vyskytující se v Layoutu. Všechny takové objekty se potom deli na 4 skupiny:

  - 9 hlavních Itemů, které můžeme ručně přidat v Layout Editoru, jsou reprezentovány rozhraním `MainItem`.

- Rozhraní `ReportItemContainer` popisuje všechny objekty, které se používají jako kontejnery pro další položky Layoutu (příkladem jsou jednotlivé části stránky Layoutu popsané v předchozí kapitole).
- Některé Report Itemy a jejich části obsahují odkazy na elementy datové struktury v Data Modelu. Takový odkaz se nazývá Field a odsud pochází i název příslušného rozhraní – `FieldItem`.
- Poslední skupinu tvoří všechny ostatní Report Itemy a jsou popsány obecným rozhraním `ReportItem`.

Tyhle skupiny nejsou disjunktní a některé Report Itemy mohou patřit současně do dvou skupin (viz obrázek 4.1).

Vytvořená struktura skoro přesně odpovídá skutečné vnitřní struktuře stažených souborů, což velmi usnadňuje celkový proces jejich zpracovávání a vytváření Modelu. Díky tomu dostaneme ve výsledku také podrobnější graf datových toků, ve kterém se zobrazí co nejvíc objektů a jejich částí. Přebytek položek v grafu není problém, protože částí softwaru Manta Flow jsou i speciální filtrační moduly, které dokážou odstranit z grafu všechny zbytečné uzly.

4.3 Generator

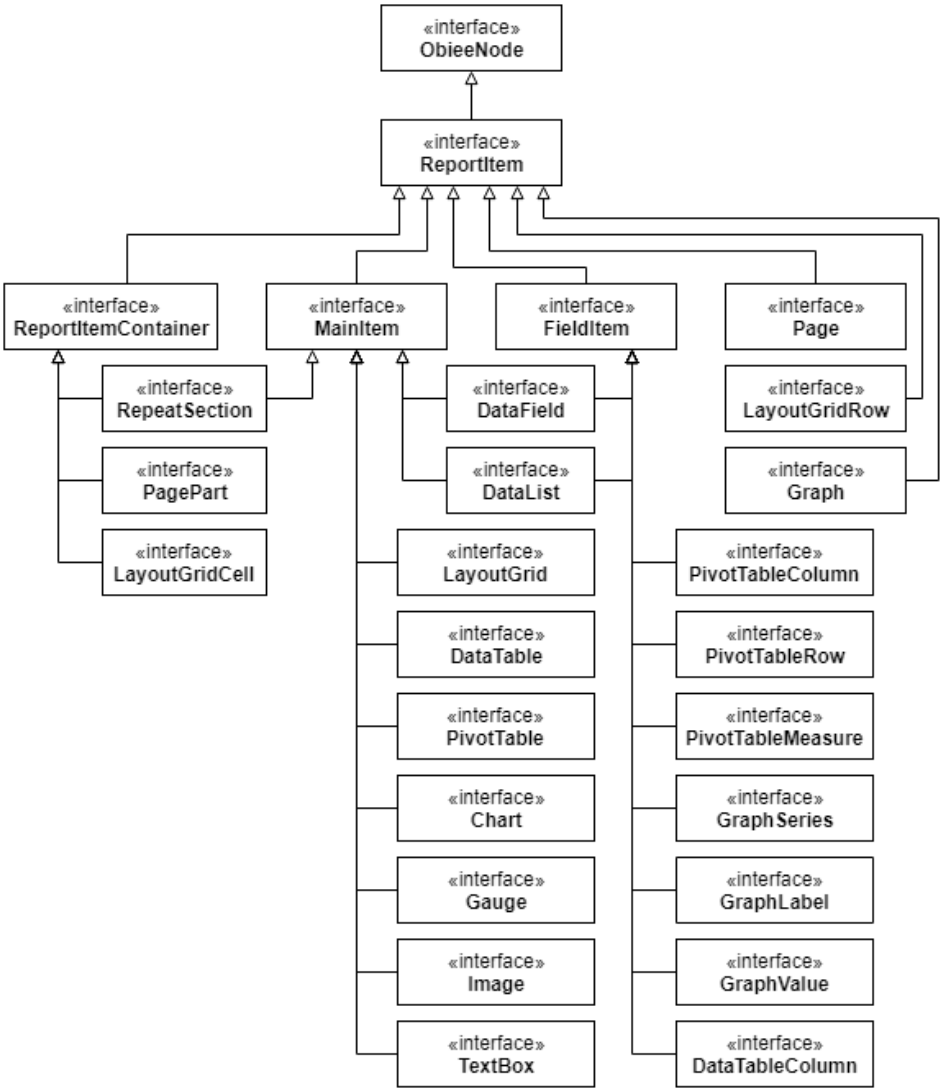
Účelem Generatoru je vytvořit graf datových toků na základě modelu získaného od Resolveru. Celkově třídy tohoto modulu jsou rozděleny do třech skupin:

- **analyzers** – speciální třídy (tzv. analyzátory), které analyzují Report Itemy a jejich jednotlivé části a získávají data potřebné pro sestavení výsledného grafu.
- **connectors** – skupina rozhraní a tříd, které se zabývají vytvářením grafu datových toků v rámci datových zdrojů.
- **utils** – pomocné třídy sloužící k vytváření výsledného grafu datových toků.

Dále se podrobněji podíváme na každou ze skupin a řekneme o jejich zajímavostech.

4.3.1 Analyzers

Úkolem tříd této skupiny je vytvořit první část grafu datových toků: sestavit strom všech objektů získaných od Resolveru a přidat datové toky mezi jednotlivými Report Itemy a Data Modelem. Ke splnění toho úkolu slouží speciální



Obrázek 4.1: Hierarchie rozhraní `ReportItem`

analyzátory, které dokážou ze vstupního objektu získat informaci použitelnou pro následné vytváření uzlů grafu. Každý analyzátor odpovídá jenom za jeden přesně určený objekt nebo jeho část, a po její zpracování předá ostatní částí jiným příslušným analyzátorům.

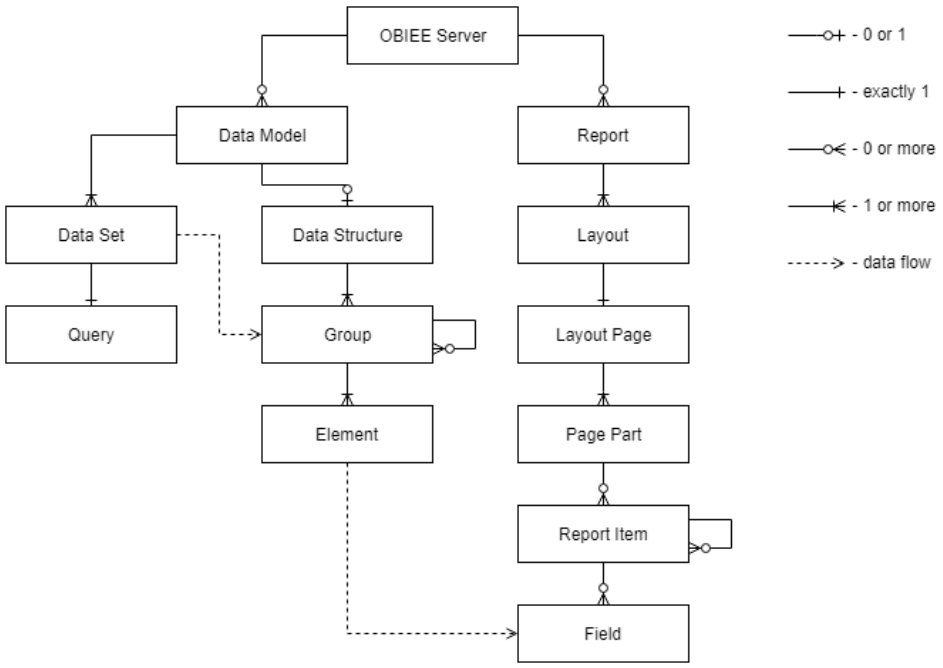
Pro větší pochopení rozebereme následující příklad práce analyzátoru serveru, který se volá na celý vstupní model. Třída `ServerAnalyzer` dostává na vstup objekt `ObieeServer` reprezentující celkový model, který obsahuje v sobě dva seznamy objektů – Reporty a Data Modely. Analyzátor nejdřív projde obě dvě kolekce a potom každý získaný objekt pošle dalšímu odpovídajícímu analyzátoru (`ReportAnalyzer` nebo `DataModelAnalyzer`). Tenhle proces se opakuje, pokud nedojde k analýze objektu typu `FieldItem`. Ve výsledném grafu takový objekt je vždycky listem a musíme ho propojit s příslušným datovým zdrojem. K tomu slouží metoda `connectToDataStructure`, která se nachází ve třídě `AbstractAnalyzer`. Tato abstraktní třída vystupuje základem pro všechny ostatní analyzátory a obsahuje v sobě všechny jejich společné funkce. Metoda `connectToDataStructure` dostane na vstup `FieldItem`, pomocí Fieldu z toho objektu nalezne odpovídající element datové struktury v Data Modelu a do výsledného grafu přidá hranu popisující datový tok mezi nimi (Data Modely se analyzují před Reporty a máme zajištěno, že příslušná Data Structure již bude zpracována). Stejným způsobem procházíme všechny položky Reportu a postupně je přidáváme do výstupního grafu.

Výsledkem práci analyzátorů je tedy část grafu datových toků, která obsahuje všechny objekty získané od Resolveru a jejich jednotlivé části. Tyto objekty jsou reprezentovány uzly a jsou propojeny hranami představujícími datové toky mezi nimi (struktura grafu je podrobněji popsána na obrázku 4.2).

4.3.2 Connectors

Následující skupina tříd se zabývá datovými toky v rámci databází, což je druhá část výsledného grafu. K vytváření grafu slouží `DatabaseConnector` a implementující ho třída `DatabaseConnectorImpl`. Hlavní jejich činnost je představena v metodě `findQuerySources`, která nejenom zanalyzuje datový zdroj a vytvoří odpovídající graf datových toků, ale i vrátí seznam datových sloupců potřebných pro spojení dvou částí grafu. Metoda dostane na vstup informaci o datovém zdroji a SQL dotaz, které získáme z příslušného Data Setu. Po provedení analýzy dotazu a databáze vrátí seznam sloupců, které jsou výsledkem SQL dotazu a přesně odpovídají jednotlivým elementům v Data Structure.

`DatabaseConnector` se používá ve třídě `DataStructureAnalyzer` a díky tomu jsme schopni propojit jednotlivé elementy objektu Data Structure a jejich datové zdroje již během analýzy Data Modelu. Výsledkem spolupráci rozhraní `DatabaseConnector` a analyzátorů je kompletní graf datových toků.



Obrázek 4.2: Struktura Data Flow grafu

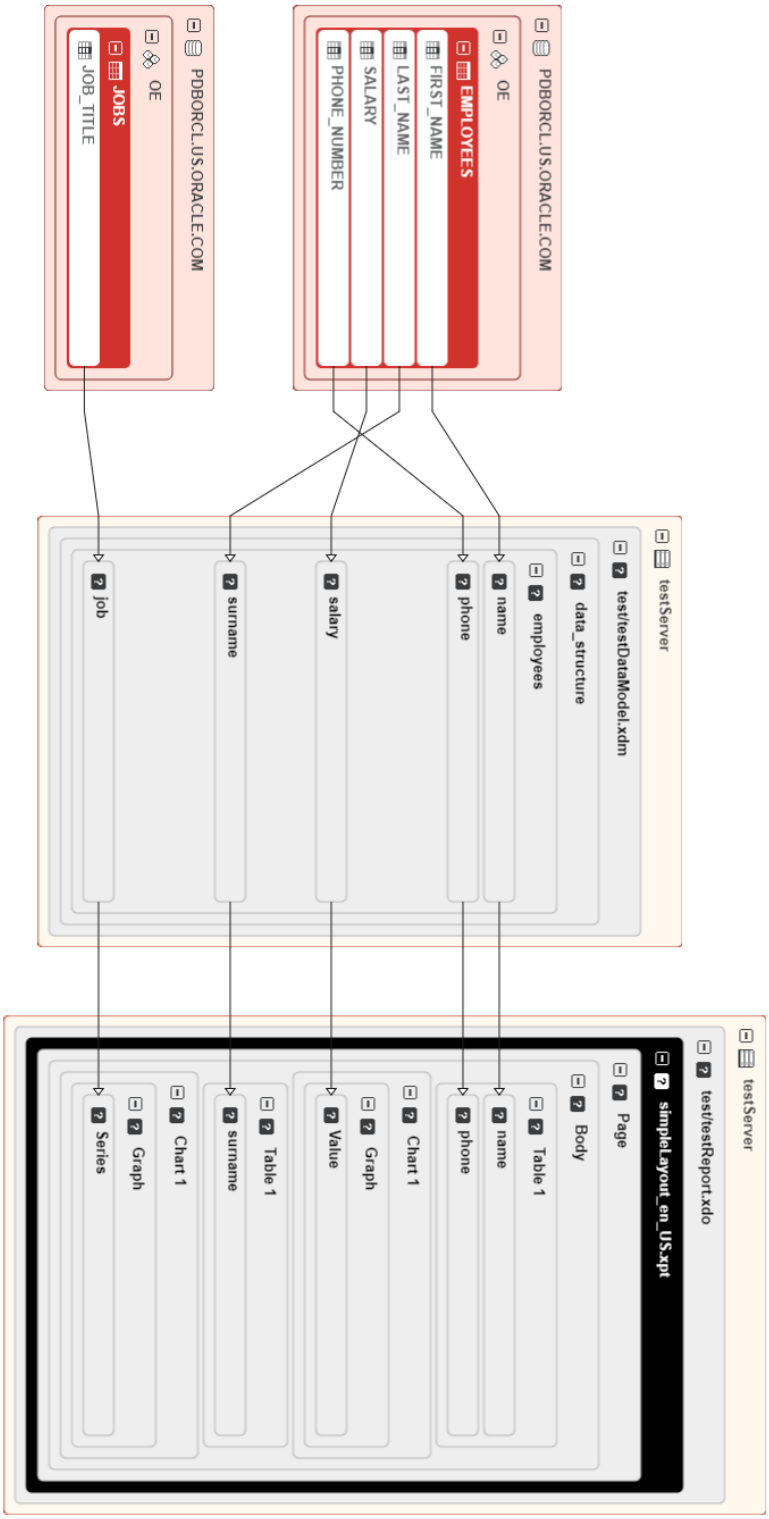
4.3.3 Utils

Poslední skupina obsahuje dvě třídy `ObieeUtils` a `ObieeGraphHelper`, které jsou pomocnými třídy pro práci se vstupním modelem a výstupním grafem. První třída pracuje s objekty modulu a slouží k nalezení jejich rodičů. Tuto funkci jsme často využíváme při propojování jednotlivých Report Itemů a odpovídajících jim elementů v Data Modelu, který můžeme získat pouze pomocí rodičovského Reportu položky.

Druhá třída slouží k sestavení výsledného grafu a udržovaní jeho aktuální verze během vytváření. Nejčastěji používána metoda je `buildNode`, která dokáže na základě vstupního objektu typu `ObieeNode` vytvořit a přidat do grafu uzel odpovídajícího typu. Zároveň tato metoda vytváří uzly i pro všechny rodiče objektu do samého kořenu. Z toho důvodu stačí zavolat metodu `buildNode` pouze na objekt typu `FieldItem` (v grafu je vždycky listem) a uzly pro všechny jeho rodiče se vytvoří automaticky. Také je zajištěna unikátnost uzlu v grafu a nemusíme se starat o vícenásobné zpracovávání stejného objektu (ve výsledku se takový objekt objeví jenom jednou).

4.4 Shrnutí

Výsledkem implementační části této práce je funkční prototyp, který se skládá ze tří částí: Extractor, Resolver a Generator. Každá část je nezávislá na ostatních a dokáže pracovat samostatně, a výsledkem jejich spolupráci je graf datových toků, který je již možno zobrazit v nástroji Manta Flow (viz obrázek 4.3).



Obrázek 4.3: Ukázka datových toků jednoho Layoutu

Kapitola 5

Testování

V této kapitole se budeme zabývat popisem provedeného testování modulu pomocí frameworku JUnit. Pro každou části modulu jsme vytvořili jednotlivé třídy pro testování, účelem kterých otestovat funkčnost celého prototypu na různých typech vstupních dat a v různých situacích. Jedná se o takzvané jednotkové testy, které testují malé části aplikace nezávisle na sobě. Z toho důvodu každá testovací třída obsahuje v sobě několik metod, každá z kterých testuje jenom jednu určitou věc. Pomocí vytvořených testů jsme schopni ověřit celkovou funkčnost modulu a jeho jednotlivých částí a také urychlit proces nalezení chyb v případě změn nebo přidání dalších funkcionalit.

Dále podrobněji popíšeme samotné třídy a jejich metody, které jsou vytvořeny pro každou ze třech částí modulu.

5.1 Extractor

K testování první části modulu slouží třída `ExtractorTest`, která má za účel otestovat správnost Extractoru při práci s podporovanými a nepodporovanými typy objektů a také při špatném připojení nebo nedostupnosti serveru. Jednotlivé testy se dělí na dvě skupiny, kde v první jsou testy, které předpokládají úspěšné připojení k serveru a testuji jenom samotné zpracování objektů, a ve druhé jsou testy zabývající se případy špatného připojení k serveru. Třída má v sobě následující metody patřící do první skupiny:

`setUpExtractor`
: Tato metoda je označena JUnit anotací `@Before`, což znamená, že se metoda automaticky volá před spouštěním každého testu ve třídě. Metoda slouží k vytváření a konfiguraci instance Extraktoru, nastavují se cesta ke složce pro stahování souborů, jméno a heslo uživatele, adresa a port pro připojení k serveru OBIEE.

`extractAllItems`
: Jedná se o test, účelem kterého otestovat celkovou funkčnost Extractoru. Metoda nemá žádné specifické podmínky a jednoduše se zkouší stáhnout a uložit na disk všechny soubory ze serveru. Pomocí tohoto testu můžeme zajistit schopnost modulu pokračovat práci v případě naražení na nepodporované objekty, rychle odhalit jednoduché a očividné chyby. Test se také používá pro ladění aplikace během implementace, jelikož na prvních fázích tvorby je jediným způsobem spustit aplikaci.

  Test končí úspěchem, pokud nedojde k vyhození žádné výjimky za běhu programu a pokud ve složce s výstupy se objeví alespoň jeden soubor.

`extractSupportedItems`
: Tahle metoda reprezentuje test, cílem kterého prokázat schopnost Extractoru správně stáhnout a uložit všechny podporované typy objektů OBIEE. Jedná se o 4 soubory (Report, Data Model a 2 Layouty), které představují sebou skupinu všech momentálně podporovaných OBIEE objektů.

  Test se počítá za úspěšný, pokud nedojde k žádné chybě během zpracování objektů a uloží se právě 4 soubory.

`extractUnsupportedItems`
: Tenhle test je velmi podobny předchozímu ale již pracuje s nepodporovanými objekty (v rámci tohoto testu využijeme objekty typu Analysis, jelikož chování aplikace je stejné pro všechny typy nepodporovaných souborů). Cílem testu je zkontrolovat, že program nespadne při naražení na neznámý objekt ale jednoduše ho přeskočí a zaznamená to do logu.

  Pro započítání testu mají být v logu záznamy o nezpracovaných souborech a výstupní složka má být prázdná.

`extractWithWrongIncludeParametr`
: Úkolem testu je zkontrolovat správné chování Extractoru v případě, kdy byla zadána špatná cesta pro stažení objektů a nemůže se stáhnout žádný soubor.

  Tenhle test končí úspěchem, pokud nevyhodí žádnou výjimku za běhu, zaznamená chybné zadání cesty a nestáhne žádný soubor.

Následující dvě metody patří do druhé skupiny, která se zabývá testováním připojení k serveru:

`extractWithWrongUsernameOrPassword`
: Tato metoda slouží k otestování Extractoru v případě špatně zadaných uživatelských údajů.

  Test se počítá za úspěšný, pokud nespadne a zaznamená do logu, že byli použity špatné přihlašovací údaje.

`extractFromWrongAddress`
: Poslední test kontroluje správnost chování modulu při zadání chybné adresy pro připojení k serveru nebo při jeho nedostupnosti.

  Podmínky pro úspěšné ukončení jsou stejné jako v předchozím testu.

5.2 Resolver

Zpracování souborů do Java objektů Resolverem se testuje současně s jejich načítáním z disku ve třídě `ReadInputTest`. Testy mají za úkol zkontrolovat správnost načítání souborů z disku a jejich následného zpracování, otestovat chování programu v případě zadání nesprávných konfigurací, chybějících souborů nebo souborů s porušenou strukturou.

Testovací třída `ReadInputTest` obsahuje následující metody:

`setUpReaders`
: Stejně jako v testu Extractoru tahle funkce patří mezi `@Before` metody a spouští se automaticky pred každým jednotlivým testem ve třídě. Metoda vytváří instance tříd `XmlStreamReader`, `ObieeFileReader` a `ObieeProjectReader`, které jsou potřeba pro načtení souborů z disku a převedení XML textů do Java objektů. Také se nastavují jejich konfigurace včetně cesty ke složce se vstupními soubory.

`readExtractorOutput`
: Jednoduchý test, který stahuje všechny soubory z výstupní složky Extractoru. Nejedná se o test kontrolující správnost funkčnosti modulu ale o test pro spouštění třídy `ObieeProjectReader` a následné ladění Resolveru. Tento test by se měl pouštět pouze po proběhnutí testu Extractoru, který stáhne alespoň jeden soubor, a bude ignorován, pokud vstupní složka je prázdná.

`readCorrectFiles`
: Pozitivní[26] test, účelem kterého zajistit, že modul správně načte všechny vstupní soubory a bez chyb je zpracuje do Java tříd.

  Test končí úspěchem, pokud načte přesné číslo souborů (1 Report, 1 Data Model a 2 Layouty) a správně je převede do odpovídajících Java objektů.

`readReportWithMissingDataModel`
: Následující metoda testuje schopnost modulu správně zpracovat Report, pokud chybí odpovídající Data model. Tohle se může stát, pokud dojde k nějaké chybě Extractoru při stažení souboru, nebo pokud uživatel ručně odmaže soubor z disku.

[26] Pozitivní testy kontrolují správnost chování programu v případě správných vstupních dat.

Pro úspěch testu se má správně načíst Report a všechny jeho Layouty, i když Data Model chybí. V takovém případě metoda `getDataModel` ve třídě `Report` reprezentující skutečný Report musí vrátit `null`.

`readReportWithMissingLayout`
Tahle metoda je velmi podobná předchozí, jenom slouží k otestování případu chybějícího Layoutu.

Správným výsledkem je načtení Reportu, Data Modulu a prvního Layoutu (druhý chybí). Chybějící Layout musí chybět i v seznamu Layoutů třídy `Report`.

`readFilesWithWrongStructure`
Tenhle test má za úkol zaručit, že soubory s porušenou vnitřní strukturou se nebudou zpracovávat. Tady se používají soubory z výše uvedeného testu `readCorrectFiles`, které byli předem změněny a mají strukturu nepoužitelnou pro Resolver.

Test se počítá za úspěšný, pokud se nestáhne žádný soubor a všechny nastalé chyby se zaznamenají do logu.

`readFilesFromWrongPath`
Následující metoda slouží k otestování případu, když chybí složka zadaná jako vstupní.

Test končí úspěchem, pokud modul vyhodí výjimku o špatně zadané cestě vstupního souboru.

`readFilesFromEmptyDirectory`
Poslední metoda pouze kontroluje, že program nespadne při spouštění nad prázdnou složkou. Test nesmí oznámit žádnou chybu nebo spadnout, a výsledná instance serveru nesmí mit v sobě žádné objekty.

5.3 Generator

Pro testování Generatoru používáme třídy `ObieeTestBase` a `FilesFlowTest`.

`ObieeTestBase` slouží jako základ pro všechny testy modulu a obsahuje v sobě metody, které jsou společné pro všechny testovací třídy. Ve třídě se nastavuje Spring kontext, vytvářejí se speciální slovníky, které jsou potom použity při simulaci databází, a načítají se soubory pro následnou analýzu datových toků. Použití společného předka pro testovací třídy dovoluje snadno přidávat další testy bez zbytečného opakování kódu. Zatím implementována pouze jedna třída pro testování modulu, ale do budoucna rozhodně počítáme s přidáním i dalších.

Samotné testování Generatoru se provádí ve třídě `FilesFlowTest`, která obsahuje jedinou testovací metodu `testFlow`. Tato metoda má za úkol načíst

soubory z disku a zpracovat je do Java modelu, potom vytvořit graf odpovídající struktuře modelu a nakonec uložit výsledný graf na disk ve formě textového souboru a obrázku. Pro úspěšný průchod testem musí výsledná textová reprezentace grafu přesně odpovídat předem vytvořenému souboru, ve kterém je správný popis očekávaného grafu.

Závěr

Hlavním cílem této práce bylo navrhnout a implementovat funkční prototyp modulu pro analýzu a zpracování datových toků nástroje OBIEE. V rámci práce jsme provedli analýzu reportovacího nástroje OBIEE a prozkoumali existující možnosti zpracovávání jeho objektů. Na základě provedené analýzy jsme navrhli a implementovali modul, který dokáže stáhnout ze serveru specifické OBIEE dokumenty, následně je zpracovat a převést do Java modelu, ze kterého lze snadno vygenerovat graf pro vizualizaci datových toků. Výsledný modul je zcela zdokumentován prostřednictvím JavaDoc a řádně otestován pomoci JUnit. Tím způsobem jsme splnili všechny hlavní a dílčí cíle této práce.

Vytvořený modul ma zatím omezenou funkcionalitu, jelikož se jedná pouze o prototyp, ale je připraven ke zlepšením. Máme v plánu pokračovat v jeho rozšíření, přidat možnost zpracovávání jiných specifických dokumentů než Report (Dashboard, Analysis) a umožnit práci s jinými datovými zdroji než SQL databáze. Modul se vytvářel s předpokladem následného zařazení do softwaru Manta Flow a již od začátku implementace používá všechny potřebné prostředky systému. Proto můžeme již v nejbližší době nasadit prototyp do produkce jako demo verzi.

Tato práce stala pro mě velkým osobním přínosem, neboť je mým prvním významným projektem, na kterém jsem horlivě pracoval samostatně dlouhou dobu. Během implementace prototypu jsem rozšířil své znalosti jazyka Java, naučil jsem se pracovat s takovými technologiemi jako Spring a Maven a získal jsem mnoho zkušeností s vytvářením softwaru užitečného pro velké organizace. Velice mě bavil celkový proces vytváření modulu a určitě bych rád pokračoval v jeho zlepšení a rozšíření o další funkcionalitu.

Literatura

1. *What is Business Intelligence? Definition and Example* [online]. © 2019 [cit. 2019-05-04]. Dostupné z: https://www.guru99.com/business-intelligence-definition-example.html.
2. Oracle. *Fusion Middleware Uživatelská příručka k produktu Oracle Business Intelligence Enterprise Edition* [online]. 2017 [cit. 2019-04-27]. Dostupné z: https://docs.oracle.com/middleware/12213help/biee/cs/BIEUG/toc.htm.
3. Manta Tools, s.r.o. *Co je to Manta Flow?* [online]. 2018 [cit. 2019-04-27] [Interní zdroje firmy].
4. Techopedia. *Data Lineage* [online]. © 2019 [cit. 2019-04-23]. Dostupné z: https://www.techopedia.com/definition/28040/data-lineage.
5. *Manta Flow Pricing, Features, Reviews and Comparison of Alternatives* [online]. 2019 [cit. 2019-05-14]. Dostupné z: https://www.getapp.com/business-intelligence-analytics-software/a/manta-flow/.
6. The Apache Software Foundation. *Apache Maven Project* [online]. © 2002–2019 [cit. 2019-04-23]. Dostupné z: http://maven.apache.org/.
7. Techtarget. *SOAP (Simple Object Access Protocol)* [online]. 2019 [cit. 2019-05-04]. Dostupné z: https://searchmicroservices.techtarget.com/definition/SOAP-Simple-Object-Access-Protocol.
8. Oracle. Managing Objects in the Oracle BI Presentation Catalog. *Fusion Middleware User's Guide for Oracle Business Intelligence Enterprise Edition* [online]. 2017 [cit. 2019-05-04]. Dostupné z: https://docs.oracle.com/middleware/12213/biee/BIEUG/GUID-5E1F2DFE-794A-455F-AEA3-B4C1DD6234BB.htm#BIEUG2889.

9. Oracle. SAWSessionService Service. *Fusion Middleware Integrator's Guide for Oracle Business Intelligence Enterprise Edition* [online]. 2015 [cit. 2019-05-08]. Dostupné z: https://docs.oracle.com/middleware/12213/biee/BIEIT/methods.htm#BIEIT240.

10. Oracle. WebCatalogService Service. *Fusion Middleware Integrator's Guide for Oracle Business Intelligence Enterprise Edition* [online]. 2015 [cit. 2019-05-08]. Dostupné z: https://docs.oracle.com/middleware/12213/biee/BIEIT/methods.htm#BIEIT286.

11. Oracle. Introduction to Designing Reports. *Fusion Middleware Report Designer's Guide for Oracle Business Intelligence Publisher* [online]. 2017 [cit. 2019-05-09]. Dostupné z: https://docs.oracle.com/middleware/12213/bip/BIPRD/GUID-18EAD29F-EE79-4685-9FF2-01B50570C88A.htm#BIPRD2068.

12. Oracle. What Is a Data Model? *Fusion Middleware Data Modeling Guide for Oracle Business Intelligence Publisher* [online]. 2017 [cit. 2019-05-09]. Dostupné z: https://docs.oracle.com/middleware/12213/bip/BIPDM/GUID-F4CC0AF3-32D0-4F9C-AED2-5D08DE5015D0.htm#BIPDM124.

13. Oracle. About the Layout Types. *Fusion Middleware Report Designer's Guide for Oracle Business Intelligence Publisher* [online]. 2017 [cit. 2019-05-10]. Dostupné z: https://docs.oracle.com/middleware/12213/bip/BIPRD/GUID-18EAD29F-EE79-4685-9FF2-01B50570C88A.htm#BIPRD3222.

Příloha A

Seznam použitých zkratek

BI Business Intelligence

CSV Comma Separated Values

ČVUT České vysoké učení technické v Praze

EDI Electronic Data Interchange

EFT Electronic Funds Transfer

FIT Fakulta informačních technologií

HTML Hypertext Markup Language

IBM International Business Machines

IDE Integrated Development Environment

JEE Java Enterprise Edition

OBIEE Oracle Business Intelligence Enterprise Edition

PDF Portable Document Format

RTF Rich Text Format

SOA Service Oriented Architecture

SOAP Simple Object Access Protocol

SSRS SQL Server Reporting Services

SQL Structured Query Language

SVN Subversion

WSDL Web Services Description Language

XML Extensible Markup Language

XSL Extensible Stylesheet Language

Příloha B

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `BP_Buldyk_Yauheniy_2019.pdf` ..... text práce ve formátu PDF