VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2016 Josef Swaczyna

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student:	**Josef Swaczyna**
Studijní program:	B2647 Informační a komunikační technologie
Studijní obor:	2612R025 Informatika a výpočetní technika
Téma:	Absolvování individuální odborné praxe Individual Professional Practice in the Company
Jazyk vypracování:	čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: IDC CEMA, s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

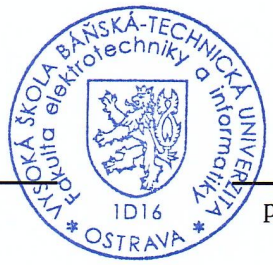
Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Mgr. Pavla Dráždilová, Ph.D.**

Konzultant bakalářské práce: Ing. David Vach

Datum zadání: 01.09.2015

Datum odevzdání: 29.04.2016



doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 16. dubna 2016 .....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 16. dubna 2016

IDC CEMA s.r.o.
Malé náměstí 13, 110 00 Praha
IČ: 26482347, DIČ: CZ26482347

Na tomto místě bych velice rád poděkoval všem, kteří mi byli s prací nápomocni, především mé poděkování patří společnosti International Data Corporation (IDC), kde mi bylo umožněno bakalářskou praxi vykonávat. Dále bych chtěl poděkovat vedoucí bakalářské práce paní Mgr. Pavle Dráždilové, Ph.D. za odborné vedení, za pomoc a rady při vypracování této práce.

Abstrakt

Tato bakalářská práce popisuje průběh odborné praxe ve firmě International Data Corporation (IDC), která je přední světový poskytovatel obchodních informací a poradenských služeb v oblasti informačních technologií a telekomunikací. V průběhu praxe byly řešeny problémy z oblasti webových aplikací se zaměřením na implementaci v programovacím jazyce Java. Bylo nutno seznámit se s technologiemi používanými ve firmě, mezi které patří například Apache Maven, PostgreSQL, AngularJS, Spring MVC a další. V rámci praxe jsme byli zapojeni do tří projektů, Edison, Data Warehouse a Kettle Server, které sloužily k analýze velkého množství dat, jejich sběru a transformacím. Tyto projekty byly určeny pouze pro vnitřní potřeby společnosti.

Klíčová slova: International Data Corporation, Java, Maven, Spring MVC, Hibernate, PostgreSQL, Oracle, AngularJS, JavaScript, HTML, CSS, odborná praxe

Abstract

This thesis describes the course of professional practice in company International Data Corporation (IDC), which is the world's leading provider of business information and consultancy services in the field of information technology and telecommunications. During practice were solved the problems of web applications with a focus on implementation in the Java programming language. It was necessary to become familiar with the technology used in the company, which include, for example, Apache Maven, PostgreSQL, AngularJS, Spring MVC and others. Within the practice, we have been involved in three projects, Edison, Data Warehouse and Kettle Server, which served to analyze large amounts of data, their collection and transformation. These projects were intended only for internal needs of the company.

Key Words: International Data Corporation, Java, Maven, Spring MVC, Hibernate, PostgreSQL, Oracle, AngularJS, JavaScript, HTML, CSS, professional practice

Obsah

Seznam použitých zkratek a symbolů

IDC	–	International Data Corporation
IDG	–	International Data Group
MVC	–	Model View Controller
SQL	–	Structured Query Language
JS	–	JavaScript
HTML	–	Hyper Text Markup Language
CSS	–	Cascading Style Sheet
SCM	–	Software Configuration Management
JavaEE	–	Java Enterprise Edition
Scrum	–	Iterativní a inkrementální metodologie agilního vývoje softwaru
REST	–	Representational State Transfer

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Obsahem této práce je přiblížit absolvování odborné praxe ve společnosti International Data Corporation (IDC). Odborné praxe mají studentům přiblížit fungování ve skutečných firmách, kde si student může vyzkoušet teoretické znalosti nabyté během studia a zároveň nabrat praktické zkušenosti z řešení reálných problémů, se kterými se v průběhu studia nesetkal.

IDC je mezinárodní společností, která se zabývá průzkumem trhu a poskytuje svým zákazníkům poradenství v oblasti informačních technologií a telekomunikací. Firma zaměstnává přes tisíc zaměstnanců a má pobočky ve více než padesáti zemích světa. Působil jsem zde v Ostravském vývojovém centru, kde jsem byl součástí týmu interních nástrojů, který se zabývá vývojem interních aplikací, které pokrývají veškeré potřeby firmy. Projekty, na kterých jsme pracovali byly zaměřeny na zpracovávání a analýzu dat, modelování statistik a zobrazování přehledů.

Bylo mi umožněno nastoupit zde na pozici aplikačního vývojáře se zaměřením na programovací jazyk Java. V průběhu působení ve firmě IDC jsem byl součástí šesti členného týmu vývojářů, kde mi bylo umožněno aplikovat nabyté teoretické znalosti jako například programování v jazyce Java, práce s databázemi, práce s operačním systémem Linux a další.

Součástí této práce je popis firmy, kde byla odborná praxe vykonávána, její harmonogram, popis používaných technologií a také popis vlastní práce ve společnosti IDC. V závěru je stručné zhodnocení průběhu odborné praxe ve firmě IDC a shrnutí nabytých zkušeností.

2 Popis firmy

IDC [1] je jedna z nejvýznamnějších analytických a poradenských společností. Poskytuje průzkumy a analýzu trhu, předpovídá vývoj a trendy v oblasti informačních a komunikačních technologií a pořádá odborné konference.

Zaměstnává přes 1000 analytiků a analyzuje data z více než 110 zemí světa. Díky rozsáhlé síti poboček v desítkách zemí se může chlubit skutečnou, nikoli pouze zprostředkovanou znalostí místního prostředí. I proto se již přes 50 let spoléhají manažeři úspěšných ICT firem, investoři i organizace veřejného sektoru na informace a doporučení IDC při rozhodování o investicích do IT (Information Technology) a tvorbě úspěšných obchodních strategií.

Ve svých studiích IDC zkoumá trhy IT, hodnotí postavení jednotlivých hráčů, posuzuje obecné ekonomické i společenské faktory a interpretuje obecné trendy. Na základě výsledků podrobných analýz poskytuje praktická doporučení pro investiční rozhodování, strategické plánování, obchodní strategie, volbu technologií a marketingové aktivity.

Společnost IDC je součástí mezinárodní skupiny IDG, která se zabývá publikační činností, výzkumem trhu a organizováním odborných konferencí v oblasti informačních technologií.

Vývojové týmy firmy IDC se již 6 let nachází na území České republiky. Hlavní vývojové týmy sídlí v Praze a Ostravě. Za Ostravskou pobočku je zde cca 30 lidí, kteří jsou rozděleni na 5 týmů. Z toho každý tým pracuje na rozdílných projektech.

3 Harmonogram bakalářské práce

1. - 2. týden Seznámení se s technologiemi

- Během prvních dvou týdnů působení ve společnosti IDC jsme se seznámili s technologiemi potřebnými pro práci na pozici aplikačního vývojáře. Vedení společnosti bylo velice otevřené a nápomocné, během tohoto času jsme absolvovali různé online kurzy a školení, ve kterých jsme se naučili pracovat s technologiemi, které se ve firmě standardně používají.

2. - 3. týden Seznámení se s projekty a business strukturou společnosti

- V průběhu tohoto období jsme absolvovali několik business školení, kde jsme se více seznámili s fungováním společnosti, jejím postavením na trhu, klientelou, business strukturou a samozřejmě také s jednotlivými projekty na kterých jsme později pracovali.

4. - 8. týden Práce na jednodušších úlohách

- V tomto období jsme se ve společnosti IDC začali aktivně podílet na vývoji softwaru. Nejprve nám byly přiděleny jednodušší úlohy, zabývající se spíše uživatelským rozhraním aplikace Edison, která byla v té době ve fázi raného vývoje. Během této fáze působení ve společnosti IDC jsme byli zapojeni do týmového vývoje, kde jsme se mohli blíže seznámit se všemi kroky agilního vývoje Scrum.

3. - 7. měsíc Práce na komplexních úlohách

- Větší část našeho působení ve společnosti IDC jsme řešili komplexní úlohy ve spolupráci s mnoha vývojáři na mnoha projektech. Během tohoto období jsme se zdokonalili v používání různých moderních technologií a postupů. Naučili jsme se také mnoho užitečných aspektů týmové spolupráce a týmového verzovaného vývoje.

8. měsíc Začlenění do údržby a uživatelské podpory na vybraných projektech

- Během tohoto období jsme byli začleněni do údržby a uživatelské podpory na vybraných projektech, které byly již nasazeny do ostrého provozu. V této fázi našeho působení ve společnosti IDC jsme se mimo jiné zdokonalili v komunikaci v anglickém jazyce, protože komunikace se zadavateli a uživateli systémů probíhala pouze v anglickém jazyce. Vyzkoušeli jsme si také práci pod výrazným časovým tlakem.

4 Popis použitých technologií

4.1 Java

Java [3] je objektově orientovaný programovací jazyk nezávislý na platformě. Vyvíjí ho společnost Oracle a je zdarma dostupný pro různé operační systémy. Jde o jeden z nejpoužívanějších programovacích jazyků na světě. Podle Tiobe indexu je Java druhý nejpopulárnější programovací jazyk.

Nezávislost na operačním systému a na hardwaru počítače zajišťuje způsob kompilace. Zdrojové kódy programu nejsou překládány do strojového kódu procesoru, ale pouze předzpracovávány do tzv. byte-kódu. Ten ještě není závislý na konkrétním procesoru, ale časově náročné fáze kompilace jsou již provedeny. Takto předzpracovaný kód je pro člověka nečitelný. Při spuštění Java programu je byte-kód velmi rychle převeden na strojový kód daného procesoru (s ohledem na použitý operační systém) – to provádí tzv. Java Virtual Machine (JVM).

4.2 JavaScript

JavaScript [4] je objektově orientovaný programovací jazyk, využívaný při tvorbě webových stránek. Na rozdíl od serverových programovacích jazyků sloužících ke generování kódu samotné stránky, JavaScript běží na straně klienta, tedy v prohlížeči až po stažení do jeho počítače.

JavaScript se používá především pro vytváření interaktivních webových stránek. Příkladem použití mohou být nejrůznější kontroly správného vyplnění formulářů, obrázky měnící se po přejetí myší, rozbalovací menu atd. JavaScript se také často používá k měření statistik návštěvnosti.

4.3 Apache Maven

Apache Maven [5] je rozšířený systém pro správu a sestavování aplikací postavených nad platformou Java. Jeho využitím odpadá závislost na konkrétním IDE, protože všechny informace potřebné ke kompilaci a sestavení programu jsou přímo obsaženy ve speciálních souborech pom.xml (POM = project object model).

Principem systému Maven je vytvoření objektového modelu nad zdrojovým kódem, se kterým lze provádět různé operace. Nejčastěji se jedná o kompilaci, kontrolu, vytvoření balíků, atd. Model projektu je definován v souborech pom.xml, které se nachází v kořenovém adresáři každého projektu. V těchto adresářích lze spouštět příkazy mvn s parametry, které načtou model z odpovídajícího souboru pom.xml a provedou zadanou akci. Klíčovou funkcí systému Maven je řešení závislostí. To znamená, že není nutné ručně kopírovat knihovny a umisťovat je na classpath. Zvláště u velkých projektů složených z mnoha podprojektů je to neocenitelné zpřehlednění a zjednodušení vývoje.

4.4 Spring Framework

Spring Framework [6] je populární open-source aplikační rámec neboli framework (označován také jako kontejner) pro vývoj JavaEE aplikací.

Jádro Springu je postaveno na využití návrhového vzoru Inversion of Control a je označován jako IoC kontejner. Tento návrhový vzor funguje na principu přesunutí zodpovědnosti za vytvoření a provázání objektů z aplikace na framework. Objekty lze získat prostřednictvím vsazování závislostí, což je speciální případ Inversion of Control. Dependency Injection řeší vlastní způsob vložení objektů. Základní tři způsoby vložení objektů jsou Setter Injection, Constructor Injection a Interface Injection. Objekty vytvořené kontejnerem jsou nazývány Beans. Objekty jsou frameworkem vytvořeny typicky na základě načtení konfiguračního souboru ve formátu XML, který obsahuje definice těchto Beans.

Spring Framework se nezabývá řešením již vyřešených problémů. Místo toho využívá prověřených a dobře fungujících existujících open-source nástrojů, které v sobě integruje. Tím se stává jejich použití často jednodušším.

Spring je modulární framework a umožňuje využít třeba jen část, která se zrovna hodí k řešení daného problému. Jeho účelem je zjednodušení návrhu JavaEE aplikací se zaměřením na architekturu aplikace místo na technologii, jednoduchou testovatelnost, neinvazivnost a modulárnost.

4.5 Hibernate

Nástroj Hibernate [7] se používá k zajištění persistence dat, což je vlastnost umožňující uchovávat data nebo informace i když program skončí. Persistence dat má několik vlastností. Jednou z nich je současný přístup více uživatelů k datům. Jenom proto, že si jeden uživatel prohlíží data ve webovém prohlížeči, nemusí ostatní uživatelé ztratit možnost si tato data také prohlížet a případně je modifikovat. Další vlastností persistence je transakční přístup. Právě transakce umožňují více uživatelům pracovat se stejnými daty nebo programy bez starostí souvisejících s porušením dat. Transakční zpracování se stará o konzistenci dat, která by mohla být narušena vícenásobným přístupem k jedněm datům. Třetí vlastností persistence je přenositelnost a snadnost použití. Ve většině případů, když se zahájí implementace programu nebo aplikace, často se přenositelnost omezuje pouze na určitý typ specifického prostředí

4.6 PostgreSQL

PostgreSQL [8] je robustní databázový server s mnoha pokročilými funkcemi. Často bývá srovnáván s databázovým systémem MySQL, podobně jako on je PostgreSQL dostupný pro všechny hlavní platformy a je také šířen jako open-source.

Na rozdíl od MySQL je PostgreSQL někdy trochu pomalejší, vynahrazuje to však svou dlouhodobou bezpečností a spolehlivostí, s níž mohou soupeřit pouze poslední verze MySQL.

K výhodám PostgreSQL patří také vynikající rozšiřitelnost či možnost spouštět procedury psané v jazycích Perl, Python a dalších.

4.7 AngularJS

AngularJS [9] je javascriptový webový framework, který se zaměřuje na tvorbu single-page aplikací. Tyto aplikace jsou tvořeny pomocí HTML kódu, do kterého jsou vloženy speciální formátovací značky, které určují, jaké operace či data mají být na dané místo vloženy. Tento způsob se označuje jako data-binding.

AngularJS je volně dostupný na webu[1] pod MIT licencí. V současné době je největším přispěvatelem společnost Google, která tento framework v roce 2009 založila. Aktuálně se chystá nová verze pod označením AngularJS 2, která by měla přinést mnohé změny a taktéž má být nekompatibilní s předchozími verzemi.

4.8 Apache Lucene Core

Apache Lucene Core [10] je vysoce výkonná, plně vybavená knihovna pro full-textové vyhledávání napsaná kompletně v Javě. Je to technologie vhodná pro téměř všechny aplikace, které vyžadují fulltextové vyhledávání, zejména pak cross-platformí.

Mezi její hlavní přednosti patří rychlost indexace, která dosahuje až 150GB za hodinu, nízké nároky na operační pamět a malá velikost výsledných indexů.

4.9 HTML5

Hypertext Markup Language (HTML) [11] je jednoduchý značkovací jazyk používaný k vytváření hypertextových dokumentů, které jsou platformně nezávislé. HTML Dokumenty jsou SGML dokumenty s generickou sémantikou, které jsou vhodné pro reprezentaci informace z široké škály domén.

HTML byl používán World Wide Web (WWW), globální informační iniciativou již od roku 1990. Aktuální verze HTML je verze 5, která přináší mnoho pozitivních změn.

HTML verze 5 se od verze 4 liší novými, zkrácenými a rychlejšími zápisy značek. Autoři dávají důraz na jednoduchost a zároveň účinnost. HTML5 je též možné vytvořit aplikaci, která funguje v prohlížeči i tehdy, když uživatel nemá internetové připojení, a která ukládá data do lokálního úložiště na uživatelově počítači. Je-li internetové připojení k dispozici, může aplikace synchronizovat data se vzdáleným serverem.

4.10 CSS3

CSS [12] vzniklo někdy kolem roku 1997. Je to kolekce metod pro grafickou úpravu webových stránek. Zkratka CSS znamená Cascading Style Sheets, česky "kaskádové styly". Kaskádové, protože se na sebe mohou vrstvit definice stylu CSS3 (Cascading Style Sheets 3) je už třetí verzí

kaskádových stylů CSS, a to již od roku 2005, kdy byl vývoj této technologie zahájen konsorciem W3C.

4.11 GIT

Git [13] je decentralizovaný systém pro správu verzí (SCM), za kterým stojí Linus Torvalds. Používá se například při vývoji Linuxového jádra a operačního systému Google Android. Samotný název "git"je britský slang, označující hloupou nebo nepříjemnou osobu. Torvalds často vtipkuje, že i tento software pojmenoval podle sebe (podobně jako Linux).

5 Popis vlastní práce

V této části práce následuje popis vlastní práce, která byla prováděna v rámci bakalářské praxe ve společnosti IDC

5.1 Projekt DataWarehouse

Data Warehouse je projekt, který slouží jako datový sklad. Jsou do něj slučovány data ze všech našich aplikací, které se později využívají v jiných aplikacích. Zdroje mohou mít různá fyzická umístění a mohou být v různých datových formátech.

Celý projekt je napsán v programovacím jazyku Java a obsahuje pouze minimalistickou prezentační vrstvu, která slouží pouze pro administrátory. Srdcem celé aplikace je databáze, která obsahuje obrovské množství dat. Jedná se o zvláštní typ relační databáze, která umožňuje řešit úlohy zaměřené převážně na analytické dotazování nad rozsáhlými soubory dat. Zvláštním typem relační databáze rozumíme databázi, která obvykle používá definici datového skladu od Billa Inmona [2]. Pro integraci dat je používán nástroj Kettle od společnosti Pentaho, který pomocí takzvaných transformací sbírá, upravuje a ukládá data z různých zdrojů do datového skladu.

5.1.1 Google Analytics data mining

Našim úkolem bylo transformovat data z Google Analytics do našeho datového skladu.

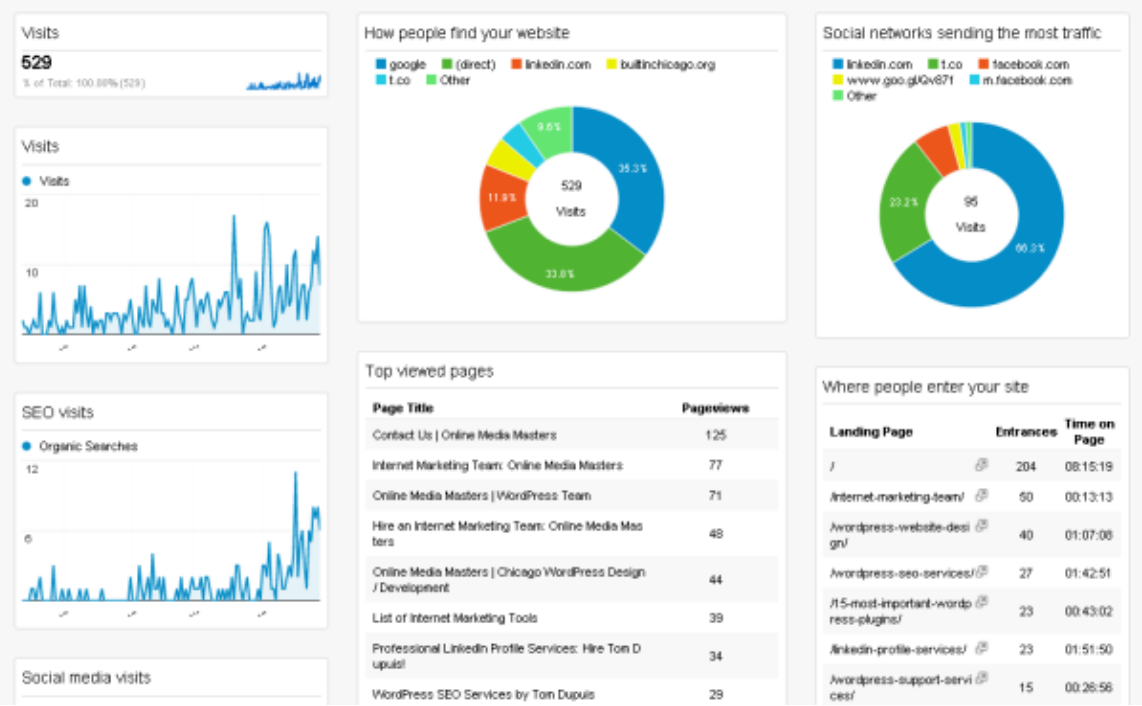
Google Anayltics je služba, kterou poskytuje společnost Google zcela zdarma a slouží pro zaznamenávání komplexních anylytických statistik návštěvnosti webových stránek. Pro zaznamenávání stačí vložit do hlavičky stránky malý JavaScriptový sledovací kód, který odesílá podrobné informace o návštěvníkovi webu přímo na servery Googlu.

K těmto statistikám lze přistupovat dvojím způsobem, buďto přes webové rozhraní aplikace (viz obrázek 1), nebo přes REST rozhraní.

Řešení úkolu

Většina datových transformací na projektu DataWarehouse probíhala pomocí řešení Kettle od společnosti Pentaho. Pentaho Kettle je nástroj pro integraci dat z mnoha zdrojů. Jedna z hlavních výhod využívání tohoto nástroje je možnost data různě transformovat, spojovat a přetvářet. Jednotlivé transformační kroky jsou rozděleny do dílčích kroků, dále jen transformací, které mohou být shlukovány do skupin, které jsou nazývány transformační práce. Jako další výhodu můžeme označit také fakt, že všechny kroky v rámci jedné transformace probíhají paralelně (viz obrázek 2).

Nejprve bylo zapotřebí ověřit, zda již v databázi nemáme nekompletní data z předchozího dne, protože stahování dat z Google Analytics bylo možno provádět pouze po celých dnech a docházelo by tak k duplicitám. Pro rozpoznání nekompletních dat byl použit atribut Date-



Obrázek 1: Webová aplikace Google Analytics

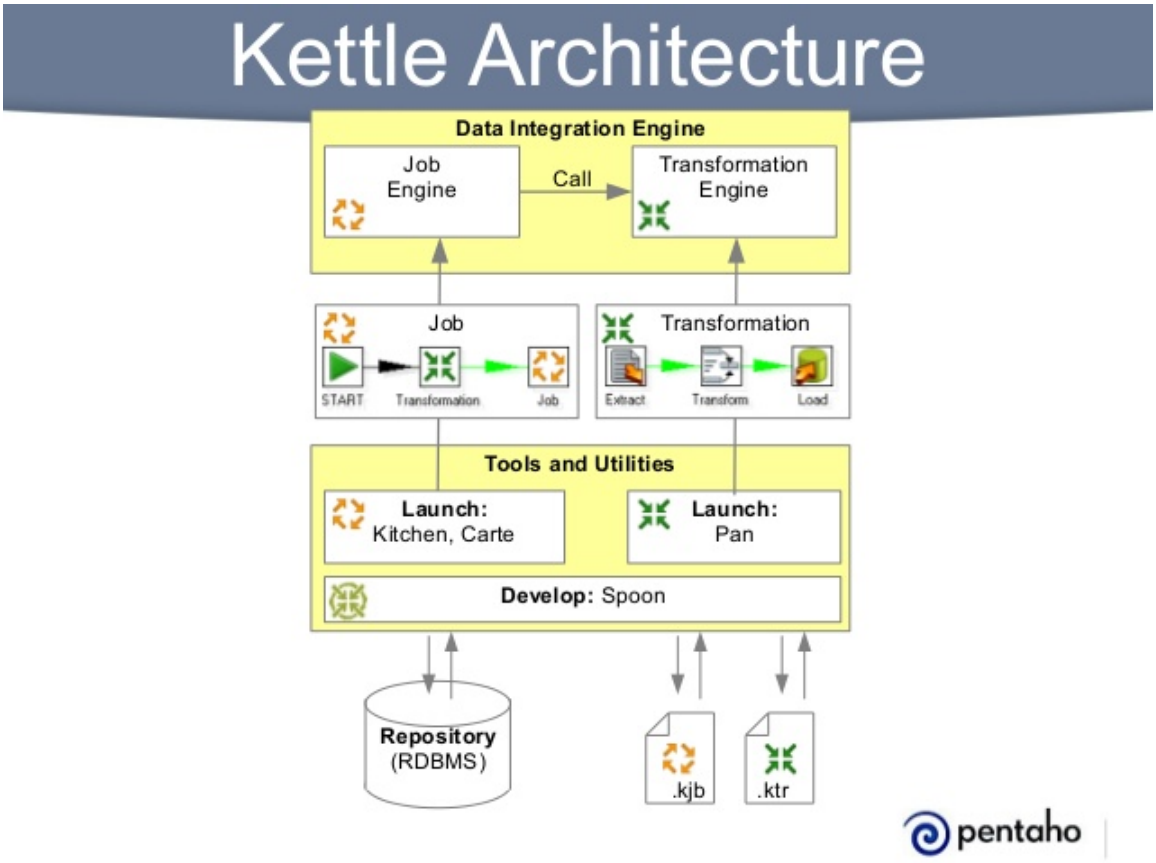
Hour, který nám specifikoval přesnou hodinu získání záznamu. Na základě tohoto atributu jsme z databáze smazali záznamy za předchozí den.

V dalším kroku jsme pomocí integrovaného doplňku Google Analytics v nástroji Pentaho Kettle nastavili stahování dat na základě atributu DateHour z předchozího kroku a tyto data jsme uložili do databáze.

Posledním krokem bylo nahrání transformační práce na Kettle Server a následné zařazení práce do plánované fronty, která se spouštěla každý den ve 2:00.

Problémy při řešení úkolu

Během řešení tohoto úkolu jsme narazili na několik problémů. Jedním z nich byla agregace dat na straně serveru Google Analytics, která nám znemožňovala stahovat data ve své nejčistší formě. Dále jsme se také potýkali s problémy spojenými s vlastními dimenzemi, které mohou obsahovat například klíčová slova stránky, ale musí mít vždy přiřazenou alespoň implicitní hodnotu, protože v důsledku agregace dat Google Analytics odstraňuje celé řádky záznamů, pokud je obsažena alespoň jedna prázdná hodnota. Mezi další problémy můžeme zařadit také omezený počet dotazů na servery Google Analytics a jejich značnou nestabilitu, která se projevovala častými výpadky služby.



Obrázek 2: Architektura aplikace Pentaho Kettle

5.2 Projekt Kettle Server

Kettle Server je projekt, zaměřený na integraci dat z různých zdrojů do datového skladu. Jak již jeho název napovídá, projekt je založen na aplikaci Kettle společnosti Pentaho, která je naimplementována do serverové podoby.

Projekt je napsán v programovacím jazyce Java, s velice minimalistickým uživatelským rozhraním, které slouží pouze pro nezbytnou administraci a přidávání nových transformačních kroků. Aplikace shromažďuje datové transformace, vytvořené pomocí nástroje Pentaho Kettle, z mnoha zdrojů pomocí napojení na distribuované systémy správy verzí GIT a SVN. Jednotlivé transformace jsou shlukovány administrátory do logických skupin, které jsou následně spouštěny automaticky v periodických intervalech.

5.2.1 Migrace z SVN na GIT

Z důvodu postupného přechodu všech projektů na distribuovaný systém správy verzí GIT bylo zapotřebí přizpůsobit také strukturu projektu Kettle Server, který doposud uměl pracovat pouze s SVN repositáři. Projekt nepoužívá pouze jedno centralizované úložiště pro transformace, ale shlukuje transformace z mnoha jiných repositářů pomocí správce repositářů, kde je umožněno

administrátorům přidávat jednotlivé repositáře obsahující požadované transformace, které jsou poté spouštěny periodicky.

Řešení úkolu

Nejprve bylo zapotřebí oddělit veškerý stávající SVN závislý kód do separátních tříd, které obsluhovaly pouze SVN repositáře, protože tyto třídy bylo nutno zachovat v rámci zpětné kompatibility.

Poté bylo nutné vytvořit sadu abstraktních tříd, které se staraly o obecný přístup k jednotlivým metodám, které byly implementovány v rámci konkrétních distribuovaných systémů správy verzí.

Závěrem byla vytvořena zcela nová sada tříd obsluhujících pouze GIT repositáře, která byla napojena na abstraktní třídy vytvořené v předchozím kroku.

Problémy při řešení úkolu

V průběhu řešení tohoto úkolu jsme narazili na problém rozdílného pojetí verzování mezi GIT a SVN. Distribuovaný systém správy verzí SVN se při verzování zaměřuje na jednotlivé soubory a jejich fyzickou strukturu, kdežto systém GIT je zaměřen na obsahovou část a souborová struktura jej prakticky nezajímá.

Tento problém se projevoval při stahování dat z repositářů. U verzovacího systému SVN byl projekt nastaven tak, aby stahoval pouze jednu složku obsahující požadovaná data, která měla často pouze zlomkovou velikost celého repositáře. Verzovací systém GIT tuto funkcionalitu nenabízel, proto bylo zapotřebí nalézt vhodné řešení (viz výpis kódu 1).

```
1  private void checkoutOrUpdateKettleFilesFromGIT(final VersionControlLocation
       versionControlLocation) throws GitAPIException {
2      vcPropertiesCache.remove(versionControlLocation.getId());
3      File localWorkingCopy = new File(KettleUtil.KETTLE_LOCAL_GIT_FOLDER +
           versionControlLocation.getName());
4      Git gitRepo = null;
5      if (!localWorkingCopy.exists()) {
6          localWorkingCopy.mkdirs();
7          try {
8              gitRepo = Git.cloneRepository().setURI(versionControlLocation.getUrl().contains("
                   http://") ? versionControlLocation.getUrl() : "http://" +
                   versionControlLocation.getUrl()).setDirectory(localWorkingCopy)
9                  .setCredentialsProvider(new UsernamePasswordCredentialsProvider(
                       versionControlLocation.getUser(), VCSPasswordUtil.decryptPassword(
                       versionControlLocation.getPassword())))
10                 .setNoCheckout(true).call();
11             gitRepo.checkout().setStartPoint(versionControlLocation.getGitBranchName()).
                   addPath(versionControlLocation.getGitKettleFolderPath()).call();
12             gitRepo.getRepository().close();
13         } catch (GitAPIException e) {
14             logger.error(e.getMessage(), e);
```

```
15              // we need to delete the working copy directory manually
16              FileUtils.deleteQuietly(localWorkingCopy);
17              throw new KettleRuntimeException("Could not checkout Git repository: " +
                    versionControlLocation.getName(), e);
18          } catch (Exception e) {
19              e.printStackTrace();
20          }
21      } else {
22          try {
23              gitRepo = Git.open(localWorkingCopy);
24              gitRepo.fetch().setCredentialsProvider(new UsernamePasswordCredentialsProvider(
                    versionControlLocation.getUser(), VCSPasswordUtil.decryptPassword(
                    versionControlLocation.getPassword()))).call();
25              gitRepo.checkout().setStartPoint(versionControlLocation.getGitBranchName()).
                    addPath(versionControlLocation.getGitKettleFolderPath()).call();
26              gitRepo.getRepository().close();
27          } catch (IOException e) {
28              logger.error(e.getMessage(), e);
29              throw new KettleRuntimeException("Could not update local Git repository: " +
                    versionControlLocation.getName(), e);
30          } catch (Exception e) {
31              e.printStackTrace();
32          }
33      }
34  }
```

Výpis 1: Specifické nastavení stahování repositáře GIT

5.3 Projekt Edison

Edison je projekt, který je především zaměřen na analýzu a zobrazení dat, které získává z datového skladu. V rámci tohoto projektu je umožněno našim zaměstnancům procházet nejrůznější statistiky, grafy a tabulky. Aplikace je rozdělena do několika vzájemně propojených celků. Mezi ně patří například sekce s přehledem našich klientů, kde mohou naši zaměstnanci získat přehledný a velice detailní pohled na všechny naše klienty, jejich interakce, kontrakty a další detaily. Další částí aplikace je sekce analýzy výzkumu, která analyzuje průběh našich vlastích výzkumů a nabízí tak uživateli cenné statistické grafy, popisující jejich průběh a výsledky. V neposlední řadě aplikace obsahuje sekci zabývající se detailním rozkladem kontraktů našich klientů, kde je uživateli představena detailní analýza průběhu jednotlivých kontraktů nad různými časovými rámci a kritérii.

Edison je založen na technologii Java, kde je hlavní komponentou Spring Framework. Pro efektivnější práci s velkými daty je zde implementována technologie Apache Lucene. Pomocí Apache Lucene se potřebná data z datového skladu naindexují přímo na lokální disk a vytvoří

se nad nimi indexy, které napomáhají k velice rychlému přístupu. Zobrazovací část projektu je postavena na AngularJS společně s jQuery, JavaScriptem, HTML5 a CSS3.

5.3.1 Integrace Ag-Grid rozšíření s vlastním tříděním a filtrováním

V celé aplikaci Edison bylo používáno značné množství klasických HTML tabulek, které zde sloužily pro zobrazování množství analytických dat. Problémem klasických HTML tabulek byla složitá manipulace s dynamickými daty, pomalé načítání nebo také omezená možnost řazení a filtrování dat.

Řešení úkolu

Bylo nalezeno JavaScriptové rozšíření Ag-Grid, které poskytovalo dostatečně robustní rozhraní, díky kterému bylo možné jednoduše manipulovat i s dynamickými daty. Tabulky vytvořené za pomoci tohoto rozšíření jsou schopny pracovat velice rychle i s dynamickými daty v řádech desetitisíců řádků. Načítání takovéto tabulky mnohdy netrvá ani jednu vteřinu.

Pro univerzální použití rozšíření napříč celou aplikací bylo zapotřebí vytvořit obecné JavaScriptové funkce v AngularJS pro transformaci dat z JSON objektů, které byly zasílány klientovi pomocí REST rozhraní, na Ag-Grid řádky. Dále bylo nutné vytvořit obecné funkce pro dynamické vytváření sloupců tabulek (viz výpis kódu 2) tak, aby odpovídaly požadavkům ze strany serveru a také musely být vytvořeny obecné funkce pro dynamické třídění a filtrování dat. Nesmíme také zapomenout na nutnost transformace samotných dat do tvaru, který odpovídal požadavkům Ag-Grid rozšíření, pro tyto účely byly opět napsány obecné funkce, použitelné napříč celou aplikací.

```
1  function createColumnsDefForAgGrid(dataTableColumns) {
2      var columnDefs = [];
3      for (var i = 0; i < dataTableColumns.length; i++) {
4
5              var span = "<span data-toggle='tooltip' data-placement='top' title='" +
                   dataTableColumns[i].name + "' >" + dataTableColumns[i].name + "</span>";
6              var column = {headerName: span, field: dataTableColumns[i].id, comparator:
                   generalLowercaseComparator};
7
8          columnDefs.push(column);
9      }
10     return columnDefs;
11 }
```

Výpis 2: Obecná funkce pro vytváření definic sloupců

6 Zhodnocení

Pro bakalářskou praxi jsem se rozhodl z důvodu pozdějšího uplatnění v oboru. Velice se mi líbila možnost vyzkoušet si pracovní proces ve skutečné firmě, která se zabývá aplikačním vývojem analytických nástrojů. Ve společnosti IDC jsme si vyzkoušeli práci v mezinárodním kolektivu výborných vývojářů, se všemi jejími aspekty, mezi něž patří například cizojazyčná komunikace, práce pod zkušeným vedoucím týmu nebo také silné zázemí prosperující firmy. Rád bych vyzdvihl použitelnost nabytých teoretických ale i praktických znalostí během studia, mezi které patří například znalosti z předmětů zabývajících se programovacími jazyky, databázemi ale také operačními systémy. Velice důležitá byla také znalost anglického jazyka, protože samotná firma pochází ze Spojených států amerických, a tudíž byla vyžadována anglická komunikace.

Některé znalosti a dovednosti jsme si museli v průběhu praxe ještě doplnit, šlo především o znalosti konkrétních technologií, například verzovacích systémů GIT a SVN, dále pak distribučního systému Apache Maven nebo znalosti z oblasti aplikačních serverů a vývoje v týmech.

Myslím si, že je velice důležité, aby se ještě více studentů zapojovalo do odborných praxí, protože zkušenosti, které mi praxe poskytla jsou nenahraditelné.

7 Závěr

Absolvovaná praxe ve firmě IDC byla velkým obohacením mého kariérního života. V prvních týdnech našeho působení ve společnosti IDC nás firma řádně zaškolila a seznámila se všemi technologiemi, které byly používány v průběhu celé odborné praxe. Mezi ně patří například systém pro správu a sestavování aplikací Apache Maven, knihovna pro full-textové vyhledávání Apache Lucene Core, framework pro vývoj JavaEE aplikací Spring nebo JavaScriptový webový framework AngularJS. V průběhu prvního měsíce jsme byli taktéž seznámeni s týmovým vývojem a distribuovanými systémy správy verzí GIT a SVN. V neposlední řadě nám firma poskytla možnost zdokonalit se v programovacích jazycích Java a JavaScript.

V průběhu působení ve firmě IDC jsme se účastnili týmového vývoje na projektech Edison, Data Warehouse a Kettle Server, kde jsme se věnovali především úpravám serverové části aplikací. Mezi tyto úpravy patřila například transformace dat pomocí nástroje Pentaho Kettle, úprava struktury databázových tabulek, úpravy výpočtů nad analytickými daty nebo třeba také integrace nového systému správy verzí.

Během absolvování odborné praxe bylo zapotřebí doplnění chybějících dovedností a zkušeností, ale díky vstřícnému přístupu ze strany vedení firmy a všech kolegů, byla celá odborná praxe velice příjemnou zkušeností, kterou zakončila nabídka další spolupráce na plný pracovní úvazek.

Literatura

[1] About IDC [online]. [cit. 2015-12-12].
Dostupné z: http://www.idc.com/about/

[2] Data Warehouse definition by Bill Inmon [online]. [cit. 2016-04-01].
Dostupné z: https://www.1keydata.com/datawarehousing/

[3] Programovací jazyk Java [online]. [cit. 2016-04-01].
Dostupné z: https://www.interval.cz/clanky/naucte-se-javu-uvod/

[4] Programovací jazyk JavaScript [online]. [cit. 2016-04-01].
Dostupné z: http://www.adaptic.cz/znalosti/slovnicek/javascript/

[5] Apache Maven [online]. [cit. 2016-04-06].
Dostupné z: http://voho.cz/wiki/maven/

[6] Spring Framework [online]. [cit. 2016-04-06].
Dostupné z: http://www.multimediaexpo.cz/mme121/index.php/Spring_Framework

[7] STRAKOŠ, M. Architektura intranetového řešení aplikovaná na evidenci projektů. VŠB-TU Ostrava, 2005

[8] PostgreSQL [online]. [cit. 2016-04-06].
Dostupné z: http://www.adaptic.cz/znalosti/slovnicek/postgresql/

[9] AngularJS [online]. [cit. 2015-12-12].
Dostupné z: https://cs.wikipedia.org/wiki/AngularJS

[10] Apache Lucene Core [online]. [cit. 2015-12-12].
Dostupné z: https://lucene.apache.org/core/

[11] Hypertext Markup Language [online]. [cit. 2016-04-20].
Dostupné z: https://tools.ietf.org/html/rfc1866

[12] CSS Tutorial [online]. [cit. 2016-04-20].
Dostupné z: http://www.w3schools.com/css/default.asp

[13] GIT [online]. [cit. 2016-04-20].
Dostupné z: http://voho.cz/wiki/git/