FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Dokumentace datových toků v datovém skladu ČVUT
Student:	Bc. David Hajčiar
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 2018/19

Pokyny pro vypracování

Cílem práce je analyzovat, navrhnout a ve formě funkčního prototypu implementovat dokumentaci datových toků při provozu datového skladu ČVUT.

1. Analyzujte a popište datové toky při provozu datového skladu ČVUT.
2. Navrhněte vhodné způsoby jejich dokumentace, diskutujte a zvolte vhodná metadata, která dokumentaci doplní.
3. Navrhněte způsob prezentace těchto datových toků a příslušných metadat v portálu EBIE.
4. Po dohodě s vedoucím práce zvolte část návrhu z předchozího bodu, kterou implementujete formou funkčního prototypu.
5. Zvolenou část implementujte, zdokumentujte a řádně 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 18. prosince 2017

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Dokumentace datových toků v datovém skladu ČVUT

Bc. David Hajčiar

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

6. května 2018

Poděkování

Chtěl bych poděkovat svému vedoucímu Ing. Michalu Valentovi, Ph.D. za vedení mé práce, cenné rady a konzultace. Také bych chtěl poděkovat vývojářům datového skladu za spolupráci a pomoc. Na závěr děkuji své snoubence a celé rodině za podporu a trpělivost v průběhu celého studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) 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., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 6. května 2018 .....

Odkaz na tuto práci

Hajčiar, David. *Dokumentace datových toků v datovém skladu ČVUT*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Tato diplomová práce se zabývá analýzou datových toků v datovém skladu ČVUT a návrhem vhodných způsobů jejich dokumentace za využití souvisejícího portálu EBIE (Extended Business Intelligence Encyclopedia). Pro navržené oblasti dokumentace jsou v jazyce Ruby implementovány a otestovány podpůrné generátory vytvářející nový obsah a aktualizující obsah stávající.

Klíčová slova Datový sklad, datové toky v datovém skladu, dokumentace, portál EBIE, generátory, Ruby

Abstract

This master thesis is dealing with analysis of data flows in data warehouse of ČVUT and with the design of suitable documentation, using portal EBIE (Extended Business Intelligence Encyclopedia). For the proposed areas of documentation are implemented a tested supporting generators, written in Ruby, which generate new content and update existing ones.

Keywords Data warehouse, data flows in data warehouse, documentation, portal EBIE, generators, Ruby

Obsah

Seznam obrázků

Úvod

V dnešní době velké organizace disponují více informačními systémy, v nichž vzniká mnoho různorodých dat. Za účelem jednotného zpracování a přístupu k těmto datům jsou vytvářeny datové sklady, které informace z přítomných systémů získávají, transformují do požadovaných formátů a prezentují koncovým uživatelům.

Získané informace jsou prezentovány uživatelům datových skladů a slouží např. pro podporu manažerských rozhodnutí.

Nezbytnou součástí prezentace výstupů datového skladu musí být jasná a přehledná dokumentace údajů, které se uživateli zobrazují. Pokud takový popis není dostupný, může docházet ke špatnému (nebo nepřesnému) výkladu přítomných dat, která tím ztrácejí na své hodnotě.

Cíl práce

Cílem mé práce je těmto problémům předejít a zabránit.

V první kapitole analyzuji aktuální stav datového skladu, přítomných datových toků a existující dokumentace.

V druhé kapitole na základě této analýzy navrhuji vhodný způsob dokumentace pro datový sklad ČVUT a to jak z hlediska obsahu, tak podpůrných procesů.

Ve třetí kapitole ukazuji a vysvětluji způsob, kterým jsem navržené procesy implementoval.

V poslední části přibližuji vytvořené testy, kterými jsem ověřil funkčnost implementovaných řešení.

Kapitola 1

Analýza

V této kapitole se věnuji analýze datového skladu ČVUT, jeho datových toků a existující dokumentace, zejména pak v souvislosti s portálem EBIE.

Dále se věnuji aktuálním procesům pro tvorbu této dokumentace a jejich nedostatkům.

1.1 Projekt Datová čistota

V rámci IP projektu DU20 Datová čistota začal v roce 2015 vznikat datový sklad ČVUT [1].

Projekt ukázal, že stávající Manažerský informační systém (MIS), který je součástí Portálu ekonomických služeb (PES), má neuspokojivý stav dat (čistotu, úplnost a kvalitu). Zkoumán byl nejvíce modul Studium.

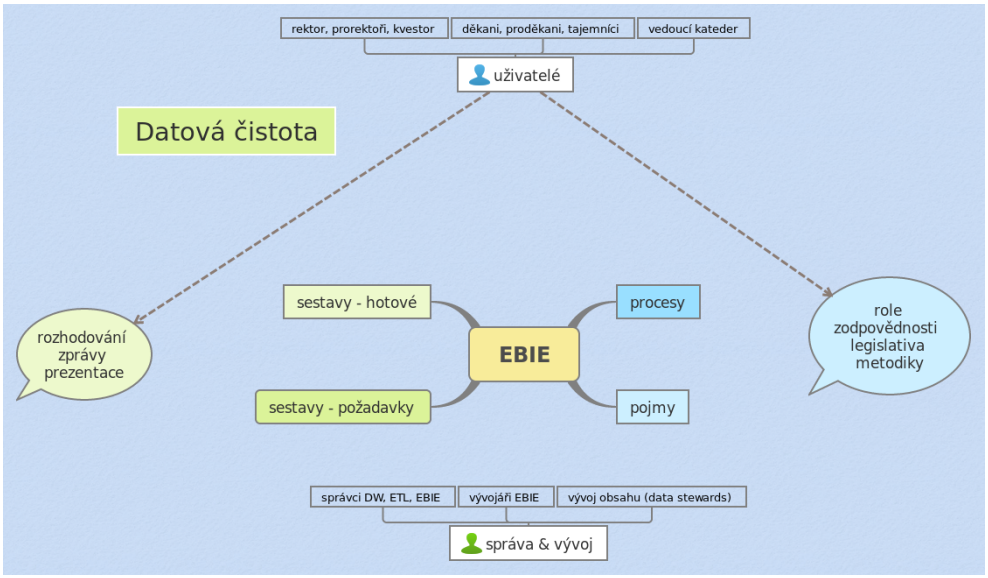
Hlavní příčiny byly tyto:

1. Chybějící metodici - pro oblast studia neexistuje role metodika dat („data steward“), který by garantoval kvalitu dat, spravoval datový model a byl autoritou ohledně schvalování změn pro zdrojové a manažerské systémy.

2. „Děravé“ systémy zdrojových dat - například databáze a systém KOS (KOmponenta Studium), u kterého nelze realizovat kompletní kontrolu vkládaných dat.

3. Neexistuje systém (proces) kontroly kvality dat - proces, který by zahrnoval mechanismus kontroly dat, legislativní podporu pro vynucení nápravy a kompetentní role, které by za něj zodpovídaly.

Jako řešení těchto problémů bylo navrženo vytvoření nového datového skladu ČVUT, který by realizoval jednotlivé moduly formou datových tržišť. Návrh architektury je vidět na obrázku 1.1.



Obrázek 1.1: Návrh řešení na základě projektu Datová čistota DU20

1.1.1 Datové sklady

Ve světě datových skladů jsou nejrozšířenější 2 koncepty, jejichž autory jsou:

- William H. Inmon, který je nazýván *„otcem datových skladů“*, jelikož jako první zavedl ve své knize *„Building the data warehouse“* pojem *„Data Warehouse“* [2].
- Rudolf Kimball, podle jehož řešení byl navržen Datový sklad ČVUT (viz diplomová práce Ing. Stanislava Kuznetsova [3])

Vzhledem k použití architektury Rudolfa Kimballa v datovém skladu ČVUT se dále zaměřím pouze na tento koncept.

1.1.2 Datový sklad podle Kimballa

Kimball definuje datový sklad takto: *„Data Warehouse is a copy of transaction data specifically structured for querying and reporting“* [4].

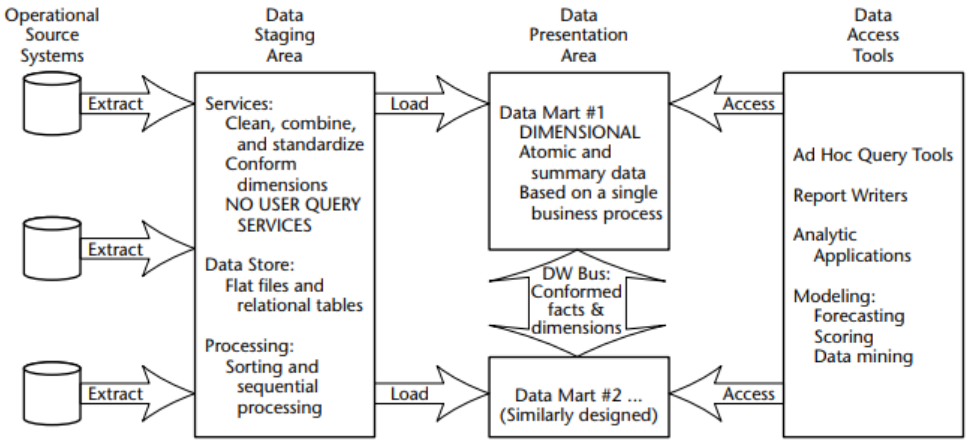
Vymezuje základní 4 komponenty (viz obrázek 1.2):

- Operation Source Systems - Zdrojové systémy bývají mimo datový sklad a obvykle nad obsahem a formátem jejich dat nemáme téměř žádnou kontrolu.
- Data Staging Area - Oblast, ve které probíhá transformace dat ETL (Extract Transformation Load) procesy ze zdrojových systémů do prezentační oblasti.

Kimball tuto část datového skladu připodobňuje ke kuchyni v restauraci, kde je syrové jídlo „transformováno“ na chutný pokrm, kuchyň je zároveň běžným zákazníkům (uživatelům) nepřístupná. [5].

- Data Presentation Area - zde jsou uskladněna organizovaná data, nad nimiž může uživatel zadávat dotazy. Typicky tuto oblast označujeme jako seskupení integrovaných datových tržišť (v nejjednodušším případě datové tržiště prezentuje jediný obchodní proces).

- Data Access Tools - nástroje poskytované uživatelům za účelem využití prezentační oblasti k analytickému rozhodování.Může se jednat např. o jednoduchý dotazovací nástroj nebo sofistikované „dolování dat“.



Obrázek 1.2: Architektura datového skladu podle Kimballa [5]

1.2 Datový sklad ČVUT

V následujících sekcích čerpám z dokumentace k datovému skladu ČVUT, kterou vytváří jeho vývojáři [6].

1.2.1 Vývoj datového skladu ČVUT

První verze datového skladu vznikla jako diplomová práce Ing. Stanislava Kuznetsova [7] v roce 2013, v té době se týkala pouze Fakulty informačních technologií. Sklad obsahoval studijní výsledky studentů spolu se studijními výsledky z fakultního systému Progtest.

Druhá verze, kterou byl prototyp datového skladu ČVUT, vznikla v rámci projektu Datové čistoty (viz 1.1) a integrovala data ze systému KOS (kvůli

omezenému přístupu pouze pro Fakultu informačních technologií) a z Ankety ČVUT. Cílem bylo především demonstrovat užitečnost datového skladu.

Třetí, aktuální verze, je stále evidovaná jako prototyp. Datový sklad nyní obsahuje:

- Plný přístup do systému KOS
- Datová tržiště ohledně studií, studijních výsledků, výsledků přijímacího řízení a vyhodnocení Ankety ČVUT
- Integraci Systému pro odevzdávání závěrečných prací (BPM ČVUT), studijní systémy FIT (EDUX, Progtest)
  - V přípravě je integrace pro UserMap (systém pro evidenci a správu uživatelů v rámci informačních systémů ČVUT), Portál pro spolupráci s průmyslem a V3S (aplikace evidující výsledky vědy a výzkumu na ČVUT)
- Reporty pro podporu rozhodování
- Automatizované nahrávání do datového skladu
- Implementaci vlastního způsobu historizace záznamů
- Vlastní jmennou konvenci

Ve své práci se budu věnovat právě této aktuální verzi datového skladu, kterou budu označovat, podle jeho verze, zkratkou DWH3 (Data WareHouse 3).

1.2.2 Architektura datového skladu ČVUT

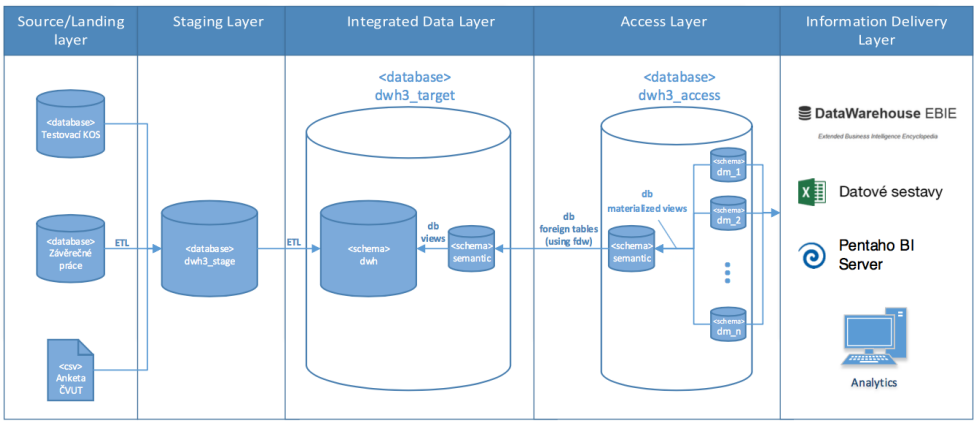
Pro vymezení datových toků při provozu datového skladu ČVUT je přínosné schéma aktuální architektury DWH3, uvedené na obrázku 1.3.

Jednotlivé části ze schématu 1.3 nyní popíši podrobněji.

Source Layer
Pro systémy KOS a Závěrečné práce jsou data ETL procesem nahrávána přímo ze zdrojové databáze do Staging vrstvy. Anketa ČVUT se nahrává pouze jednou za semestr datovými exporty ve formátu CSV (Comma Separated Values).

Staging Layer
Staging Layer je tvořena databází dwh3_stage na databázovém systému PostgreSQL. Zde je vytvořeno 5 různých schémat pro jednotlivé zdrojové systémy:

- PRE_STAGE – datový otisk tabulek ze zdrojového systému.



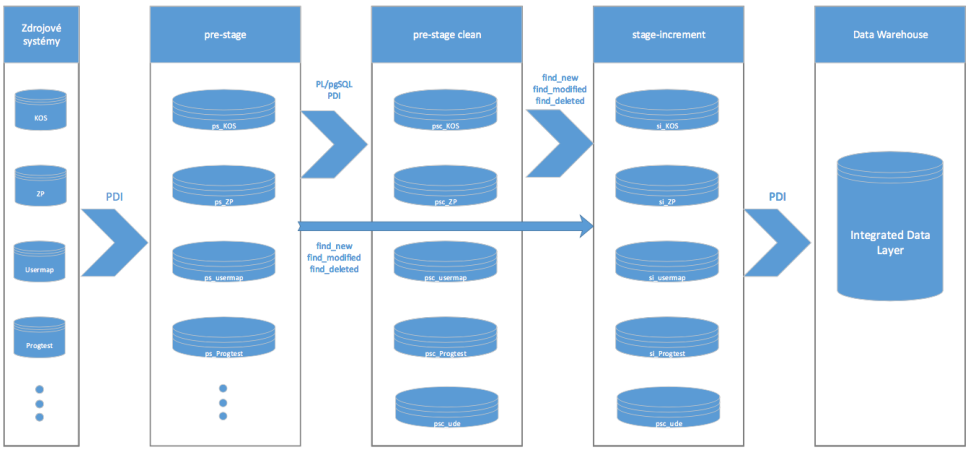
Obrázek 1.3: Architektura DWH3 [6]

- PRE_STAGE_CLEAN – technicky vyčištěné tabulky ze zdrojových systémů.
- PRE_STAGE_CLEAN_UDE – technicky vyčištěné tabulky pro uměle definované entity.
- STAGE_INCREMENT – poslední otisk zdrojového systému, nahraný do datového skladu (obsahuje tabulky z PRE_STAGE i PRE_STAGE_CLEAN).
- STAGE_INCREMENT_UDE – speciální typ STAGE_INCREMENT pouze pro tabulky z PRE_STAGE_CLEAN_UDE.

Proces nahrávání dat do Integrated Data Layer je přiblížen na obrázku 1.4.

Integrated Data Layer
V této vrstvě se nachází PostgreSQL databáze dwh3_target, která obsahuje 2 hlavní schémata:

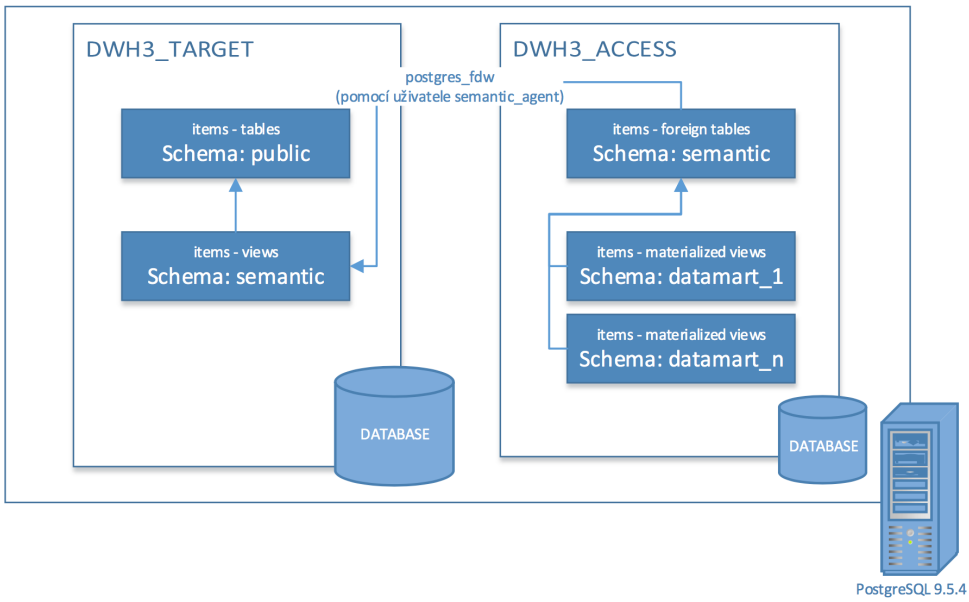
- dwh - reprezentuje centrální databázi, kde jsou uloženy datové struktury definované fyzickým datovým modelem. Data jsou do této databáze nahrávána pouze pomocí ETL procesů ze Staging Layer.
- semantic - sémantická databáze je tvořena databázovými pohledy nad centrální databází. Každý pohled definuje určitou obchodní entitu nad tabulkami z centrální databáze.



Obrázek 1.4: Architektura a proces nahrávání dat do Integrated Data Layer [6]

Access Layer

Přístupová vrstva je tvořena databází dwh3_access, která slouží k ukládání datových tržišť (datamartů) a přístupu většího množství uživatelů. Propojení mezi databázemi dwh3_access a dwh3_target (viz 1.5) je zajištěno pomocí technologie Postgres Foreign Data Wrapper.



Obrázek 1.5: Způsob propojení Target a Access vrstev [6]

Datová tržiště představují podmnožinu dat důležitých pro potřeby jed-

notlivých oddělení. Jsou tvořena materializovanými pohledy, což má pozitivní dopad na dobu odezvy při získávání dat z datových tržišť („materializované pohledy ukládají výsledky ve formě podobné tabulkám“ [8]).

Information Delivery Layer
Tato vrstva se zabývá poskytováním informací koncovým uživatelům, momentálně skrze datové exporty v Excelu, reporty v EBIE (viz 1.4) a Pentaho BI Server, který umožňuje analýzy nad datovými tržišti pomocí technologie OLAP.

1.3 Stávající dokumentace datového skladu ČVUT

K dokumentaci datového skladu jsou primárně využívány tyto stránky:

- Gitlab wiki vývojářů datového skladu
- Datová Logická Mapa
- Portál EBIE

V následujících sekcích tyto pojmy patřičně popíši.

1.3.1 Dokumentace vývojářů datového skladu

Vývojáři datového skladu udržují ve svém soukromém repozitáři tzv. „wiki“, která pro jejich potřeby dokumentuje datový sklad ČVUT. Obsah této stránky jsem již využil v sekci popisující datový sklad (viz 1.2).

Pro potřeby mé práce zkráceně uvedu jmennou konvenci, zavedenou na těchto stránkách:

Název každé tabulky je uveden v jednotném čísle a velkými tiskacími písmeny, kde jednotlivá slova jsou oddělena podtržítkem:

```
[prefixy][doména]\_NAZEV\_TABULKY[suffixy]
Př.: T\_OSOB\_OSOBA
```

Na výběr je z těchto předpon:

- T - klasická tabulka
- MV - materializovaný pohled
- V - pohled
- FT - cizí tabulka

A z těchto přípon:

- DIM - označení dimenzionální tabulky v případě dimenzionálního schématu
- FACT - označení faktové tabulky v případě dimenzionálního schématu

[6]

1.3.2 Datová Logická Mapa

Vývojáři datového skladu také udržují tzv. Datovou Logickou Mapu (DLM). Jedná se o dvě Google Docs tabulky (listy) - DLM schéma a DLM číselníky. DLM číselníky obsahují seznam a popis integrovaných číselníků, které v této práci nebudu využívat.

DLM schéma
Tato tabulka se sestává z následujících částí:

- Zdrojový systém
  - Databáze - název zdrojové databáze
  - Tabulka/soubor - název tabulky/souboru, odkud atribut pochází
  - Sloupec - název integrovaného atributu ve zdrojovém systému
  - Datový typ - datový typ atributu
  - DQ - datová kvalita
  - Typ SCD - typ historizace
- Cílový systém
  - Tabulka - název tabulky v cílovém systému
  - Sloupec - název sloupce/atributu v cílovém systému
  - Datový typ - datový typ atributu v cílovém systému
- Sémantická vrstva
  - Sémantika - název atributu v sémantické vrstvě
- ETL
  - Fáze implementace - pořadí, ve kterém jsou jednotlivé atributy integrovány
  - Název transformace
- Obchodní popis (metadata)
  - Obchodní název - název atributu (stručný popis významu)

– Obchodní definice - rozsáhlejší popis významu atributu

Schéma dále obsahuje sloupec s názvem popisované entity a Poznámku pro vývoj, která obsahuje důležité poznámky pro integraci daného atributu.

Ukázka DLM je na přiloženém obrázku 1.6.

	A	B	C	D	E
1		Zdrojový systém			
2		Databáze	Tabulka/soubor	Sloupec	Datový typ
3	OSOBA	kos	tek_osoby	peridno	number(38)
4		kos	tekusers	user_name	varchar2(20)
5		kos	tek_osoby	osoba_eid	number(38)
6		kos	tek_osoby	jmeno	varchar2(24)
7		kos	tek_osoby	prijmeni	varchar2(35)
8		kos	tek_osoby	titul	varchar2(35)
9		kos	tek_osoby	titulza	varchar2(35)
10		kos	tek_osoby	rodcis	varchar2(10)

Obrázek 1.6: Ukázka DLM, zobrazující část zdrojových systémů entity Osoba

1.4 Portál EBIE

Portál EBIE (Extended Business Intelligence Encyclopedia) vznikl společně s datovým skladem v rámci projektu Datová čistota DU20 za účelem poskytování kontextu řešení (popisy metadat, procesů, sestav a jejich souvislostí) a umožnění přístupu k jednotlivým sestavám (reportům).

Dnes zde můžeme nalézt reporty zobrazující aktuální data z připravených datamartů, popis souvisejících entit, dokumentaci k některým datamartům nebo datovým zdrojům aj.

Ve své bakalářské práci [9] jsem obsah portálu EBIE podrobně popsal, zde se proto nejvíce zaměřím na části související s datovým skladem. Další vývoj server je popsán v diplomové práci Ing. Ondřeje Batíka [10].

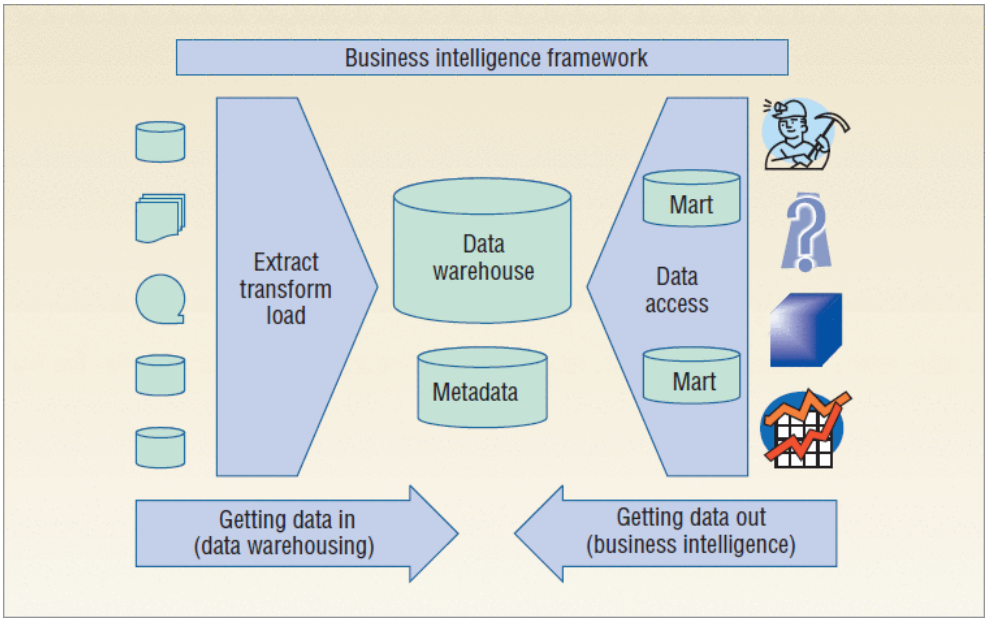
1.4.1 Extended Business Intelligence Encyclopedia

„Business Intelligence (BI) je sada procesů, aplikací a technologií, jejichž cílem je účinně a účelně podporovat rozhodovací procesy ve firmě. Podporují analytické a plánovací činnosti podniků a organizací a jsou postaveny na principu multidimenzionality.“ [11]

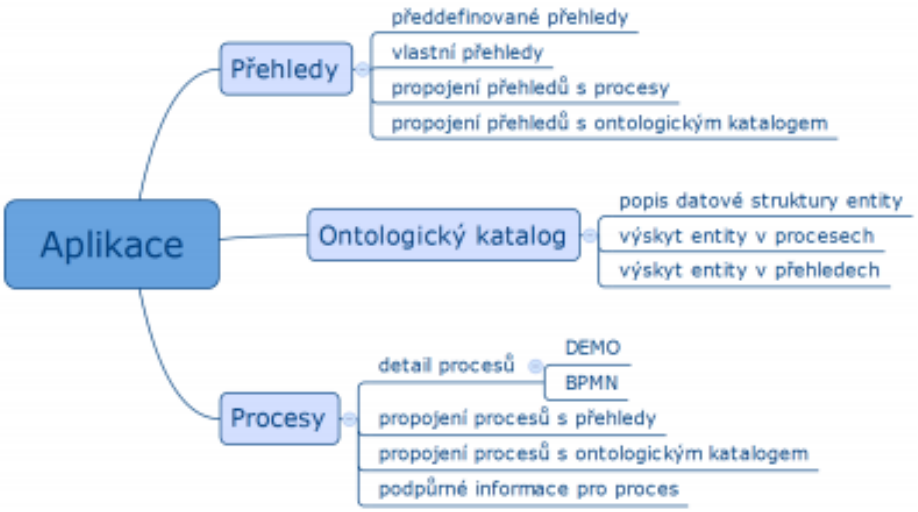
Vztah BI encyklopedií s datovými sklady je dobře patrný z obrázku 1.7.

Extended v EBIE značí rozšíření o konceptuální vrstvu OntoUML a DEMO. Začlenění těchto vrstev do EBIE navrhl Ing. Václav Jirovský ve své diplomové práci (viz [12]), navržená architektura je k vidění na obrázku 1.8.

OntoUML je rozšíření UML (Unified Modeling Language) o ontologickou vrstvu. Jeho základy položil Giancarlo Guizzardi ve své knize „Ontological Foundations for Structural Conceptual Models“ [14].



Obrázek 1.7: Proces BI nad datovým skladem. Obrázek převzat z článku [13]



Obrázek 1.8: Navržená informační architektura EBIE [12]

DEMO (Design & Engineering Methodology for Organizations) *„je analýza a modelování procesů v organizaci. Tato metodika stojí na teorii Production in Social Interaction (PSI nebo -theory), která popisuje, jak a proč lidé spolupracují v podnicích.“* [12].

K modelování procesů se v EBIE používá i notace BPMN (Business Process Model and Notation), která slouží k modelování podnikových procesů [15].

1.4.2 Portál EBIE

EBIE je webová aplikace vytvořená v Ruby on Rails. Momentálně jsou nasazené 2 servery:

- Produkční server - určený pro uživatele EBIE, jeho verze odpovídá master větvi ebie repozitářů
- Testovací server - označovaný jako ebie-vyvoj, vývojáři jej využívají k testování nových rozšíření nebo změn obsahu.

Ruby on Rails (RoR) je framework pro tvorbu webových aplikací v jazyce Ruby. Ruby je objektově orientovaný, dynamický programovací jazyk [16].

Ke správě vývoje se používá technologie Git na fakultním Gitlab serveru. Git je distribuovaný systém správy verzí softwaru, spravované projekty se nazývají repozitáře [17]. Gitlab je webový správce git repozitářů, který podporuje např. sledování chyb (issue tracker), wiki, průběžnou integraci aj. [18].

EBIE se v tuto chvíli skládá ze tří částí, rozdělených do samostatných repozitářů:

- **ebie-source** - obsahuje aplikační logiku portálu (tj. vlastní RoR aplikaci), která kompletně zajišťuje provoz EBIE
- **ebie-content** - tento repozitář je připojen k ebie-source jako git submodul (*„submodul je repozitář vnořený do jiného repozitáře“* [19]).

  Jsou zde uloženy zdrojové soubory všech dokumentů zobrazovaných na EBIE (ve formátu html nebo AsciiDoc)
- **ebie-data-api** (zkráceně ebie-api) je samostatně nasazená aplikace, která funguje nezávisle na EBIE a má zajištěný přístup k datamartům v DWH3.

  Její účel je poskytování dat z datového skladu pro reporty v EBIE.

Protože všechny uvedené části EBIE používají Ruby nebo RoR, považuji za nezbytné alespoň zkráceně uvést základní syntaxi tohoto jazyka a strukturu RoR aplikací.

1.4.3 Syntaxe jazyka Ruby

Ruby je dynamicky typovaný, čistě objektový jazyk. Jeho syntaxe je navržena s ohledem na příjemné a přirozené používání, přesto může být Ruby kód pro neznalého čtenáře obtížně pochopitelný (jako např. řádek „target_table.split('_')[2..-1]&.join('_')“).

V této sekci uvádím základní syntaxi a postupy pro práci s tímto jazykem. Výstup jednotlivých řádků zapisuji ve formátu „# => výstup“, znak „#“ má v Ruby funkci začátku komentáře.

Proměnné, řetězce, regulární výrazy
Inicializace lokální proměnné:

```
s = 'string'
```

Konstanty jsou zapisovány velkými písmeny:

```
CONS = 55
```

Instanční proměnné objektů začínají znakem @:

```
@array = 'text'
```

Ruby má garbage collector, proměnné tedy není nutné uvolňovat. Středník na konci řádku je nepovinný.

Řetězce zapsané pomocí dvojitých uvozovek ("") vyhodnocují speciální znaky (např. „\n“ jako nový řádek), a umožňují přímé vkládání jiných řetězců, pokud jsou vloženy pomocí operátoru „uv#{}“.

Řetězce lze skládat i dalšími způsoby, např. pomocí „+“, tyto způsoby jsou však z hlediska výkonu pomalejší než první uvedený. Jednoduché uvozovky (') text nevyhodnocují a např. „\n“ vypíší jako „\n“, ne jako nový řádek

```
s = 'test'
s2 = "druhy #{s}" #  => druhy test
s2 == 'druhy ' +
s
```

String, standardní knihovna definující řetězce, obsahuje velký počet užitečných funkcí, např:

```
'10-20-30'.split('-') # => ["10", "20", "30"]
'David'.gsub('vid', 'niel') # => "Daniel"
```

Regulární výrazy se zapisují do „/ /“ a je možné je využít např. při nahrazování určitých míst v řetězci:

```
/.*regex$/
'a12a13a14'.gsub(/[0-9]/, 'B') # => "aBBaBBaBB"
```

Funkce
Funkce jsou definovány klíčovým slovem „def“ a ukončeny „end“. Ruby neumožňuje přetěžování funkcí, jako například c++. Návratová hodnota funkce je určena buď klíčovým slovem „return“ nebo výsledkem posledního vykonaného řádku.

```
def sum(a, b)
   a + b
end
```

Psaní závorek není při volání funkce vyžadované, předchozí funkci můžeme použít následnými způsoby:

```
sum(2, 4) # => 6
sum 2, 4  # => 6
```

Podmínky
Standardní podmínky se sestávají z klíčových slov „if“, „else“ a „end“. If může být nahrazeno slovem „unless“, které znamená „if not“. Pokud jsou podmínky jednořádkové, neukončují se slovem end:

```
if true
   p 'TRUE' # p je alias pro puts, vypíše vstup + \n
else
   p 'FALSE'
end

p 'TRUE' if 1 == 1 # => 'TRUE'
```

U proměnných je nutné kontrolovat, jestli jejich hodnota není nil (Ruby verze pro NULL), a až poté jim posílat metodu. S pomocí operátoru & je toto možné udělat na jednom řádku. „||=“ pak do proměnné přiřadí hodnotu pouze tehdy, pokud je v ní nil.

V následujících ukázkách uvádím ekvivalentní zápisy vždy napřed bez použití zmíněných operátorů a poté s nimi:

```
unless array.nil?
    array.each { ... }
end
array&.each { ... }

s = '' if s.nil?
s ||= ''
```

Pole, Hash, symboly
Pole může obsahovat libovolné objekty. Je možné přistupovat pouze k prvkům v zadaném intervalu. Pro přidání nových prvků do pole lze použít operátor „«“. Poslední nebo první člen pole je možné získat funkcemi „first“ a „last“:

```
a = ['string', 15, 9, []]
b = a[1..2] # => [15, 9]
b << 3 # => [15, 9, 3]
b.last == 3 # => true
```

Symboly jsou objekty reprezentující určité řetězce. Zapisují se jako lokální proměnné nebo řetězce, před nimiž je „:“. Za běhu programu je pro stejný text vytvořen vždy stejný objekt (na rozdíl od například opakovaného vytváření stejného řetězce).

```
'a',object_id # => 17971700
'a',object_id # => 17937860

:a.object_id # => 744988
:a.object_id # => 744988
```

Třída Hash značí hash (rozptylovací) tabulku, která obsahuje unikátní klíče a jejich hodnoty. Jako klíče se často využívají symboly (kvůli nižší paměťové náročnosti a efektivnějšímu porovnávání). K hodnotě, spojené s určitým klíčem, přistupujeme podobným způsobem jako u pole:

```
h = { a: 1, b: 'string' }
h[:a] == 1
```

Bloky, yield
Bloky jsou samostané části kódu, je možné je ohraničit buď závorkami „{ }“ nebo klíčovými slovy „do“ a „end“. Názvy vstupních proměnných jsou zavedeny v místě ohraničeném „| |“. Používat se mohou např. u iterátorů:

```
a = [1,5,3]
a.each { |num| print num } # => 153

sum = 0
a.each do |num|
   sum += num
end
sum == 9
```

Pokud funkce očekává, že obdrží blok jako vstupní argument, jeho provedení vyvolá klíčovým slovem yield (funkce tak může být využita mnoha různými způsoby): (pozn. funkce „to_s“ v tomto kontextu převede číslo na řetězec)

```
def each_number(numbers)
   numbers.each { |num| yield num }
end

sum = 0
each_number([1, 5, 3]) { |n| sum += n }
sum == 9

text = ''
each_number([1, 5, 3]) { |n| sum += n.to_s }
text == '153'
```

Třídy, moduly, require
Definování a použití nové třídy:

```
class Person
   attr_accessor :name

   def initialize(name)
      @name = name
   end

   def hello
      p "Hello, my name is #{@name}"
   end

   def self.hello_world
      p 'Hello world from Person'
   end
end

person = Person.new('David')
person.hello # => "Hello, my name is David"
Person.hello_world # => "Hello world from Person"

person.name = 'Karel'
parson.name # => "Karel"
```

Třídy mají metody třídní nebo instanční. Třídní metody začínají na „self.“ a jsou vyvolávány přes název třídy (viz metoda hello_world), instanční metody je pak možné použít pro vytvořenou instanci třídy. Attr_accessor vytvoří metody pro čtení a změnu proměnné.

Při potřebě použití vytvořené třídy v jiném souboru s Ruby kódem je možné využít require, require_relative nebo load: (předpokládám, že třída Person je v souboru person.rb)

```
require_relative 'person'
Person.new('David').hello # => "Hello, my name is David"
```

Třída může být rozšířena o tzv. moduly. Jedná se o kolekci funkcí a konstant, které lze do třídy přidat. Ke vnořeným modulům, třídám nebo konstantám se přistupuje operátorem „::“:

```
module Hello
   def hello
      p 'Hello world'
   end
end

class Test
   include Hello
end
Test.new.hello # => "Hello world"

module A
   class B
      def self.test
         p 'test'
      end
   end
end

A::B.test # 'test'
```

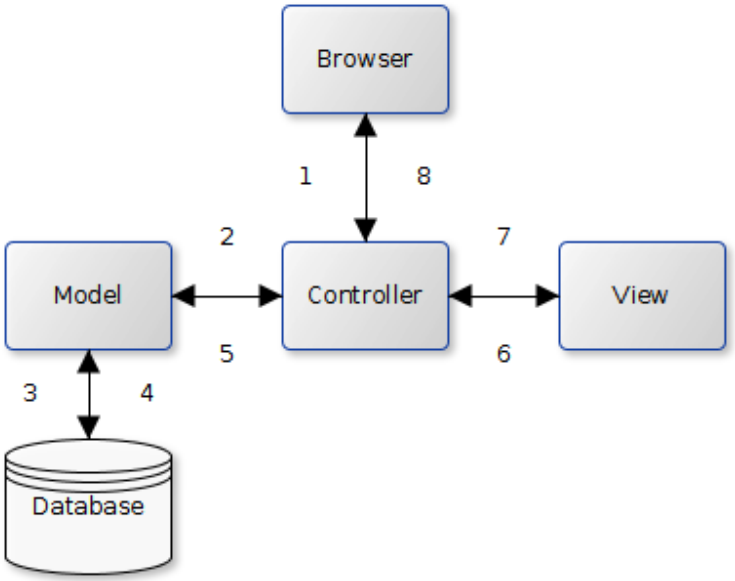
Výjimky
Pokud při vykonávání kódu dojde ke kritické chybě, je vyvolána tzv. výjimka (error). Taková situace může nastat např. při volání nedefinované metody. K zachycení a zpracování vyvolané výjimky slouží „rescue“: (NoMethodError je potomek třídy StandardError, proto je tato výjimka zachycena):

```
nil[0]
rescue StandardError => e
p e # => #<NoMethodError: undefined method `[]' for nil:NilClass>
```

1.4.4 Ruby on Rails

Aplikace psané RoR se řídí architekturou MVC (Model View Controller). Jedná se o architektonický vzor, který dělí aplikaci do tří logických vrstev (viz 1.9):

- Modely - reprezentují entity v databázi a poskytují logiku (funkce) k jejich načítání, upravování a dalším s nimi spojeným operacím



Obrázek 1.9: Diagram standardní architektura MVC aplikací

- View - generují požadované stránky
- Controller - řídí komunikaci mezi uživatelem (prohlížečem), modely a pohledy

1.4.5 Obsah EBIE

Obsah portálu EBIE je uložen v repozitáři ebie-content.

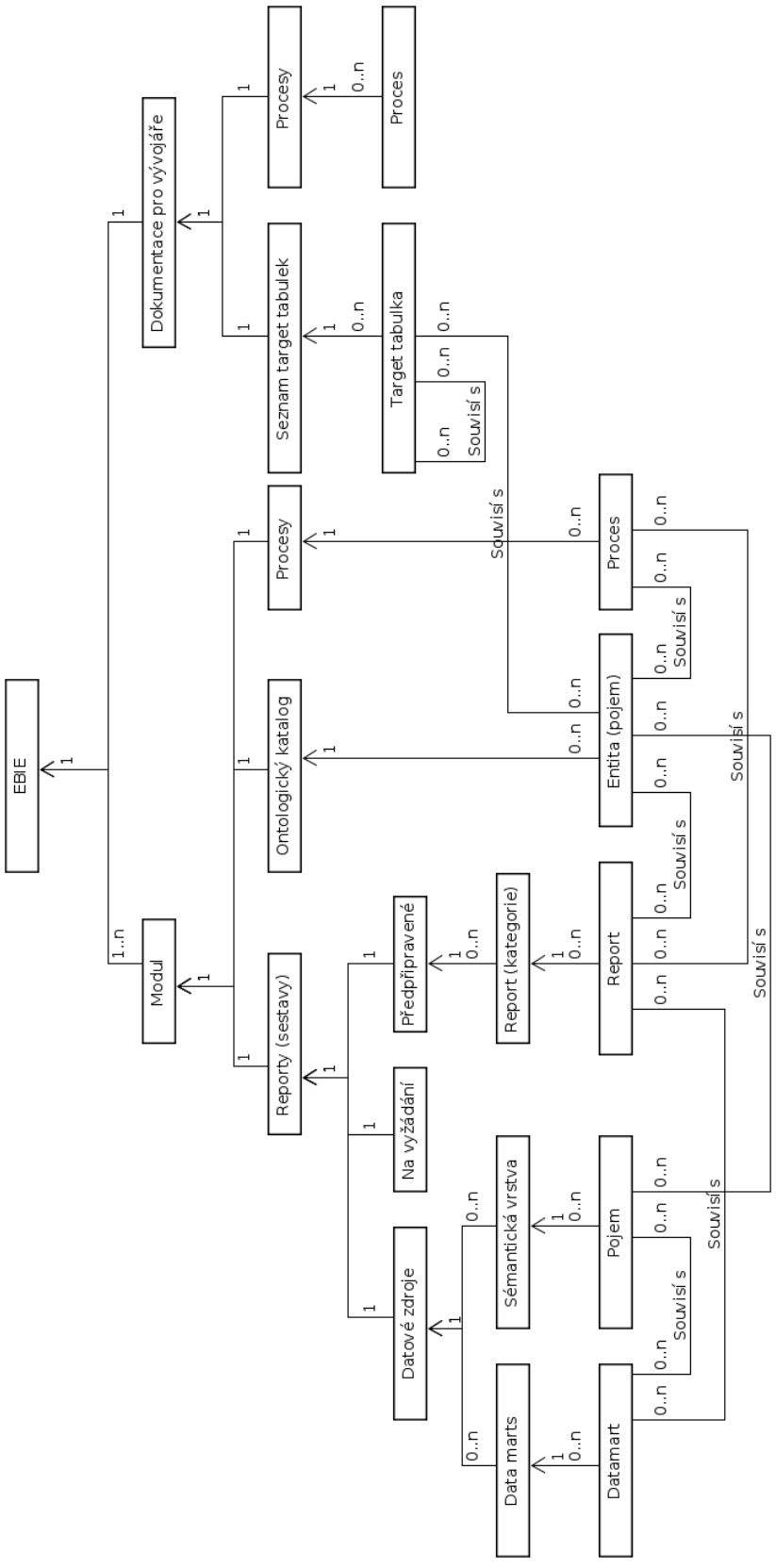
Jeho momentální podoba je viditelná na diagramu 1.10. Na nejvyšší úrovni se portál dělí na moduly a Dokumentaci pro vývojáře (kterou popisuji v 1.4.7. V tuto chvíli je vytvořený jediný modul - Studium, který se dělí na 3 sekce:

- Ontologický katalog - tato sekce obsahuje zavedené byznys entity, které slouží uživatelům zejména pro přesné pochopení reportů a specifikaci úprav stávajících reportů, popř. vytváření požadavků na nové.

  Entity mají přímou vazbu na procesy a reporty.

- Procesy - popisují činnosti organizace (jako je např. uznávání předmětů z jiné fakulty ČVUT). Každý proces je jasně definován pomocí metodik DEMO a BPMN a vždy jsou uvedeny entity, které v daném procesu vystupují.

- Reporty - dělí se na 3 podsekce:



Obrázek 1.10: Architektura EBIE v září 2017

  - Na vyžádání - zde bude možné požádat o vytvoření nového reportu.
  - Předpřipravené - vytvořené sestavy, které graficky zobrazují data z datamartů. Uživatel má k dispozici různé filtry, pomocí nichž je možné upravit výsledný graf.
  - Datové zdroje - obsahují popis Datamartů a Sémantické vrstvy. Protože se přímo týkají dokumentace DWH3, podrobněji se jim věnuji v sekci 1.4.6.

1.4.6 Dokumentace datových zdrojů na EBIE

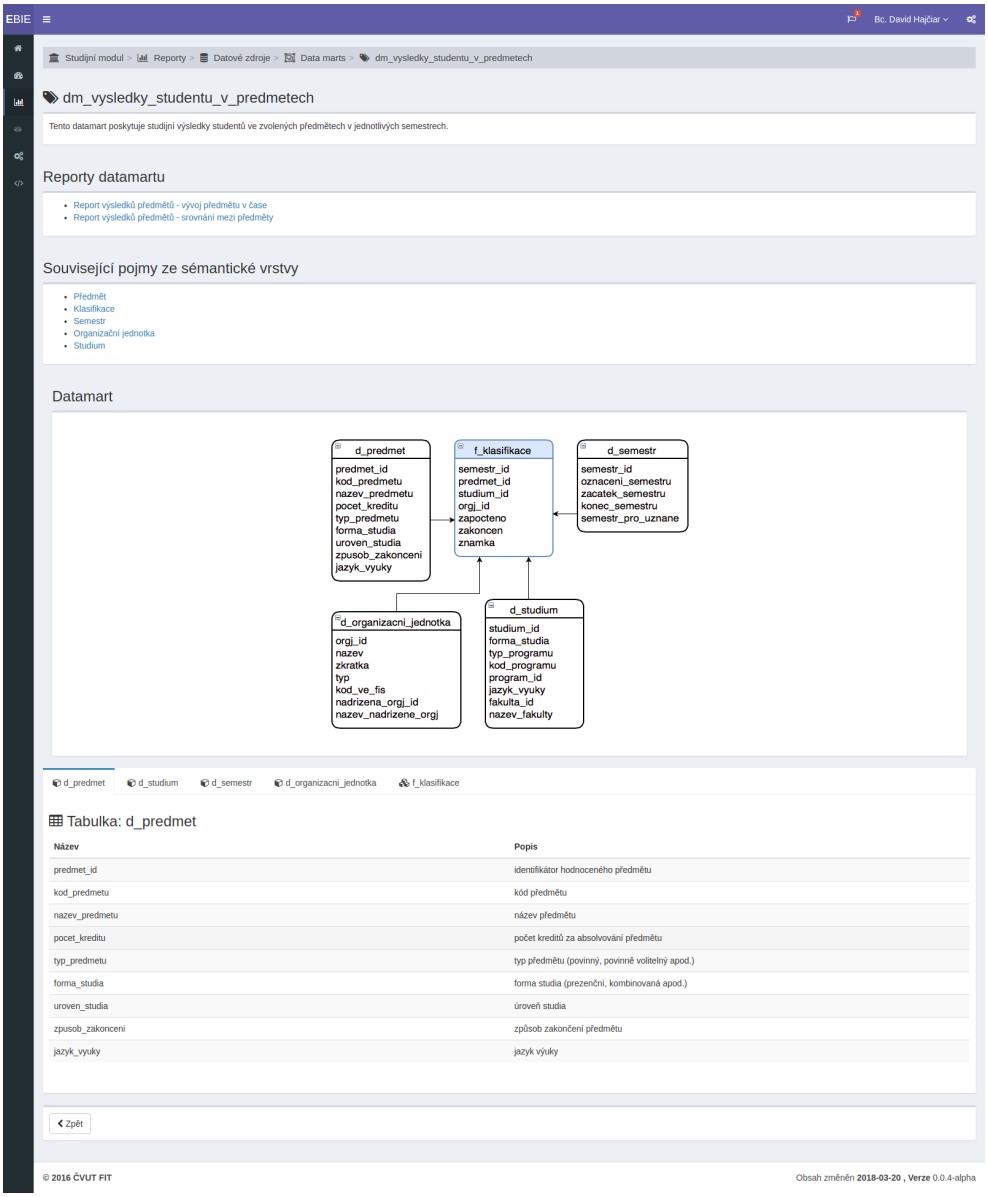
Na snímku obrazovky 1.11 je vidět vzhled stránky popisující datamart v oddílu Data marts. Obsahem je:

- Popis
- Odkazy na reporty, které používají data z tohoto datamartu
- Odkazy na související pojmy ze sémantické vrstvy
- Diagram zobrazující tabulky datamartu společně s jejich vazbami
- Tabulku s názvy sloupců všech tabulek a s jejich popisy

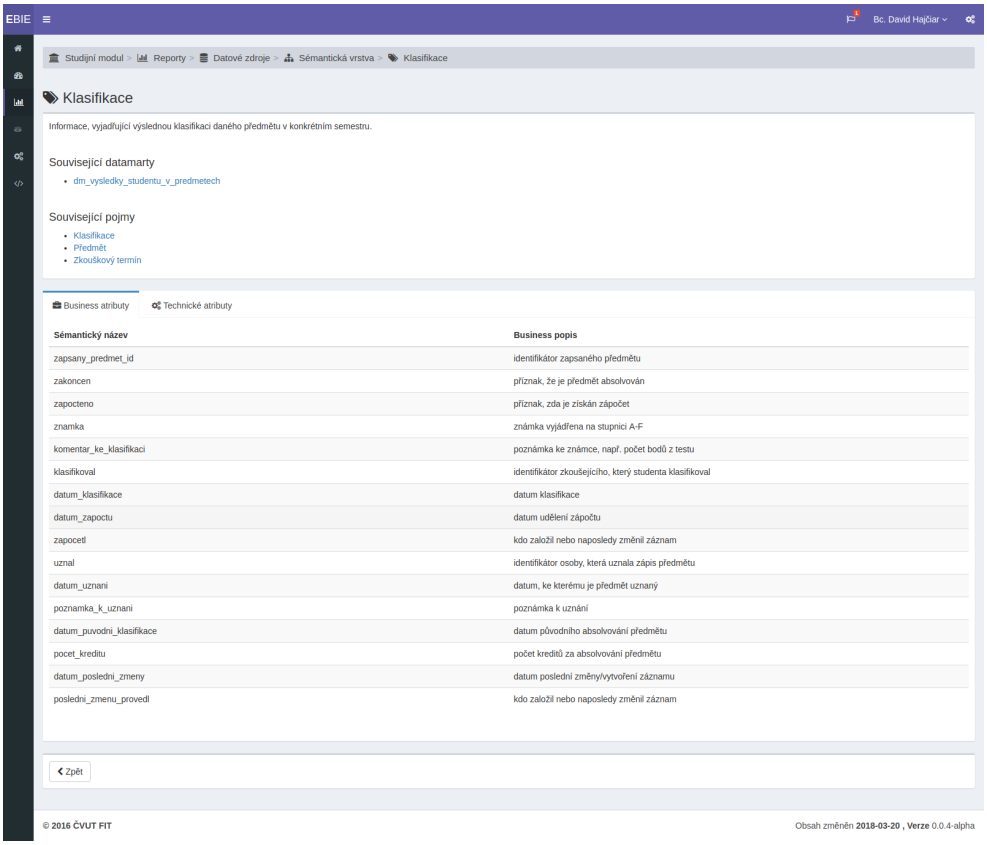
Ukázka pojmu ze Sémantické vrstvy je na obrázku 1.12. Stránka se sestává z částí:

- Popis
- Odkazy na související datamarty
- Odkazy na související pojmy (entity) Ontologického katalogu
- Tabulka se záložkami pro business a technické atributy, která pro každý sémantický název obsahuje:
  - business popis
  - mapování v target tabulkách
  - název zdrojové tabulky

Pojem ze sémantické vrstvy vždy popisuje Sémantické názvy sloupců určité target tabulky.



Obrázek 1.11: Ukázka popisu datamartu Výsledky studentů v předmětech



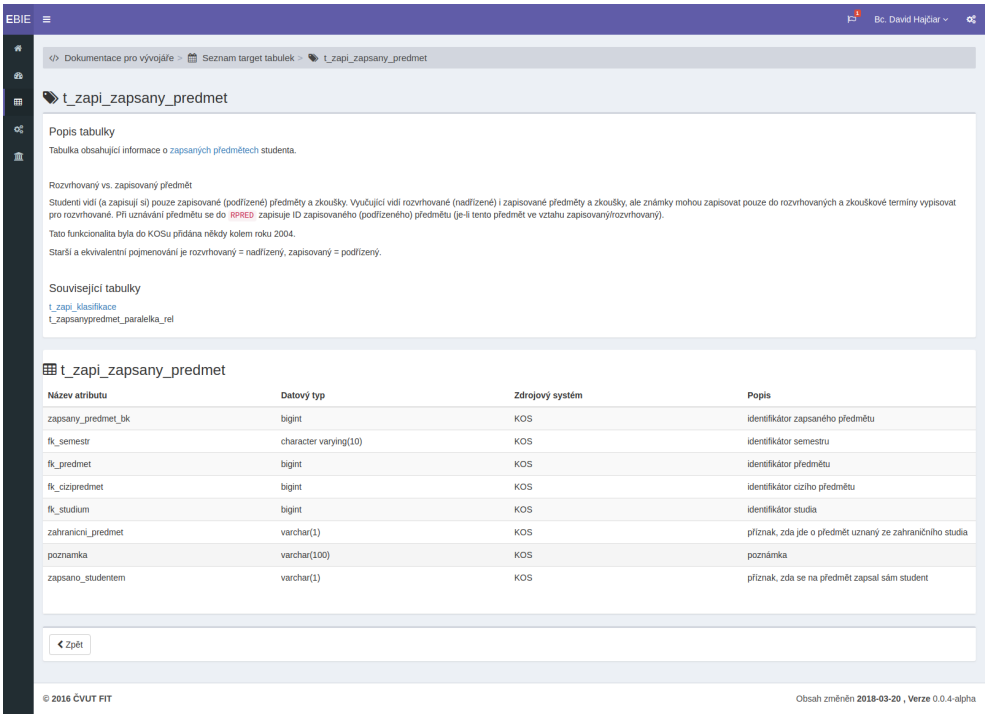
Obrázek 1.12: Ukázka popisu pojmu klasifikace v Sémantické vrstvě

1.4.7 Dokumentace pro vývojáře na EBIE

Dokumentace pro vývojáře dokumentuje zdrojové tabulky z DWH3 a různé procesy. Dělí na 2 sekce - Seznam target tabulek a Procesy.

Seznam target tabulek obsahuje jednotlivé tabulky (viz 1.13), které se sestávají z těchto částí:

- Popis
- Odkaz na související tabulky
  - Související tabulky patří v tabulce DLM schéma pod stejnou entitu
- Html tabulka s vybranými atributy z DLM:
  - Název atributu
  - Datový typ
  - Zdrojový systém



Obrázek 1.13: Ukázka tabulky t_zapi_zapsany_predmet v sekci Seznam target tabulek

– Popis

V sekci Procesy jsou dokumentovány procesy týkající se spravování obsahu na EBIE, datového skladu či kontroly čistoty dat. Z důvody velikosti uvádím pouze ukázku části stránky - 1.14. Procesy obsahují:

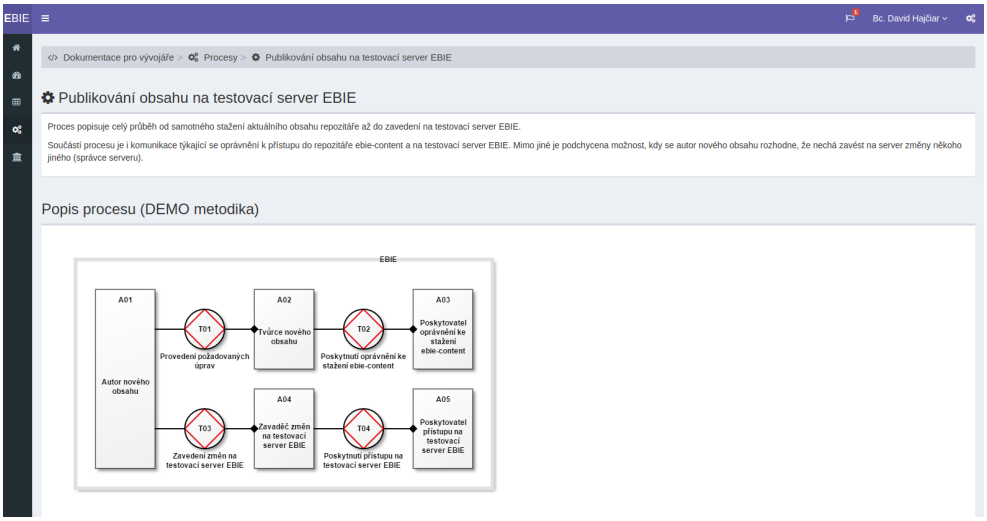
- Popis
- Popis procesu pomocí DEMO metodiky
- Popis procesu metodikou BPMN

1.4.8 Ebie-content

Na závěr popisu obsahu EBIE je nezbytné objasnit, jakým způsobem je repozitář ebie-content navržen a jak funguje jeho propojení s aplikačním serverem.

Adresářová struktura v repozitáři je členěna podle jednotlivých modulů, jejich sekcí, podsekcí atd. Na každé úrovni (s výjimkou podsekce předpřipravené reporty) je soubor info.json, který popisuje přítomné složky a soubory.

Aplikace ebie-source soubor info.json načte a na jeho základě zobrazí na stránce požadovaný obsah. Data jsou v těchto souborech uložena ve formátu



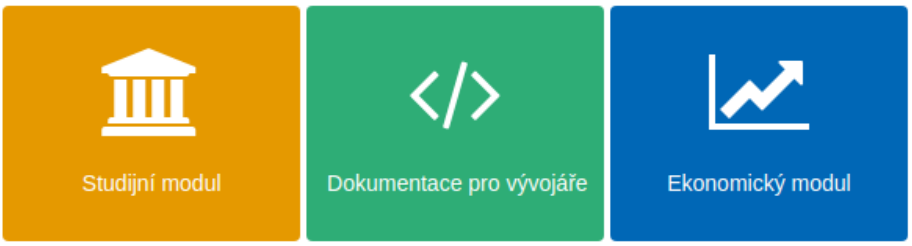
Obrázek 1.14: Ukázku dokumentace procesu pro publikování obsahu na testovací server EBIE

JSON (JavaScript Object Notation), což je *„textový, na jazyce zcela nezávislý formát pro výměnu dat“* [20].

Jako ukázku uvádím soubor info.json v kořenovém adresáři složky s obsahem v ebie-content, ve kterém jsou zavedeny zobrazované moduly (viz 1.15, Ekonomický modul je v tuto chvíli prázdný):

```
{
  "modules" : [
    {
      "title": "Studijní modul",
      "description": "Informace o~studiu",
      "icon": "fa fa-university",
      "background_color": "#e59900",
      "directory": "studium"
    },
    {
      "title": "Ekonomický modul",
      "description": "Ekonomický přehled",
      "icon": "fa fa-line-chart",
      "background_color": "#0067B6",
      "directory": "ekonomika"
    },
    {
      "title": "Dokumentace pro vývojáře",
      "description": "Informace pro vývojáře",
```

```
      "icon": "fa fa-code",
      "background_color": "#2ead76",
      "directory": "dokumentace"
    }
  ]
}
```



Obrázek 1.15: Ukázka modulů EBIE, zobrazených na základě souboru info.json

Kromě podsekce Předpřipravené reporty, která je uložena v čistém html kódu, jsou všechny ostatní dokumenty psané ve formátu AsciiDoc.

AsciiDoc je *„značkovací jazyk, psaný čistě textově, který je možné konvertovat např. do html5“* [21]. V případě EBIE se používá jeho implementace pro Ruby, která se nazývá Asciidoctor a je volně ke stažení jako Ruby gem. Soubory v tomto formátu jsou ukládány s koncovkou .adoc.

Příklad zdrojového AsciiDoc textu a z něj vygenerovaného html kódu:

```
==== Práva fakulty
Mezi práva fakulty patří:

* tvorba a uskutečňování studijních programů,

...
```

```
<h4>Práva fakulty</h4>

<p>Mezi práva fakulty patří:</p>

<ul>
<li>
tvorba a uskutečňování studijních programů,
</li>
<li>
```

Ebie-source obsahuje generátor, který po spuštění vytvoří z existujících .adoc souborů nové .html soubory, jež jsou zobrazitelné v prostředí EBIE.

Pro vytváření nových AsciiDoc souborů jsou pro každý typ obsahu (datamarty, target tabulky atd.) připraveny šablony, který definují vzhled a rozložení stránky stejných dokumentů.

1.4.8.1 Struktura AsciiDoc dokumentů v EBIE

S využitím datamartu Výsledky studentů v předmětech (viz 1.11) ukáži používaný zápis ve formátu AsciiDoc.

Komentáře (které nejsou po převedení do html zobrazeny) se vkládají zapsáním znaků „//“ na začátek řádku:

```
// Popis datamartu
```

Nadpis je na stránce zapsán ve vloženém html kódu. AsciiDoc podporuje přímé vkládání html kódu do označených částí dokumentu. Tyto části jsou ohraničeny buď znaky „++++ libovolný víceřádkový html ++++“ pro víceřádkové oblasti, nebo „+++ jeden řádek +++“ pro zápis na jediném řádku.

Tímto způsobem je zapsán i nadpis stránky:

```
+++<h3><i class="fa fa-tags"></i>dm_vysledky/h3>+++
```

Popis datamartu je vložen jako čistý text:

```
Tento datamart poskytuje studijní výsledky studentů ve zvolených
předmětech v jednotlivých semestrech.
```

Odkazy na reporty datamartu využívají nečíslovaný seznam (znak „*“ na začátku řádku), a poté interní odkazy (odkazy na jiné stránky v EBIE), zapisované jako: „link:url[text odkazu]“.

Hodnoty v závorkách „{}“ jsou nazývány atributy a fungují jako proměnné. Atributy jsou v ebie-content definovány v souboru logic.adoc:

```
* link:{subject_results}subject.html[Report - vývoj]
* link:{subject_results}semester.html[Report - srovnání]
```

Odkaz na vkládaný obrázek, který zobrazuje diagram datamartu, je vytvořen pomocí jeho názvu (image::nazev.pripona[], v tomto případě se jedná o obrázek dm1.png).

```
image::dm1.png[]
```

V souboru logic.adoc je atributem „imagesdir“ definována cesta k obrázku ve formátu vyžadovaném aplikačním serverem EBIE, při konvertování do html je tato cesta přidána do odkazu na obrázek („:imagesdir: img?path=.&file=“).

Sloupce v tabulkách jsou odděleny znakem „|“. Úvodní řádek určuje html třídy tabulky, vlastní řádky jsou ohraničeny řetězcem „|===“:

```
[role="table table-hover table-striped"]
[options="header"]
|===
| Název | Popis
| predmet_id | identifikátor hodnoceného předmětu
| kod_predmetu | kód předmětu
...
|===
```

Převod AsciiDoc dokumentů do html je prováděn s využitím tzv. „backendů“. Každý modul má k dispozici soubory backendu ve složce .html.

1.5 Procesy dokumentování datového skladu na portálu EBIE

Proces vytváření dokumentace se sestává z šesti kroků:

1. Příprava lokálního prostředí
2. Provedení úprav
3. Kontrola změn
4. Nahrání změn do repozitáře ebie-content
5. Zavedení změn na testovací server ebie-vyvoj
6. Zavedení změn na produkční server ebie

V následující sekcích tyto kroky podrobně popíši.

1.5.1 Příprava lokálního prostředí

Jedná se o jednorázovou činnost, při které si uživatel připraví software nezbytný pro používání repozitáře; podmínkou je nainstalovaný git a přístup do ebie-content.

Požadované systémové závislosti, společně s minimálními verzemi, které jsou uvedeny v souboru README.adoc:

- Ruby 2.2
- Asciidoctor 1.5
- Tilt 2

1.5.2 Provedení úprav

EBIE se dělí na mnoho částí (viz 1.10), pro něž je možné dokumenty vytvářet, upravovat nebo odstranit (při přidávání nebo odebírání obsahu je nutné patřičně upravit příslušný soubor info.json). Možné úpravy se tak týkají těchto oblastí:

- Entity Ontologického katalogu
- Procesy
- Reporty
  - Předpřipravené reporty
  - Datové zdroje
    - Datová tržiště
    - Sémantické vrstvy
- Seznam target tabulek
- Procesy v Dokumentaci pro vývojáře
- Moduly
- Sekce

Vzhledem k zaměření této diplomové práce se dále věnuji provádění úprav v sekci Datové zdroje - Datová tržiště a Sémantická vrstva a v modulu Dokumentace pro vývojáře Seznamu target tabulek.

1.5.3 Úpravy Dokumentace pro vývojáře

Data pro jednotlivé target tabulky jsou, kromě popisu tabulky, získávána z DLM schéma (viz 1.3). Zdrojem pro tyto části jsou:

- Související tabulky - tabulky jsou související, pokud patří pod stejnou entitu (DLM schéma sloupec A)
- Dokumentovaná target tabulka - obsahuje Název atributu (DLM schéma sloupec I), Datový typ (sloupec J), Zdrojový systém (sloupec B) a Popis (sloupec N).

Při úpravě je nutné procházet DLM, kopírovat data ze správných sloupců a řádků a následně je převádět do finálního AsciiDoc formátu.

1.5.4 Úpravy Datových zdrojů

Data pro datamarty v podsekci Datová tržiště:

- Reporty a Související pojmy - z přítomných dokumentů v EBIE
- Tabulky - informace jsou obsaženy v materializovaných pohledech v databázi dwh3_access. Je nutné vzdáleně se připojit k databázi a vyčíst informace ze schématu zvoleného datamartu.

  Popis jednotlivých sloupců (jejichž pojmenování odpovídá Sémantickým názvům) je uložen v DLM schéma (sloupec N)
- Datamart - diagram, který obsahuje tabulky v materializovaném pohledu a jejich vazby; využívá tedy podobná data jako tabulky (bez nutnosti připojení k DLM)

Pro Reporty, Související pojmy a Tabulky je třeba získané informace převést do formátu AsciiDoc. Tabulky jsou rozděleny záložkami, ty je třeba zadávat v html kódu společně s úpravami pro jejich názvy a ikony.

Diagram datamartu je pak generován libovolným modelovacím nástrojem.

Výsledný obrázek je nutné vložit do složky assets/img v adresáři s datamarty a poté na něj odkázat v upravovaném souboru.

Zdroje dat pro Sémantickou vrstvu:

- Související datamarty a Související pojmy - z obsahu EBIE
- Tabulka - DLM schéma. Sémantický název odpovídá sloupci K, Business popis sloupci N, Mapování v targetu sloupci I a Zdrojová tabulka sloupci H

Formátování dat probíhá stejným způsobem jako u Datových tržišť.

1.5.5 Kontrola změn

Po upravení obsahu má uživatel k dispozici ruby skript (napodobení generátoru z ebie-source, viz 1.4.8), který ze všech existujících .adoc souborů vygeneruje html soubory do adresáře test.

Struktura adresářů se zde vytvoří stejně jako je tomu v adresáři ebie_content (např. souboru ebie_content/studium/ontology/anketa.adoc je vytvořen odpovídající test/studium/ontology/anketa.html).

Složka source obsahuje css soubory, hlavičky a zápatí z EBIE, které jsou přidávány testovaným souborům, aby jejich vzhled co nejvíce odpovídal výslednému vzhledu na serveru EBIE.

1.5.6 Nahrání změn do ebie-content

V případě, že chce uživatel nahrát své lokální úpravy do repozitáře ebie-content, musí vytvořit git commit a poté git push (na větev master_vyvoj). Posloupnost příkazů, zadaných v terminálu, může vypadat takto:

```
git add -A
git commit -m'Popis změn'
git push origin master_vyvoj
```

Operace je dokončena, pokud má uživatel oprávnění nahrát změny do repozitáře. V opačném případě se na Gitlabu automaticky vytvoří merge request, který musí schválit osoba s dostatečnými pravomocemi.

1.5.7 Zavedení změn na testovací server

Změny se zavádí následujícími kroky:

1. Přihlášení se na server ebie-vývoj
   - *root@ebie-vyvoj.is.cvut.cz*
   - Uživatel musí mít zajištěný přístup na server
2. Změna uživatele na ebie a přesun do adresáře ebie-web
3. Stažení nových dat
   - *git submodule update –remote*
4. Vygenerování nového obsahu
   - *bundle exec rails generate content*

1.5.8 Zavedení změn na produkční server

Správce repozitáře ebie-content provede git merge z větve master-vyvoj na master a poté provede update produkčního serveru stejným způsobem, jako je tento proces prováděn na testovacím serveru.

1.6 Problémy s vytvářením dokumentace na portálu EBIE

V analýze procesů dokumentování datového skladu na portál EBIE (viz 1.5) jsem podrobně popsal všechny kroky, které musí uživatel vykonat při jakékoli úpravě.

Nyní se zaměřím na rozbor funkčnosti a problémů jednotlivých kroků těchto procesů při reálném používání.

1.6.1 Příprava lokálního prostředí

Tento krok je jasně nastaven a dobře zdokumentován, uživatel má přesné informace o potřebných nástrojích i požadovaných verzích. Jediný problém může nastat při aktualizaci některého z používaných programů na vyšší verzi (např. kvůli nalezené chybě ve staré verzi).

Uživatel se tuto informaci může dozvědět pouze zprávou od vývojáře nebo pravidelnou kontrolou README repozitáře.

1.6.2 Provedení úprav

Pro úpravy jak Datových zdrojů (viz 1.5.4) tak Dokumentace pro vývojáře (viz 1.5.3) je nutné manuálně vyhledávat a kopírovat velké množství dat.

Tento proces může být jednak únavný, jednak náchylný k těžko odhalitelným chybám (DLM má v době psaní této práce zhruba 1300 řádků a 16 sloupců, např. pojem Předmět ze Sémantické vrstvy popisuje 54 sémantických názvů, z nichž každý obsahuje 3 dodatečné informace (viz 1.4.6), celkem tedy 216 hodnot na kontrolování).

Přidané hodnoty navíc musí být převedeny do formátu AsciiDoc, pro tabulky s více záložkami je nutné správně formátovat html kód a tak zajistit jejich správné zobrazení.

Další časově náročnou věcí je tvorba diagramů datových tržišť (viz 1.4.6). Pro jejich vytvoření je nutné převést obsah tabulek do formátu požadovaného nástrojem, který tyto diagramy vytváří.

Pokud dojde ke změně v některém z datamartů na DWH3, popř. v DLM, je nutné tuto změnu neprodleně zavést do EBIE, protože vzhledem k množství zobrazovaných informací je zpětné dohledávání změny časově velmi náročné.

To ovšem znamená, že pro předejití nekonzistenci dat musí být všichni vývojáři datového skladu detailně seznámeni s procesem dokumentování na EBIE. Při změně datamartu je navíc nutné znovu generovat zobrazovaný diagram, a pokud uživatel vykonávající změny nemá naposledy použitá data pro tento diagram uložena, musí je znovu ručně převést do potřebného formátu.

1.6.3 Kontrola změn

Kontrola provedených změn je velmi náročná. Existující skript sice umožní přibližné zobrazení výsledné podoby, není však možné otestovat funkčnost vytvořených interních odkazů a správné zobrazení přiložených obrázků (to zajišťuje aplikační server).

Pokud v ebie-source dojde ke změně html kódu (či souvisejícího css), je nutné tyto úpravy zpropagovat i do ebie-content. Pokud by se tak nestalo, kontrolní zobrazení bude nepřesné i vizuálně.

1.6.4 Zavedení změn na testovací server

Zavedení změn na testovací server se sestává z příliš mnoha kroků, které je často nutné opakovat v krátkém časovém rozpětí. Vykonání commitu se změnami do repozitáře ebie-content je nezbytné, uživatel však musí také vykonat všechny kroky popsané v sekci 1.5.7.

S tím je spojeno hned několik problémů:

- Uživatel musí mít zajištěný přístup na server, kde je nasazen EBIE-vývoj
- Uživatel musí znát instrukce pro nahrání změn a restart serveru
- Provedení změn je časově náročné.

  Když se tyto komplikace spojí s problémy při kontrole změn, osoba provádějící změny má šanci zkontrolovat jejich výslednou podobu až za několik minut po nahrání, což je při řešení problémů se zobrazováním značně nešťastné.

  Tyto problémy se reálně projevily při přidávání obsahu v minulých letech.

1.6.5 Zavedení změn na produkční server

Nahrávání změn na produkční server funguje bez komplikací. Je zde nezbytná ruční aktualizace dat na serveru, avšak vzhledem k tomu, že se jedná o méně častou operaci, to nepřináší výrazné problémy.

Kapitola 2

Návrh

V této kapitole se věnuji návrhu dokumentování datových toků na základě analýzy v předchozí kapitole, dále návrhu generátorů pro podporu dokumentování a nakonec volbě vhodných technologií pro navržené procesy.

2.1 Dokumentace datových toků v datovém skladu

Obrázek 2.1 zobrazuje přehledný pohled na datové toky v DWH3. Vyznačil jsem na něm klíčové oblasti pro procesy dokumentování:

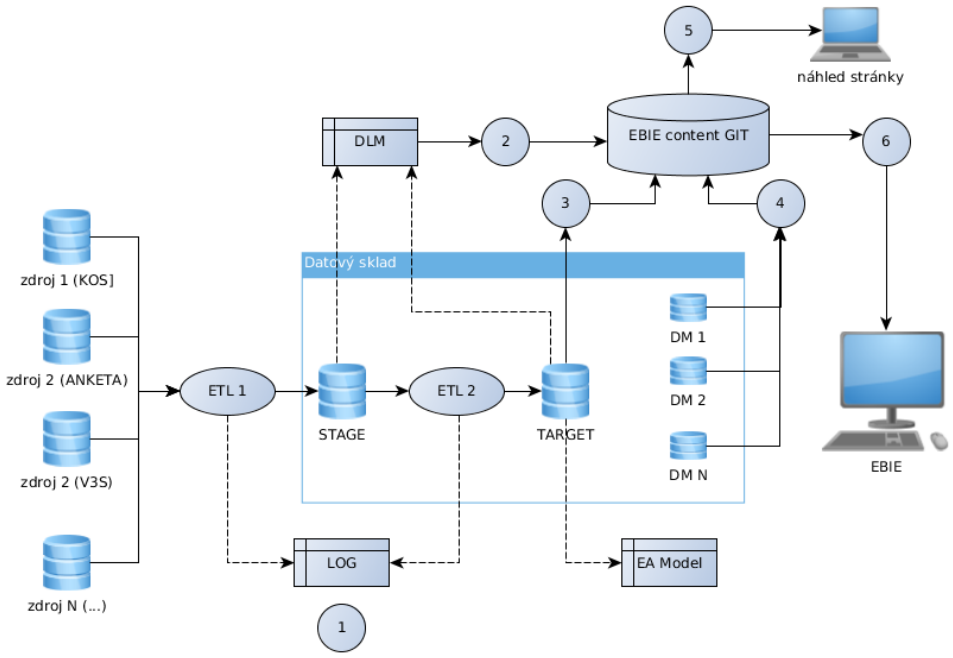
1. Dokumentace ETL procesů
2. Dokumentace s využitím DLM
3. Dokumentace target tabulek
4. Dokumentace datových tržišť
5. Náhled na upravovaný obsah
6. Nahrání změn na portál EBIE

Dokumentací ETL procesů se v této práci nezabývám, neboť v tomto roce se ji věnuje bakalářská práce Ondřeje Pletichy [22].

U zbylých částí uvádím v následující sekcích jejich význam a navrhuji způsob nových procesů dokumentace.

Jako vhodnou platformu pro zobrazení této dokumentace vidím portál EBIE, a to z následujících důvodů:

- Portál EBIE byl od počáku zamýšlen pro dokumentování datového skladu



Obrázek 2.1: Datové toky v datovém skladu ČVUT

- Z existujících platforem pro dokumentaci (viz 1.3) je EBIE pro uživatele nejlépe dostupná
- Uživatelé EBIE budou mít na jednom místě jak reporty, tak dokumentaci jejich zdrojů

2.1.1 Dokumentace DLM

DLM schéma dokumentuje mnoho důležitých informací ohledně provozu datového skladu - zdrojové systémy, target tabulky, sémantickou vrstvu, datové entity atd.

Kromě zdrojových systémů jsou v tuto chvíli v EBIE přítomné všechny ostatní části. Protože se tato práce, a také portál EBIE, týká primárně vlastního datového skladu, po domluvě s vedoucím mé práce se zdrojovým systémům více nevěnuji.

Zbylé údaje slouží k pochopení souvislostí a struktury těchto částí datového skladu. Vývojáři si v modulu pro Vývojáře mohou přehledně zobrazit aktuální target tabulky.

Target tabulky obsahují základní potřebné údaje o svém zdroji - jednotlivé atributy společně s jejich datovým typem, zdrojovým systémem a popisem,

rychlý přehled souvisejících tabulek, viz 1.4.7.

Pojmy ze sémantické vrstvy také obsahují všechny podstatné informace - související datamarty a pojmy, sémantické názvy společně s business popisem, mapováním v target tabulkách a zdrojovou tabulkou, viz 1.4.6.

Navrhuji tyto části target tabulek a pojmů ze sémantické vrstvy, dokumentované na základě DLM schéma, zachovat ve stávající podobě.

Jak jsem ukázal v sekci 1.6, aktuální způsob vytváření dokumentace s využitím DLM schéma je nevyhovující.

Jako řešení uvedených problémů navrhuji vytvořit podpůrné generátory, které by vykonávaly následující činnosti:

- Vytváření nových souborů pro Pojmy ze sémantické vrstvy a Target tabulky:
  - Vytváření tabulek ve formátu AsciiDoc
  - Vyhledávání souvisejících tabulek pro Target tabulky a vytváření interních odkazů na tyto tabulky
  - Vložení informací o novém souboru do správného info.json
- Aktualizace existujících souborů v těchto sekcích

Tyto generátory pro DLM vyřeší problémy popsané v sekci 1.6.2:

- Ve vytvořených tabulkách budou vždy správné údaje (nehledě na obsáhlost DLM)
- Nebudou hrozit špatně přepsaná slova
- Informace budou vhodně převedeny do AsciiDoc formátu
- Aktualizace přítomných dokumentů bude snadná, rychlá a přesná

Konkrétním návrhům jednotlivých generátorů se věnuji v sekci 2.2.

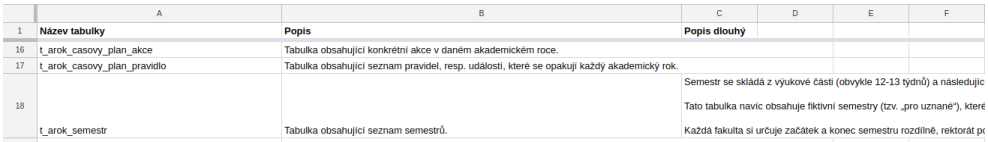
Rozšíření DLM

Po diskuzi s Ing. Magdou Friedjungovou z týmu vývojářů datového skladu jsme se rozhodli do DLM přidat novou tabulku s názvem „tabulky“.

Tato tabulka (dále ji budu označovat jako „DLM tabulky“) bude obsahovat názvy target tabulek a jejich popisy (viz 2.2). Generátor target tabulek bude moci popisy stahovat a automaticky vkládat do nových souborů, tím bude uživateli ještě více ulehčena práce.

Pokud se tento experiment ukáže jako vhodný, bude tímto způsobem možné ukládat také popisy ostatních sekcí.

„DLM tabulky“ se bude skládat ze sloupců:



	A	B	C	D	E	F
1	**Název tabulky**	**Popis**	**Popis dlouhý**			
16	t_arok_casovy_plan_akce	Tabulka obsahující konkrétní akce v daném akademickém roce.				
17	t_arok_casovy_plan_pravidlo	Tabulka obsahující seznam pravidel, resp. událostí, které se opakují každý akademický rok.				
18	t_arok_semestr	Tabulka obsahující seznam semestrů.	Semestr se skládá z výukové části (obvykle 12-13 týdnů) a následujíc Tato tabulka navíc obsahuje fiktivní semestry (tzv. „pro uznané"), které Každá fakulta si určuje začátek a konec semestru rozdílně, rektorát p			

Obrázek 2.2: Ukázka obsahu „DLM tabulky“

- Název target tabulky
- Popis - text, který se zobrazí u popisu target tabulky v souboru info.json
- Popis dlouhý - text zobrazený jako popis v target tabulce.

U sloupce Popis dlouhý je možné používat stejnou AsciiDoc syntaxi jako v dokumentech v ebie-content:

- Interní odkazy - v standardním formátu „link:{nazev_modulu-nazev_sekce}nazev_souboru.html[zobrazený text]“
- Externí odkazy - „URL[zobrazený text]“, např. „https://ebie.is.cvut.cz[EBIE]“
- Nadpisy - „==== Nadpis“, počet „=“ značí úroveň nadpisu, vzhledem k pozici na stránce by se mělo jednat o minimálně 4 =
- Nová řádka - Buď „ +“ na konci řádky nebo „{nl}“ na samostatné řádce
- Text tučně a kurzívou - „*tučně*“ a „_kurzívou_“
- Zvýraznění textu (monospace) - „‘text‘“
- Nečíslované seznamy
  - Nepovinný nadpis seznamu - „.Nadpis“
  - Položky - „* Položka“, „** Položka“ - druhá úroveň víceúrovňového seznamu, místo „* “ je možné používat „- “
- Číslovaný seznam - používá se podobně jako nečíslovaný seznam, s rozdílem znaku pro zápis položek - „. “

2.1.2 Dokumentace target tabulek

Struktura target tabulek je dobře zdokumentovaná v modulu Dokumentace pro vývojáře, není však dostupná informace o aktuálnosti dat v jednotlivých tabulkách.

V datovém skladu je možné na aktuálnost dat nahlížet ze dvou pohledů:

- Čas posledního ETL procesu týkajícího se určité tabulky (entity) - čas nahrání dat, která nemusela změnit, přidat či odebrat žádnou hodnotu tabulky
- Čas poslední úpravy v tabulce

Oba uvedené pohledy přinášejí užitečný vhled. Prostředí pro získání času posledního proběhnutého ETL procesu je v době vytváření této práce teprve ve vývoji, proto jsem rozhodl zaměřit pouze na čas poslední úpravy (s možností pozdějšího doplnění dokumentace o čas nahrání dat pro určité entity).

Všechny target tabulky obsahují informaci o své poslední změně. Navrhuji v EBIE zobrazovat tento časový údaj pro přítomné target tabulky společně s několika dříve zaznamenanými časy změny.

Tyto informace budou sloužit nejen k zjištění data poslední aktualizace tabulky, ale také bude na základě předchozích časů možné získat představu o četnosti změn (to bude dobře viditelné např. u tabulek týkajících se Ankety ČVUT, jejíž nahrání probíhá zhruba jednou za 6 měsíců).

Navrhuji, aby aktuálnost zobrazovaného času poslední změny automaticky kontroloval pravidelně spouštěný skript, který informace o nových změnách uloží do databáze EBIE. Na stránce target tabulky pak bude zobrazena také informace o posledním a příštím naplánovaném spuštění tohoto skriptu.

Jako prostředníka pro získávání těchto dat navrhují použít server ebie-api, který obstarává načítání dat z datových tržišť v dwh3_access. Tento proces tak přirozeně doplní jeho využití.

Pro dotazy z EBIE bude nutné vytvořit výjimku při autentizaci na serveru ebie-api. Navrhuji, aby byl aplikacím ebie-api a ebie, popř. ebie-vyvoj, vytvořen tajný, silný klíč, kterým budou požadavky ověřovány. (Momentálně používáné ověřování není možné použít z toho důvodu, že využívá session uživatele, navržená komunikace však bude probíhat na pozadí, nezávisle na uživatelích.)

2.1.3 Dokumentace datamartů

Pro uživatele EBIE je pro správné porozumnění reportů důležité mít možnost vidět strukturu a popis jednotlivých datových tržišť, odkazy na související pojmy ze Sémantické vrstvy i přehled všech reportů, které z daného datamartu čerpají data.

Datová tržiště jsou již dostatečně dokumentována v portálu EBIE (viz 1.4.6), navrhuji proto stávající řešení zachovat a pouze vylepšit proces vytváření a kontroly těchto dokumentů.

Pro jejich dokumentaci navrhuji podobné řešení jako pro dokumentaci DLM schéma, tedy naprogramované generátory, které by umožňovaly:

- Vytvoření nového datamartu:
  - Vytvoření tabulky ve formátu AsciiDoc, společně s jejími vnitřními tabulkami a sloupci s popisy
  - Vygenerování diagramu
  - Vložení informací o novém souboru do příslušného souboru info.json
- Aktualizace stávajících datamartů

Tyto generátory umožní snadnou aktualizaci uložených dat, zamezí chybám při vkládání informací a velmi usnadní údržbu diagramů.

Pro vzniklé generátory je třeba zajistit přístup k databázi dwh3_access, ve které jsou datová tržiště uložená. Pro přihlášení je třeba zadat mimo jiné uživatelské jméno a heslo. Tyto údaje nejsou veřejné a ne všichni uživatelé s přístupem do ebie-content mají přístup na servery s databází, není tedy možné tyto údaje předvyplnit do veřejně přístupného kódu.

Uživatel bude muset potřebné údaje generátoru předat a tím mu umožnit přístup k dwh3_access. Zadávat je při každém spuštění generátoru není vhodné z těchto důvodů:

- Další zdržení pro uživatele - základní myšlenkou generátorů je jednoduchost a rychlost používání, zadávání všech potřebných hodnot (celkem jich je 5 - „user, password, host, port, database“) by uživatele zdrželo při každém spuštění
- Nemožnost automatických testů

Jako vhodné řešení navrhuji konfigurační soubor, do kterého budou potřebné informace zadány a generátor je při startu načte. Tento soubor musí být ignorován v prostředí git (uveden v souboru .gitignore), aby jej vývojář omylem nenahrál do repozitáře.

2.1.4 Náhled stránky

Při popisu problémů současného procesu vytváření dokumentace jsem se věnoval i problému kontroly vytvořeného/upraveného obsahu (viz 1.6.3).

Přidání generátorů obsahu tuto komplikaci neřeší, navrhuji proto umožnit nahrání zvoleného souboru z ebie-content (primárně z datamartů, sémantických pojmů a target tabulek) na testovací server EBIE, kde bude tento soubor zobrazen s aktuálním vzhledem stránky, fungujícími interními odkazy a obrázkem v datamartu.

Nabízí se 2 odlišná řešení:

1. Vytvoření nové části EBIE, do které uživatel přes webový prohlížeč vstoupí a zvolí soubor (pro datamarty i obrázek), který chce zobrazit. Server požadovanou stránku v případě potřeby přeloží do formátu html a zobrazí.

   - Výhody
     - Přívětivé uživatelské prostředí
   - Nevýhody
     - Nelze používat společně s ostatními generátory
     - Narušení stávající struktury portálu funkcionalitou určenou pouze pro omezenou skupinu uživatelů
     - Nemožnost rychlého testování změn v backendu, který generuje html soubory

2. Vytvoření generátoru (v tomto kontextu by přesnější označení bylo „zobrazovače“), který na pozadí požadovaný soubor nahraje na server EBIE a uživateli v prohlížeči otevře novou stránku s tímto souborem

   - Výhody
     - Možnost napojení na jiné generátory (pokud by uživatel např. vytvořil generátorem novou target tabulku, mohl by si vygenerovaný soubor rovnou zobrazit v prostředí EBIE, bez nutnosti zadávání dalšího příkazu)
     - Toto řešení je rychlejší (i v kontextu bezpečnostních opatření zmíněných později), uživateli stačí zadat pouze jeden příkaz v příkazové řádce
   - Nevýhody
     - Ne všichni uživatelé preferují používání příkazové rádky
     - Nutnost zabezpečení spojení se serverem (při přístupu přes prohlížeč se uživatel musí přihlásit a tím ověřit, pro toto řešení bychom museli vymyslet vlastní způsob ověření)

Na základě nalezených výhod a nevýhod jsem se rozhodl pro druhé řešení, které bude spojitelné s ostatními generátory.

Navrhuji vytvořit generátor, který zadaný soubor nahraje na serveru ebievyvoj a vytvořenou stránku následně otevře v prohlížeči uživatele. Tato stránka bude mít omezený čas platnosti, po jeho uběhnutí bude ze serveru odstraněna. (tím se zamezí zaplňování úložného prostoru serveru a zároveň bude zachována přehlednost existujících dat.)

Bezpečnostní problémy

Tohoto řešení se ovšem týkají 2 problémy v oblasti zabezpečení:

1. Komunikace se serverem - viz nevýhody 2. řešení

2. Stránka je přímo vytvořena uživatelem, bez kontroly a bez uložení v repozitáři (jediná informace budou logy v ebie-vyvoj). Hrozí tedy např. xss (Cross Site Scripting) útok, kdy uživatel na stránku umístí škodlivý javascript kód a odkaz pošle jinému uživateli, který je takto skrytě napaden.

Tyto hrozby navrhuji vyřešit následujícími způsoby:

1. Před vytvořením stránky s náhledem na vytvořený obsah se uživateli otevře stránka s unikátním kódem, který bude nutné vložit do terminálu se spuštěným generátorem. Stránka bude vyžadovat přihlášení uživatele a server bude ověřovat platnost zadaného kódu před nahráním vybraného souboru.

   Tato ochrana zamezí vkládání obsahu na server nepovolaným osobám a také znemožní zahlcení serveru velkým množstvím nevyžádaných souborů v krátkém časovém úseku

2. Vytvořená stránka bude přístupná pouze uživateli, který ji vytvořil. Ostatní uživatelé tuto stránku nebudou moci zobrazit, budou tak chráněni před potenciálním škodlivým kódem na této stránce.

   Možnost náhledu stránky bude navíc k dispozici jen na testovacím serveru EBIE, na produkčním serveru tato funkce nebude umožněna.

2.1.5 Nahrání změn na portál EBIE

V sekci 1.6.4 jsem ukázal problémy spojené s nahráním změn na testovací server EBIE. Tyto problémy vyřeší zavedení „Průběžné integrace“ (anglicky „Continuous integration“), kdy budou nové změny v repozitáři ebie-content automaticky nasazeny na testovací server.

Je třeba zajistit, aby se proces týkal pouze větve master-vývoj a ostatní vývojové větve byly ignorovány.

2.2 Generátory

V této sekci se zaměřím na návrh jednotlivých generátorů.

2.2.1 Úprava šablon

Generátory budou v dokumentech opakovaně kontrolovat nebo měnit specifická místa (např. při aktualizaci tabulky datamartu). Aby bylo možné tyto operace vykonávat automaticky, dotčená místa musí splňovat následující podmínky:

- Jednoznačná identifikovatelnost
- Perzistence (pokud by se během vývoje změnila, nebylo by možné provádět aktualizace těchto míst generátorem)
- Žádný vliv na vzhled dokumentu

Navrhuji tato místa označit komentáři tímto způsobem:

```
// Sekce
...
...
// Konec sekce
```

Komentáře nezasahují do vzhledu výsledné stránky (z AsciiDoc komentářů se ani negenerují html komentáře), pro uživatele bude navíc tento způsob jednoduchý na používání - stačí neupravovat stávající komentáře (toto pravidlo zdůrazním v dokumentaci repozitáře). Vyjma komentářů tak uživatel bude moci dokument libovolně upravovat.

I přes upozornění může nastat situace, kdy je komentář omylem odstraněn, přidám proto do generátorů kontrolu přítomnosti komentářů.

Pro vývoj generátorů je tento přístup výhodný. Generátory vytvářející nový obsah i aktualizující stávající mohou k místům označeným komentáři přistupovat stejným způsobem.

2.2.2 Generátor target tabulek

Target tabulky obsahují mnoho části, které je možné vyplnit automatickým procesem (viz 1.4.7) - název, související target tabulky a html tabulka. Díky přidání „DLM tabulky“ je možné získat i popis target tabulky a popis do info.json.

Tento generátor po zadání jména target tabulky stáhne z řádků DLM schéma tyto hodnoty:

- Cílový systém
  - Sloupec
  - Datový typ
- Zdrojový systém
  - Databáze
- Metadata
  - Obchodní název (popis)

Hodnoty bude brát z těch řádků, ve kterých bude Tabulka v části Cílový systém odpovídat zadanému názvu target tabulky.

Názvy souvisejících target tabulek generátor získá také z DLM schéma - jsou to všechny Tabulky v části Cílový systém, které mají stejnou entitu jako vytvářená target tabulka.

Popis se získá z „DLM tabulky“, sloupce Popis dlouhý, popis do info.json ze sloupce Popis. Hledaný řádek je ten, kde Název tabulky odpovídá zadanému názvu.

V šabloně tables.adoc budou jednoznačně určitelná místa, na která se získané informace doplní:

- Název - „// Nazev tabulky“
- Popis - mezi „// Popis tabulky“ a „// Konec popisu“
- Související tabulky - mezi „// Odkazy“ a „// Konec odkazu“
- Html tabulka - oblast mezi „// Tabulka“ a „// Konec tabulky“

Generátor tato místa vyhledá pomocí regulárních výrazů a následně je přepíše (stejný postup je i u všech ostatních generátorů, a proto jej nebudu znovu uvádět).

2.2.3 Generátor datových tržišť

Do nových datamartů je možné generovat jejich diagramy a html tabulku.

Tabulka obsahuje dimenzionální a faktové tabulky, které jsou v materializovaném pohledu daného datového tržiště. Každá tabulka obsahuje názvy svých sloupců společně s jejich popisem.

Názvy sloupců odpovídají sloupci Sémantika v části Sémantická vrstva v DLM schéma. Popis pak je Obchodní název z části Metadata.

Diagramy

Pro Asciidoctor je dostupné rozšíření „Asciidoctor-Diagram“, které umožňuje v AsciiDoc dokumentu vytvářet textově popsané diagramy (tato knihovna podporuje širokou škálu programů, např. Erd, GraphViZ, PlantUML, UMLet aj.) [23].

Na generování diagramů, které obsahují informace o tabulkách a jejich vzájemných vazbách, je nejvhodnější nástroj PlantUML.

Používání tohoto rozšíření je jednoduché a intuitivní, jako ukázku uvedu následující kód, ze kterého vznikne diagram 2.3: (text v závorce říká knihovně, že má použít program PlantUML a vytvořený obrázek typu png pojmenovat test)

```
[plantuml, test, png]
....
```

```
class A~{
    col1
    col2
}

class B {
    col1
    col2
}

A~---> B
....
```

Obrázek 2.3: Ukázka diagramu vygenerovaného knihovnou Asciidoctor Diagram a programem PlantUML

PlantUML podporuje konfiguraci výsledného vzhledu diagramů pomocí konfiguračního souboru. Navrhuji vytvořit takový soubor, který bude upravovat vzhled všech vytvářených diagramů. Jejich podobu tak bude možné snadno upravovat na na jediném místě.

Datamarty se vytváří z šablony data_marts.adoc. Doplňovaná místa budou:

- Název - „// Nazev datamartu“
- Diagram - mezi „// Image datamartu“ a „// Konec image datamartu“
- Html tabulka - Od řádku „<!– Tabulka –>“ až do konce dokumentu

U tabulky se komentář bude nacházet v oblasti html kódu, proto se jedná o komentář ve formátu html.

Pro nově vytvořený datamart je také možné vytvořit záznam do správného souboru info.json.

2.2.4 Generátor pojmů ze sémantické vrstvy

Pro pojmy ze sémantické vrstvy lze automaticky vytvářet jejich html tabulku s daty z DLM schéma. Tato tabulka se sestává ze dvou záložek s atributy:

- Business atributy
  - Sémantický název - Sémantická vrstva, Sémantika
  - Business popis - Metadata, Obchodní název
- Technické atributy
  - Sémantický název - stejně jako u Business atributů
  - Mapování v targetu - Cílový systém, Sloupec
  - Zdrojová tabulka - Cílový systém, Tabulka

Z DLM schéma budou vybrány řádky, ve kterých Tabulka v části Cílový systém odpovídá zadané target tabulce, pro kterou hledáme informace ze Sémantické vrstvy.

Při generování nového pojmu budou v šabloně semantic_layer.adoc nahrazena tato místa:

- Název - komentář „// Nazev pojmu“
- Html tabulka - Od řádku „<div class="nav-tabs-custom»“ do konce dokumentu

  Pro nový pojem bude také vytvořen záznam v souboru info.json.

2.2.5 Generátory aktualizující obsah

Tyto generátory budou zajišťovat aktualizaci stávajících souborů. Kromě názvů dokumentu jsou aktualizována stejná místa, která upravují generátory vytvářející nový obsah:

- Target tabulky:
  - Popis (v info.json i v target tabulce)
  - Související tabulky
  - Html tabulka
- Datamarty
  - Diagram

  - Html tabulka

- Pojmy ze sémantické vrstvy

  - Html tabulka

Aktualizace popisů target tabulek nebude (dočasně) povolena ve výchozím nastavení. Je to z toho důvodu, že obsah „DLM tabulky“ zatím není ve finální podobě (viz 2.1.1), je tedy možné, že uživatel bude muset popisy ručně upravit, a proto je nebude chtít přepsat hodnotami z „DLM tabulky“.

2.2.6 Zobrazení v prostředí ebie-vyvoj

Generátor zobrazující dokument v prostředí serveru ebie-vyvoj bude fungovat na základě návrhu 2.1.4.

Na ebie-vyvoj bude možné zobrazit html a AsciiDoc soubory, AsciiDoc soubory budou přeloženy do html před odesláním. Uživatel tak bude moci otestovat např. změnu v backendu (ve složce .html5), aniž by tuto změnu musel zvlášť nahrávat na testovací server.) Vygenerované html soubory budou také přímo dostupné v lokálním adresáři ve složce test.

Html kód bude posílán jako řetězec, server jej tak bude moci snadno uložit do databáze a při zobrazení přidat do šablony pro náhledovou stránku.

V případě posílání datamartu generátor sám nahraje i diagram.

Základní úpravy RoR aplikace EBIE budou následující:

- Vytvoření nového kontroleru, který bude mít na starost komunikaci při procesu zobrazování stránky

- Přidání výjimky do ověřování v aplikačním kontroleru za účelem umožnění nahrání souboru generátorem na server

  - Tuto podmínku bude možné zakázat proměnnou prostředí (Ruby ENV) na jiném serveru než testovací serveru EBIE

- Vytvoření nového modelu, který bude ukládat html kód, cestu k diagramu aj.

2.3 Volba technologií

Pro navržené úpravy a rozšíření je nezbytné zvolit vhodné technologie pro implementaci.

2.3.1 Generátory

Jako první možnost pro implementaci generátorů se nabízí jazyk Ruby, který je v prostředí ebie-content již používán - gem Asciidoctor. V Ruby je navíc napsaný i aplikační server ebie. Uživatelé přidávající obsah do ebie-content by tak nemuseli instalovat další programovací jazyk.

Je nutné ověřit, zda má Ruby potřebné knihovny (gemy) pro navržené procesy:

- Čtení informací z Google Docs dokumentu - Ruby obsahuje několik gemů, které to umožňují. Nejvíce mě zaujal „google_drive“, aktivně udržovaná knihovna umožňující čtení/zápis z Google Drive/Docs [24].
- Přístup k databázi s Datovými tržišti - na výběr je ze dvou populárních knihoven:
  - ActiveRecord - velmi používaný, protože je součástí Ruby on Rails. To je zároveň důvod, proč není pro potřeby ebie-content ideální; ActiveRecord je robustní knihovna závisející na dalších RoR knihovnách a její nasazení mimo prostředí RoR může přinést komplikace
  - Sequel - poskytuje jednoduchý, flexibilní a výkonný SQL databázový přístup pro Ruby [25]. Pro plánované využití je plně vyhovující - umožňuje přímý přístup do databáze, okamžité používání a velké množství přídavných „adaptérů“, které rozšiřují funkčnost této knihovny.

    Pro přístup k PostgreSQL databázím je doporučeno rozšíření:

    * sequel_pg - Sequel adaptér zrychlující SQL SELECT operace [26].
- Správa přihlašovacích údajů na dwh3_access - dotenv je „Ruby gem sloužící k nahrávání proměnných prostředí“ (environment variables) ze souboru .env [27]. Tyto proměnné jsou po spuštění programu přístupné přes třídu ENV, která patří mezi standardní knihovny Ruby.

  Použitím dotenv bude zajištěno připojení generátorů k dwh3_access.
- Automatické generování diagramu pro datamarty - gem Asciidoctor-diagram, viz 2.2.3
- Nahrání souboru na server EBIE - na výběr je z velkého množství knihoven. Nutná podmínka je možnost poslání souboru.

  Zvolil jsem gem rest-client, Jednoduchý HTTP a REST klient pro Ruby [28], pro jeho snadné používání a podporu posílání souborů (ukázka zaslání souboru:
  „RestClient.post ’https://example.com/data’,
  myfile: File.open(’/path/to/image.png’, ’rb’)“).

- Otevření stránky v prohlížeči - tuto funkcionalitu řeší gem Launchy, pro otevření stránky stačí zadat „launchy URL“.
- Testování vytvořených procesů - pro tvorbu testů, ověřujících funkčnost vytvořených procesů, jsem vybral gem Rspec, který umožňuje jednoduše přidávat a spouštět testy.
- Vytváření unikátních ověřovacích kódů - Ruby obsahuje systémovou knihovnu SecureRandom pro vytváření bezpečných náhodných řetězců.
- Tvorba generátorů - gem Thor je jednoduchý a efektivní nástroj pro vytváření sebe-dokumentujících rozhraní příkazové řádky [29]. Při jeho použití bude možné:
  - Spouštět generátory z příkazové řádky
  - Jednoduchým způsobem vytvořit zobrazitelnou nápovědu k používání
  - Snadno interagovat s uživatelem (např. dotazem na přepsání souboru atd.)
  - Přidat generátorům volitelné argumenty, které ovlivní jejich funkčnost

Pro podporu vývoje generátorů jsem přidal gem byebug, jednoduchý, funkčně bohatý debugger pro Ruby [30].

Z předchozího seznamu je očividné, že jazyk Ruby obsahuje knihovny na všechny požadované procesy. Proto jsem jej zvolil jako implementační jazyk pro generátory.

2.3.2 Kontrola verzí knihoven

V předchozí sekci jsem navrhl přidání značného množství knihoven (celkem 11) do repozitáře ebie-content. Oproti původnímu počtu 3 to je pro uživatele velké navýšení počtu instalovaných knihoven. Problém s aktualizacemi těchto knihoven (viz sekce 1.6.1) se s vyšším počtem potenciálně aktualizovaných knihoven ještě zhorší.

Pro řešení komplikací spojených s instalací gemů a udržováním jejich verzí existuje gem bundler [31]. Jedná se o nejpoužívanější Ruby knihovnu, v době psaní této sekce (duben 2018) má zhruba 237 000 000 stažení [32].

Bundler využívá soubory Gemfile a Gemfile.lock. Gemfile obsahuje všechny používané gemy společně s jejich verzemi, které je možné detailně definovat (např. označení „>=“ v řádku „gem 'dotenv', '>= 2.3'“ říká, že používáme gem dotenv ve verzi minimálně větší rovno 2.3.0).

Gemfile.lock „uzamyká“ používené verze gemů.

Příkaz „bundle install“ ověří, že knihovny uvedené v Gemfile existují a jejich verze nejsou v neřešitelné kolizi (gem A závisí na gemu B verze >= 1, gem C závisí na gemu B verze < 0.5), a poté chybějící gemy nainstaluje. Aktuální používané verze jsou zapsány do souboru Gemfile.lock.

Při přístím spuštění „bundle install“, potenciálně jiným vývojářem, se bundler pokusí nainstalovat ty verze knihoven, které jsou uvedeny v Gemfile.lock. Tak je zajištěno, že všichni vývojáři užívají stejné verze všech gemů.

Navrhuji proto rozšířit vybrané knihovny o bundler, který zajistí jednoduché udržování a instalování verzí přítomných knihoven.

2.3.3 Přístup k DLM

Vybraný gem google_drive zajistí přístup ke Google Doc tabulkám, nikoli přímo čtení z DLM. Proto je nutné podniknout ještě několik kroků (následující popis vychází z oficiální dokumentace Googlu [33]):

- Vytvoření vývojářské aplikace v Google API konzoli
- Aktivování Google Sheets API
- V Google API konzoli vytvoření „credentials“ pro OAuth client ID s aplikací typu Other
- Uložení nových credentials (jedná se o klíče „client_id“ a „client_secret“) do souboru v repozitáři ebie-content (v prostředí ebie se bude jmenovat app_keys.json)
- Odkázání se na tento soubor při používání knihovny google_drive

Při prvním použití knihovna uživatele požádá o autorizaci a povolení rozsahu přístupu k požadovaným Google službám (anglicky scope). Tím aplikace získá tvz. „refresh token“, s jehož použitím může přistupovat k DLM jménem uživatele (Google pro autentizaci používá Oauth2 standard).

Tento token je soukromý a nesmí být zveřejněn. Navrhuji proto každému uživateli vytvořit lokální kopii souboru s aplikačními klíči, která bude ignorována gitem a do které bude zapsán získaný refresh token. Nebude tak hrozit, že uživatel svůj refresh token omylem zveřejní.

2.3.4 Čas poslední změny v target tabulkách

Díky ing. Jakubu Krejčímu, který je vývojářem datového skladu, vznikl v databázi dwh3_target nový materializovaný pohled na časy poslední úpravy target tabulek.

Server ebie-api využívá pro připojení k datovému skladu gem Sequel, který bude použit i pro nové generátory. Touto knihovnou zajistím načtení dat

z DWH3.

Seznam časů změn pro jednotlivé target tabulky bude uložen v databázi EBIE. Pro aktualizaci těchto záznamů je nutné na serveru EBIE vytvořit opakující se proces, který bude s využitím ebie-api požadované údaje získávat a ukládat.

K těmto účelům slouží na Unixových operačních systémech Cron. Cron je systémový démon, používaný na vykonávání úloh na pozadí v určený čas [34].

Pro snadné napojení Ruby aplikací na Cron slouží knihovna Whenever, Ruby gem, který poskytuje čistou syntaxi pro psaní a nasazování Cron jobs [35]. Ukázka použití:

```
every 5.minutes do
   rake 'task:test'
end
```

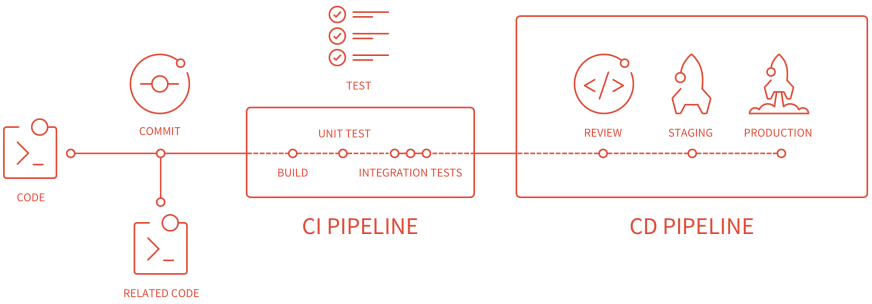
Rake úkoly se běžně využívají pro vykonávání automatických činností v Ruby, popř. RoR, kde mohou přistupovat k existujícím modelům. Navrhuji vyvořit „úkol“, který přes ebie-api načte aktuální časy změn target tabulek a v případě změny je zapíše do databáze na EBIE.

2.3.5 Průběžná integrace

Server Gitlab, na kterém je uložen repozitář ebie-content, má integrovanou podporu pro CI (Continuous Integration) [36], nabízí se tedy jako vhodné řešení. Na obrázku 2.4 jsou vidět jednotlivé části tohoto procesu.

Pro nastavení se používá konfigurační soubor .gitlab-ci.yml, který definuje vykonávané činnosti. Umožňuje také omezit integraci na vybrané git větve, tím je splněna nutná podmínka používání (viz 2.1.5).

Vykonávání nadefinovaných operací, stejně jako sledování změn v repozitáři, zajišťuje program GitLab Runner. Je dostupný i ve verzi Debian, což je operační systém testovacího serveru EBIE.



Obrázek 2.4: Diagram zobrazující proces Gitlab CI [36]

Kapitola 3

Implementace

V této kapitole popíši a ukáži, jakým způsobem jsem navržené úpravy implementoval.

3.1 Vytvoření generátorů

Generátory se nachází v souboru „content.thor“, kde definuji jejich název, atributy, dokumentaci a prováděné operace, delegované na pomocné třídy. Tento soubor poskytuje přehledný a ucelený pohled na přítomné generátory.

Příkaz „thor content:help“ vypíše základní dokumentaci generátorů (popisy jsem kvůli jejich délce přesunul na samostatný řádek a některé i zkrátil):

```
Commands:
  thor content:check_dlm_consistency
    # Zkontroluje, zda jsou DLM schéma a DLM tabulky konzistentní
  thor content:generate_datamart NAME
    # Generuje nový datamart
  thor content:generate_semantic TARGET_TABLE_NAME
    # Vytvoří nový adoc soubor pro pojem ze sémantické vrstvy
  thor content:generate_target_table NAME
    # Generuje novou target tabulku
  thor content:help [COMMAND]
    # Describe available commands or one specific command
  thor content:missing_comments
    # Zkontroluje přítomnost komentářů ve všech souborech
  thor content:print_datamarts
    # Zobrazí názvy všech datamarty, které jsou v~datovém skladu
  thor content:show_on_ebie FILE_NAME
    # Ukáže zadaný soubor na ebie-vyvoj
  thor content:sort_info_json FILE_PATH
    # Seřadí obdržený soubor info.json podle klíče 'title'
```

```
  thor content:update_datamarts
    # Aktualizuje všechny přítomné datamarty
  thor content:update_semantics
    # Aktualizuje všechny přítomné pojmy ze Sémantické vrstvy
  thor content:update_target_tables
    # Aktualizuje všechny přítomné target tabulky podle DLM
```

Pokud je příkaz zadán špatně, vypíše se vysvětlující chybová hlášení a popis správného použití:

```
> thor content:generate_target_table
ERROR: "thor generate_target_table" was called with no arguments
Usage: "thor content:generate_target_table NAME"
```

Pro každý generátor je možné zobrazit i detailní popis, který ukazuje také volitelné argumenty:

```
> thor content:help generate_datamart
Usage:
  thor content:generate_datamart NAME

Options:
  -d, [--description=DESCRIPTION]
     # Popis nově vytvořeného datamartu pro info.json
  -t, [--test], [--no-test]
     # Nenabízet zobrazení na ebie-vyvoj - pro automatické testy

Generuje nový datamart
```

Na obrázku 3.1 jsou vidět nově přidané soubory, nutné pro správné fungování generátorů.

Generátory je možné rozdělit do tří skupin:

1. Generátory vytvářející nový obsah
2. Generátory aktualizující existující obsah
3. Pomocné generátory

V souboru README.adoc, repozitáře ebie-content, jsem pro uživatele generátory dokumentoval. Z této dokumentace vycházím v následujících sekcích.

3.1.1 Řešení selhání generátorů

Spuštění generátoru není povoleno, jestliže uživatel nemá nainstalované všechny požadované knihovny ve správných verzích. V takovém případě je vypsáno upozornění a vhodný postup pro nápravu:

- ebie-content
  - content_logic
    - asciidoc_html_generator.rb Třída konvertující .adoc soubory do do html
    - content_helper.rb Modul s podpůrnými funkcemi pro generátory.
    - datamart.rb..... Třída reprezentující databázové schéma pro datamarty
    - datamart_helper.rb..... Modul s podpůrnými funkcemi pro generování datamartů
    - dlm_reader.rb..... Třída zajiťující načítání dat z DLM
    - dlm_support.rb .... Modul obsahující podpůrné třídy pro operace s DLM
    - dlm_worker.rb..... Třída zajišťující operace spojené s DLM
    - dwh_worker.rb..... Třída zajišťující operace spojené s DWH3
    - semantic_helper.rb..... Modul s podpůrnými funkcemi pro generování pojmů ze sémantické vrstvy
    - target_table_helper.rb ..... Modul s podpůrnými funkcemi pro generování target tabulek
  - content.thor

Obrázek 3.1: Podpůrné třídy a moduly pro generátory v content.thor

```
> gem uninstall dotenv
> thor content:print_datamarts
Could not find gem 'dotenv (~> 2.4)' in any of the gem sources
listed in your Gemfile.
Run `bundle install` to install missing gems.
```

Některým generátorům hrozí selhání při komunikaci s DLM, DWH3 nebo ebie-vyvoj, ať už z důvodu výpadku internetu, špatné konfigurace či jiných komplikací. Ruby v takovém případě velmi podrobné vypíše tzv. „backtrace“ popisující error.

Délka tohoto výstupu může dosahovat desítek řádku, v příloze B.2.2 ukazuji výpis při selhání připojení k databázi v datovém skladu.

Pro uživatele, který není zvyklý pracovat s Ruby, v takovém případě není snadné zjistit příčinu chyby. Ve všech dotčených generátorech proto v případě chyby ve spojení s DLM, DWH3 nebo ebie-vyvoj chybu zachytávám a vypisuji jen krátké chybové hlášení, bez detailně rozepsaných tříd a metod.

Jako ukázku uvedu opět případ selhání spojení s databází:

```
Přístup k~databázi selhal z~důvodu: 'PG::ConnectionBad: could
not connect to server: Connection refused
        Is the server running on host xxx and accepting
        TCP/IP connections on port xxx?
```

Pro vývojáře a uživatele, kteří budou chtít vidět plný výpis chyby, jsem generátorům přidal přepínač „-t“. Tento přepínač využívají i vytvořené testy vyžadující nezachycenou chybu.

3.1.2 Generátory vytvářející nový obsah

Do této skupiny se řadí generátory target tabulek, datamartů a pojmů ze sémantické vrsty. Uživatel je využije při rozšiřování stávajícího obsahu EBIE. Po vytvoření nového souboru je nabídnuto jeho zobrazení na ebie-vyvoj.

Přepínač „-t“ pro tyto generátory kromě zobrazení backtrace errorů navíc uživateli nenabídne zobrazení nového souboru v prostředí ebie-vyvoj. Tím je umožněno automatické provádění testů, bez nutnosti zásahu uživatele (viz4.1.

Může se stát, že vygenerované informace nebudou úplné (např. chybějící Popis v tabulce datamartu). V takovém případě je ideálním řešením doplnění informací do DLM a následné spuštění aktualizujícího generátoru (viz 3.1.3).

Pokud soubor se stejným názvem existuje, uživatel si může vybrat pokračování nebo ukončení generátoru (zadáním „y/n“).

V případě, že došlo k úpravě šablony a byl z ní odebrán některý z komentářů nutných pro správné fungování generátorů (viz 2.2.1), vypíše generátor informaci o chybějícím komentáři:

```
"VAROVÁNÍ: V~šabloně souboru /home/ebie-content/ebie_content/
dokumentace/tables/t_stud_studium.adoc chybí komentáře označující
Tabulku"
```

Target tabulky

Příkaz „thor content:generate_target_table TARGET_TABLE_NAME“ vytvoří novou target tabulku. Dokumentovaná funkčnost a diagramy s popisem tohoto generátoru jsou dostupné v příloze B.1.1.

Do tabulky je doplněn její název, popis, odkazy na související tabulky a html tabulka. Do správného souboru info.json je přidán nový záznam společně s popisem této tabulky.

V části Odkazy na související tabulky jsou vytvořeny odkazy na ty tabulky, které jsou na EBIE přítomné, ostatní tabulky jsou vypsány bez odkazu.

Datamarty

Nový datamart je vytvořen příkazem „thor content:generate_datamart DATAMART_NAME“. V příloze B.1.2 je dokumentováno použití společně s diagramy, popisujícími tento proces.

Vytvořený datamart obsahuje svůj název, diagram ve vhodném formátu pro PlantUML a html tabulku. Do patřičného info.json je doplněn záznam o tomto datamartu. Pokud byl generátor spuštěn s atributem „-d=DESCRIPTION“, do info.json je doplněn i popis.

Pojmy ze sémantické vrstvy
Pojem ze sémantické vrstvy přidá „thor content:generate_semantics TARGET_TABLE_NAME“. Jeho podrobnější dokumentace je k nalezení v příloze B.1.3.

Vzniklý soubor má doplněný název a html tabulku. Podobně jako u datamartu je doplněn do info.json s možností zadání popisu přes atribut „-d=DESCRIPTION“.

3.1.3 Generátory aktualizující obsah

Tyto generátory aktualizují zvolenou skupinu souborů (target tabulky, datamarty nebo pojmy ze sémantické vrstvy).

Pokud dojde k úpravě souboru, jeho název je vypsán. Dále je vypsána informace o souborech, které nebyly nalezeny ve zdrojových systémech (např. soubor s target tabulkou, jejíž název se nenachází v DLM schema).

Generátory aktualizující obsah jsou užitečné pro zajištění konzistence informací v EBIE a DLM či DWH3 a rychlé zpropagování změn z těchto zdrojových systémů do EBIE.

Pokud v některém z aktualizovaných souborů chybí komentáře pro označení míst, uživatel je o tom informován:

```
"VAROVÁNÍ: V~souboru /home/ebie-content/ebie_content/dokumentace/
tables/t_stpl_forma_studia.adoc chybi komentare oznacujici Odkazy
(Souvisejici tabulky)"
```

Target tabulky
Target tabulky je možné aktualizovat příkazem
„thor content:update_target_tables“. Další dokumentace se nachází v příloze B.1.4.

Při spuštění je provedena aktualizace odkazů na související target tabulky a html tabulky. Při zadání přepínače „-d“ jsou aktualizovány také popisky.

Datamarty
Aktualizace datamartů se spouští příkazem
„thor content:update_datamarts“, který je podrobněji dokumentován v příloze B.1.5.

Aktualizace se týká diagramu a html tabulky.

Pojmy ze sémantické vrstvy
Pojmy ze sémantické vrstvy lze aktualizovat zadáním
„thor content:update_semantics“, více informací o tomto procesu viz příloha B.1.6.

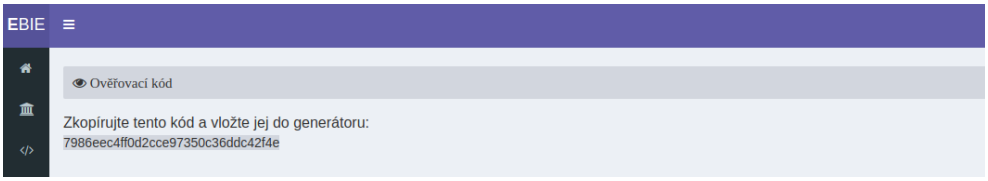
Generátor aktualizuje html tabulku.

3.1.4 Zobrazení souboru v prostředí ebie-vyvoj

Náhled stránky v prostředí ebie-vyvoj
Zobrazení souboru v prostředí ebie-vyvoj je možné příkazem „thor content:show_on_ebie FILE_PATH“. Další dokumentace je k dispozici v příloze B.1.7

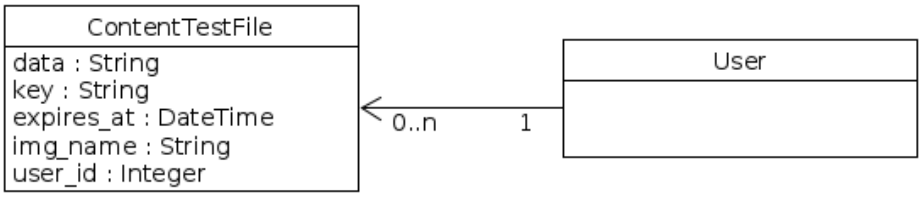
Tento generátor je možné používat samostatně nebo jako doplněk generátorů nového obsahu (viz 3.1.2). Při samostatném spuštění je možné zobrazit soubory typu .adoc nebo .html.

Podoba stránky s ověřovacím kódem, který je nutný pro nahrání dokumentu na testovací server ebie, je na obrázku 3.2.



Obrázek 3.2: Ukázka stránky s ověřovacím kódem pro generátor show_on_ebie

Pro zajištění této funkčnosti jsem upravil ebie-source. Nová entita ukládá html kód zobrazovaného souboru, ověřovací klíč, čas expirace, pro datamart název diagramu a id uživatele, kterému vytvořená entita patří. Na diagramu 3.3 je tato entita znázorněna společně s její vazbou k uživateli.



Obrázek 3.3: Entita ContentTestFile a její vztah k uživateli

Každému diagramu je na serveru nastaveno právo na čtení pro všechny uživatele v operačním systému, bez toho by webový server (nginx) nemohl diagramy zobrazovat (je totiž spuštěn pod jiným uživatelem, než aplikační server ebie-vyvoj):

```
FileUtils.chmod 'o=r', file
```

Aby bylo možné správně zobrazit obrázek diagramu, musel jsem upravit konfiguraci webového serveru. Důvodem bylo nastavení pro soubory ve složce public/assets, kam diagramy ukládám:

```
location /assets/ {
  expires     max;
  add_header  Cache-Control public;
}
```

„Hodnota max u expires nastaví obrázkům expiraci na 31. prosince 2037 a Cache-Control na 10 let“ [37]. To znamená, že pokud uživatel opakovaně provede drobné úpravy diagramu stejného datamartu a nahraje jej na server, stále se mu bude zobrazovat první zaslaný obrázek.

Přidal jsem proto toto nastavení, které nastaví Cache-Control na „no-cache“:

```
location ~* assets\/.*\.png$ {
  expires -1;
}
```

Díky těmto nastavením a úpravám se např. diagram qnewline dm_vysledky_studentu_v_predmet
zobrazí jako na obrázku 3.4. Použitý příkaz byl:

```
thor content:show_on_ebie ebie_content/studium/reports/
data_sources/data_marts/dm_vysledky_studentu_v_predmetech.adoc
```

3.1.5 Pomocné generátory

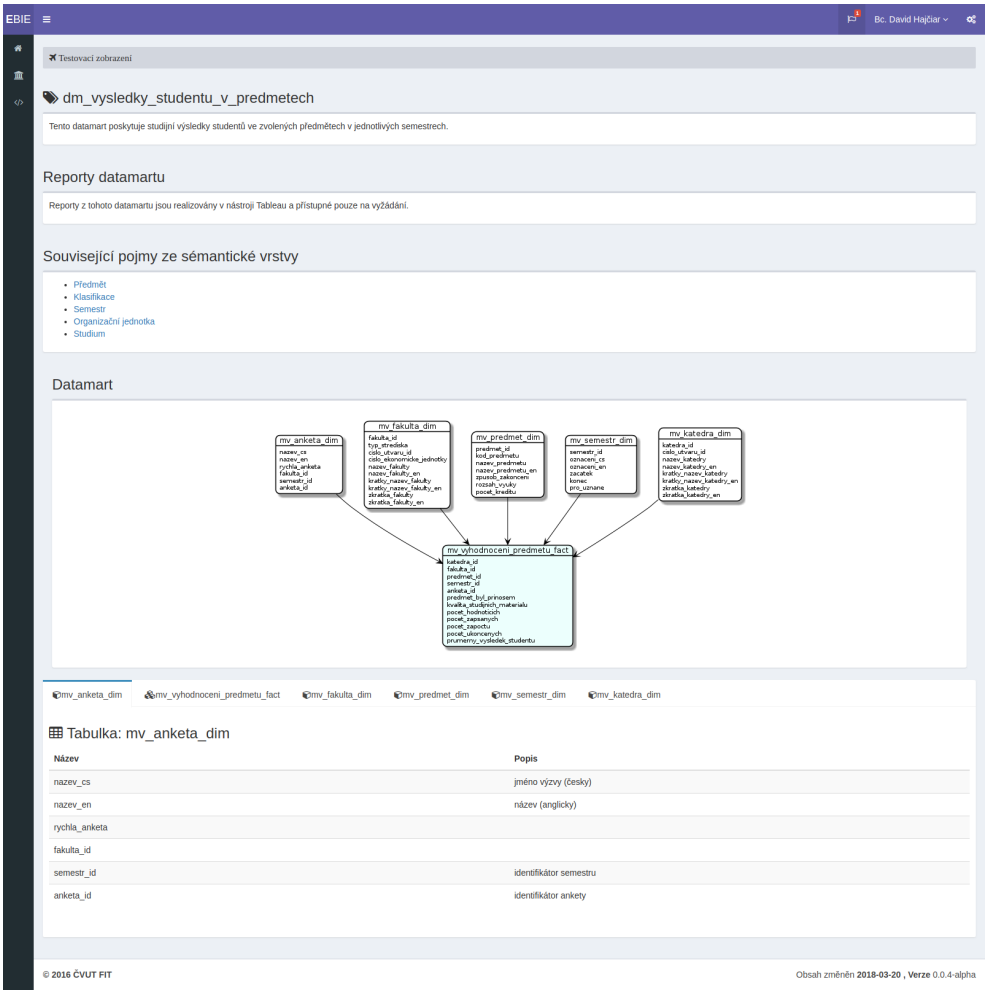
Tyto generátory jsem vytvořil za účelem podpory vývoje a usnadnění používání ostatních generátorů. Jejich podrobnější dokumentace je k dispozici jako příloha B.1.8.

Vypsání přítomných datamartů
Příkaz „thor content:print_datamarts“ vypíše datamarty, které jsou přítomny v datovém skladu. Uživatel tak může rychle a pohodlně zjistit přesný název datamartu, který chce přidat.

Kontrola konzistence DLM
„thor content:check_dlm_consistency“ vypíše názvy target tabulek, které chybí buď v DLM schéma, nebo v „DLM tabulky“. Tato informace je důležitá pro udržování konzistentních dat v DLM schéma a DLM tabulky.

Kontrola chybějících komentářů
Pro kontrolu přítomnosti komentářů v target tabulkách, datamartech a pojmech ze sémantické vrstvy jsem připravil generátor missing_comments, spouštěný příkazem „thor content:missing_comments“.

Program pro každý soubor testuje přítomnost specifických komentářů. Uživatel je informován o všech chybějících komentářích tímto způsobem:



Obrázek 3.4: Ukázka náhledu datamartu dm_vysledky_studentu_v_predmetech v prostředí ebie-vyvoj

```
"VAROVÁNÍ: V~souboru /ebie-content/ebie_content/dokumentace/
tables/t_arok_semestr.adoc chybi komentare oznacujici Tabulku"
```

Seřazení souboru info.json

Příkazem „thor content:sort_info_json FILE_PATH“ je možné seřadit zadaný soubor info.json.

Tento generátor je vhodné použít při ruční úpravě některého souboru info.json, navíc jej lze využít pro všechny sekce ebie. (To se v tuto chvíli vyplatí především pro ontologický katalog, který čítá 25 entit.)

Řazení probíhá podle klíče 'title', v případě porovnávání českých slov jsou znaky s diakritikou nahrazeny znaky bez diakritiky:

```
CZECH_CHARS = 'áčďěéíňóřšťúůýž'
ENGLISH_CHARS = 'acdeeinorstuuyz'

def sortable_word(word)
  word.downcase.tr CZECH_CHARS, ENGLISH_CHARS
end
```

3.1.6 Důležité pomocné třídy a funkce

Nyní více přiblížím důležitá a zajímavá místa v implementaci generátorů.

Načítání dat z DLM

Data z DLM načítá třída DlmReader.

Při prvním použití je vytvořen soubor .config.json, obsahující client_id a client_secret, po autorizování uživatele (viz 3.5) google_drive uloží do daného souboru nový refresh token.

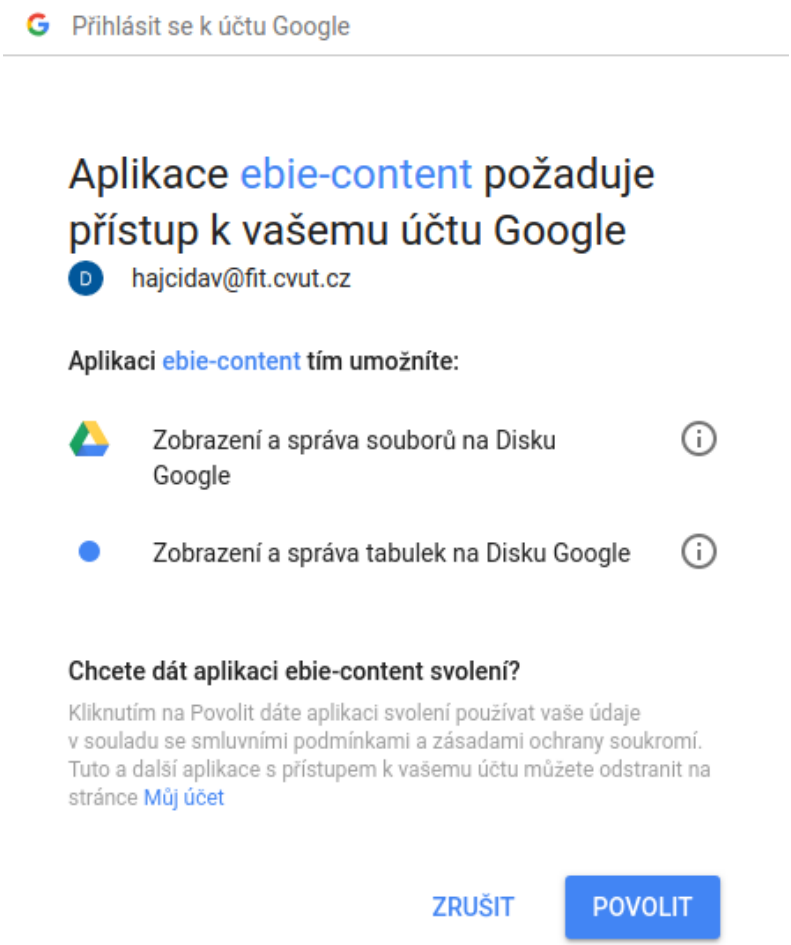
Vytvořil jsem funkce pro načítání z dat DLM schéma a „DLM tabulky“, které uloží všechny obdržené řádky do instanční proměnné třídy DlmReader (řádky jsou reprezentovány třídami DlmSchemaRow a DlmTablesRow z modulu DlmSupport).

Data z DLM jsou tak stahována a převáděna pouze jednou. Tyto funkce fungují zároveň jako iterátory (ukázka pro DLM schéma):

```
def each_useful_schema_row
  initialize_schema_rows if @dlm_schema_rows.nil?
  @dlm_schema_rows.each { |row| yield row }
end

def initialize_schema_rows
  @dlm_schema_rows = []
  (3..@dlm_schema.num_rows).each do |row|
    @dlm_schema_rows <<
     DlmSupport::DlmSchemaRow.new(whole_dlm_schema_row(row))
  end
end
```

Načítání dat pro target tabulky

Při vytváření nebo aktualizování target tabulek získávám potřebné informace funkcí „read_target_tables“ v modulu TargetTableHelper. Funkce jako výsledek vrátí hash obsahující názvy target tabulek jako klíče ukazující na pole jejich řádků (reprezentované třídou TargetTableRow v modulu DlmSupport):



Obrázek 3.5: Stránka, na které uživatel povolí generátoru přistupovat k DLM.

```
def read_target_tables(tables)
  symbols = tables.map(&:to_sym)
  {}.tap do |dlm_tables|
    each_useful_schema_row do |r|
      i = tables.find_index row.target_table
      next if i.nil?
      dlm_tables[symbols[i]] ||= []
      dlm_tables[symbols[i]] << DlmSupport::TargetTableRow.new(r)
    end
  end
end
```

Načítání pojmů ze sémantické vrstvy probíhá obdobným způsobem.

Získávání informací o datových tržištích
Spojení s datovým skladem je vytvořeno při inicializaci třídy DwhWorker na základě uživatelem zadaných hodnot v souboru .env (viz 2.3.1:

```
@DB = Sequel.connect(adapter: :postgres, user: ENV['DWH_USER'],
password: ENV['DWH_PASSWORD'], host: ENV['DWH_HOST'],
port: ENV['DWH_PORT'], database: ENV['DWH_DATABASE'])
```

Tímto spojením generátor přistupuje k dwh3_target a získává názvy schémat, tabulek a sloupců. Jako ukázku uvádím část kódu, kde sql dotazem získám názvy sloupců materializovaného pohledu a uložím je do datamartu:

```
@DB.fetch(
"SELECT a.attname
FROM pg_attribute a
   JOIN pg_class t ON a.attrelid = t.oid
   JOIN pg_namespace s~ON t.relnamespace = s.oid
WHERE a.attnum > 0
   AND NOT a.attisdropped
   AND t.relname = ?
   AND s.nspname = ?", matview, schema)  do |column|
   datamart.add_column matview, column[:attname]
end
```

Zapsání dat do dokumentu

Zapisování zpracovaného textu ve formátu AsciiDoc probíhá pro všechny generátory stejným způsobem. Na ukázku postupu uvedu funkci „fill_datamart“ modulu DatamartHelper:

```
def fill_datamart(file, name, datamart_table_adoc, diagram_adoc,
 update = false)
  adoc_file = File.read file, encoding: 'utf-8'

  missing_datamart_comments file, adoc_file, update

  adoc_file.gsub! DETECT_NAME, name unless update
  adoc_file.gsub! DETECT_DIAGRAM, diagram_adoc
  adoc_file.gsub! DETECT_TABLE, datamart_table_adoc

  File.open(file, 'wb') { |f| f.write adoc_file }
  adoc_file
end
```

Konstanty DETECT_NAME atd. obsahují regulární výraz, který v načteném souboru vyhledává správná místa, funkce „gsub!“ je poté nahradí připraveným obsahem. DETECT_NAME je definovaná takto:

```
DETECT_NAME = /\/\/ Název datamartu/
```

Přidání záznamu do info.json
Tuto funkcionalitu využívají generátory pro přidání target tabulky, datamartu a pojmu ze sémantické vrstvy, funkce „add_to_info_json“ se proto nachází ve sdíleném modulu ContentHelper.

Po nalezení správného souboru info.json je zápis proveden funkcí update_json. Ta vyhledá správnou pozici v poli uložených položek (pole se nachází pod klíčem 'list'), na kterou je zapsán nový záznam (popř. je přepsán starý záznam.)

Pro zajištění správného pořadí jsou před návratem všechny záznamy seřazeny:

```
def update_json(json, name, desc)
  position = json['list'].index {
                        |file_info| file_info['title'] == name }
  position ||= json['list'].size
  json['list'][position] = info_json_object(name, desc)
  sort_info_json_by_title json
end
```

Zobrazení souboru na ebie-vyvoj
Funkce show_on_ebie je další sdílená funkce v modulu ContentHelper. Zajišťuje otevírání stránek v prohlížeči a komunikaci s testovacím serverem EBIE:

```
  def show_on_ebie_vyvoj(file_path, rep_dir)
    Launchy.open "#{EBIE_VYVOJ}/content_tests/new"

    key = $stdin.gets.chomp
    file = read_file_for_ebie file_path, rep_dir
    params = { key: key, content_test: { data: file } }

    if datamart? file_path
      response = JSON.parse(RestClient.post
         "#{EBIE_VYVOJ}/content_tests",
         params.merge(image: datamart_image(file_path, rep_dir),
         multipart: true))
    else
      response = JSON.parse(RestClient.post
         "#{EBIE_VYVOJ}/content_tests",
```

```
        params.to_json, content_type: :json, accept: :json)
    end

    Launchy.open response['test_file_url']

    rescue StandardError => e
      p "Došlo k~chybě, generátor bude ukončen z~důvodu: '#{e}'"
  end
```

V případě dosazení řetězce „http://localhost:3000/“ do konstanty EBIE_VYVOJ bude komunikace probíhat s lokálně spuštěným serverem EBIE.

3.2 Průběžná integrace

Gitlab runner, nutný pro nahrávání dat z repozitáře na testovací server, jsem nakonfiguroval podle návodu v dokumentaci gitlabu ([38]).

Při instalaci jsem jej pojmenoval „ebie-content-ci“ a pro napojení na repozitář ebie-content jsem použil instrukce na stránce settings/ci_cd (označené jako pipelines) tohoto repozitáře - viz 3.6.

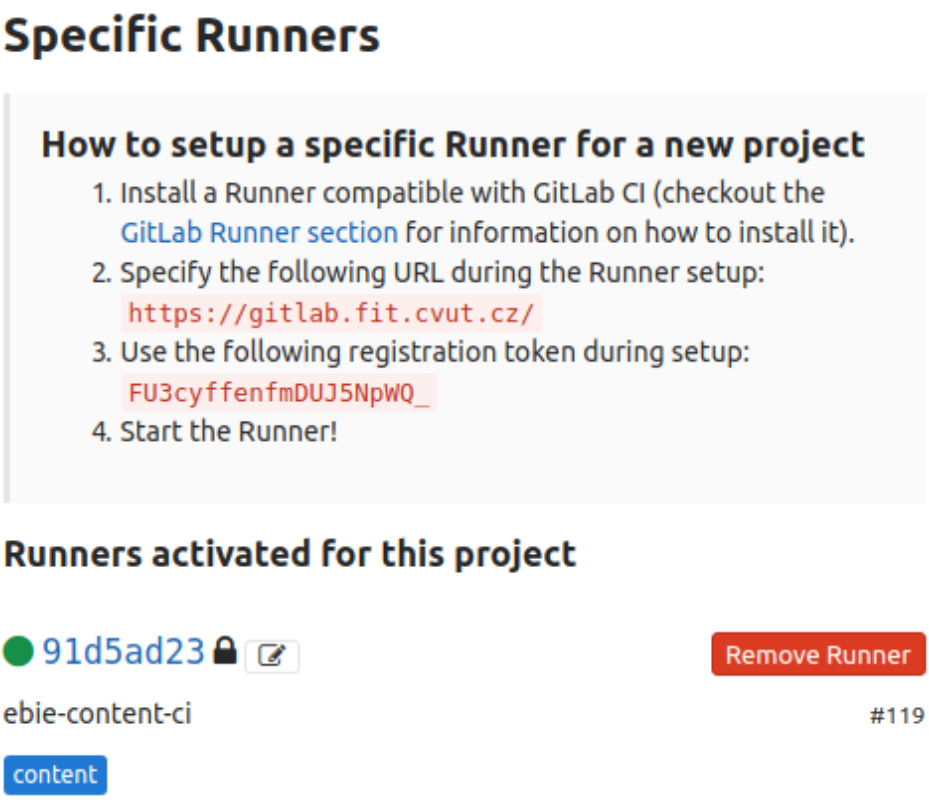
Pro zajištění průběžné integrace na server ebie-vyvoj jsem v repozitáři vytvořil konfigurační soubor .gitlab-ci.yml s tímto obsahem:

```
before_script:
  - cd /var/opt/ebie/ebie-web/
  - git submodule update --remote

content:
  only:
    - master_vyvoj
  script:
    - RAILS_ENV=production bundle exec rails generate content
    - bundle exec thin restart -P /run/ebie-web/thin.pid
```

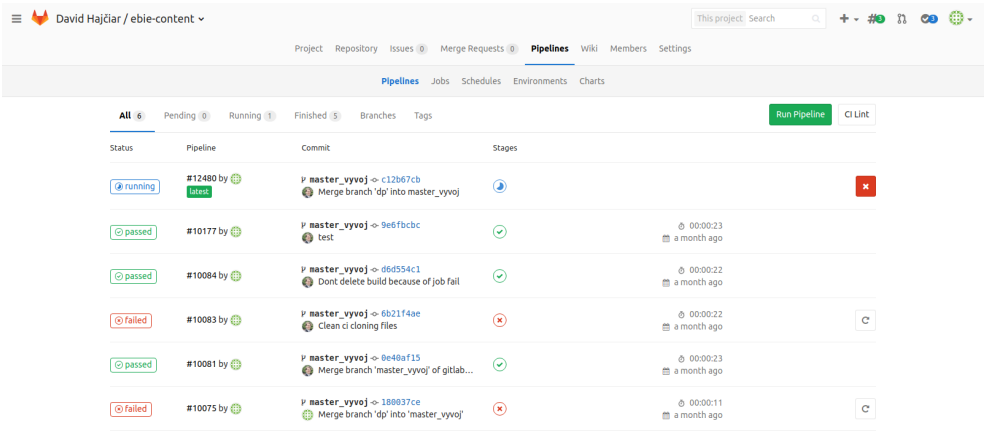
Pokud je detekována aktualizace repozitáře ve vývojové větvi master_vyvoj, gitlab runner se přesune do adresáře ebie a stáhne si aktuální data. Poté je spuštěn generátor obsahu, který z AsciiDoc souborů vygeneruje odpovídající html soubory, pro datamarty vygeneruje také diagramy. Po těchto operacích je proveden restart aplikačního serveru.

Na stránce repozitáře je možné proces nahrávání na ebie-vyvoj sledovat v záložce „pipelines“ (viz 3.7). Uživatel je tak informován, zda byla operace úspěšná nebo došlo k selhání.

U každého proběhlého pokusu o nahrání je možné zobrazit jeho detail, kde jsou uvedeny zadané příkazy, výstup a případná chybová hlášení.



Obrázek 3.6: Propojení programu Gitlab runner s repozitářem ebie-content.

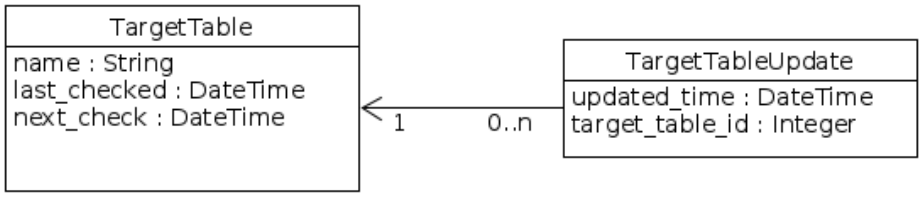


Obrázek 3.7: Přehled proběhlých automatických nahrání na ebie-vyvoj

3.3 Čas poslední změny v target tabulkách

V ebie-api zajišťují příjem požadavků a načtení dat z DWH3 tzv. „Endpointy“ (koncové body). Vytvořil jsem nový Endpoint CasyZmenTargetTabulek.rb (jako zavedená jmenná konvence jsou v tomto repozitáři používány české názvy), který přistupuje k databázi dwh3_access.

Do ebie-source jsem přidal modely TargetTable a TargetTableUpdate (viz 3.8), které reprezentují target tabulky a jejich aktualizace.



Obrázek 3.8: Entity TargetTable, TargetTableUpdate a jejich vazba

Nový záznam TargetTableUpdate je vytvořen vždy, kdy se čas poslední aktualizace změní.

Za pomoci knihovny Whenever jsem definoval spouštění „rake tasku“ pro aktualizaci časů změn:

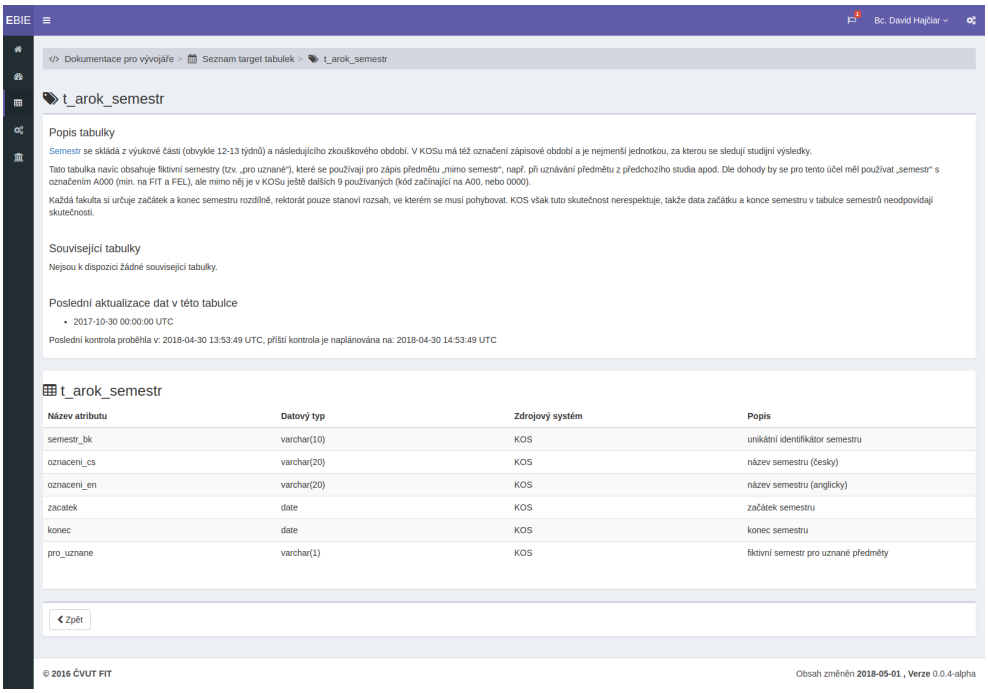
```
every 1.hour do
  rake "target_tables:load_update_times period=#{1.hour}"
end
```

Uvedený příkaz vyvolá funkci update_from_dwh třídy TargetTable, která data získá přes ebie-api a následně uloží:

```
def self.update_from_dwh(period)
  response = HTTP.get("#{ENV['DATA_API_URL']}/casy-zmen",
   json: { ebie_auth_key: ENV['EBIE_API_KEY'] })
  return unless response.code == 200
  t = Time.now

  response.parse.each do |target_table_info|
    target_table = TargetTable.find_or_create_by name:
                        target_table_info['table_name']
    update_time = target_table_info['last_update']
    target_table.target_table_updates.find_or_create_by
                                          updated_time:
         update_time.to_datetime if update_time.present?
```

```
    target_table.update last_checked: t, next_check: t + period
  end
end
```

Při spuštění generátoru obsahu, který převádí AsciiDoc soubory do html, jsou do target tabulek přidány získané časové údaje. Na obrázku 3.9 je vidět výsledná podoba tabulky t_arok_semestr.



Obrázek 3.9: Výsledná podoba target tabulky t_arok_semestr

Jelikož jsou údaje získávány pro všechny target tabulky v datovém skladu, ukládají se i pro ty, které zatím nejsou dokumentované v EBIE. Po jejich přidání tak budou získané časy úprav okamžitě k dispozici.

3.4 Úpravy obsahu

Do všech target tabulek, diagramů, pojmů ze sémantické vrstvy a zároveň do jejich šablon jsem doplnil komentáře označující generovaná místa:

- Target tabulky

```
==== Popis tabulky

// Popis tabulky
```

```
// Konec popisu
...
==== Související tabulky

// Odkazy

// Konec odkazu
...
// Tabulka

// Konec tabulky
```

- Datamarty

```
// Image datamartu

// Konec image datamartu
...
<!- Tabulka ->
```

- Pojmy ze sémantické vrstvy

```
...
<!- Tabulka ->
```

Tabulku „DLM tabulky“, zmíněnou v sekci 2.1.1, Magda Friedjungová vytvořila a předvyplnila. Já jsem, podle návrhu ze stejné sekce, popis target tabulek převedl do AsciiDoc formátu.

Aktualizujícími generátory jsem obnovil obsah datamartů a pojmů ze sémantické vrstvy. V obou sekcích došlo k opravě několika překlepů a doplnění malého počtu sloupců tabulek - V tabulce t_zapi_zapsany_predmet je jeden atribut bez popisu, u dvou řádků tabulky pojmu Studium nebyl nalezen sémantický název.

Tyto drobnosti bude snadné v DLM schéma napravit.

U datamartů jsem se setkal s většími komplikacemi, oba přítomné datamarty, reprezentované materializovanými pohledy, mají zhruba polovinu sloupců pojmenovanou názvy, které se v DLM schéma nevyskytují. Abych předešel zobrazení poloprázdných tabulek, jejich obsah jsem prozatím neaktualizoval.

Doplnil jsem alespoň potřebné komentáře. Až bude tato nekonzistence vyřešena, spuštění generátoru pro aktualizaci datamartů vytvoří správné tabulky i diagramy.

Do všech target tabulek a jejich šablony jsem přidal novou část pro zobrazení časů jejich aktualizace:

```
==== Poslední aktualizace dat v~této tabulce

// Zajišťuje ebie-source

{table_updates}
```

V souboru logic.adoc jsem přidal řádek „:imagesoutdir: assets/img “, který určuje lokaci nově vygenerovaným obrázkům (diagramům).

Nový soubor plantuml.config určuje výslednou podobu diagramu. Objektům jsem nastavil mírně zaoblené rohy, bílou barvu (faktovým tabulkám světle modrou) a černé ohraničení:

```
!define FACT_COLOR #ECFFFD
skinparam roundcorner 15
skinparam object {
    BackgroundColor white
    ArrowColor black
    BorderColor #000000
}
```

Další úpravy a opravy:

- Seřazení info.json u procesů v Dokumentaci pro vývojáře
- Oprava obrázků - Obrázky, přiložené přes AsciiDoc dokument, nebyly responzivní, upravil jsem proto backend generující html kód pro zobrazení obrázků.

  Nově lze přidat obrázek řádkem „image::1.png[role="img-responsive"]“, který dosadí zadanou třídu.
- Složku sémantická vrstva a šablonu se stejným názvem jsem přejmenoval na „semantic_layer“, aby korespondovaly se jmennou konvencí u ostatních složek/adresářů repozitáře.
- README.adoc, dokumentující repozitář ebie-content, jsem doplnil informacemi o nových generátorech.

  Předchozí README, popisující ruční tvorbu dokumentů, jsem uložil pod názvem add_file.adoc a odkazuji na něj z hlavního dokumentu.

- Všechny generátory, které upravují obsah, jsem dokumentoval v modulu Dokumentace pro vývojáře, v sekci Procesy (viz příloha B.1).

  Dokumentace se sestává z DEMO a BPMN diagramů a krátkého popisu.

- Soubor .gitignore jsem doplnil o .config.json a .env. Uživateli tak nebude hrozit, že tajné informace, obsažené v těchto souborech, nahraje do repozitáře.

Kapitola 4

Testování

Všechna implementovaná řešení jsem vyzkoušel a otestoval vytvořenými testy.

4.1 Testování generátorů

Funkčnost vytvořených generátorů jsem důkladně otestoval itegračními testy s využitím knihovny RSpec.

Do repozitáře ebie-content jsem přidal nové soubory (viz 4.1).

Účelem vytvořených testů je:

- Kontrola pro vývojáře generátorů, zda se implementací nové funkčnosti nenarušil dříve implementovaný proces
- Uživatel si po instalaci a nastavení repozitáře ebie-content může snadno ověřit, že mu generátory fungují
- Prokázání funkčnosti generátorů

```
spec ..... adresář obsahující testy a jejich podpůrné soubory
├── content_logic ..... Adresář s testy pro soubory v content_logic
│   ├── dlm_worker_spec.rb ..... Testy související s DLM
│   ├── dwh_worker_spec.rb ..... Testy související s DWH
│   └── generator_helpers_spec.rb ..... Testy zbývajících pomocných
│       generátorů
├── helpres ..... Pomocné funkce pro testy
└── spec_helper ..... Výchozi konfigurační soubor pro RSpec
```

Obrázek 4.1: Adresáře a složky pro testování frameworkem RSpec

Testy se spouští příkazem „rspec spec“. Je možné spustit test jediného souburu - „rspec spec/content_logic/dlm_spec.rb“, popřípadě pouze jediný test podle čísla řádku - příkaz „rspec spec/content_logic/dlm_worker_spec.rb:44“ spustí ze souboru dlm_spec.rb test na řádce 44.

Základní syntaxe zápisu testů v Rspec:

- Testy jsou ohraničeny bloky „describe“ a „it“, které popisují jejich obsah a zaměření
- Ověření (otestování) předpokladu se zapisuje pomocí funkce „expect“, např.

  ```
  expect(1).to eql 1
  ```

  Pokud podmínka není splněna, daný test je ukončen a uživatel je informován o chybě.
- Bloky „before“ a „after“ jsou spouštěny před a po každém testu

Způsob testování
Vytvořené testy ověřují funkčnost připojení k databázi, čtení dat z DLM a správné výsledky operací spojených s jednotlivými generátory.

Jako ukázku uvedu (zkráceně) část testů ověřujících funkčnost aktualizace target tabulek. Protože testy manipulují s existujícími soubory, před jejich spuštěním soubory zálohuji a po ukončení je opět obnovuji:

```
describe 'update_target_tables' do

    before do
      Dir["#{target_tables_path}/*.adoc"].each do |file_path|
        backup_file file_path, file_path + '.bkp'
      end
      Content.start(['update_target_tables', '-t'])
        # Spustí content:update_target_tables -t
    end

    after do
      Dir["#{target_tables_path}/*.adoc.bkp"].each do |file_path|
        backup_file file_path, file_path[0..-5], true
      end
    end
```

Před vykonáním testů target tabulky aktualizuji. Úspěšnost této operace ověřuji prvním testem, který sleduje, zda generátor nevypsal zprávu oznamující aktualizaci některého ze souborů:

```
it 'should not update when nothing changed' do
  expect { Content.start(['update_target_tables', '-t'])
  }.not_to output(/byl aktualizován/).to_stdout
end
```

Další test ověřuje, že při změně obsahu tabulky generátor tento soubor upraví do stejné podoby jako před změnou (ověřuji jak vypsání informace do konzole, tak stejný obsah kopie target tabulky a zálohovaného originálu):

```
it 'should update when table is changed' do
  file_to_edit = read_file_in_utf file
  old_file = file_to_edit.dup

  file_to_edit.gsub! TargetTableHelper::DETECT_TABLE,
        "// Tabulka\nneaktualni tabulka\n// Konec tabulky"
  write_to_file file, file_to_edit

  expect { Content.start(['update_target_tables', '-t'])
}.to output(/#{name}\.adoc byl aktualizován/).to_stdout
  expect(read_file_in_utf(file)).to eql old_file
 end
```

Pokud některý z testů selže, uživateli je zobrazen podrobný výstup vysvětlující, v jakém testu a na jaké řádce se neúspěšný test nachází a z jakého důvodu byl takto vyhodnocen.

Pokud by selhal v této sekci uvedený test aktualizace tabulky, konec výpisu by vypadal takto: (v příloze B.2.1 uvádím ukázku plného výpisu)

```
rspec ./spec/content_logic/dlm_worker_spec.rb:46 # Content
update_target_tables should update when table is changed
```

Dalšími testy poté ověřuji správnou detekci změny popisu, souvisejících odkazů i souboru info.json. Podobné testy jsem vytvořil také pro ostatní generátory.

Při správném nastavení spojení s DWH3 a DLM je po spuštění „rspec spec“ úspěšných všech 18 testů - viz obrázek 4.2.

```
Finished in 2 minutes 34 seconds (files took 0.60017 seconds to load)
18 examples, 0 failures
```

Obrázek 4.2: Výsledná obrazovka při úspěšném průchodu všemi testy

4.1.1 Testování v ebie-source

Na otestování přidaných tříd v ebie-source jsem použil knihovnu minitest, která je defaultně k dispozici v RoR aplikacích. Oproti RSpec má jednodušší syntaxi.

U modelů jsem vytvořil unit testy kontrolující funkčnost jejich základních metod. Ukázka ověřuje, že ověřovací klíč je vždy vytvořen a že je zaručena jeho unikátnost:

```
test 'creates key' do
  test_file = ContentTestFile.create
  assert_not test_file.key.nil?
end

test 'key can not be duplicated' do
  test_file = ContentTestFile.create
  another_test_file = ContentTestFile.create(key: test_file.key)
  assert another_test_file != test_file.key
end
```

U kontroleru ContentTestController jsem ověřil, že přijímá požadavek pouze při obdržení validního klíče:

```
test 'requests with invalid keys are not authorized' do
  assert_raises(StandardError) { post :create, key: 'invalid' }
end

test 'requests with valid keys pass' do
  post :create, key: ContentTestFile.create.key,
                content_test: { data: 'html' }, format: :json
  assert_response :success
end
```

Testy jsou spouštěny příkazem „rake test“, před jejich spuštěním je nutné připravit testovací databázi - „rake db:test:prepare“.

Úspěch ve všech testech je možný pouze při spuštění v prostředí EBIE. Testuji totiž i načítání dat z datového skladu, které je závislé na tajném klíči, uloženém pouze na EBIE.

Závěr

V této práci jsem, v souladu se zadáním, podrobně analyzoval a popsal datové toky v Datovém skladu ČVUT, společně s existujícími způsoby dokumentace těchto toků a jejich nedostatky.

Na základě analýzy jsem navrhl vhodné způsoby dokumentace těchto toků, a to jak využitím existujících procesů, tak vytvořením nových.

Dále jsem ukázal a doplnil způsob prezentace těchto dat v prostředí portálu EBIE za využití navržených generátorů a získávání časových údajů z datového skladu.

Navržené generátory jsem implementoval a současně jsem upravil aplikaci ebie-source. Tato řešení jsem zdokumentoval (společně s návodem na používání) v README repozitáře ebie-content.

Na závěr jsem pro implementované řešení vytvořil unit a integrační testy, kterými jsem ukázal funkčnost implementovaných částí a zároveň zajistil možnost jejich kontroly při dalším vývoji.

Vytvořené generátory zlepšují tvorbu a údržbu obsahu na EBIE takto:

- Rychlé a bezchybné vytvoření souborů s obsahem z DLM nebo DWH
- Snadná aktualizace existujících souborů a diagramů
- Zrychlené přidávání záznamů do info.json spojené se zajištěním abecedního řazení položek
- Odhalení chybějících pojmů v DLM
- Rychlá kontrola vzhledu a funkčnosti souborů
- Jednotný formát dokumentů

Další rozvoj

Vytvořené generátory jsou snadno upravitelné a rozšiřitelné, případným požadavkům na změnu struktury nebo obsahu dokumentovaných částí tak bude možné vyhovět.

Pokud se ukáže, že uživatelům vyhovuje uložení popisů pro info.json a dokumenty v „DLM tabulky“, je možné „DLM tabulky“ rozšířit o pojmy ze sémantické vrstvy a datamarty.

Dalším krokem může být také zobrazení času poslední změny dat v datamartech a času provedení ETL procesů.

Literatura

[1] Valenta, M.: IP 2015 DU 20 – Datová čistota - manažerské shrnutí, 2016, dílčí výstupní zpráva projektu na ČVUT.

[2] Inmon, W.: *Building the Data Warehouse*. New York: A Wiley-QED Publication, 1992, ISBN 04-715-6960-7.

[3] Kuznetsov, S.: *Datový sklad fakulty*. Diplomová práce, České vysoké učení v Praze, Fakulta informačních technologií, Praha, 2013.

[4] Kimball, R.: *The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses*. New York: Wiley, 1996, ISBN 04-711-5337-0.

[5] Kimball, R.; Ross, M.: *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*. New York: Wiley, druhé vydání, 2002, ISBN 04-712-0024-7.

[6] Dokumentace k datovému skladu ČVUT [online]. 2018, [cit. 2018-04-10]. Dostupné z: `https://gitlab.fit.cvut.cz/big/BIG_D/wikis/home`

[7] Kuznetsov, S.: *Datový sklad fakulty*. Diplomová práce, České vysoké učení technické v Praze, Fakulta informačních technologií, 2013.

[8] Group, P. G. D.: PostgreSQL: Documentation: 9.5: Materialized Views [online]. `https://www.postgresql.org/docs/9.5/static/rules-materializedviews.html`, 2018, [cit. 2018-04-28].

[9] Hajčiar, D.: *Návrh a implementace backend projektu EBIE*. Bakalářská práce, České vysoké učení v Praze, Fakulta informačních technologií, Praha, 2016.

[10] Batík, O.: *Portál EBIE - nasazení pilotní verze v prostředí ČVUT*. Diplomová práce, České vysoké učení v Praze, Fakulta informačních technologií, Praha, 2017.

[11] Novotný, O.; Pour, J.; Slánský, D.: *Business intelligence*. Praha: Grada, první vydání, 2005, ISBN 80-247-1094-3.

[12] Jirovský, V.: *Konceptuální analýza datových domén Studium a Hodnocení kvality výuky s ohledem na datovou čistotu*. Diplomová práce, České vysoké učení v Praze, Fakulta informačních technologií, Praha, 2015.

[13] Watson, H. J.; Wixom, B. H.: The Current State of Business Intelligence [online]. *Computer*, ročník 40, č. 9, Sept 2007: s. 96–99, ISSN 0018-9162, doi:10.1109/MC.2007.331.

[14] Guizzardi, G.: *Ontological foundations for structural conceptual models [online]*. Enschede Enschede: Centre for Telematics and Information Technology Telematica Instituut, 2005, ISBN 90-75176-81-3.

[15] Group, O. M.: BPMN Specification - Business Process Model and Notation [online]. `http://www.bpmn.org/`, 2018, [cit. 2018-02-05].

[16] About Ruby [online]. 2018, [cit. 2018-04-11]. Dostupné z: `https://www.ruby-lang.org/en/about/`

[17] Git [online]. 2018, [cit. 2018-04-08]. Dostupné z: `https://git-scm.com/`

[18] GitLab: Features [online]. 2018, [cit. 2018-04-20]. Dostupné z: `https://about.gitlab.com/features/`

[19] Git - gitsubmodules Documentation [online]. 2018, [cit. 2018-04-08]. Dostupné z: `https://git-scm.com/docs/gitsubmodules/`

[20] JSON [online]. 2018, [cit. 2018-04-08]. Dostupné z: `http://www.json.org/json-cz.html`

[21] What is Asciidoc? Why do we need it? [online]. 2016, [cit. 2018-04-09]. Dostupné z: `http://asciidoctor.org/docs/what-is-asciidoc/#the-zen-of-writing-asciidoc`

[22] Pleticha, O.: *Webová aplikace pro monitoring datového skladu ČVUT*. Bakalářská práce, České vysoké učení v Praze, Fakulta informačních technologií, Praha, 2018.

[23] Eeckhoudt, P. V.: Asciidoctor diagram extension, with support for AsciiToSVG ... [online]. 2018, [cit. 2018-04-21]. Dostupné z: `https://github.com/asciidoctor/asciidoctor-diagram`

[24] Ichikawa, H.: A Ruby library to read/write files/spreadsheets in Google Drive/Docs [online]. 2018, [cit. 2018-04-20]. Dostupné z: `https://github.com/gimite/google-drive-ruby`

[25] Evans, J.: Sequel: The Database Toolkit for Ruby [online]. 2018, [cit. 2018-04-20]. Dostupné z: https://github.com/jeremyevans/sequel

[26] Evans, J.: Faster SELECTs when using Sequel with pg [online]. 2018, gitHub repository. Dostupné z: https://github.com/jeremyevans/sequel_pg

[27] Keepers, B.: A Ruby gem to load environment variables from '.env'. [online]. 2018, [cit. 2018-04-20]. Dostupné z: https://github.com/bkeepers/dotenv

[28] Wiggins, A.: Simple HTTP and REST client for Ruby [online]. 2018, [cit. 2018-04-21]. Dostupné z: https://github.com/rest-client/rest-client/

[29] Yehuda Katz, J. V.: Thor [online]. 2017, [cit. 2018-04-21]. Dostupné z: https://github.com/erikhuda/thor

[30] David Rodriguez, M. M., Kent Sibilev: Debugging in Ruby 2 [online]. 2018, [cit. 2018-04-21]. Dostupné z: https://github.com/deivid-rodriguez/byebug

[31] Bundler: the best way to manage a Ruby application's gems [online]. 2018, [cit. 2018-04-20]. Dostupné z: http://bundler.io/

[32] Together, R.: Stats [online]. 2018, [cit. 2018-04-21]. Dostupné z: https://rubygems.org/stats

[33] Google: Authorize Requests | Sheets API | Google Developers [online]. https://developers.google.com/sheets/api/guides/authorizing, duben 2018, [cit. 2018-04-24].

[34] Project, U. D.: CronHowto - Community Help Wiki [online]. https://help.ubuntu.com/community/CronHowto, 11 2016, [cit. 2018-04-30].

[35] Makhmali, J.: javan/whenever: Cron jobs in Ruby [online]. https://github.com/javan/whenever/, 1 2018, [cit. 2018-04-30].

[36] GitLab: GitLab Continuous Integration (GitLab CI/CD) [online]. 2018, [cit. 2018-04-20]. Dostupné z: https://docs.gitlab.com/ee/ci/

[37] NGINX: Module ngx_http_headers_module [online]. http://nginx.org/en/docs/http/ngx_http_headers_module.html#expires, 4 2018, [cit. 2018-04-50].

[38] GitLab: Registering Runners | GitLab [online]. https://docs.gitlab.com/runner/register/index.html, 2018, [cit. 2018-04-25].

Příloha **A**

Seznam použitých zkratek

API Application Programming Interface

BPMN Business Process Model and Notation

CI Continuous Integration

CSV Comma Separated Values

ČVUT České vysoké učení technické

DEMO Design & Engineering Methodology for Organizations

DLM Datová logická mapa

DWH3 Data WareHouse 3

EBIE Extended Business Intelligence Encyclopedia

ETL Extract Transformation Load

HTML HyperText Markup Language

FIT Fakulta informačních iechnologií

JSON JavaScript Object Notation

KOS Komponenta studium

MIS Manažerský informační systém

MVC Model View Controller

PES Portál ekonomických služeb

RoR Ruby on Rails

UML Unified Modeling Language

XSS Cross Site Scripting

Příloha B

Přílohy

B.1 Dokumentace generátorů

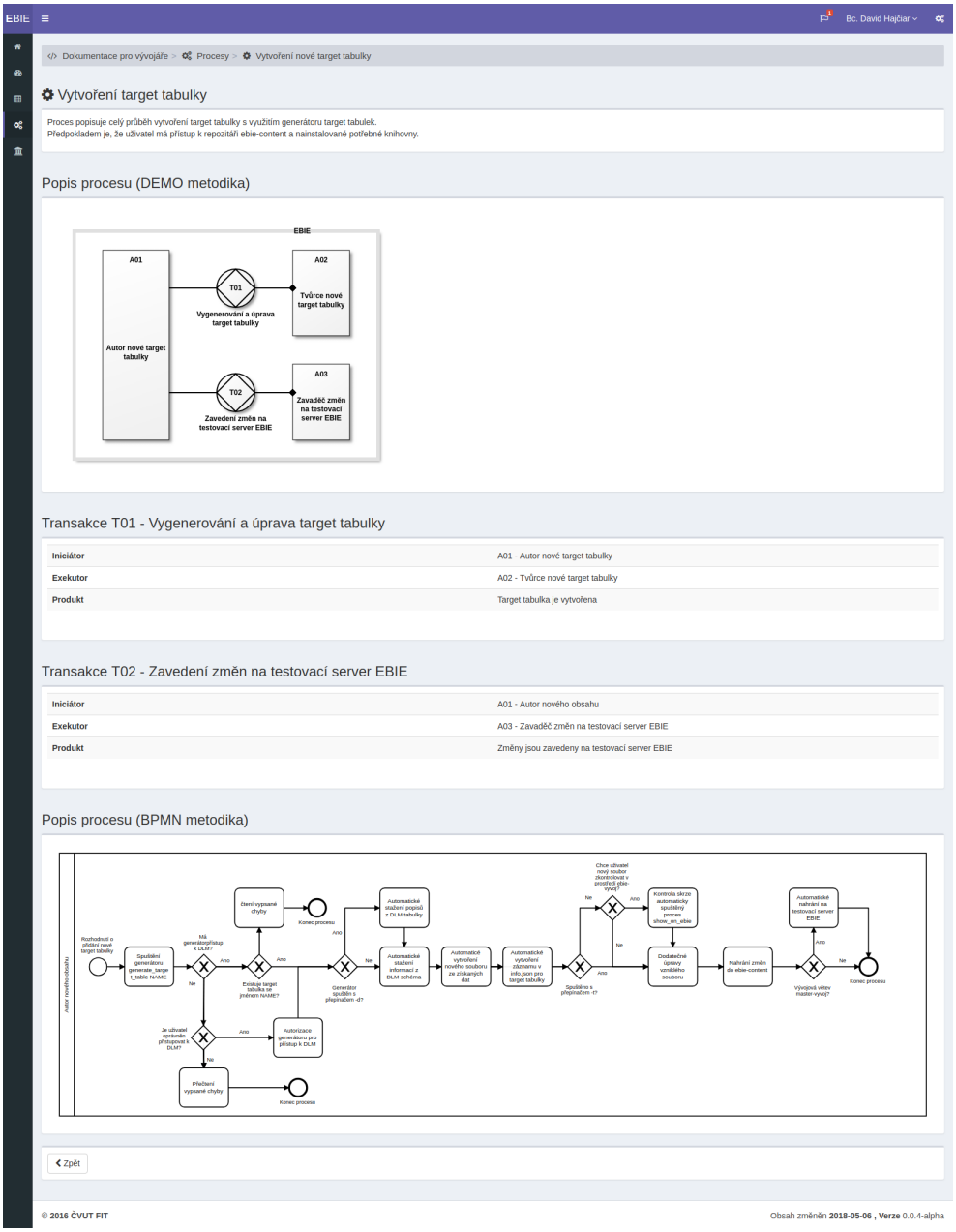
B.1.1 Generátor target tabulek

```
> thor content:help generate_target_table
Usage:
  thor content:generate_target_table NAME

Options:
  -t, [--test], [--no-test]
     # Nenabízet zobrazení na ebie-vyvoj a při výjimce
      nevykonat abort - pro vývojáře a automatické testy

Generuje novou target tabulku
```

Na snímku stránky B.1 ukazuji dokumentaci tohoto generátoru v sekci Procesy v Dokumentaci pro vývojáře.

B.1.2 Generátor datamartů

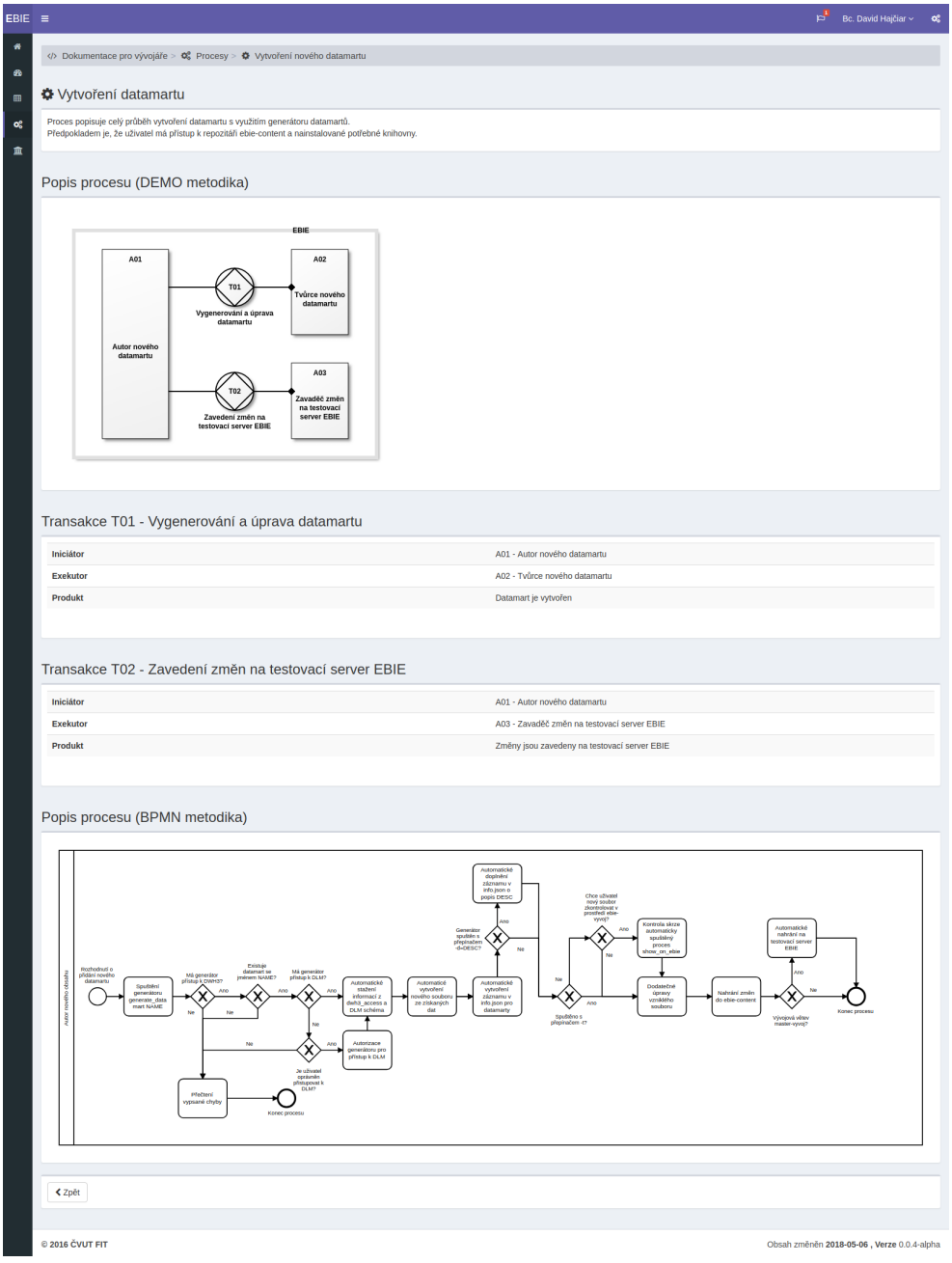
```
> thor content:help generate_datamart
Usage:
  thor content:generate_datamart NAME

Options:
  -d, [--description=DESCRIPTION]
     # Popis nově vytvořeného datamartu pro info.json
  -t, [--test], [--no-test]
     # Nenabízet zobrazení na ebie-vyvoj a při výjimce
      nevykonat abort - pro vývojáře a automatické testy
```



Obrázek B.1: Proces v Dokumentaci pro vývojáře, který popisuje vytvoření nové target tabulky

`Generuje nový datamart`

Obrázek B.2 zachycuje vytvořenou dokumentaci.



Obrázek B.2: Proces v Dokumentaci pro vývojáře, který popisuje vytvoření nového datamartu

B.1.3 Generátor pojmů ze sémantické vrstvy

```
> thor content:help generate_semantic
Usage:
  thor content:generate_semantic TARGET_TABLE_NAME

Options:
  -d, [--description=DESCRIPTION]
     # Popis nově vytvořeného datamartu pro info.json
  -t, [--test], [--no-test]
     # Nenabízet zobrazení na ebie-vyvoj a při výjimce
      nevykonat abort - pro vývojáře a automatické testy

Vytvoří nový adoc soubor pro pojem ze sémantické vrstvy
```

Snímek stránky B.3 zachycuje stránku popisující vytvoření nového pojmu ze sémantické vrstvy.

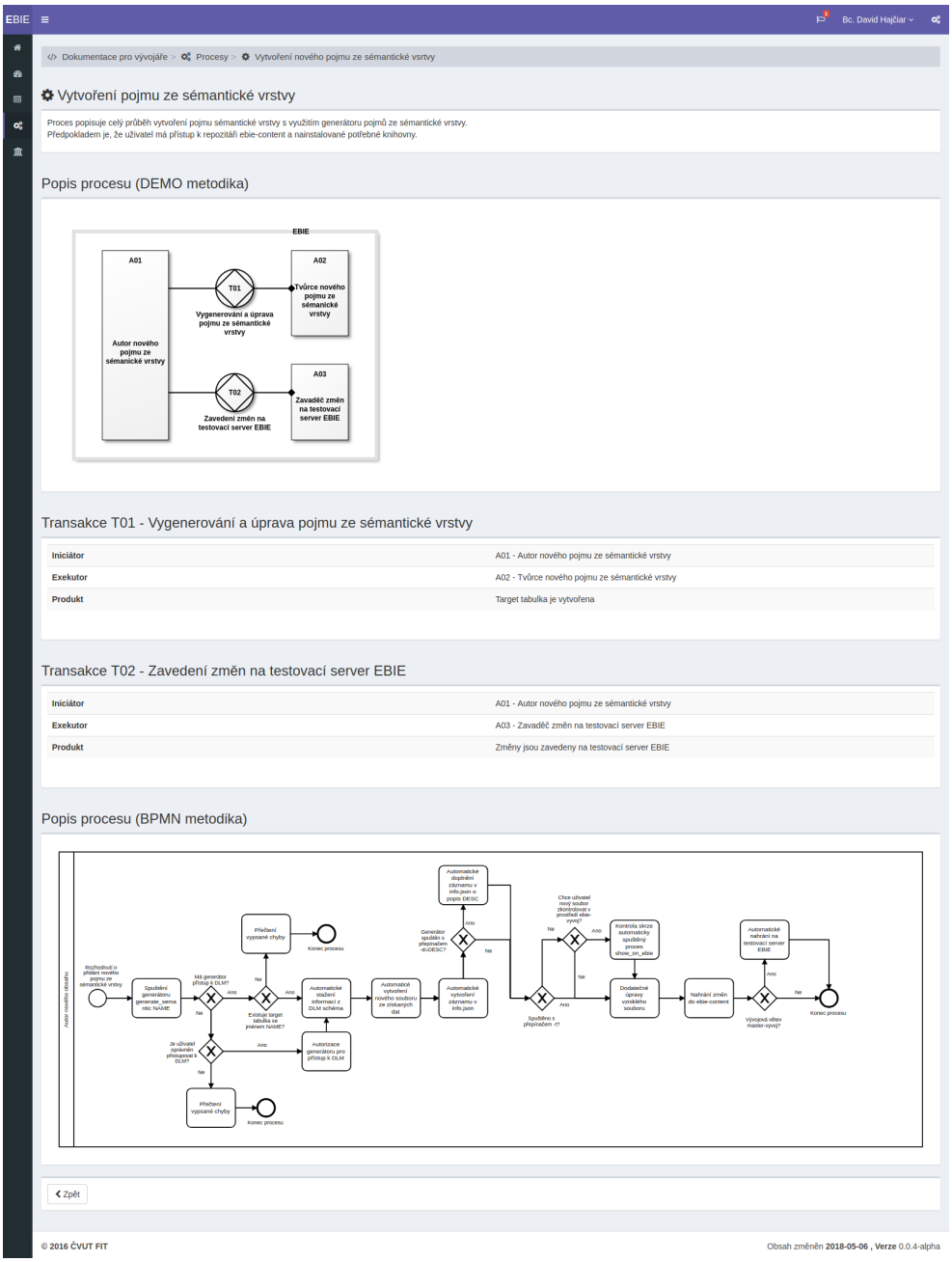
B.1.4 Generátor aktualizující target tabulky

```
> thor content:help update_target_tables
Usage:
  thor content:update_target_tables

Options:
  -d, [--descriptions], [--no-descriptions]
     # Aktualizovat také
   popisy v info.json a v target tabulkách)
  -t, [--test], [--no-test]
     # Při rescue výjimky neprovede abort - pro vývojáře
      a automatické testy

Aktualizuje všechny přítomné target tabulky podle DLM
```

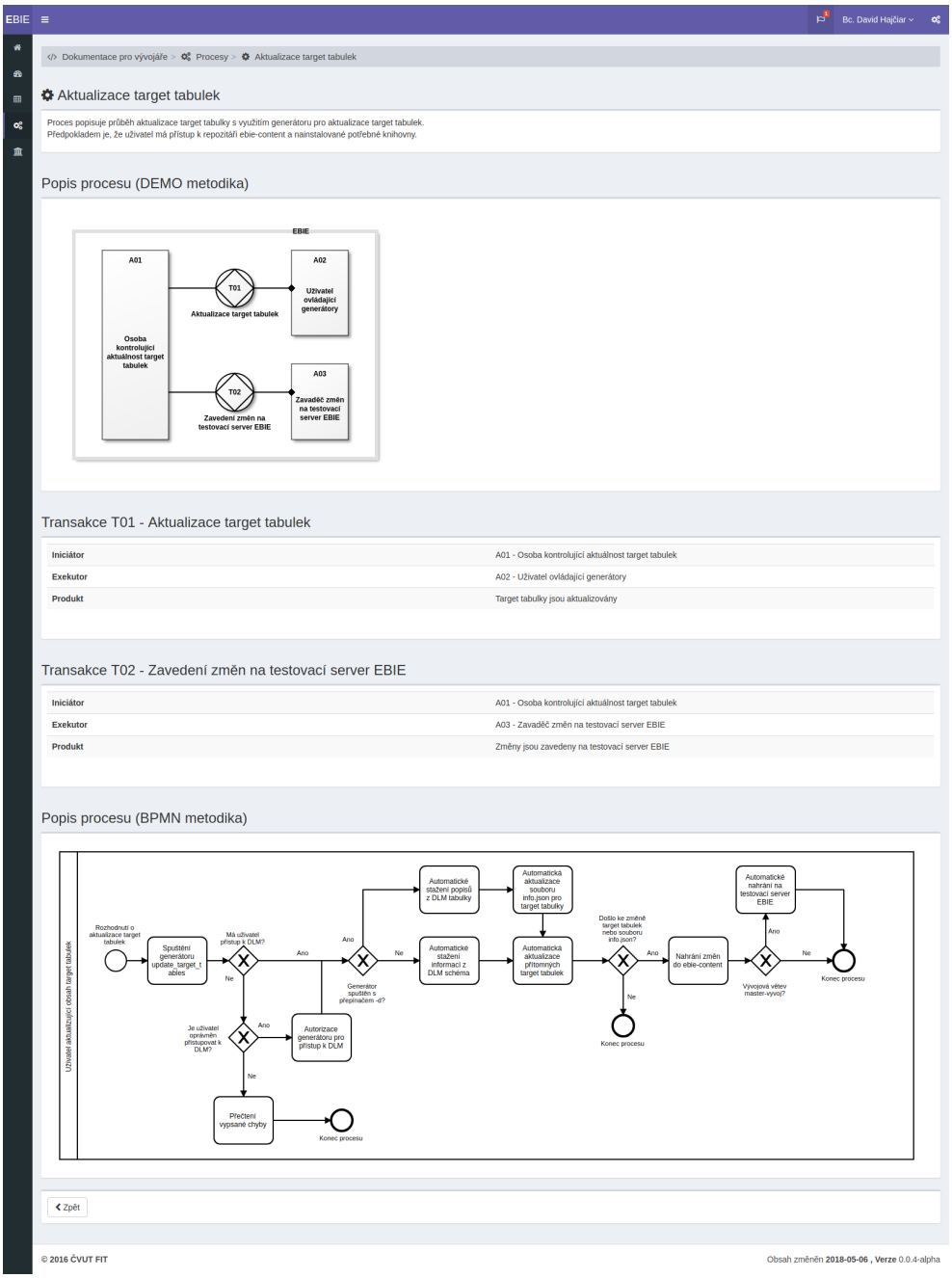
Dokumentace tohoto procesu je k vidění na obrázku B.4.

B.1.5 Generátor aktualizující datamarty

```
> thor content:help update_datamarts
Usage:
  thor content:update_datamarts

Options:
  -t, [--test], [--no-test]
     # Při rescue výjimky neprovede abort - pro vývojáře
```



Obrázek B.3: Proces v Dokumentaci pro vývojáře, který popisuje vytvoření nového pojmu ze sémantické vrstvy



Obrázek B.4: Proces v Dokumentaci pro vývojáře, který popisuje aktualizaci target tabulek

```
      a automatické testy

Aktualizuje všechny přítomné datamarty
```

Dokumentace tohoto generátoru - viz obrázek B.5.

B.1.6 Generátor aktualizující pojmy ze sémantické vrstvy

```
> thor content:help update_semantics
Usage:
  thor content:update_semantics

Options:
  -t, [--test], [--no-test]
     # Při rescue výjimky neprovede abort - pro vývojáře
      a automatické testy

Aktualizuje všechny přítomné pojmy ze Sémantické vrstvy
```

Obrázek B.6 zachycuje vytvořenou dokumentaci v EBIE.

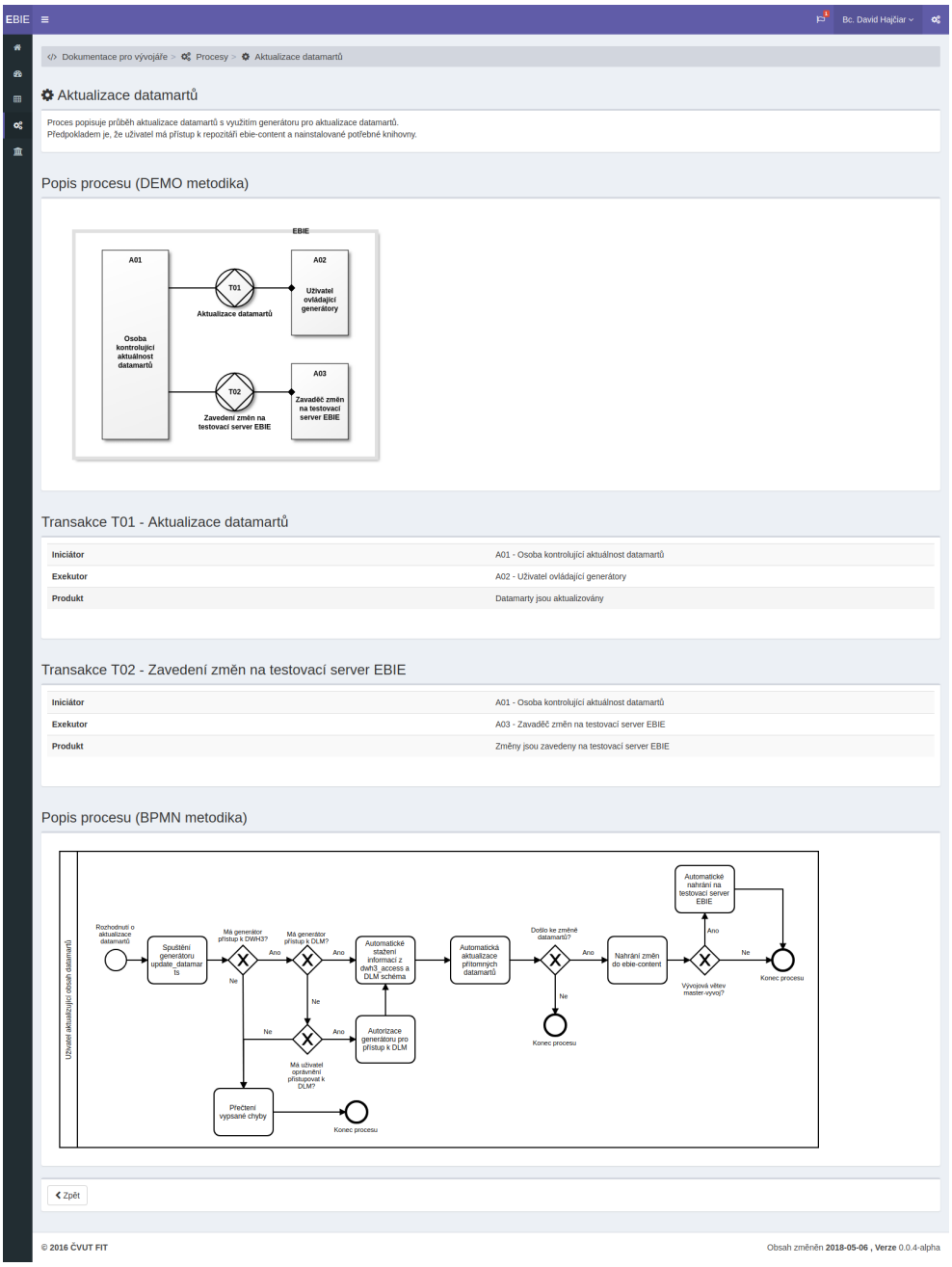
B.1.7 Generátor zobrazující soubor v ebie-vyvoj

```
> thor content:help show_on_ebie
Usage:
  thor content:show_on_ebie FILE_PATH

Options:
  -t, [--test], [--no-test]
  # Při rescue výjimky neprovede abort - pro vývojáře
   a automatické testy

Ukáže zadaný soubor na ebie-vyvoj
```

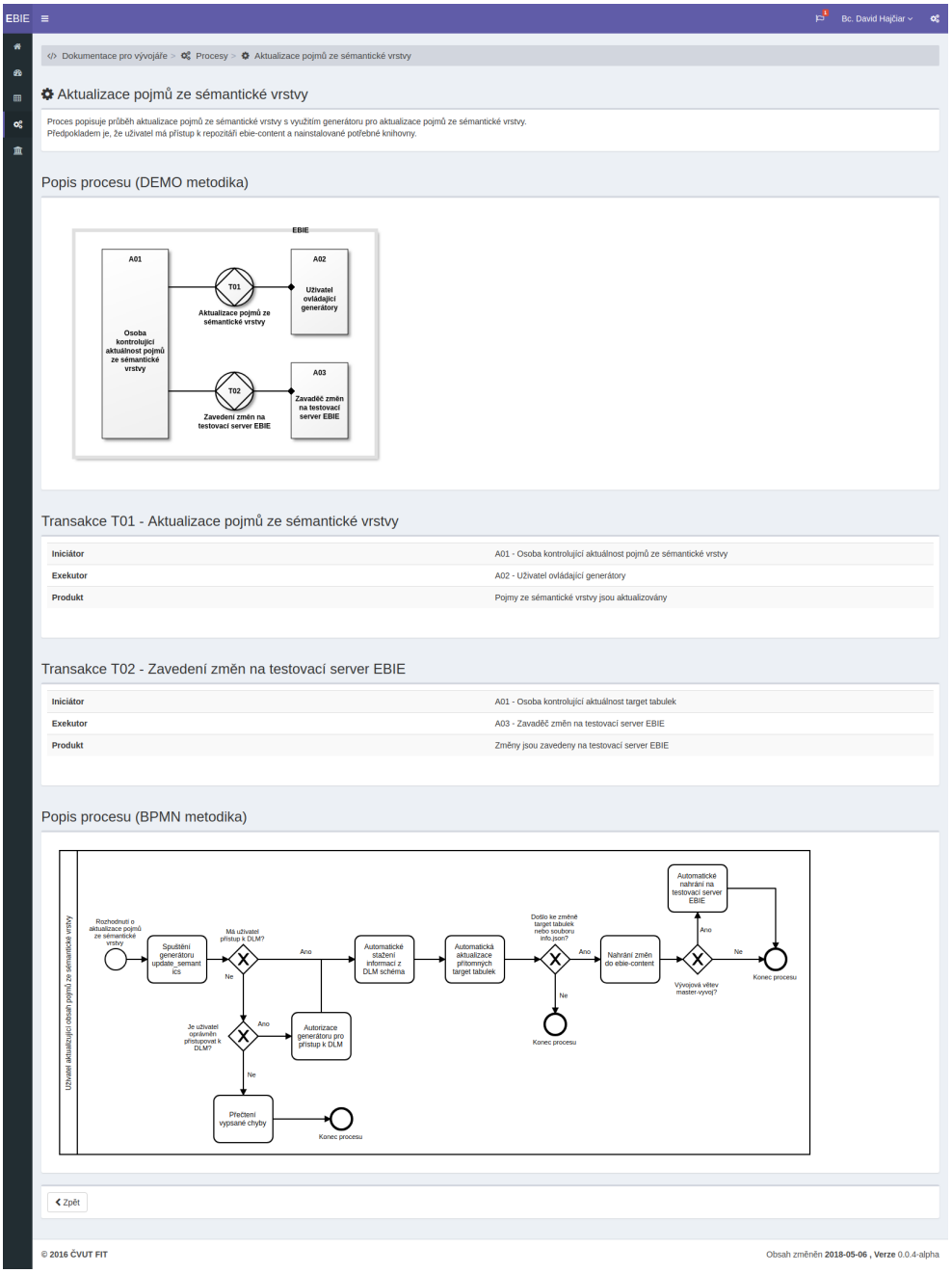
B.1.8 Pomocné generátory

Kontrola přítomnosti komentářů

```
> thor content:help missing_comments
Usage:
  thor content:missing_comments

Zkontroluje přítomnost potřebných komentářů ve všech souborech,
ve kterých jsou komentáře nezbytné
```



Obrázek B.5: Proces v Dokumentaci pro vývojáře, který popisuje aktualizaci datamartů



Obrázek B.6: Proces v Dokumentaci pro vývojáře, který popisuje aktualizaci pojmů ze sémantické vrstvy

Výpis datamartů v datovém skladu

```
> thor content:help print_datamarts
Usage:
  thor content:print_datamarts

Options:
  -t, [--test], [--no-test]
  # Při rescue výjimky neprovede abort - pro vývojáře
   a automatické testy

Zobrazí názvy všech datamartů, které jsou v datovém skladu
```

Seřažení info json

```
thor content:help sort_info_json
Usage:
  thor content:sort_info_json FILE_PATH

Seřadí obdržený soubor info.json podle klíče 'title'
```

Kontrola konzistence DLM

```
thor content:help check_dlm_consistency
Usage:
  thor content:check_dlm_consistency

Options:
  -t, [--test], [--no-test]
     # Při rescue výjimky neprovede abort - pro vývojáře
      a automatické testy

Vypíše target tabulky, kterou nejsou buď v DLM schéma nebo
 DLM tabulky
```

B.2 Chybové výpisy

B.2.1 Neúspěšný test v RSpec

```
Run options: include {:locations=>{"./spec/content_logic/
dlm_worker_spec.rb"=>[46]}}
"Aktualizace dokončena"
```

```
F

Failures:

  1) Content update target_tables should update when table is
   changed Failure/Error: expect {
      Content.start(['update_target_tables', '-t']) }.to
      output(/#{name}\.adoc byl aktualizován/).to_stdout

       expected block to output /t_prih_prihlaska\.adoc byl
       aktualizován/ to stdout, but output "\"VAROVÁNÍ:
       V~souboru /ebie-content/ebie_content/dokumentace/tables/
       t_prih_prihlaska.adoc chybi komentare oznacujici
       Tabulku\"\n\"Aktualizace dokončena\"\n"
       Diff:
       @@ -1,2 +1,3 @@
       -/t_prih_prihlaska\.adoc byl aktualizován/
       +"VAROVÁNÍ: V~souboru
       /ebie-content/ebie_content/dokumentace/tables/
       t_prih_prihlaska.adoc chybi komentare oznacujici Tabulku"
       +"Aktualizace dokončena"

     # ./spec/content_logic/dlm_worker_spec.rb:51:in `block
      (3 levels) in <top (required)>'

Finished in 16.1 seconds (files took 0.61533 seconds to load)
1 example, 1 failure

Failed examples:

rspec ./spec/content_logic/dlm_worker_spec.rb:46 # Content
update target_tables should update when table is changed
```

B.2.2 Backtrace při selhání připojení k DWH3

```
/home/david/.rvm/gems/ruby-2.4.3/gems/sequel-5.7.1/lib/sequel/
adapters/postgres.rb:191:in `initialize': PG::ConnectionBad:
could not connect to server: Connection refused
 (Sequel::DatabaseConnectionError)
        Is the server running on host xxx and accepting
        TCP/IP connections on port xxx?

from /home/david/.rvm/gems/ruby-2.4.3/gems/sequel-5.7.1/lib/
```

```
sequel/adapters/postgres.rb:191:in ‘new’
from /home/david/.rvm/gems/ruby-2.4.3/gems/sequel-5.7.1/lib/
sequel/adapters/postgres.rb:191:in ‘connect’
from /home/david/.rvm/gems/ruby-2.4.3/gems/sequel-5.7.1/lib/
sequel/connection_pool.rb:126:in ‘make_new’
from /home/david/.rvm/gems/ruby-2.4.3/gems/sequel-5.7.1/lib/
sequel/connection_pool/threaded.rb:192:in ‘assign_connection’
from /home/david/.rvm/gems/ruby-2.4.3/gems/sequel-5.7.1/lib/
sequel/connection_pool/threaded.rb:133:in ‘acquire’
from /home/david/.rvm/gems/ruby-2.4.3/gems/sequel-5.7.1/lib/
sequel/connection_pool/threaded.rb:90:in ‘hold’
from /home/david/.rvm/gems/ruby-2.4.3/gems/sequel-5.7.1/lib/
sequel/database/connecting.rb:264:in ‘synchronize’
from /home/david/.rvm/gems/ruby-2.4.3/gems/sequel-5.7.1/lib/
sequel/database/connecting.rb:279:in ‘test_connection’
from /home/david/.rvm/gems/ruby-2.4.3/gems/sequel-5.7.1/lib/
sequel/database/connecting.rb:58:in ‘connect’
from /home/david/.rvm/gems/ruby-2.4.3/gems/sequel-5.7.1/lib/
sequel/core.rb:116:in ‘connect’
from /ebie-content/content_logic/
dwh_worker.rb:12:in ‘initialize’
from /ebie-content/
content.thor:104:in ‘new’
from /ebie-content/
content.thor:104:in ‘print_datamarts’
from /home/david/.rvm/gems/ruby-2.4.3/gems/thor-0.20.0/lib/
thor/command.rb:27:in ‘run’
from /home/david/.rvm/gems/ruby-2.4.3/gems/thor-0.20.0/lib/
thor/invocation.rb:126:in ‘invoke_command’
from /home/david/.rvm/gems/ruby-2.4.3/gems/thor-0.20.0/lib/
thor.rb:387:in ‘dispatch’
from /home/david/.rvm/gems/ruby-2.4.3/gems/thor-0.20.0/lib/
thor/base.rb:466:in ‘start’
from /home/david/.rvm/gems/ruby-2.4.3/gems/thor-0.20.0/lib/
thor/runner.rb:42:in ‘method_missing’
from /home/david/.rvm/gems/ruby-2.4.3/gems/thor-0.20.0/lib/
thor/command.rb:29:in ‘run’
from /home/david/.rvm/gems/ruby-2.4.3/gems/thor-0.20.0/lib/
thor/command.rb:128:in ‘run’
from /home/david/.rvm/gems/ruby-2.4.3/gems/thor-0.20.0/lib/
thor/invocation.rb:126:in ‘invoke_command’
from /home/david/.rvm/gems/ruby-2.4.3/gems/thor-0.20.0/lib/
thor.rb:387:in ‘dispatch’
from /home/david/.rvm/gems/ruby-2.4.3/gems/thor-0.20.0/lib/
```

```
thor/base.rb:466:in ‘start’
from /home/david/.rvm/gems/ruby-2.4.3/gems/thor-0.20.0/bin/
thor:6:in ‘<top (required)>’
from /home/david/.rvm/gems/ruby-2.4.3/bin/thor:23:in ‘load’
from /home/david/.rvm/gems/ruby-2.4.3/bin/thor:23:in ‘<main>’
from /home/david/.rvm/gems/ruby-2.4.3/bin/
ruby_executable_hooks:15:in ‘eval’
from /home/david/.rvm/gems/ruby-2.4.3/bin/r
uby_executable_hooks:15:in ‘<main>’
```

Příloha C

Obsah přiloženého CD

- `DP_Hajčiar_David_2018.pdf` ..... text práce ve formátu PDF
- `DP_Hajčiar_David_2018.tex` ..... text práce ve formátu LaTeX
- `odkazy.txt` .. textový soubor obsahující odkazy na upravované repozitáře