FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

# ZADÁNÍ DIPLOMOVÉ PRÁCE

| | |
|---|---|
| **Název:** | Nástroj pro správu číselníků pro datové sklady |
| **Student:** | Bc. Šárka Weberová |
| **Vedoucí:** | RNDr. Ondřej Zýka |
| **Studijní program:** | Informatika |
| **Studijní obor:** | Webové a softwarové inženýrství |
| **Katedra:** | Katedra softwarového inženýrství |
| **Platnost zadání:** | Do konce letního semestru 2020/21 |

## Pokyny pro vypracování

Cílem diplomové práce je vytvořit komponentu pro správu číselníků pro datové sklady. Komponenta musí zvládnout na uživatelské úrovni definici struktury číselníků, stejně tak jako správu jejich obsahu, a také napojení na ostatní komponenty řešení.

Postupujte v těchto krocích:
1. Seznamte se s kontextem budoucího řešení.
2. Definujte funkční a nefunkční požadavky na modul a interface na ostatní moduly.
3. Navrhněte generalizovaný datový model pro komponentu.
4. Návrh implementujte formou funkčního prototypu, řádně jej zdokumentujte a otestujte.
5. Diskutujte přínosy řešení a navrhněte případná další rozšíření.

## Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

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

V Praze dne 31. října 2019

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

# Nástroj pro správu číselníků pro datové sklady

***Bc. Šárka Weberová***

Katedra Softwarového inženýrství
Vedoucí práce: RNDr. Ondřej Zýka

24. května 2020

# Poděkování

V první řadě bych chtěla poděkovat firmě Profinit EU s.r.o., že mi umožnila vytvořit diplomovou práci v rámci jednoho z jejich projektů. Dále bych chtěla poděkovat vedoucímu diplomové práce RNDr. Ondřeji Zýkovi za jeho odborné rady a pomoc při tvorbě této práce. Poděkování patří také mé rodině a přátelům za podporu během tvorby této práce, ale i během celého studia.

# Prohlášení

Prohlašuji, že jsem předloženou práci vypracovala samostatně a že jsem uvedla veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při 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, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 24. května 2020 . . . . . .

### Odkaz na tuto práci

Weberová, Šákra. *Nástroj pro správu číselníků pro datové sklady.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.

# Abstrakt

Tato práce dokumentuje analýzu, návrh a implementaci prototypu nové komponenty pro správu číselníků nástroje DATA_FRAME Application. DATA_FRAME Application je nástroj podporující automatizaci datových skladů, který je vyvíjen v platformě Oracle APEX firmou Profinit EU, s.r.o. Komponenta pro správu číselníků bude umožňovat správu definice jejich struktury a jejich obsahu pomocí grafického uživatelského rozhraní. Úpravy číselníků budou historizovány a bude možné je zpětně dohledat.

**Klíčová slova** Správa referenčních dat, číselník, referenční data, pomalu se měnící dimenze, DATA_FRAME Application, Profinit EU s.r.o., PL/SQL, Oracle APEX

# Abstract

This diploma thesis documents analysis, design and prototype implementation of new component for reference data management of DATA_FRAME Application. DATA_FRAME Application is a tool supporting data warehouse automation, which is developed in the Oracle APEX platform by Profinit EU, s.r.o. The reference data component will allow you to manage the definition

of reference data structure and their content using a graphical user interface. Changes to reference data will be historized and traceable.

**Keywords** Reference data management, code list, reference data, slowly changing dimension, DATA_FRAME Application, Profinit EU s.r.o., PL/SQL, Oracle APEX

# Obsah

# Seznam obrázků

# Seznam tabulek

# Seznam zdrojových kódů

# Úvod

Data jsou v dnešní době nedílnou součástí našeho života. Hlavně větší firmy musejí uchovávat velká množství dat a musejí umět z těchto dat získat pro ně prospěšné informace. Pro bližší představu si můžeme vzít příklad nějaké bankovní společnosti. Ta musí uchovávat mimo jiné veškerá data o zákaznících a jejich transakcích. Takových dat je mnoho a aby bylo možné se v nich vyznat, je potřeba věnovat čas a prostředky jejich správě. Pro automatizaci tohoto procesu existuje v dnešní době mnoho nástrojů a další vznikají.

Jedním z takovýchto nástrojů je například DATA_FRAME Application, který vznikl pod záštitou firmy Profinit EU, s.r.o. Nástroj umožňuje zjednodušit a automatizovat správu dat především v rámci datového skladu. Uživatelské rozhraní aplikace nabízí možnost spravovat data i uživatelům, kteří nemají znalosti databázového programovacího jazyka. Aplikace je navržena jako sestava několika komponent zaměřených na různé části databáze. Některé komponenty jsou již implementovány a používány, jiné jsou pouze navrženy jako budoucí součást aplikace.

Značná část tabulek v databázi (20-50%) jsou číselníky. Ty slouží ke kategorizaci dalších údajů v databázi a na rozdíl od kmenových dat jsou upravována jen zřídka a většinou ručně, ne automatizovaně. Protože jsou číselníky nedílnou součástí datových skladů a jejich správa je důležitá stejně, jako správa ostatních dat, byl nástroj DATA_FRAME Application navržen s vizí komponenty i na jejich správu. Analýza, návrh a implementace takové komponenty je předmětem této diplomové práce. Komponenta bude umožňovat na uživatelské úrovni spravovat číselníky v rámci DATA_FRAME Application. Také bude uchovávat historii veškerých změn provedených na číselnících a tyto změny bude uživatel moci v aplikaci dohledat.

## Cíl práce

Cílem diplomové práce je vytvoření nové komponenty nástroje DATA_FRAME Application. Výsledkem práce je funkční prototyp této komponenty vytvořený ve vývojové platformě Oracle APEX s ohledem na již hotové části nástroje DATA_FRAME Application.

Součástí práce je analýza současné verze nástroje DATA_FRAME Application a jiných již existujících řešení pro práci s číselníky. Také jsou v analytické části definovány funkční a nefunkční požadavky pro vytvářenou komponentu. Dále je vytvořen návrh nové komponenty, jehož součástí je datový model, definice způsobu historizace číselníků, definice rozhraní pro ostatní moduly aplikace a vytvoření návrhu uživatelského rozhraní. Tento návrh je uplatněn při realizaci prototypu, který bude řádně otestován. Implementace rozhraní poskytovaného pro ostatní moduly není součástí této diplomové práce.

Komponenta pro správu číselníků zvládne práci se strukturou číselníků v podobě úpravy jejich popisu a atributů. Dále bude umožňovat správu jejich obsahu a historizaci všech provedených změn.

Kapitola 1

# Analýza

V analytické části této práce je nejdříve popsán produkt DATA_FRAME Application, jeho již implementované i teprve plánované komponenty. Následně jsou analyzovány existující nástroje pro správu číselníků. Na základě této analýzy jsou definovány funkční a nefunkční požadavky na nově vznikající komponentu. Součástí této části práce je také popis případů užití nové komponenty.

## 1.1 Vymezení pojmů

V této sekci je uvedeno několik základních pojmů používaných v této diplomové práci a potřebných pro její pochopení.

**Kmenová data** „Kmenová data (anglicky Master data) jsou data o zdrojích, produktech, subjektech, místech nebo věcech a popisují jejich vlastnosti a parametry.“[2] „Kmenová data jsou klíčový zdroj informací pro fungování každé organizace.“[2]

**Referenční data** Referenční data (anglicky Reference data) jsou jakýkoli druh dat, který se používá ke kategorizaci dalších údajů uvedených v databázi nebo za účelem přiřazení dat v databázi k informacím z jiné domény.[3]

Referenční data jsou jednodušší než kmenová data a pomaleji se mění. Kategorizovat či propojovat můžou mimo jiné i různá kmenová data. Vytvoření nového prvku kmenových dat může vyžadovat vytvoření nových referenčních dat.[4]

Referenční data jsou často považována za podmnožinu kmenových dat. Úplný název této kategorie dat je kmenová referenční data.[5]

**Číselník** Číselník je uspořádaný seznam entit, kde je každé konkrétní entitě přiřazen jednoznačný kód. V kontextu této práce bude tento pojem

používán jako forma referenčních dat. Pojmy číselník a referenční data lze použít i jako synonyma.

Mezi číselníky patří například:

- kódy států,
- daňové identifikační čísla,
- kódy společností,
- kódy měn.

## 1.2 Správa dat

Správa dat (anglicky Data Governance – DG) je proces správy dostupnosti, použitelnosti, integrity a bezpečnosti dat v podnikových systémech na základě interních datových standardů a politik. Tento proces také řídí využití dat. Efektivní správa dat zajišťuje, že jsou data konzistentní, důvěryhodná a jsou správně používaná.[6]

Bez efektivní správy dat může dojít k jejich nekonzistenci v různých systémech napříč organizací. To může zkomplikovat snahu o integraci dat a vytvářet problémy s integritou dat, které ovlivňují přesnost analýzy dat. Špatná správa dat může také bránit iniciativám v oblasti dodržování předpisů.[6]

Správa dat se týká všech jejich kategorií jako jsou například již definovaná kmenová data, referenční data nebo také transakční data či metadata. Jelikož je cílem této práce vytvořit komponentu pro správu referenčních dat, bude zde blíže popsána tato kategorie. Pro lepší přiblížení situace bude uvedeno i porovnání se správou kmenových dat.

### 1.2.1 Správa kmenových dat

Správa kmenových dat (anglicky Master Data Management – MDM) jsou procesy, které řídí správu hodnot kmenových dat k umožnění konzistentního, sdíleného a kontextového použití té nejpřesnější, aktuální a relevantní verze pravdy o základních obchodních entitách napříč systémy.[7]

Polde DAMA-DMBOK[7] existují tři primární oblasti zájmu MDM:

- Identifikace duplicitních záznamů v rámci a napříč datovými zdroji pro vytváření a udržování globálních ID a přidružených křížových odkazů pro umožnění integrace informací.
- Sjednocení napříč datovými zdroji a poskytnutí „zlatého záznamu“ nebo nejlepší verze pravdy. Tyto konsolidované záznamy poskytují společný pohled na informace napříč systémy a snaží se řešit nekonzistence názvů a adres.
- Poskytování přístupu ke zlatým datům napříč aplikacemi, buď přímým čtením, nebo replikačními kanály do OLTP a DW / BI databází.

### 1.2.2 Správa referenčních dat

Správa referenčních dat (anglicky Reference Data Management – RDM) je kontrola nad definovanými hodnotami domény (také známými jako číselníky), včetně kontroly standardizovaných termínů, hodnot kódů a jiných unikátních identifikátorů. Také je kontrolou nad obchodními definicemi pro jednotlivé hodnoty, nad obchodními vztahy v rámci a napříč číselníky. Dále se správa referenčních dat stará o konzistentní sdílené používaní přesných, aktuálních a relevantních referenčních dat pro klasifikaci a kategorizaci dat.[7]

Na rozdíl od kmenových dat se hodnoty referenčních dat mění pomalu. Hodnoty referenčních data jsou udržována jejich správci, kteří by měli o datech udržovat několik informací, mezi které patří hodnoty jejich kódů, obchodních definic, popisů, čas poslední úpravy a další.[7]

Špatná správa referenčních dat může mít velký dopad na databázové operace, business inteligence a výsledky analýzy dat. Zde je pět osvědčených postupů pro správu referenčních údajů podle deníku Lightsondata[8]:

1. Formalizace správy referenčních dat

   Referenční data často nejsou udržována, pokud není stanovena žádná odpovědná osoba. Většinou proběhne pouze počáteční načtení dat do aplikací, ale nikdo se nestará o jejich aktualizaci. Když už jsou u referenčních dat prováděny potřebné změny, dochází k tomu většinou na úrovni systémů a nikoliv na úrovni podniku.

   Nejlepším řešením tohoto problému je vytvoření jednotky pro správu referenčních dat, která bude dohlížet na správu referenčních dat v celém podniku.

2. Přihlášení se k odběru externích referenčních dat

   Pokud to jde, doporučuje se nejprve nahlédnout do referenčních dat poskytovaných třetími stranami, jako jsou ISO, SWIFT, ACORD atd. Napojení na tato data interně ušetří spoustu času a zdrojů, vyjma zdrojů potřebných k jejich nákupu a zakomponování do interních systémů.

3. Řízení interních referenčních dat

   Interní referenční data jsou specifická pro daný podnik. Na rozdíl od externích dat ne potřeba se u těch interních zaměřit na jejich kvalitu. Je potřeba definovat procesy, pokyny a zodpovědné osoby, jak bylo zmíněno v bodu jedna.

4. Správa referenčních dat na podnikové úrovni

   Jelikož se referenční data používají v systémech napříč podnikem, je potřeba řešit distribuci úprav a nově přidaných záznamů. Klasickým řešením jsou způsoby poskytující centrální umístění referenčních dat, ze kterého si může libovolná aplikace importovat svá referenční data.

5. Verzování referenčních dat

   Jelikož mají referenční data v systémech převahu nad ostatními typy dat, musí být zajištěna nejen jejich aktuální verze v celém podniku, ale také sledování jejich změn. To je obzvláště užitečné v projektech integrace dat, řešení správy kmenových dat a výstupů business inteligence. K tomu je potřeba znát datum účinnosti změny a i původní verzi dat.

## 1.3 Nástroj DATA_FRAME Application

DATA_FRAME Application je soubor nástrojů navržený a vytvořený firmou Profinit EU, s.r.o. Podporuje automatizaci vývoje datového skladu na základě modelem řízené metodologie.[9] Modelem řízená metodologie spočívá ve vytváření modelů nebo abstrakcí systému nebo dat za účelem zvýšení základní kompatibility mezi systémy.[7]

„DATA_FRAME Application neurčuje typ databáze, nad kterou má běžet cílový datový sklad nebo data mart. Aplikace předpokládá vytvoření řešení založené na konceptu Profinit profilu.“[9]

DATA_FRAME Application se skládá ze čtyř komponent, z nichž některé jsou již implementovány a některé jsou pouze navrženy jako budoucí součást aplikace.

### 1.3.1 Designer

Nástroj DF_DESIGNER slouží jako uložiště metadat s jednotným modelem umožňující popis struktury a transformací cílového řešení. Dále obsahuje uložiště šablon a generátor jednotlivých konfiguračních jednotek.[9] Tato komponenta je již implementovaná a je používána na několika projektech.

Mezi základní funkce tohoto modulu patří správa definice profilu a modelu. U profilu se definuje typ pro lokalizaci, tabulku, sloupec a transformaci, QA požadavky a šablona. Pro definici modelu je možné uvést umístění tabulek v datovém skladu, definovat tabulky a sloupce a jejich atributy a definovat transformace. Komponenta také umožňuje vtvořit vazby mezi jednotlivými tabulkami či indexy nad nimi.[9]

Mimo základní funkce nabízí komponenta také správu verzí modelů a porovnávání mezi nimi, dále export a import aktuální verze profilu nebo modelu, generátor objektů, QA validace aktuální verze a vytvoření dodávky připravené pro nahrání do cílového systému.[9]

### 1.3.2 Scheduler

DF_SCHEDULER představuje universální nástroj na řízení procesů v datovém skladu. „Umožňuje definovat a spouštět procesy složité, procesy sestavené z jednotlivých tasků, řídit návaznosti jednotlivých tasků, řešit chybové stavy

a sbírat operativní metadata.“[9] Realizace této části je tématem bakalářské práce Ondřeje Závodného.

Pomocí nástroje Scheduler by mělo být možné definovat a upravovat definice procesů a jejich plány spuštění, plánování úkolů podle daných požadavků, vyhledávání v historii procesů i úkolů nebo například řešení chybových stavů.

### 1.3.3 Reference data

DF_REFERENCE_DATA je komponenta pro správu a historizaci číselníků cílového řešení. Vytvoření této komponenty je cílem této diplomové práce. Podrobnější popis funkčních i nefunkčních požadavků je uveden dále v tomto dokumentu.

### 1.3.4 Metadata

Poslední nástroj slouží k „ukládání, verzování a analýzu metadat cílového řešení“[9]. Tato komponenta ještě není implementována.

## 1.4 Existující řešení

Byla provedena rešerše mnoha existujících řešení pro správu číselníků. V této sekci jsou uvedeny řešení dvou velkých dodavatelů, kteří se tímto problémem zabývají. Je popsán jejich pohled na číselník jako takový a funkcionality, které uživateli nabízejí.

### 1.4.1 InfoSphere MDM Ref DM Hub

InfoSphere MDM Ref DM Hub neboli IBM InfoSphere Master Data Management Reference Data management Hub je nezávislá komponenta nástroje IBM Master Data Management Product ID (PID) vytvořená firmou IBM. Komponenta poskytuje možnost centralizovaně definovat a spravovat referenční data. Také umožňuje mapování mezi různými reprezentacemi jednotlivých číselníků.[10]

#### 1.4.1.1 Struktura

Nástroj od IBM obsahuje tři hlavní typy referenčních dat. Mezi ně patří sety, mapy a hierarchie.

**Sety** Hlavním stavebním kamenem domény referenčních dat jsou sety. Set je seznam hodnot datového prvku, který se používá ke klasifikaci dat. Hodnoty v setu mají dva povinné atributy – kód a jméno. Dále mohou obsahovat hodnotu s překladem jména a mohou být rozšířeny o uživatelem definované vlastnosti a to jak na úrovni hodnot, tak na úrovni celého setu.

**Mapy** Jedním typem vztahu mezi dvěma sety představují mapy. Ty jsou používány k přiřazení hodnot z jednoho setu k hodnotám z druhého. Jedná se o spojení dvou setů reprezentujících významově stejná data, pouze v jiném formátu. Tento typ je používán pro provázání setů mezi různými systémy.

**Hierarchie** Stejně jako mapy, i hierarchie představují vztah mezi hodnotami. Na rozdíl od map ale nespojují dva různé sety, ale definují vztah mezi hodnotami v jednom setu. Hierarchie může využívat řada podnikových informačních systémů, ale většinou jsou používány v reportech systémů pro business inteligence.

Nástroj dále obsahuje několik podpůrných objektů. Mezi ně patří typy, složky a řízené systémy a subskripce. Typy slouží k upřesnění atributů a vlastností setů a map. Nástroj nabízí několik výchozích typů. Možnost vytvoření složek usnadňuje uživateli organizaci setů. Každá ze složek může mít nastavená jiná přístupová práva, čímž je možné ovlivnit přístupnost setů pro různé role. Referenčním datům je možné také nastavit subskripce systémům, které je využívají.[10]

#### 1.4.1.2 Funkce

InfoSphere MDM Ref DM Hub je navržena jako ready-to-run aplikace. Její instalace je rychlá a její použití je velice jednoduché. Mezi její klíčové funkce patří:

- na rolích založené uživatelské rozhraní zajišťující bezpečnost a nastavení přístupových práv včetně integrace na LDAP,
- správa setů referenčních dat a jejich hodnot,
- správa mapování a vztahů mez sety referenčních dat,
- import a export referenčních dat ve formátu CSV a XML za použití dávkového souboru či uživatelského rozhraní,
- podpora verzování setů a map referenčních dat,
- nastavení kontroly procesu pomocí konfigurovatelných životních cyklů,
- správa hierarchií.[10]

Jedna z hlavních funkcí uvedených výše, která zde bude více rozveden, je nastavení kontroly procesu pomocí životního cyklu. Vždy při vytváření setu mu správce přiřadí jeden z životních cyklů. V aplikaci jsou v základu dostupné čtyři životní cykly popsané níže, další mohou být vytvořeny při počáteční konfiguraci aplikace.

**Jednoduché schvalování** Mezi základní životní cykly patří jednoduchý proces schvalování. Při něm musí být každá úprava setu nejdříve schválena jiným uživatelem, než se stane oficiální.

**Stavový automat** Druhým životním cyklem je stavový automat, u kterého je možné provádět změny pouze před publikováním celého setu, poté již hodnoty v setu upravovat nelze.

**Aktivně editovatelný** Další životní cyklus umožňuje měnit set kdykoliv. Provedené změny jsou ihned provedeny, nemusí být nikým schváleny.

**Dvojité schvalování** Posledním základním životním cyklem je proces dvojitého schvalování. Tento proces se od jednoduchého procesu schvalování liší pouze v tom, že každá změna musí být schválen dvěma různými uživateli namísto jednoho.[10]

### 1.4.2 MDM – Reference 360

Informatica MDM – Reference 360 je cloudová služba od firmy Informatica, která organizacím slouží k vytváření, správě a řízení referenčních dat. Aplikace umožňuje spravovat různé reprezentace stejných hodnot pomocí přechodů. Také nabízí zobrazení historických informací o provedených změnách v referenčních datech.[11]

#### 1.4.2.1 Struktura

Koncept nástroje Informatica MDM – Reference 360 je postaven na několika základních komponentách.

**Sety referenčních dat** Všechna referenční data jsou kategorizována v setech. Set referenčních dat je logické uskupení referenčních dat. Sety obsahují číselníky, kterým slouží jak jako obal, tak jako předloha. Každému setu je při vytváření nastavena definice struktury a atributů, po vytvoření již měnit nejdou. Sety mají dva speciální typy – hierarchické a se závislostí. Hierarchický set umožňuje vytvořit hierarchii hodnoty v rámci číselníků, které obsahuje. Sety se závislostí mohou mít rodičovskou závislost na jiném referenčním setu.

**Číselníky** Číselníky jsou seznamy unikátních hodnot. Hodnoty v číselníku reflektují číselníkové hodnoty, které jsou použité ve zdrojové aplikaci. Číselník dědí definici svojí struktury a atributů ze setu referenčních dat. Stejně jako sety mohou být číselníky hierarchické nebo se závislostí. Chování funguje obdobně jako u setů.

**Přechody** Set referenčních dat může obsahovat několik číselníků a každý z nich obsahuje variantu stejných dat. Přechody jsou způsob, jakým

je možné vytvořit jednosměrné mapování mezi hodnotami v takovýchto dvou číselnících. Pro vytvoření ekvivalentního vztahu je potřeba vytvořit dva přechody, každý jedním směrem.

**Atributy** Při vytváření číselníku je možné definovat, jaké atributy budou muset hodnoty splňovat. V základu má každý číselník tři připravené atributy a to jméno, kód a popis, z nichž jméno a kód jsou povinné.[11]

#### 1.4.2.2 Funkce

Reference 360 nabízí následující funkcionalitu:

- vytvoření, správa a řízení referenčních dat,
- vytvoření přechodů mezi číselníky k přeložení rozdílných reprezentací stejných termínů v různých aplikacích,
- schválení změn v číselnících předtím, než se stanou součástí publikované verze referenčních dat,
- zobrazení historie změn v referenčních datech.[11]

Jednou z klíčových funkcí systému, která zde bude rozvedena je schvalovací proces. Akce vytvoření či editace číselníku prochází přes schvalovací proces, který je v aplikaci předpřipraven. Nejdříve se vytvoří navrhovaná verze číselníku, kterou může upravovat pouze autor tohoto návrhu. Po potvrzení změn je odeslána notifikace správci Reference 360, který může změnu publikovat nebo vrátit k přepracování.

Při každé změně v číselníku je vytvořena jeho nová verze. Uživateli je vždy zobrazena poslední publikovaná verze. Pokud existuje ještě neschválená verze setu, je možné si zobrazit její porovnání s verzí publikovanou. V porovnání jsou zobrazeny upravené i nezměněné hodnoty.

Další z klíčových funkcí je zobrazení historie. V aplikaci je možné si zobrazit změny provedené v referenčních datech, číselnících i přechodech. V historii je zobrazeno o každé změně několik údajů, a to původní a nová hodnota, událost, která nastala, ID uživatele, který hodnotu změnil a datum a čas změny.

Nástroj také nabízí možnost vyhledávání v referenčních datech, přiřazení přístupových rolí jednotlivým uživatelům, import a export hodnot číselníku a REST API, které může být použito například na import a export modelů setů referenčních dat, import číselníků a jejich mapování nebo načtení změn a jejich podrobností.[11]

## 1.5 Funkční požadavky

Na základě provedených rešerší a požadavků od zadavatele bylo stanoveno několik funkčních požadavků na komponentu pro správu referenčních dat. Tyto požadavky budou popsány v této kapitole a bude podle nich následně probíhat návrh a realizace nástroje.

### 1.5.1 Správa číselníku

Nástroj musí umožňovat vytvoření číselníku. Každý číselník bude mít nastavitelný název, u kterého bude kontrolována unikátnost, dále popis číselníku, schvalovací cyklus, který se bude na číselník aplikovat a umístění číselníku ve složkové struktuře.

Každý číselník bude možné editovat nebo smazat. V rámci editace bude umožněna i změna názvu číselníku.

### 1.5.2 Správa atributů

Ke každému číselníku bude možné vytvořit atributy. Každý z nich, nezávisle na sobě, bude moci být nastaven jako primární (jeden číselník může mít primární klíč složený z více atributů) nebo jako cizí klíč odkazující na jiný číselník v rámci aplikace. Atributy budou mít také nastavitelný datový typ a validaci platnou pro všechny jeho hodnoty.

U každého z atributů bude možnost jeho smazání nebo editace názvu či jeho popisu. Vždy při vytváření nového atributu nebo při přejmenování jednoho z již existujících se bude kontrolovat unikátnost jeho názvu v daném číselníku. Stejný název atributů ve dvou rozdílných číselnících bude povolen.

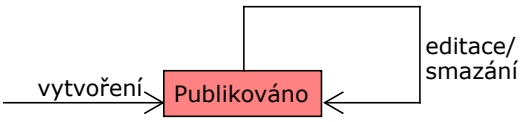
### 1.5.3 Správa záznamů

V každém číselníku bude možné přidávat, mazat a editovat záznamy. Každý nově vytvářený záznam bude mít strukturu podle aktuální podoby číselníku a jeho atributů. Každá nově vkládaná nebo editované hodnota bude validována podle vlastností atributu, kterému přísluší, neboli bude zkontrolován její datový typ a validace, pokud je u atributu uvedena. Také bude kontrolována unikátnost primárního klíče a existence cizího klíče v odkazovaném číselníku.

Každá operace se záznamy bude podléhat schvalovacímu procesu, který je přiřazen danému číselníku. Při zobrazení číselníku uživateli bude zobrazena vždy aktuální platná verze záznamů – záznamy v neschváleném stavu nebudou součástí aktuální verze.

### 1.5.4 Hierarchizace

V nástroji bude možné vytvářet složkovou strukturu a s její pomocí hierarchizovat číselníky. Složky budou editovatelné i mazatelné. Smazat složku půjde,




Obrázek 1.1: Stavový diagram procesu bez schvalování

jen pokud v ní nebude žádný číselník, ani složka. Složky bude možné do sebe libovolně vkládat a přesouvat i spolu s jejich obsahem.

### 1.5.5 Schvalovací proces

Nástroj bude pro každý číselník umožňovat nastavení schvalovacího procesu. Budou přednastaveny dva procesy.

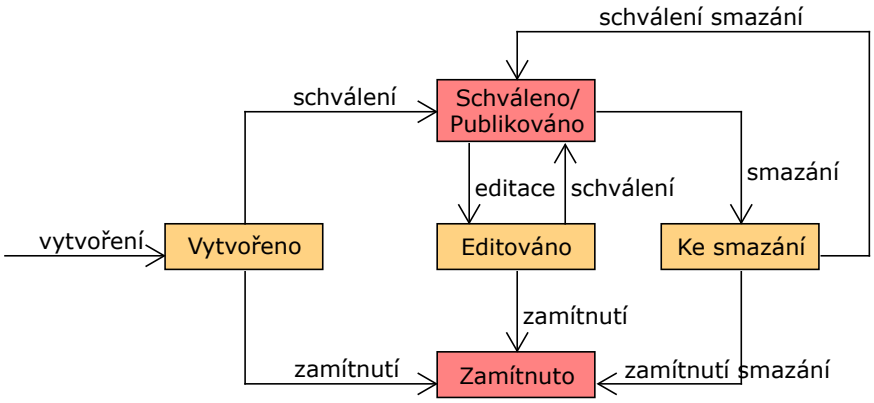
**Bez schvalování** První proces, který bude dostupný je proces bez schvalování. V tomto případě bude každá editace záznamu ihned publikována. Stejně tak i při vytvoření a smazání záznamu nebude vyžadováno žádné schválení jinou osobou. Stavový diagram tohoto procesu je zobrazen na obrázku 1.1.

**Se schvalováním** Druhým procesem je proces se schvalováním. Při každé změně záznamu v číselníku, ať už při editaci, vytvoření nového záznamu nebo smazání, bude vytvořen návrh dané změny. Tento návrh musí být následně schválen nebo zamítnut jiným uživatelem, který k tomu má oprávnění. Nová hodnota bude publikována až po jejím schválení. Podrobnější popis přechodů mezi stavy záznamu jsou znázorněny ve stavovém diagramu procesu se schvalováním na obrázku 1.2.

### 1.5.6 Uživatelské role

V aplikaci bude možné vytvářet a upravovat uživatelské role. Každý uživatel aplikace bude moci mít přeřazeno více rolí. Každé roli bude možné nastavit různá přístupová práva k jednotlivým číselníkům číselníku. Aplikace bude podporovat čtyři typy přístupových práv.

**Čtení** Základním právem pro přístup k číselníku bude právo pro jeho čtení. Role s tímto oprávněním bude moci vidět obsah i strukturu číselníku, nebude však moci číselník jakkoliv upravovat. Také nebude mít přístup k neschváleným změnám. Role s jakýmkoli jiným oprávněním pro práci s číselníkem má automaticky přiřazeno i toto oprávnění.




Obrázek 1.2: Stavový diagram procesu se schvalováním

**Úpravy záznamů** Role s tímto oprávněním bude moci upravovat, přidávat a mazat záznamy číselníku. Také bude mít možnost zobrazit si seznam neschválených změn. S neschválenými změnami a strukturou číselníku nebude moci role s tímto oprávněním provádět žádné operace.

**Schvalování změn** Toto oprávnění umožní uživateli schvalovat a zamítat změny provedené v záznamech číselníku. Oprávnění nebude uživateli umožňovat záznamy editovat.

**Administrace** Role s oprávněním k administraci číselníku bude moci upravovat strukturu daného číselníku, tedy upravovat název číselníku, jeho detaily a spravovat jeho atributy. Uživatel s tímto oprávněním nemusí mít nutně oprávnění i pro schvalování záznamů či jejich úpravu.

### 1.5.7 Vyhledávání záznamů

Nástroj bude podporovat vyhledávání záznamů podle jejich hodnot.

### 1.5.8 Historizace

Nástroj bude automaticky historizovat všechny úpravy číselníků. Změny detailu číselníků a atributů nebudou podléhat schvalovacímu procesu, ale budou kompletně historizovány. Pro číselníky s procesem se schvalováním budou historizovány i všechny návrhy na změnu včetně procesu jejich schválení.

V historii bude možné filtrovat podle data změny. Také bude možné si zobrazit verzi číselníku ke konkrétnímu datu v minulosti, či provedené změny za časové období. V uživatelském rozhraní musí být možné zobrazit výsledky těchto dotazů:

- Jak vypadal obsah číselníku ke konkrétnímu datu?
- Jaké změny byly provedeny v číselníku od data k datu?
- Jak se v čase měnily informace o číselníku?
- Jaké atributy měl číselník ke konkrétnímu datu?
- Ukaž změny provedené konkrétním uživatelem v daném číselníku.
- Ukaž všechny nevyřešené změny v číselníku s procesem se schvalováním.

## 1.6 Nefunkční požadavky

Pro nově vznikající komponentu pro správu číselníků byly definovány také nefunkční požadavky. Ty jsou spojeny zejména s faktem, že nástroj vzniká jako součást již existující aplikace a je vhodné, aby jeho vývoj respektoval již zaběhnuté postupy a konvence.

### 1.6.1 Použité technologie

Jelikož je aplikace DATA_FRAME Application vytvořena v nástroji APEX, musí být tento nástroj použit i pro vytvoření její nové komponenty. V nástroji APEX budou pouze volání služeb databáze. V databázi budou obsažena kompletní funkcionalita celé komponenty.

### 1.6.2 Stabilita datového modelu

Je dán požadavek na to, aby byl datový model komponenty stabilní – neměl by se měnit v průběhu provozu aplikace.

### 1.6.3 Uživatelské rozhraní

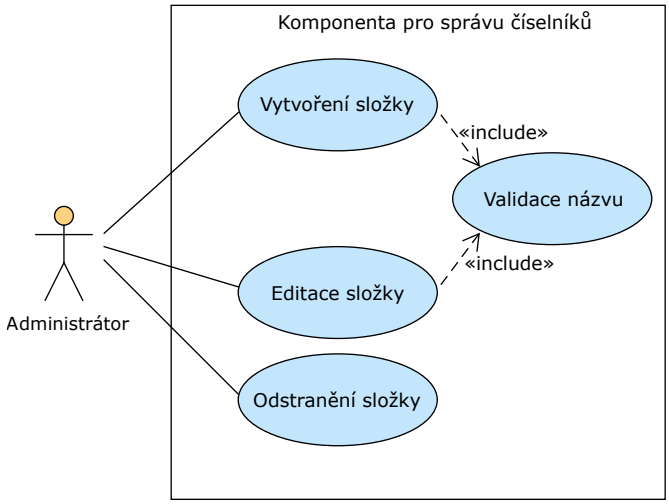
Uživatelské rozhraní nově vznikající komponenty by mělo být graficky sladěné s již existující částí aplikace. Pro uživatele tak bude jednodušší se v jednotlivých komponentách aplikace orientovat.

### 1.6.4 Autentizace

Pro autentizaci a správu uživatelů bude použito rozhraní již hotové základní komponenty aplikace. Pro přístup do komponenty pro správu číselníků bude vytvořena nová role na úrovni celé aplikace.

### 1.6.5 Autorizace

V komponentě bude umožněna autorizace pomocí uživatelských rolí. Bude možné nastavit jinou autorizaci pro každý z číselníků.




Obrázek 1.3: Případy užití pro správu složkové struktury

## 1.7 Případy užití

V této části budou popsány způsoby, jakými bude možné nástroj na správu číselníků využívat. Protože je možností využití více, byly rozděleny do několika částí podle logických celků aplikace. Pro každý celek byl zvlášť vytvořen diagram případů užití.

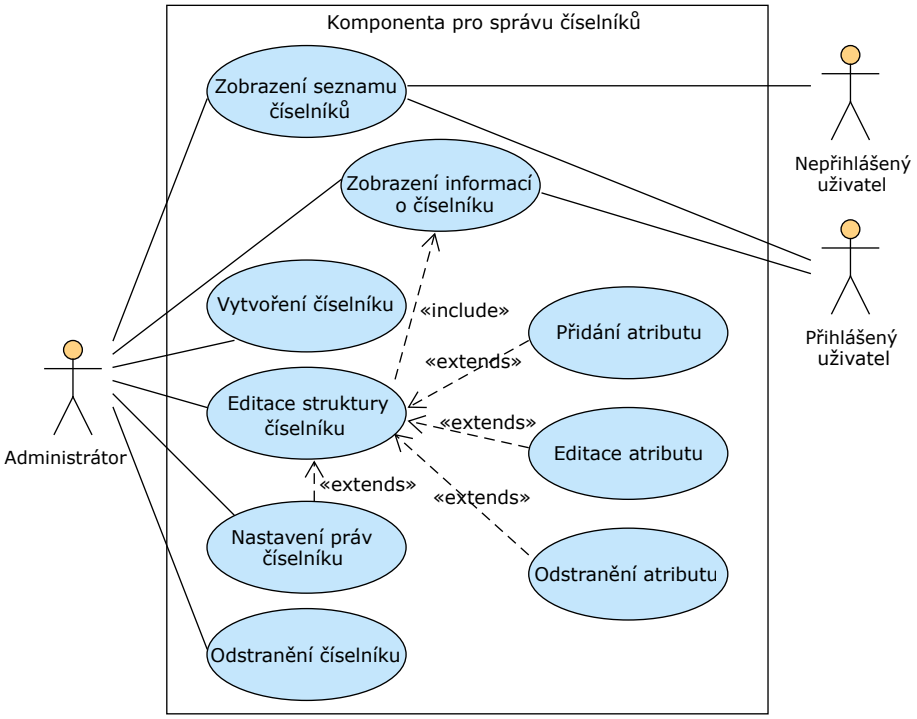
### 1.7.1 Správa složkové struktury

Jedním z logických celků aplikace je správa složkové struktury. Jak je vidět na diagramu na obrázku 1.3, jedná se o základní akce vytvoření, úpravy a odstranění složky. Vykonávání všech těchto akcí je podmíněno oprávněním pro administraci komponenty.

**Vytvoření složky** Složky bude možné vytvářet. Při vytváření složky bude uživatelem zadán její název a umístnění v již existující složkové hierarchii. Při potvrzení hodnot bude provedena kontrola jedinečnosti názvu složky ve složce, do které je nová složka vytvářena.

**Editace složky** U složek bude možné upravit její název a umístění. U obou změn bude prováděna stejná kontrola jako při vytváření složky, tedy kontrola jedinečnosti názvu složky v cílové složce. Přesunutí složky bude probíhat společně s jejím obsahem.

**Odstranění složky** Nástroj bude umožňovat odstranění existujících složek. Odstranit bude možné pouze prázdnou složku, tedy složku bez číselníků či dalších složek.




Obrázek 1.4: Případy užití pro správu číselníků

### 1.7.2 Správa číselníků

Dalším z logických celků aplikace je správa číselníků. V této části jsou uvedeny akce, které je možné vykonávat za účelem nastavení struktury číselníku včetně jeho vlastností a atributů. Jak můžete vidět na obrázku 1.4, vykonávání některých akcí je podmíněno přístupovými rolemi, které má uživatel přiřazené.

**Zobrazení seznamu číselníků** V nástroji bude možné zobrazit si stromovou strukturu složek a číselníků v nich vytvořených. Při kliknutí na detail složky bude zobrazen seznam číselníků, které obsahuje. Tyto informace budou dostupné pro všechny uživatele bez ohledu na jejich oprávnění.

**Zobrazení informací o číselníku** Každý přihlášený uživatel, jež má práva alespoň pro čtení číselníku si bude moci zobrazit jeho podrobnější informace. Mezi tyto informace patří název, popis, umístění a typ schvalovacího cyklu číselníku, dále také veškeré informace o jeho aktuálních atributech. V rámci těchto informací budou pouze aktuální data, historie změn nebude zobrazena.

**Vytvoření číselníku** Administrátorovi komponenty správy číselníků bude umožněno vytváření nových číselníků. Při vytváření musí uživatel za-

dat název, popis, umístění a typ schvalovacího cyklu číselníku. Při potvrzení hodnot uživatelem, bude provedena kontrola unikátnosti názvu číselníku.

**Editace struktury číselníku** Uživatelům, kteří jsou administrátory komponenty nebo číselníku, bude umožněno editovat jeho strukturu. V rámci editace bude uživatel moci provést několik úkonů.

Prvním z nich je editace podrobností o číselníku, jako je jeho popis a umístění. Číselník bude také možné přejmenovat, v tom případě bude probíhat kontrola unikátnosti jeho nového názvu stejně, jako u vytváření číselníku nového.

Další částí editace je správa atributů. Uživatel bude moci vytvořit, upravit či smazat atributy. U každého atributu bude probíhat kontrola jedinečnosti jeho názvu v rámci číselníku.

Poslední možností při editaci číselníku je nastavení jeho práv, to je podrobněji popsáno v následujícím případu užití.

**Nastavení práv číselníku** Administrátoři komponenty a číselníku budou také moci nastavit a upravovat přístupová oprávnění pro daný číselník. Každé uživatelské roli komponenty pro správu číselníků můžou být nastavena různá práva pro různé číselníky.

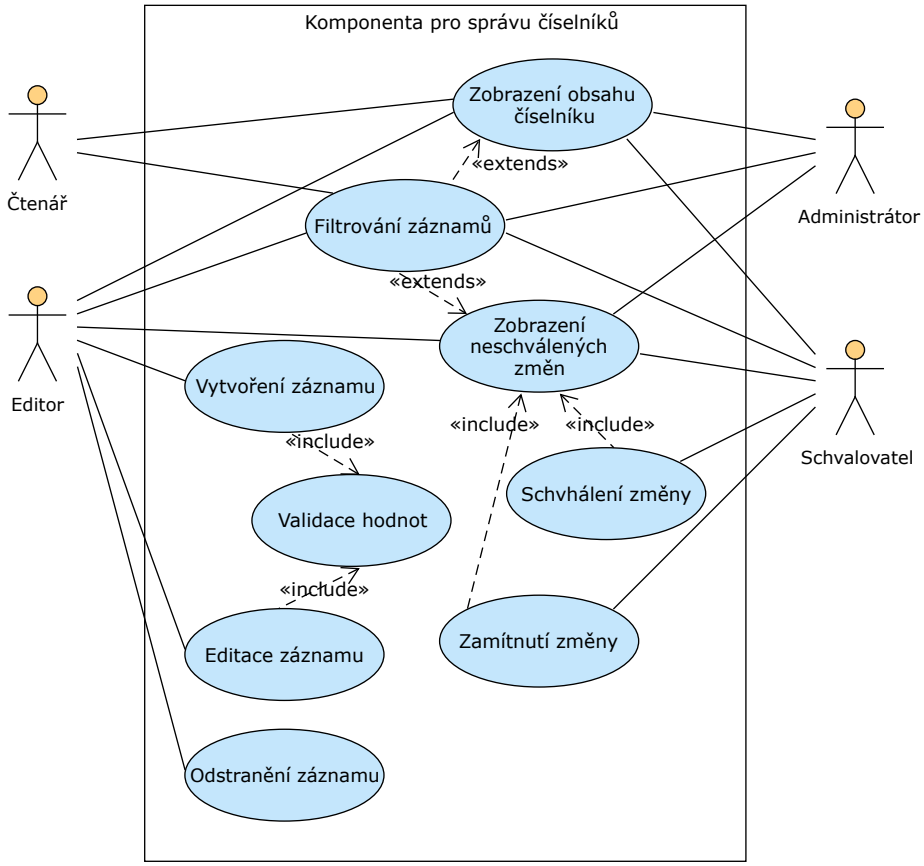
**Odstranění číselníku** Kterýkoliv číselník musí být možné odstranit. Práva na jeho odstranění budou mít, stejně jako na jeho editaci, administrátoři daného číselníku nebo komponenty. Při mazání nebude číselník smazán z paměti, bude pouze označen jako nevalidní a bude možné ho zpětně dohledat v historii.

### 1.7.3 Správa záznamů

Následujícím logickým celkem používání nástroje je správa záznamů v jednotlivých číselnících. V této části jsou aktivity zaměřeny na vytváření a úpravu záznamů a na práci se změnami v číselníku s procesem se schvalováním. Jejich diagram je na obrázku 1.5. I v této části mohou některé aktivity vykonávat pouze oprávněné role. Žádná z těchto akcí není dostupná pro nepřihlášeného uživatele.

**Zobrazení obsahu číselníku** U každého číselníku bude uživateli, který má práva alespoň na jeho čtení, umožněno zobrazit si seznam všech jeho aktuálních záznamů. V seznamu je možné filtrovat či řadit záznamy podle hodnot jejich atributů.

**Vytvoření záznamu** Každý uživatel s právy pro editaci číselníku bude moct přidat do něj nový záznam. Uživatel vyplní všechny potřebné atributy nového záznamu. Při potvrzení hodnot proběhne jejich validace včetně

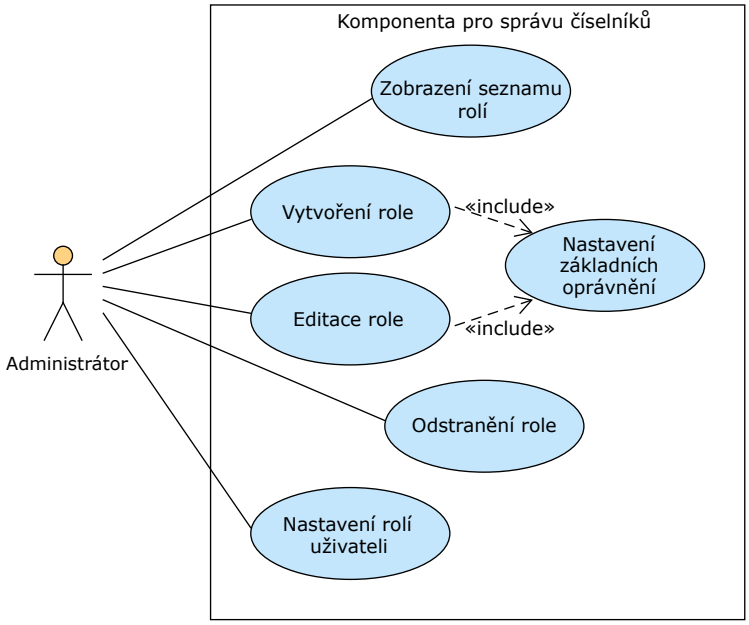



Obrázek 1.5: Případy užití pro správu záznamů

kontroly unikátnosti primárního klíče. Při úspěšném vytvoření záznamu bude buď záznam přidán do číselníku, nebo bude vytvořena žádost o jeho přidání. Která z těchto akcí bude provedena se rozhodne podle typu schvalovacího cyklu daného číselníku.

**Editace záznamu** Editace záznamu bude probíhat obdobně, jako jeho vytvoření. V rámci editace bude možná i změna hodnoty primárního klíče. Při potvrzení změny proběhne validace hodnot, stejně jako při vytváření nového záznamu, stejně tak bude probíhat i publikování změny, které se řídí typem schvalovacího procesu číselníku.

**Odstranění záznamu** Editorovi číselníku bude také umožněno smazat libovolný záznam. Proces jeho smazání bude také podléhat procesu schvalování daného číselníku. Stejně jako u číselníků, ani záznamy nebudou při smazání úplně odstraněny, vždy je bude možné dohledat v historii číselníku.




Obrázek 1.6: Případy užití pro správu rolí

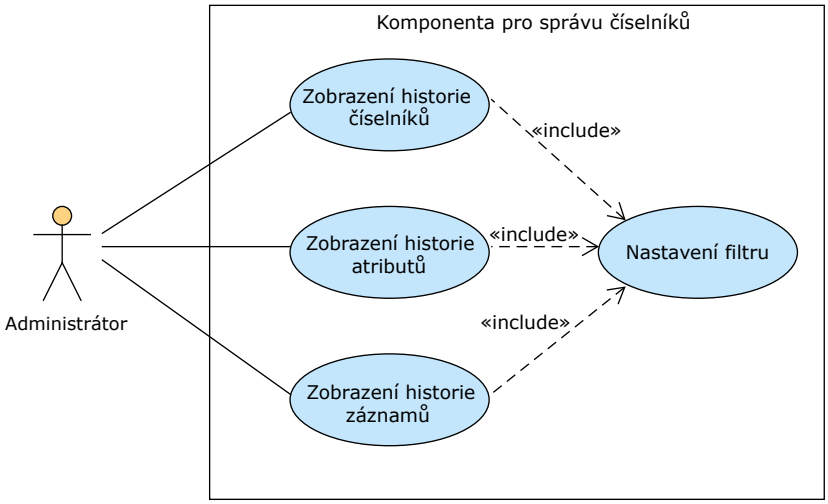
**Zobrazení neschválených změn** Pro editora, administrátora a schvalovatele číselníku bude přístupný seznam neschválených změn jeho záznamů. V tomto seznamu budou zobrazeny vždy původní hodnoty záznamu a hodnoty nově navrhované. V seznamu bude možné filtrovat a řadit podle hodnot v záznamech.

Uživatelé s oprávněním schvalovat změny v daném číselníku budou mít možnost schválit či zamítnout každou ze změn zvlášť nebo i všechny najednou. Po schválení bude změna publikována, po jejím zamítnutí bude odstraněna a její hodnoty nebudou nijak využity. Zamítnutou změnu, stejně tak, jako tu schválenou, bude možné dohledat v historii záznamů.

### 1.7.4 Správa rolí

V této části budou rozebrány případy užití spojené se správou rolí. Ke všem těmto akcím má přístup pouze přihlášený uživatel v roli administrátora komponenty. Všechny aktivity jsou zachyceny v diagramu na obrázku 1.6.

**Zobrazení seznamu rolí** Nástroj bude nabízet zobrazení seznamu rolí komponenty. V seznamu bude zobrazen vždy název, popis role a seznam základních oprávnění.




Obrázek 1.7: Případy užití pro zobrazení historie

**Vytvoření role** Administrátor bude moci přidat novou roli do komponenty. U nové role se bude zadávat její název, popis a nastavení základních oprávnění. Základní oprávnění budou používána při vytvoření nového číselníku jako základní oprávnění dané role pro daný číselník.

**Editace role** Uživatel bude mít možnost role upravovat. Součástí úpravy bude změna názvu, popisu i nastavení základních oprávnění. Upravená základní oprávnění budou platná pouze pro nově vytvářené číselníky, pro ty již vytvořené práva rolí zůstanou tak, jak byla.

**Odstranění role** Jednotlivé vytvořené role bude možné odstranit. Při jejich odstranění bude role odebrána všem uživatelům, kterým byla dříve přiřazena, a bude odebrána ze všech oprávnění pro práci s číselníky.

**Nastavení rolí uživateli** Každému uživateli bude možné přiřadit libovolný počet existujících rolí. Role budou moci být přiřazeny i odebrány. Bude-li mít uživatel více rolí, bude mít práva k přístupu, pokud je bude mít alespoň jedna z jeho rolí.

### 1.7.5 Zobrazení historie

Posledním logickým funkčním celkem komponenty, jehož případy užití můžete vidět na obrázku 1.7, je zobrazení historie číselníků. Tato část je, stejně jako správa rolí, dostupná pouze pro uživatele s rolí administrátora komponenty.

**Zobrazení historie číselníků** Nástroj umožní zobrazení historie všech provedených změn vlastností číselníků. Zobrazen bude seznam číselníků s je-

jich vlastnostmi, datem vytvoření, datem změny/odstranění a jmény uživatelů, kteří danou verzi číselník vytvořili a upravili. V rámci filtrování těchto záznam bude možné nastavit, jestli chce uživatel vidět platné záznamy v daném období, nebo jenom provedené změny. V seznamu bude možné vyhledávat podle jednotlivých vlastností číselníku, autora, editora i času vytvoření a změny.

**Zobrazení historie atributů** Další historizovanou částí jsou atributy. Bude možné zobrazit si historii změna atributů v každém číselníku. Seznam změn atributů bude možné filtrovat stejným způsobem jako seznam změn číselníků, který je popsán výše.

**Zobrazení historie záznamů** Stejně jako v případě atributů a číselníků, i u záznamů bude možné zobrazit si jejich historii. Filtrování v seznamu historie záznamů bude fungovat stejným způsobem jako i u atributů a číselníků. U záznamů bude navíc možnost zobrazení buď pouze publikovaných změn nebo i návrhů na změny, které byly následně schváleny, ale i zamítnuty.

Kapitola 2

# Návrh

Následující kapitola této práce se věnuje návrhu nástroje. Návrh byl vytvořen s ohledem na provedené analýzy požadavků a případů užití. Je zde uveden datový model celého nástroje na práci s číselníky, je definován způsob, jakým se budou data do databáze ukládat a jak se bude provádět jejich historizace. Dále je také popsáno chování aplikace při práci se záznamy a je vytvořen návrh budoucího uživatelského rozhraní.

## 2.1 Způsob datové reprezentace číselníku

Jednou z věcí, které je potřeba v této práci určit, je, jakým způsobem bude reprezentován číselník v datovém modelu. Na entitu číselníku je možné se dívat z několika úrovní:

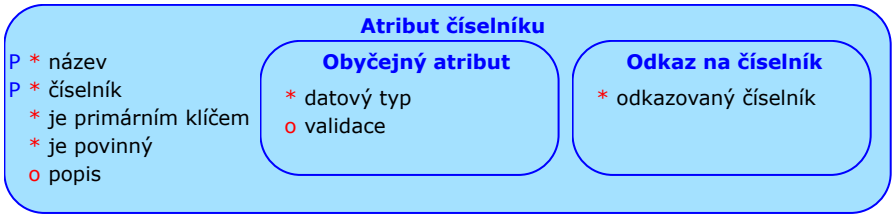
1. Informace o číselníku

   Prvním nejobecnějším pohledem je pohled na číselník jako na celek. U číselníku lze definovat několik jeho základních charakteristik, jako je název, popis a konkrétněji v případě této práce ještě typ jeho schvalovacího cyklu a umístění ve složkové hierarchii.

2. Informace o jeho struktuře

   Další informací o číselníku je jeho struktura. Tou jsou myšleny jeho atributy a informace o nich. Každý číselník může mít libovolný počet atributů a každý z nich má různé parametry, které jsou důležité buď pro číselník jako celek nebo pro jeho hodnoty. Mezi tyto informace patří název atributu, jeho popis, datový typ jeho hodnot a validace, kterou musí hodnoty tohoto atributu splňovat. Více atributů číselníku může být určeno jako identifikátor jednotlivých uložených referenčních dat. Atribut také může představovat odkaz do jiného číselníku.

3. Informace o jeho obsahu




Obrázek 2.1: Konceptuální model – atributy číselníku

Poslední částí číselníku je jeho obsah. Obsahem číselníku jsou konkrétní data. Data jsou ve formě záznamů, kde každý záznam má hodnoty pro jednotlivé atributy číselníku.

Veškeré informace uvedené v těchto třech bodech je nutné ve vznikajícím nástroji uchovávat a to i s historií jejich změn. Jelikož je možné tyto tři body od sebe oddělit, budou v datovém modelu řešeny zvlášť. Konkrétní řešení jednotlivých částí je popsáno níže.

### 2.1.1 Informace o číselníku

Základní informace o číselníku budou v datovém modelu reprezentovány jednou tabulkou. V tabulce budou sloupce pro jednotlivé vlastnosti číselníku. Jelikož jde jen o jednoduchý výčet vlastností, nemusí se v této úrovni pohledu řešit nic složitějšího.

### 2.1.2 Atributy

V případě informací o atributech se jedná z hlediska konceptuálního modelu o ISA hierarchii (podtypy), kterou můžete vidět na obrázku 2.1. Při transformaci na relační model existují tři možnosti, jak tento případ vyřešit:

1. Vytvořit pouze jednu tabulku, která bude obsahovat atributy nadtypu i obou podtypů. V tomto případě nebude při získávání množiny atributů jednoho číselníku nutné propojovat více tabulek, všechny informace budou v jedné. Nevýhodou této metody je to, že pro každý záznam v tabulce budou některé atributy nevyužity – budou prázdné.

2. Rozdělit hierarchii na tři tabulky. Jedna tabulka bude obsahovat informace společné pro obě entity a další dvě tabulky budou představovat dva podtypy. V tabulkách podtypů budou jejich atributy a cizí klíč do tabulky nadtypu spojující danou entitu. V tomto případě nejsou v modelu žádná nevyužité hodnoty, jak tomu bylo v první variantě, ale při získání informací o jedné entitě je vždy nutné provázat dvě tabulky.

3. Vytvořit dvě tabulky, každou pro jeden typ entity. V tomto případě by obě tabulky obsahovaly jak atributy specifické pro podtyp, tak i atributy nadtypu. Touto variantou se od sebe typy kompletně oddělí, a pokud je potřeba přistoupit k oběma typům dohromady, neboli je k nim potřeba přistupovat jako k nadtypu, je nutné nějakým způsobem tabulky spojit.

Jelikož v rámci vytvářeného nástroje bude potřeba přistupovat k oběma podtypům dohromady jako k atributům, byla k převedení na relační datový model vybrána první varianta. Vznikne tedy jedna tabulka obsahující atributy obou typů dohromady. Jelikož se předpokládá, že většina z atributů číselníku budou obyčejné, nebudou to odkazy, bude tak i většina polí tabulky využita. Existence hodnoty `odkazovaný číselník` v záznamu bude navíc určovat typ atributu číselníku.

### 2.1.3 Záznamy

Poslední částí číselníku, kterou je potřeba v databázi reprezentovat, je jeho obsah neboli záznamy. Byly navrženy dva způsoby, jak záznamy do databáze ukládat. Podrobný popis obou a výběr toho, který bude použit v řešení, je popsán níže.

#### 2.1.3.1 Tabulky

Prvním způsobem, jak obsah číselníků do databáze ukládat, je tak, jak je zobrazen uživateli, neboli jako tabulky. Každý číselník by v tomto případě měl svojí vlastní tabulku, která by byla vytvořena při vytvoření číselníku. Název tabulky by se shodoval s identifikátorem číselníku a názvy sloupců by byly tvořeny identifikátory jednotlivých atributů číselníku.

Výhody tohoto řešení jsou následující:

- Jelikož je číselník uložen v databázi ve stejném formátu, jako je reprezentován uživateli, je jednoduché ho tímto způsobem zobrazit.
- Přidání nového záznamu je jednoduché vložení jednoho řádku do tabulky.
- Je možné zajistit automatickou kontrolu datového typu vkládaných hodnot.

Řešení má samozřejmě i nějaké nevýhody, mezi které patří:

- Při vytvoření číselníku je nutné vytvořit novou tabulku, čímž dojde ke změně datového modelu.
- Každá úprava atributů číselníku znamená změnu struktury tabulky.

#### 2.1.3.2 Seznam hodnot

Druhou možností, jak reprezentovat záznamy, je rozdělit je na jednotlivé hodnoty a ty vložit do jedné tabulky společné pro všechny číselníky. Vznikla by jedna tabulka, která by měla sloupce s identifikátorem atributu, kterému hodnota přísluší, s identifikátorem záznamu, který je danou hodnotou tvořen a se samotnou hodnotou atributu. Pro jednodušší práci s celými číselníky by byl u každé hodnoty uveden i identifikátor číselníku. Vždy při přidání nového záznamu do číselníku by se vložila každá z hodnot v řádku do uvedené tabulky zvlášť.

Mezi hlavní výhody této varianty patří:

- Vytvoření nového číselníku znamená pouze přidání záznamy do tabulky s jeho informacemi, není nutné provádět žádné jiné změny v databázi.
- Při změně atributů dochází pouze ke změně v tabulce s informacemi o atributech.
- Při uživatelské práci s programem nedochází ke změně datového modelu, žádné nové tabulky nejsou vytvářeny.

Nevýhody uložení všech hodnot všech číselníků do jedné tabulky jsou:

- Jelikož jsou záznamy v číselníku rozdělené na jednotlivé hodnoty, které jsou uloženy zvlášť, jejich sestavení do formátu tabulky, která bude prezentována uživateli, není triviální.
- Veškeré validace hodnot, včetně jejich datového typu, musejí probíhat programově.
- Při vkládání záznamu číselníku je nutné cyklicky vložit každou z jeho hodnot zvlášť.

#### 2.1.3.3 Výběr

Pro potřeby nového nástroje byla zvolena druhá varianta reprezentace obsahu číselníku, tedy seznam hodnot. Bylo tak zvoleno kvůli požadavku na stabilitu datového modelu 1.6.2 a možnosti obejít se bez DDL operací, tedy operací umožňujících vytvářet, upravovat a mazat objekty databáze.

## 2.2 Způsob historizace dat

Podle funkčních požadavků musí být všechny změny v rámci číselníků historizovány a musí být zpětně dohledatelné. Jelikož změny číselníků probíhají velice zřídka, je jejich chování, co se týče změn přirovnatelné k pomalu se

Original row in Product dimension:

| Product Key | SKU (NK) | Product Description | Department Name | ... | Row Effective Date | Row Expiration Date | Current Row Indicator |
|---|---|---|---|---|---|---|---|
| 12345 | ABC922-Z | IntelliKidz | Education | ... | 2012-01-01 | 9999-12-31 | Current |

Rows in Product dimension following department reassignment:

| Product Key | SKU (NK) | Product Description | Department Name | ... | Row Effective Date | Row Expiration Date | Current Row Indicator |
|---|---|---|---|---|---|---|---|
| 12345 | ABC922-Z | IntelliKidz | Education | ... | 2012-01-01 | 2013-01-31 | Expired |
| 25984 | ABC922-Z | IntelliKidz | Strategy | ... | 2013-02-01 | 9999-12-31 | Current |

Obrázek 2.2: Pomalu se měnící dimenze typu 2[13]

měnícím dimenzím v kontextu dimenzionálního modelování. Pro způsoby zachycení změn těchto dat existuje několik různých variant, které jsou popsány níže.

### 2.2.1 Pomalu se měnící dimenze

„Dimenze je definována, v kontextu dimenzionálního modelování, jako typ tabulky nebo strukturální atribut datové krychle obsahující seznam členů, z nichž všechny jsou podobného typu v uživatelském vnímání dat. Například všechny měsíce, čtvrtletí, roky atd. tvoří časový rozměr, stejně jako všechna města, regiony, země atd. tvoří geografický rozměr. Dimenze funguje jako index pro identifikaci hodnot v rámci vícerozměrného pole. Dimenze nabízejí velmi výstižný a intuitivní způsob uspořádání a výběru dat pro vyhledávání, průzkum a analýzu.“[12]

„Pomalu se měnící dimenze (anglicky Slowly Changing Data – SCD) jsou dimenze obsahující data, která se v průběhu času mění tak, že míra změny je malé procento z celkového počtu řádků (tj. 1% za čtvrtletí).“[12]

Ralph Kimball[13] rozlišuje osm základních typů pomalu se měnících dimenzí podle způsobu nakládání s jejich změnami:

**Typ 0** Hodnota atributu se nikdy nemění, zůstává stále stejná již od jejího vytvoření.

**Typ 1** Stará hodnota je přepsána novou. Hodnota tedy vždy reprezentuje poslední změnu, žádné minulé verze nejsou dohledatelné.

**Typ 2** Při změně hodnoty je vložen nový záznam, který reflektuje novou verzi hodnoty. Po úpravě jsou tedy, jak je vidět na obrázku 2.2, v tabulce dva řádky patřící k jedné identitě (zde se jedná konkrétně o produkt IntelliKidz).

V případě typu 2 by se mělo do tabulky přidat několik administrativních sloupců, jak je vidět na obrázku 2.2. Data začátku a konce platnosti

Original row in Product dimension:

| Product Key | SKU (NK) | Product Description | Department Name |
|---|---|---|---|
| 12345 | ABC922-Z | IntelliKidz | Education |

Updated row in Product dimension:

| Product Key | SKU (NK) | Product Description | Department Name | Prior Department Name |
|---|---|---|---|---|
| 12345 | ABC922-Z | IntelliKidz | Strategy | Education |

Obrázek 2.3: Pomalu se měnící dimenze typu 3[13]

označující momenty, kdy byl řádek označen jako validní a kdy byl zneplatněn.

Typ 2 dále vyžaduje vytvoření náhradního primárního klíče. Přirozený klíč již nebude možné použít, protože se jeho hodnota bude objevovat ve všech verzích dané entity. Na obrázku 2.2 je uveden ještě jeden přidaný atribut určující, zda je řádek aktuální či zastaralý. Tento řádek není nutný, jeho hodnotu lze odvodit z data konce platnosti řádku.

**Typ 3** V rámci typu 3 jsou v tabulce uchovávány dvě (může ale být i více) verze hodnoty. Pro uchování druhé verze hodnoty je vytvořen nový atribut, ve kterém je hodnota uložena. Na obrázku 2.3 je uveden příklad, u kterého je dostupná aktuální a poslední předchozí verze hodnoty, všechny starší hodnoty není již možné dohledat.

**Typ 4** Tento typ je známý jako mini dimenze. Používá se pro rozsáhlé dimenze s mnoha řádky, u kterých dochází často ke změně. Tento problém lze vyřešit oddělením často analyzovaných nebo často upravovaných atributů do vlastní dimenze, které se říká mini dimenze. Protože chceme udržet co nejnižší počet řádků v mini dimenzi, používají se často místo konkrétních hodnot intervaly. Pro uchování změn se s mini dimenzí se pracuje jako s typem 2.

**Typ 5** Typ 5 je kombinací mini dimenze, tedy typu 4, a typu 1. Podstatou tohoto způsobu je přidání nového atributu do hlavní dimenze. Nový atribut je typu 1 a odkazuje na aktuální řádek v mini dimenzi. Hlavní dimenze společně s minidimenzí je v tomto případě uživateli reprezentována jako jedna tabulka.

**Typ 6** V tomto případě jde o použití typu 2 s přidaným novým atributem typu 1. Jak je vidět na obrázku 2.4, nově přidaný atribut představuje aktuální hodnotu atributu. Na obrázku 2.4 je typ 6 reprezentován atributem `Department Name`, u kterého je uvedena jeho aktuální a historická hodnota. S atributem `Historic Department Name` je nakládáno

Original row in Product dimension:

| Product Key | SKU (NK) | Product Description | Historic Department Name | Current Department Name | ... | Row Effective Date | Row Expiration Date | Current Row Indicator |
|---|---|---|---|---|---|---|---|---|
| 12345 | ABC922-Z | IntelliKidz | Education | Education | ... | 2012-01-01 | 9999-12-31 | Current |

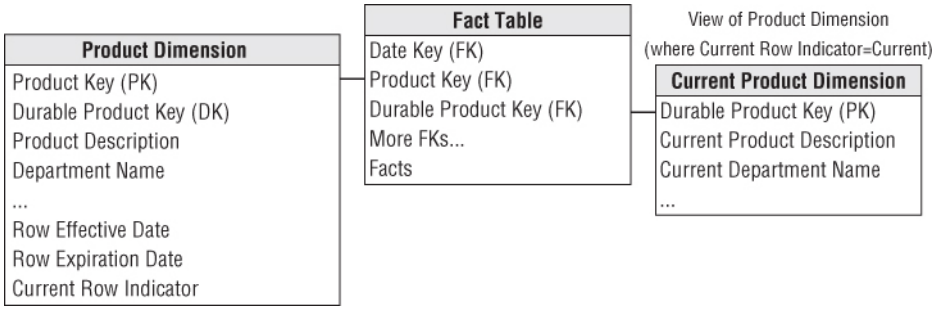
Rows in Product dimension following first department reassignment:

| Product Key | SKU (NK) | Product Description | Historic Department Name | Current Department Name | ... | Row Effective Date | Row Expiration Date | Current Row Indicator |
|---|---|---|---|---|---|---|---|---|
| 12345 | ABC922-Z | IntelliKidz | Education | Strategy | ... | 2012-01-01 | 2013-01-31 | Expired |
| 25984 | ABC922-Z | IntelliKidz | Strategy | Strategy | ... | 2013-02-01 | 9999-12-31 | Current |

Rows in Product dimension following second department reassignment:

| Product Key | SKU (NK) | Product Description | Historic Department Name | Current Department Name | ... | Row Effective Date | Row Expiration Date | Current Row Indicator |
|---|---|---|---|---|---|---|---|---|
| 12345 | ABC922-Z | IntelliKidz | Education | Critical Thinking | ... | 2012-01-01 | 2013-01-31 | Expired |
| 25984 | ABC922-Z | IntelliKidz | Strategy | Critical Thinking | ... | 2013-02-01 | 2013-06-30 | Expired |
| 31726 | ABC922-Z | IntelliKidz | Critical Thinking | Critical Thinking | ... | 2013-07-01 | 9999-12-31 | Current |

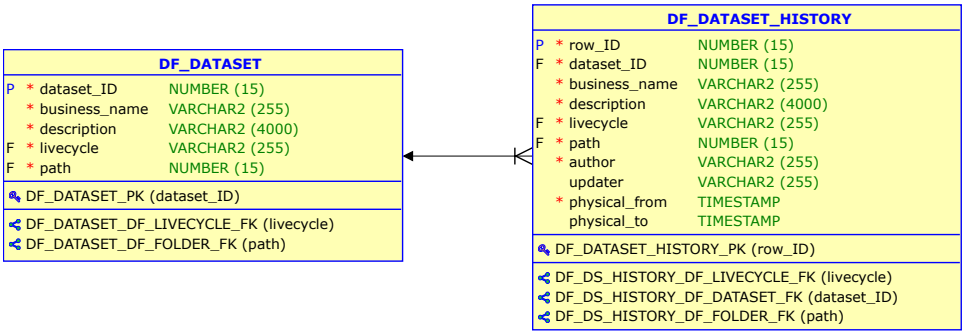
Obrázek 2.4: Pomalu se měnící dimenze typu 6[13]




Obrázek 2.5: Pomalu se měnící dimenze typu 7[13]

jako s dimenzí typu 2 a s atributem `Current Department Name` jako s typem 1.

**Typ 7** V tomto případě jde o rozšíření typu 6 tak, aby bylo možné ho použít pro mnoho atributů najednou v jedné dimenzi. Typ 7 spočívá ve vytvoření dvou tabulek pro jednu dimenzi. Jak je vidět na obrázku 2.5, jedna z tabulek obsahuje historii dat a je spravována jako typ 2, druhá tabulka obsahuje pouze aktuální data a je spravována jako typ 1. Obě tabulky obsahují stejný počet řádků, pouze s jinými hodnotami měněných atributů.




Obrázek 2.6: Datový model způsobu historizace informací o číselníku

### 2.2.2 Výběr

V rámci práce je potřeba vybrat způsob historizace zvlášť pro informace o číselníku, atributy číselníku a jeho hodnoty. U všech tří entit je nutné udržovat historii všech provedených změn. Na základě výše uvedených způsobů prací se změnami v pomalu se měnících dimenzích byly vybrány způsoby, jakým budou probíhat historizace tří uvedených entit.

Pro tabulky atributů a hodnot byl vybrán typ 2. Tento typ bude použit v souladu s potřebami vznikající aplikace. Obě tabulky budou mít navíc umělý primární klíč identifikující verzi entity neboli řádek. Dále budou podle vzoru SCD2 přidány sloupce s datem počátku a konce platnosti. Aby bylo možné sledovat nejen čas úprav, ale i autora, budou do tabulky přidány také sloupce pro autora řádku a pro editora řádku. Editor řádku musí být přidán, aby bylo možné případně zaznamenat uživatele, který entitu smazal, v tom případě se totiž nový řádek nevytváří.

Pro entitu s informacemi o číselníku je situace jiná. Protože na tuto entitu odkazují obě zmíněné tabulky, byla vybrána variace typu 7. Budou vytvořeny dvě tabulky pro tuto entitu. Jedna z tabulek bude obsahovat pouze aktuální informace o číselnících a bude tak možné přímo odkazovat na identifikační číslo číselníku. Druhá tabulka bude obsahovat veškerou historii číselníků a to pomocí SCD2. Jak je zobrazeno na obrázku 2.6, historická tabulka bude obsahovat cizí klíč do tabulky aktuálních verzí číselníků, tím budou tyto dvě tabulky propojeny. Na rozdíl od SCD typu 7 ale nebudou obě tabulky obsahovat stejný počet řádků. Tabulka s aktuálními verzemi číselníků bude vždy obsahovat jen jeden řádek příslušný jednomu číselníku.

## 2.3 Datový model

Na základě analýz způsobů řešení byl vytvořen relační datový model pro novou komponentu. Jelikož model obsahuje mnoho tabulek, byl pro přehlednost rozdělen do tří logických celků. Těmito celky jsou informace o číselnících, role

a oprávnění a schvalovací procesy. Všechny části lze dohromady propojit pomocí hlavní tabulky komponenty, kterou je `DF_DATASET` představující číselník.

### 2.3.1 Informace o číselníku

První část datového modelu, která je zobrazena na obrázku 2.7, slouží pro uložení dat o číselnících. Tato část obsahuje podrobný popis tabulek jednotlivých čísti číselníku, jako jsou například atributy či záznamy.

**DF_FOLDER** Tato tabulka představuje složku. Jelikož může existovat více složek se stejným názvem, je tato tabulka identifikována generovaným identifikačním číslem. Složky musí být možné hierarchizovat do stromové struktury, proto je zde vytvořen rekurzivní vztah, kterým je určena nadřazená složka.

**DF_DATASET** Tabulka číselníku obsahuje základní informace o něm. Mezi ty patří jeho generované identifikační číslo, název, popis, schvalovací proces a rodičovská složka. Tato tabulka neobsahuje žádné informace o historii číselníku.

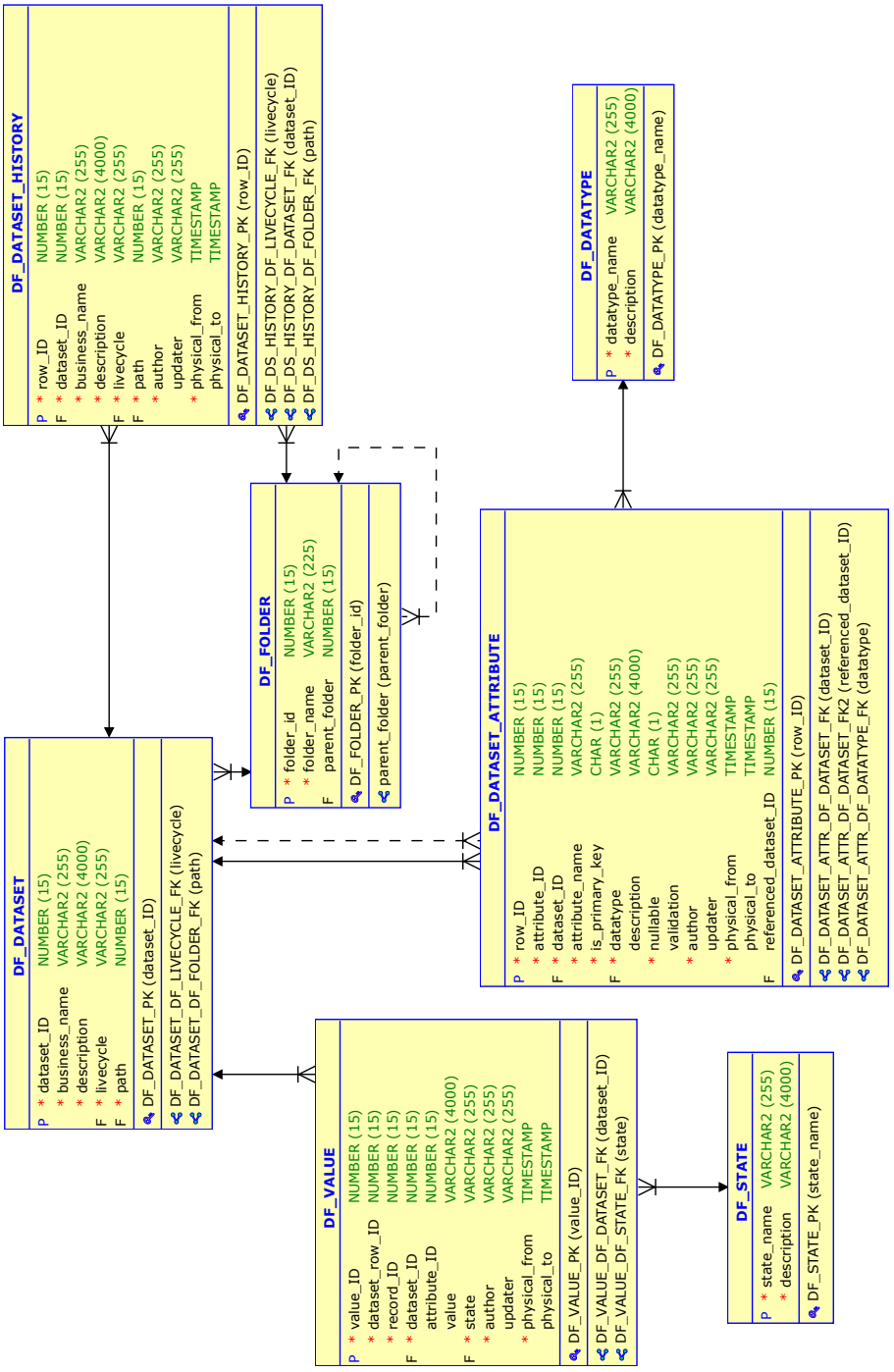
**DF_DATASET_HISTORY** `DF_DATASET_HISTORY` představuje historizovanou verzi tabulky `DF_DATASET`. Navíc tedy obsahuje informace potřebné pro SCD2 historizaci, neboli identifikaci řádku, datum platnosti od a do a autora záznamu a jeho změny. Autor změny je v tabulce uveden, aby bylo možné zaznamenat uživatele, který provedl akci smazání číselníku.

Historizace byla oddělena od `DF_DATASET`, aby bylo možné v ostatních tabulkách vytvořit cizí klíč v podobě id číselníku. V historizované tabulce id číselníku není unikátní, a proto na něj cizí klíč nelze vytvořit.

**DF_DATASET_ATTRIBUTE** Ke každému číselníku může být přiřazeno více atributů, ty představuje tabulka `DF_DATASET_ATTRIBUTE`. Každý atribute je určen generovaným identifikačním číslem, má název, datový typ a popis. Dále je určeno, jestli je primárním klíčem, jestli může být prázdný a může být uvedena validace pro jeho hodnoty. Také může být atribut cizím klíčem, v tom případě je určen číselník, který představuje referencovanou tabulku.

Stejně jakou číselníky, i atributy jsou historizovány metodou SCD2. Z této metody vyplívá, že jsou v tabulce navíc informace o času vzniku a úpravy a autor a editor záznamu, také je každý řádek identifikován svým generovaným id.

**DF_DATATYPE** V databázi je nutné uchovávat možné datové typy atributů. Každý datový typ je určen jeho názvem a je u něj uveden popis.




Obrázek 2.7: Relační datový model – informace o číselníku

**DF_VALUE** Poslední částí číselníku, kterou je potřeba uložit v databázi, je tabulka `DF_VALUE` neboli hodnoty. Každá hodnota musí mít uvedeno, do jakého číselníku patří a k jakému atributu se vztahuje. K definici číselníku je použit cizí klíč do tabulky `DF_DATASET`. Definice atributu nelze vytvořit jako cizí klíč do tabulky `DF_DATASET_ATTRIBUTE`, jelikož identifikátor atributu není v cílové tabulce unikátní. Existence atributu v tabulce `DF_DATASET_ATTRIBUTE` bude tedy kontrolována programově.

I u hodnot je použita historizace pomocí metody SCD2, jsou zde tedy navíc stejné informace, jako jsou i v tabulkách `DF_DATASET_HISTORY` a `DF_DATASET_ATTRIBUTE` potřebné pro tuto metodu. V `DF_VALUE` je navíc uveden stav, který udává stav schvalovacího cyklu číselníku.

Pro identifikaci hodnoty je uvedeno několik atributů. První z nich je `value_ID`, ten slouží jako identifikace řádku v tabulce `DF_VALUE` a je součástí atributů potřebných pro metodu SCD typu 2. Dalším identifikátorem je `record_ID`, který propojuje hodnoty příslušející k jednomu záznamu v číselníku. Posledním atributem potřebným pro identifikaci je `dataset_row_ID`. Tento atribut vznikl také kvůli historizaci záznamů pomocí metody SCD2 a představuje identifikaci jednotlivých verzí jednoho záznamu.

### 2.3.2 Role a oprávnění

Další část datového modelu, která je uvedena na obrázku 2.8, se zabývá rolemi a přístupovými oprávněními k číselníku.

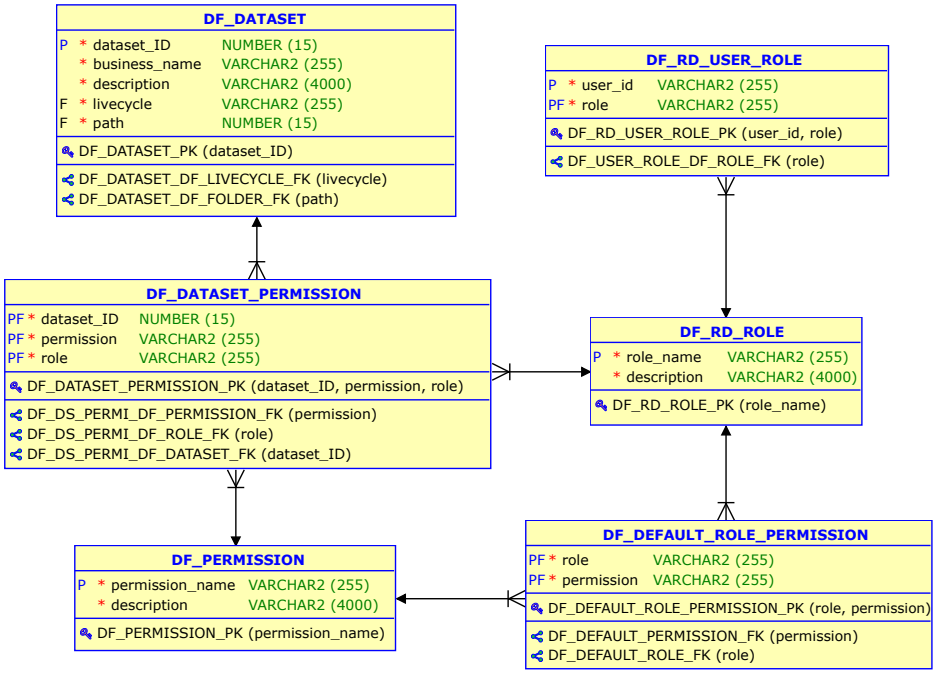
**DF_RD_ROLE** Tabulka `DF_RD_ROLE` představuje uživatelské role v komponentě. Každá role je určena jejím názvem a je u ní uveden popis.

**DF_RD_USER_ROLE** Propojení rolí komponenty a uživatelů zajišťuje tabulka `DF_RD_USER_ROLE`. Tabulka uživatelů je součástí základu aplikace DATA_FRAME Application, není tedy uvedena v datovém modelu této komponenty. Každá z rolí může být přiřazena více uživatelům a každý z uživatelů může mít více rolí.

**DF_PERMISSION** Tato tabulka představuje oprávnění pro používání jednotlivých číselníků. Oprávnění je definováno názvem a je u něj uveden popis.

**DF_DATASET_PERMISSION** K propojení rolí s oprávněními a číselníky slouží tabulka `DF_DATASET_PERMISSION`. Ta udává, jaké role mají jaká oprávnění k práci s jednotlivými číselníky.

**DF_DEFAULT_ROLE_PERMISSION** Každá role má nastavena základní oprávnění, které jsou představovány touto tabulkou.




Obrázek 2.8: Relační datový model – role a oprávnění

### 2.3.3 Schvalovací procesy

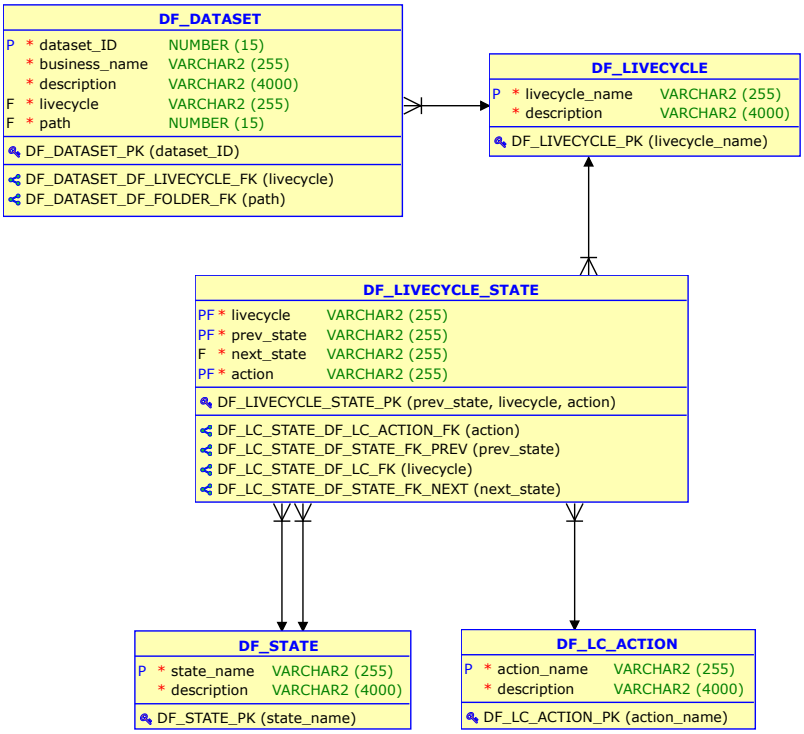
Poslední částí datového modelu jsou tabulky spojené se schvalovacími procesy. Jejich model je na obrázku 2.9.

**DF_LIVECYCLE** Tabulka `DF_LIVECYCLE` představuje schvalovací procesy v komponentě. Každá role je určena jejím názvem a je u ní uveden popis. Každému číselníku je přiřazen jeden schvalovací proces.

**DF_STATE** Tato tabulka představuje stav schvalovacího procesu. Stav je definován názvem a je u něj uveden popis.

**DF_LC_ACTION** Další tabulkou potřebnou pro definici schvalovacího procesu je `DF_LC_ACTION`. Ta představuje akci, kterou je možné provést se záznamem. Mezi akce patří například editace, smazání nebo schválení.

**DF_LIVECYCLE_STATE** K popisu přechodů mezi stavy schvalovacího procesu podle diagramů na obrázcích 1.1 a 1.2 slouží tato tabulka. V ní je uvedeno pro schvalovací cykly, ze kterého do kterého stavu se záznam dostane pomocí jaké akce.




Obrázek 2.9: Relační datový model – schvalovací procesy

## 2.4 Proces změny záznamu

V této sekci je podrobně popsán způsob, jakým bude řešena úprava záznamů s ohledem na schvalovací cykly. Krok za krokem je popsáno, jaké hodnoty se budou vkládat do kterých atributů kterých tabulek.

I když jsou jednotlivé hodnoty záznamů v databázi uloženy zvlášť, bude následující sekce popisovat chování záznamů jako celku, tedy „jak bude vypadat výsledný dataset“. Při vkládání jakékoliv nové verze záznamu budou vždy vloženy do tabulky `DF_VALUE` všechny hodnoty nové verze záznamu, i ty nezměněné. Tento způsob byl zvolen pro zjednodušení rekompozice jednotlivých verzí záznamů.

Názvy sloupců v následujících tabulkách znázorňující změny v číselnících jsou pouze pro účely popisu číselníku v rámci tohoto textu, v databázových tabulkách komponenty se názvy sloupců liší. Pro lepší orientaci, sloupce `ID`, `záznam`, `autor`, `editor`, `platnost od`, `platnost do` a `stav` odpovídaní v tabulce `DF_VALUE` sloupcům `dataset_row_id`, `record_id`, `author`, `updater`, `physical_from`, `physical_to` a `state` v uvedeném pořadí. Všechny tyto tabulky se také řídí společnými pravidly zobrazení.

- Nové záznamy v tabulce jsou zvýrazněny zelenou barvou.

Tabulka 2.1: Vytvoření záznamu v číselníku s procesem bez schvalování

| ID | záznam | A1 | A2 | autor | editor | platnost od | platnost do | stav |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | a | c | U1 | – | 1. 1. 2001 | $\infty$ | A |

Tabulka 2.2: Editace záznamu v číselníku s procesem bez schvalování

| ID | záznam | A1 | A2 | autor | editor | platnost od | platnost do | stav |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | a | c | U1 | U2 | 1. 1. 2001 | 2. 2. 2002 | A |
| 2 | 1 | b | c | U2 | – | 2. 2. 2002 | $\infty$ | A |

Tabulka 2.3: Odstranění záznamu v číselníku s procesem bez schvalování

| ID | záznam | A1 | A2 | autor | editor | platnost od | platnost do | stav |
|---|---|---|---|---|---|---|---|---|
| 2 | 1 | b | c | U2 | U3 | 2. 2. 2002 | 3. 3. 2003 | A |

- Upravené hodnoty v předešlých záznamech jsou zvýrazněny žlutě.
- Editovaná hodnota je zvýrazněna červenou barvou.
- Stavy záznamů jsou používány následovně: A – schváleno/publikováno, E – editováno, C – k vytvoření, D – ke smazání.
- Číselník obsahuje dva atributy A1 a A2.

### 2.4.1 Proces bez schvalování

U proces bez schvalování bude úprava záznamů relativně jednoduchá, jedná se jen o metodu historizace SCD2 s přidáním informace o uživateli, který verzi záznamu upravil.

V tabulce 2.1 je zobrazeno přidání nového záznamu do číselníku s procesem bez schvalování. V tomto případě vložen záznam s požadovanými hodnotami a je u něj uveden autor a čas vzniku, editor je nevyplněn a čas editace je nastaven na maximální hodnotu. Jelikož záznam nepodléhá žádnému schvalování, je rovnou ve stavu schváleno/publikováno.

Při editaci záznamu, jejíž znázornění je v tabulce 2.2, se vytvoří řádek s editovanými hodnotami, kterému je nastaven čas vzniku na aktuální a čas editace na maximální, editor není uveden. Nová verze záznamu je, stejně jako předešlá, ve stavu schválená/publikovaná. U původního záznamu je nastavena hodnota editor na uživatele, který záznam upravil a hodnota času editace na aktuální datum, tím je stará verze záznamu zneplatněna.

Poslední akcí v číselníku s procesem bez schvalování je mazání záznamu. Při této akci se, jak můžete vidět v tabulce 2.3, pouze v aktuální verzi záznamu nastaví hodnota editora na uživatele, který záznam maže a hodnota času editace na aktuální čas.

Tabulka 2.4: Vytvoření záznamu v číselníku s procesem se schvalováním

| ID | záznam | A1 | A2 | autor | editor | platnost od | platnost do | stav |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | a | c | U1 | – | 1. 1. 2001 | $\infty$ | C |

| ID | záznam | A1 | A2 | autor | editor | platnost od | platnost do | stav |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | a | c | U1 | U2 | 1. 1. 2001 | 2. 2. 2001 | C |
| 2 | 1 | a | c | U2 | – | 2. 2. 2001 | $\infty$ | A |

Tabulka 2.5: Editace záznamu v číselníku s procesem se schvalováním

| ID | záznam | A1 | A2 | autor | editor | platnost od | platnost do | stav |
|---|---|---|---|---|---|---|---|---|
| 2 | 1 | a | c | U2 | – | 2. 2. 2001 | $\infty$ | A |
| 3 | 1 | b | c | U3 | – | 3. 3. 2003 | $\infty$ | E |

| ID | záznam | A1 | A2 | autor | editor | platnost od | platnost do | stav |
|---|---|---|---|---|---|---|---|---|
| 2 | 1 | a | c | U2 | U4 | 2. 2. 2001 | 4. 4. 2003 | A |
| 3 | 1 | b | c | U3 | U4 | 3. 3. 2003 | 4. 4. 2003 | E |
| 4 | 1 | b | c | U4 | – | 4. 4. 2003 | $\infty$ | A |

### 2.4.2 Proces se schvalováním

V číselnících s procesem se schvalováním je při úpravě/vytvoření/zmazání vytvořen nejdříve záznam představující požadavek na tuto změnu. Každá z tabulek v této sekci obsahuje dvě části. První část tabulky představuje akci vytvoření požadavku a druhá část představuje schválení požadavku, či zamítnutí v případě tabulky 2.7. I když je požadavek vytvořen, musí být stále validní původní verze záznamu, takže není nastaveno datum, které by ukončovalo její platnost. Konec platnosti i editor je vyplněn až po schválení změny.

Konkrétněji v případě vytvoření záznamu, které je popsáno v tabulce 2.4, je v prvním kroku vytvořen nový záznam, podobně jako v případě bez schvalování. Rozdíl je pouze v tom, že záznam je ve stavu „k vytvoření“ a není tedy považován za validní/publikovaný. V kroku schválení se vytvoří nová verze stejného záznamu a u požadavku se jako editor uvede schvalovatel na konec platnosti se nastaví na aktuální datum.

Při editaci záznamu se, jak už bylo zmíněno dříve, při vytvoření požadavku vytvoří pouze nová upravená verze záznamu, původní záznam zůstává nezměněn. Až po schválení změny je vytvořen nový validní záznam a původnímu validnímu záznamu i požadavku na změnu je nastaven schvalovatel jako editor a konec platnosti jim je nastaven na aktuální čas. Konec platnosti původního záznamu je nastaven na čas schválení, nikoli na čas vytvoření požadavku, z toho důvodu, aby na sebe svou platností navazovaly všechny publikované verze. Znázornění těchto dvou kroků je v tabulce 2.5

Tabulka 2.6: Odstranění záznamu v číselníku s procesem se schvalováním

| ID | záznam | A1 | A2 | autor | editor | platnost od | platnost do | stav |
|---|---|---|---|---|---|---|---|---|
| 4 | 1 | b | c | U4 | – | 4. 4. 2003 | ∞ | A |
| 5 | 1 | b | c | U5 | – | 5. 5. 2005 | ∞ | D |

| ID | záznam | A1 | A2 | autor | editor | platnost od | platnost do | stav |
|---|---|---|---|---|---|---|---|---|
| 4 | 1 | b | c | U4 | U6 | 4. 4. 2003 | 6. 6. 2006 | A |
| 5 | 1 | b | c | U5 | U6 | 5. 5. 2005 | 6. 6. 2006 | D |

Tabulka 2.7: Zamítnutí změny v číselníku s procesem se schvalováním

| ID | záznam | A1 | A2 | autor | editor | platnost od | platnost do | stav |
|---|---|---|---|---|---|---|---|---|
| 2 | 2 | a | c | U2 | – | 2. 2. 2001 | ∞ | A |
| 3 | 2 | b | c | U3 | – | 3. 3. 2003 | ∞ | E |

| ID | záznam | A1 | A2 | autor | editor | platnost od | platnost do | stav |
|---|---|---|---|---|---|---|---|---|
| 2 | 2 | a | c | U2 | – | 2. 2. 2001 | ∞ | A |
| 3 | 2 | b | c | U3 | U4 | 3. 3. 2003 | 4. 4. 2003 | E |

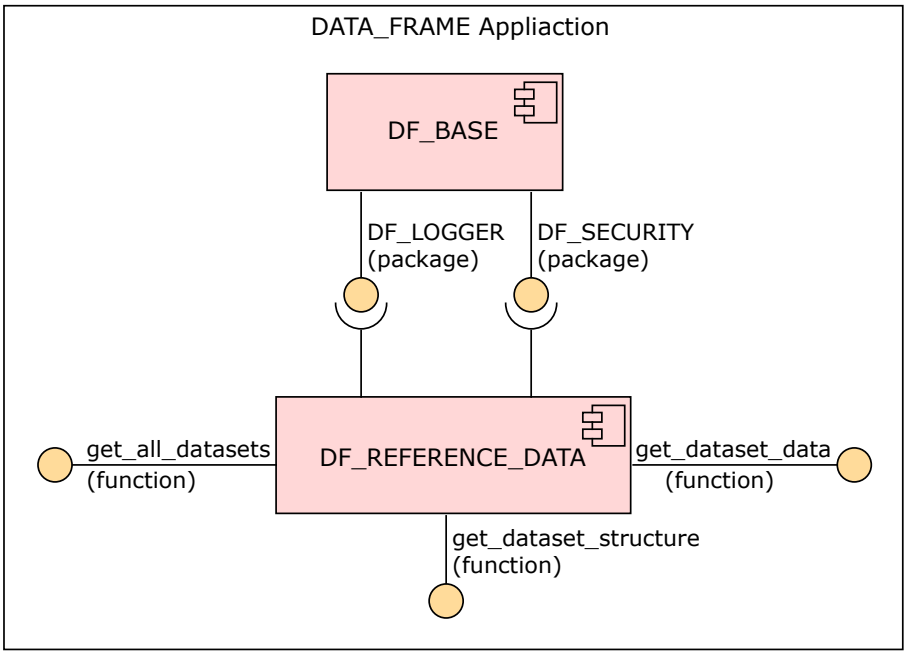
Když je záznam mazán, je nejdříve vytvořen požadavek v podobě shodného záznamu se stavem „ke smazání“. Při schválení smazání záznamu není vytvořena žádná nová verze, jen je ukončena platnost té původní a požadavku na smazání. Názorné zobrazení těchto kroků je v tabulce 2.6.

V procesu se schvalováním je možné také změnu zamítnout. Zamítnutí probíhá u všech typů úpravy stejně. V tabulce 2.7 je uveden případ editace záznamu. První krok, vytvoření požadavku na editaci, již byl popsán podrobněji výše, u editace. Druhým krokem je v tomto případě zamítnutí. Při něm se pouze nastaví uživatel, který požadavek zamítl, jako editor a konec platnosti požadavku se nastaví na aktuální datum. Původní verze záznamu se při zamítnutí požadavku nijak nemění.

## 2.5 Rozhraní

Další věcí, kterou je potřeba u nové komponenty navrhnout, je její zapojení v rámci aplikace DATA_FRAME Application. Je potřeba definovat požadavky na interface ostatních komponent a rozhraní, které bude nová komponenta vystavovat pro potřeby ostatních komponent aplikace. Na obrázku 2.10 je vidět diagram komponent potřebných pro funkčnost nově vytvářené komponenty a její rozhraní.

Co se týče rozhraní používaných komponentou pro správu číselníků, jedná se pouze o rozhraní komponenty DF_BASE, která nabízí rozhraní pro logování v rámci aplikace a rozhraní pro autentizaci uživatele. Balíček `DF_LOGGER` nabízí




Obrázek 2.10: Interface komponenty pro správu číselníků

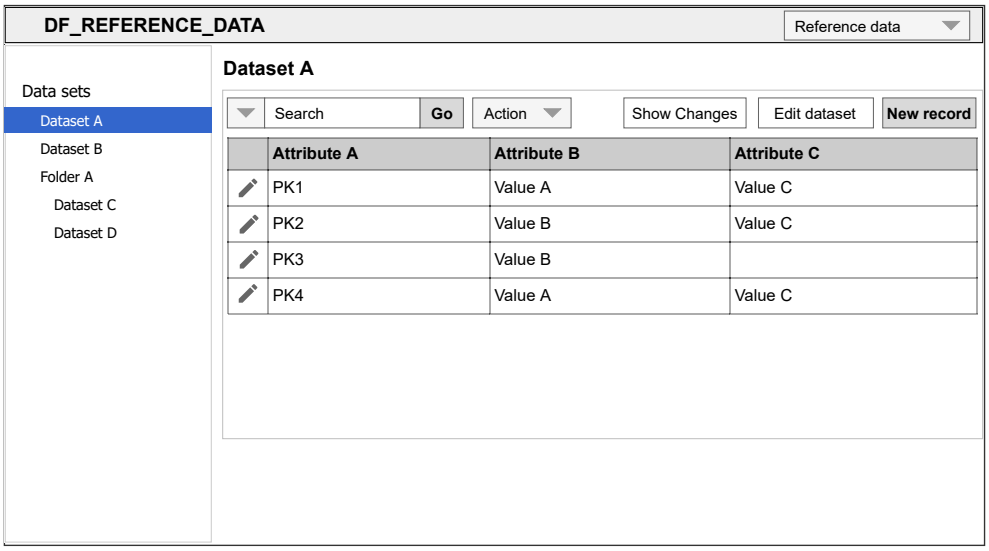
funkce a procedury pro různé úrovně logování (TRACE, DEBUG, ERROR...) a umožňuje zobrazení seznamu logovaných zpráv v uživatelském rozhraní aplikace. V balíčku `DF_SECURITY` je obsažena veškerá logika potřebná pro autorizaci a správu uživatele. Mezi její funkce patří například změna hesla, vytvoření nového uživatele nebo kontrola přihlašovacích údajů.

Komponenta pro správu číselníků bude také nabízet rozhraní pro ostatní komponenty aplikace, které potřebují s číselníky pracovat. Rozhraní se bude skládat ze tří funkcí.

**get_all_datasets** Vstupem této funkce bude pouze časové rozmezí určené dvěma daty. Výstupem bude seznam názvů číselníků, které byly validní alespoň nějakou dobu v daném časovém rozmezí.

**get_dataset_structure** Vstupem funkce pro získání struktury neboli atributů číselníku bude identifikátor požadovaného číselníku a, stejně jako u předchozí funkce, časové rozmezí určené dvěma daty. Výstupem bude tabulka ve formátu tabulky `DF_DATASET_ATTRIBUTE` s atributy odpovídajícími vstupním hodnotám funkce.

**get_dataset_data** Tato funkce bude mít stejné vstupní parametry jako ta pro získání struktury. Výstupem funkce bude tabulka ve stejném formátu, jako je databázové VIEW požadovaného číselníku a bude obsahovat pouze data validním v daném časovém rozmezí.




Obrázek 2.11: Detail číselníku

## 2.6 Uživatelské rozhraní

Návrh uživatelského rozhraní probíhal v souladu s požadavkem na jeho vzhled. Jelikož je vznikající nástroj součástí již existující aplikace, musí být v grafickém souladu s ostatními komponentami.

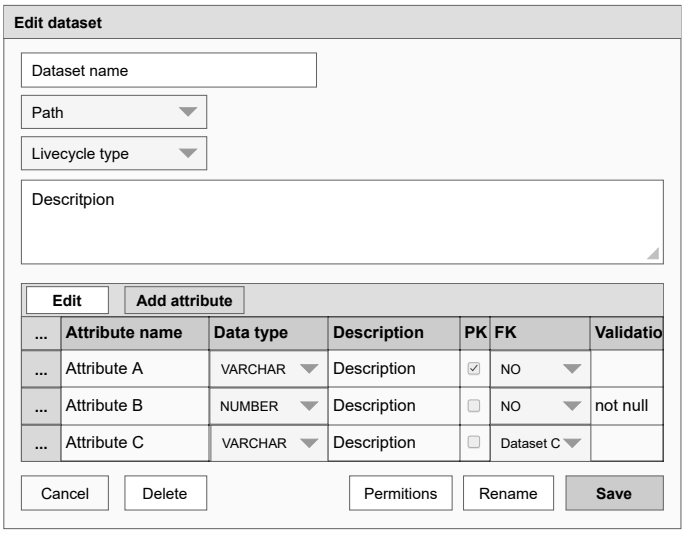
Obrazovek výsledné aplikace je více, zde je popsán návrh pouze několika z nich. Budou vynechány obrazovky pro práci s rolemi, protože se jedná z velké části o základní obrazovky nástroje APEX a budou se tedy řídit stylem tohoto nástroje. Také budou vynechány některé obrazovky pro práci s číselníky. Bude zmíněn koncept hlavních obrazovek a těch se složitější strukturou. Kompletní návrhy všech obrazovek nástroje jsou dostupné na přiloženém médiu.

Obecně platí, že na každé obrazovce aplikace je hlavní horní lišta. Ta funguje jako menu a nabízí možnost přihlášení a administrativy (součást již existující základní komponenty) a speciálně pro tuto komponentu nabízí také rozcestník mezi hlavními částmi nástroje, kterými jsou správa číselníků, správa rolí a zobrazení historie.

### 2.6.1 Detail číselníku

Jednou z hlavních obrazovek nové komponenty je detail číselníku. Jak můžete vidět na obrázku 2.11, obrazovka bude rozdělena do dvou částí.

Na levé straně obrazovky je složková struktura s existujícími číselníky. Složky jsou v seznamu obsahu složky zobrazeny jako první, po nich bude seznam číselníků v dané složce. Složky i číselníky jsou v seznamu vzájemně seřazeny podle abecedy. Na každý záznam ve struktuře bude možné kliknout, po čemž se v pravé straně obrazovky zobrazí příslušný detail.




Obrázek 2.12: Dialog pro úpravu struktury číselníku

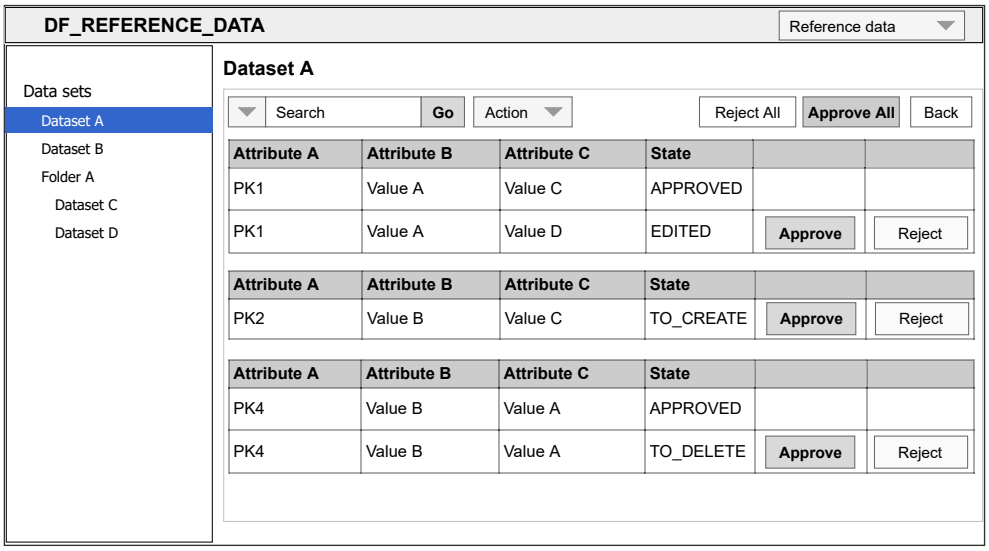
Je-li v levé části vybrána složka, zobrazí se v pravé části seznam číselníků, které obsahuje. Číselníky jsou zobrazeny ve formě tabulky a je u nich uveden jejich název, popis a typ schvalovacího procesu. V tabulce je možné vyhledávat a přidávat do ní nové číselníky. Názvy číselníků jsou interaktivní, jak tomu je i v podobných tabulkách ostatních komponent, a vedou na editaci číselníku. Obrazovka pro vytvoření či editaci číselníku je popsána níže.

Pravá strana obrazovky, která je zobrazena na obrázku 2.11, představuje detail číselníku. Ten se zobrazí, je-li v levé části obrazovky zvolen číselník. V detailu je zobrazen pouze aktuální stav číselníku. Nejsou v něm zobrazeny ani úpravy čekající na schválení, ty jsou schována pod tlačítkem „Show changes“ v pravém horním rohu tabulky. V tabulce se záznamy lze vyhledávat podle libovolných dat, záznamy lze třídit podle jednotlivých sloupců. Na levé straně každého řádku je ikona pro jeho editaci. Po kliknutí na tuto ikonu je uživatel přesměrován na obrazovku pro editaci záznamu.

### 2.6.2 Editace struktury číselníku

Při vytváření číselníku nebo úpravě jeho struktury se uživatel dostane do dialogu editace číselníku, jehož návrh je uveden na obrázku 2.12. Tento dialog je rozdělen na dvě části – informace o číselníku a atributy číselníku.

V části s informacemi o číselníku jsou textová pole pro název a popis a rozbalovací seznamy pro typ schvalovacího cyklu a složku, ve které má být číselník uložen. Jde-li o editaci číselníku, jsou tato pole předvyplněná aktuálními informacemi o něm. V rámci editace nepůjde v tomto dialogu editovat název číselníku. V případě vytváření nového číselníku jsou pole prázdná,




Obrázek 2.13: Neschválené změny

až na složku, která je předvyplněná podle místa, ze kterého se uživatel rozhodl číselník vytvořit.

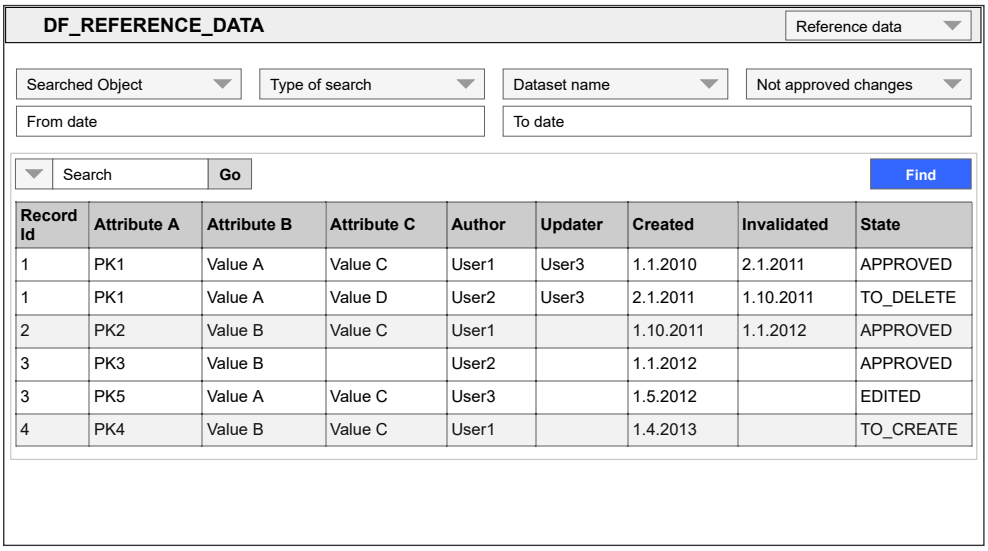
Část dialogu pro správu atributů číselníku je zobrazena, pouze pokud je číselník upravován, v případě vytváření je vynechána. V této části je uveden seznam aktuálních atributů se všemi jejich vlastnostmi. Atributy lze k číselníku přidat pomocí tlačítka „Add attribute“. Již existující atributy lze editovat. Upravit je možné všechny jejich vlastnosti až na datový typ a validaci. Vlastnosti, u kterých uživatel vybírá ze seznamu možných hodnot, jako je například datový typ nebo odkazovaná tabulka, budou vybírány pomocí rozbalovacího seznamu.

Pro potvrzení vytvořených změn slouží tlačítko „Save“ v pravém dolním rohu. Dále jsou v dialogu tlačítka pro přejmenování číselníku a nastavení přístupových práv, které obě vedou na speciální dialog pro danou akci. Uživatel může také na obrazovce pro editaci číselníku číselník smazat. Před smazáním číselníku bude po uživateli požadováno potvrzení této akce.

### 2.6.3 Neschválené změny

Po kliknutí na „Show changes“ na detailu číselníku se zobrazí obrazovka s neschválenými změnami v daném číselníku. Návrh této obrazovky je na obrázku 2.13. Levá strana obrazovka zůstává stejná jako na obrazovce detailu číselníku.

Na pravé straně obrazovky se oproti detailu číselníku změní obsah tabulky a tlačítka. Tabulka má navíc sloupec obsahující stav záznamu a na každém řádku je tlačítko pro potvrzení a pro zamítnutí změny. Tabulka bude obsahovat pouze záznamy, u kterých existuje aktuálně neschválená změna. U každého




| Record Id | Attribute A | Attribute B | Attribute C | Author | Updater | Created | Invalidated | State |
|---|---|---|---|---|---|---|---|---|
| 1 | PK1 | Value A | Value C | User1 | User3 | 1.1.2010 | 2.1.2011 | APPROVED |
| 1 | PK1 | Value A | Value D | User2 | User3 | 2.1.2011 | 1.10.2011 | TO_DELETE |
| 2 | PK2 | Value B | Value C | User1 | | 1.10.2011 | 1.1.2012 | APPROVED |
| 3 | PK3 | Value B | | User2 | | 1.1.2012 | | APPROVED |
| 3 | PK5 | Value A | Value C | User3 | | 1.5.2012 | | EDITED |
| 4 | PK4 | Value B | Value C | User1 | | 1.4.2013 | | TO_CREATE |

Obrázek 2.14: Historie

záznamu splňující tuto podmínku bude zobrazen řádek s aktuální publikovanou verzí záznamu (pokud existuje) a řádek s požadovanou změnou. Po schválení či zamítnutí změny daný záznam z tabulky zmizí.

Na stránce v pravém horním rohu tabulky budou tlačítka pro schválení a zamítnutí všech záznamů najednou. Po kliknutí na ně bude uživateli zobrazen potvrzující dialog dané akce. Také bude v pravém horním rohu tlačítko pro návrat na detail číselníku.

### 2.6.4 Historie

Další obrazovka, která zde bude popsána je historie, jejíž návrh je na obrázku 2.14. Na této obrazovce si uživatel může zobrazit historii změn číselníků, jejich atributů i jejich obsahu. K filtrování zobrazovaných změn slouží formulář v horní části obrazovky. Základními hodnotami formuláře, které jsou společné pro všechny typy vyhledávání, je datum začátku a konce požadovaného období změn.

První rozbalovací seznam určuje, zda se bude zobrazovat historie číselníků, atributů či záznamů. Při zobrazení historie číselníků je zobrazena tabulka všech číselníků se změnami provedenými v jejich popisu či názvu. Při výběru atributů či záznamů je nutné zvolit ještě číselník, pro který se mají hodnoty zobrazit. V případě atributů se zobrazí seznam všech atributů daného číselníku včetně všech informací o něm. Podobně i v případě záznamů se zobrazí obsah daného číselníku.

Další rozbalovací seznam, který je společný pro všechny typy dat, určuje typ vyhledávání. Uživatel si může zvolit mezi dvěma typy. Po zvolení jednoho

z nich budou zobrazeny všechny záznamy, které existovaly v daném období. Nezáleží na tom, kdy byly vytvořeny, či kdy byla ukončena jeho platnost, stačí, když existovaly v nějakém okamžiku z daného období. Druhou možností je zobrazit pouze změny, které byly za dané období provedeny. V tomto případě budou zobrazeny pouze záznamy, které byly v daném období vytvořeny.

Speciálně u záznamů je ještě jeden rozbalovací seznam, který určuje, zda budou zobrazeny návrhy na změnu, ať už akceptované, zamítnuté či ještě nevyřešené.

Všechny zobrazené řádky jsou vždy doplněny o autora, editora, čas začátku a čas konce platnosti. V celé tabulce je vždy možné vyhledávat a filtrovat podle jednotlivých zobrazených dat.

Kapitola 3

# Realizace

V této kapitole je popsána implementační část diplomové práce. Nejdříve jsou popsány použité nástroje. Dále je uvedeno několik implementačních částí, které bylo nutné vyřešit nějakým speciálním způsobem.

## 3.1 Nástroje

Zde jsou představeny nástroje, které byly použity pro analytickou, návrhovou a realizační část této práce. Nástroje pro testování jsou uvedené v kapitole 4 – Testování. U každého nástroje je uveden jeho název, popis a způsob, jakým byl v rámci práce využit.

**UMLet** K vytvoření stavových diagramů schvalovacích procesů a diagramů případů užití byl využit nástroj UMLet. UMLet umožňuje vytváření UML diagramů, aktivity diagramů a jejich export do různých formátů.

**Pencil** Nástroj Pencil slouží k vytvoření GUI prototypů. Umožňuje vytvoření a provázání návrhů jednotlivých obrazovek. Výsledný návrh aplikace lze exportovat do několika různých formátů, u některých z nich je zachováno provázání obrazovek, takže je možné si obrazovky aplikace proklikat bez přítomnosti fungující logiky. Pencil byl v rámci této práce použit pro vytvoření návrhů obrazovek.

**Oracle SQL Developer** Oracle SQL Developer je vývojové prostředí sloužící k vývoji a správě Oracle databází. Nástroj nabízí kompletní prostředí pro vývoj PL/SQL aplikací, list pro spouštění dotazů a skriptů, konzoli pro správu databáze a další. SQL Developer byl použit pro tvorbu backendové části aplikace v jazyku PL/SQL.

Oracle SQL Developer nabízí několik dalších funkcí. V rámci této práce byl využit ještě Data Modeler. Ten umožňuje vytvoření konceptuálního a relačního datového modelu. V tomto nástroje byl vytvořen relační datový model aplikace – obrázky 2.7, 2.8 a 2.9.

**Oracle APEX** Oracle Application Express (APEX) je platforma společnosti Oracle, která umožňuje vytvoření aplikace pro Oracle databáze s minimální nutností psát programový kód. Výsledkem je responzivní, databází řízená webová aplikace. Tento nástroj byl použit pro vytvoření forntendové části nástroje.

## 3.2 Struktura aplikace

Aby byl splněn požadavek na použité technologie 1.6.1, je aplikace sestavena ze dvou částí – z databázové části napsané v jazyce PL/SQL, která obsahuje veškerou funkcionalitu aplikace, a z uživatelského rozhraní vytvořeného v nástroji APEX. Tato sekce se zabývá hlavně strukturou databázové části aplikace, uživatelskému rozhraní a nástroji APEX je věnována kapitola 3.4.

Funkcionalita programu je pro přehlednost členěna do několika balíčků („packages“). Seznam všech vytvořených balíčků a jejich popis je uveden v tabulce 3.1. Jak je z tabulky možné vyčíst, pro práci s každou z oblastí aplikace (číselníky, role, složky) jsou vytvořeny dva balíčky. Jeden z balíčků, nazvaný jako manager, obsahuje funkce a procedury implementující potřebné funkcionality programu, jako je například vytvoření číselníku nebo přidělení role uživateli. Tyto procedury a funkce jsou poté používány jako rozhraní pro práci z daty z uživatelského rozhraní. Druhý balíček, označený přívlastkem control, obsahuje pomocné funkce a procedury. Příklady těchto funkcí jsou validace vkládaných hodnot nebo kontrola existence hodnot v databázi.

Mimo funkce a procedury vznikl také v databázi nový typ. Ten představuje jeden záznam a je tvořen tabulkou se dvěma sloupci – identifikátor atributu a hodnota záznamu jemu příslušná. V tomto formátu jsou získávána data o záznamu z uživatelského rozhraní.

## 3.3 Sestavení číselníku

V návrhu bylo stanoveno, že záznamy číselníků budou ukládány do jedné tabulky po jednotlivých hodnotách. Při tomto modelu je problematická část sestavení číselníku tak, jak bude zobrazován uživateli, tedy sestavit záznamy z jednotlivých hodnot. Způsob, jakým byl tento problém vyřešen je popsán v této části.

### 3.3.1 Klauzule PIVO

Klauzule `PIVOT` bere data jednotlivých řádků, agreguje je a převádí je do sloupců. Základní syntaxe této klauzule je v ukázce 3.1. Ve své základní podobě je operátor `PIVOT` poměrně omezený. Výčet výsledných sloupců, tedy hodnot původní tabulky, je v příkazu nutné uvést explicitně, nelze je uvést dynamicky pomocí dotazu.

Tabulka 3.1: Seznam balíčků v databázi

| Název | Popis |
|---|---|
| DF_CONTROL | Podpůrné funkce, které nelze zařadit pod žádnou z částí aplikace – získání datumu, který reprezentuje nekonečno, a kontrola datového typu a existence uživatele. |
| DF_DATASET_CONTROL | Podpůrné funkce pro práci s číselníky, jejich atributy a záznamy – kontrola existence názvů a identifikátorů, validace hodnot a získání potřebných informací o aktuální verzi číselníků. |
| DF_DATASET_MANAGER | Funkcionalita pro práci s číselníky, jejich atributy a záznamy – vytvoření, úprava a odstranění číselníků, atributů a záznamů, sestavení dat pro zobrazení číselníku. |
| DF_FOLDER_CONTROL | Podpůrné funkce pro práci se složkami – kontrola existence složky a jejího názvu, kontrola, zda je složka prázdná. |
| DF_FOLDER_MANAGER | Funkcionalita pro práci se složkami – vytvoření, úprava a odstranění složky. |
| DF_ROLE_CONTROL | Podpůrné funkce pro práci s rolemi a oprávněními – kontrola existence oprávnění a rolí, kontrola, zda má uživatel přiřazenou roli a zda má role daná oprávnění. |
| DF_ROLE_MANAGER | Funkcionalita pro práci s rolemi a oprávněními – vytvoření, úprava a odstranění rolí, nastavení oprávnění rolím a rolí uživatelům. |

Zdrojový kód 3.1: Základní syntaxe klauzule PIVOT[1]

```
SELECT
    select_list
FROM
    table_name
PIVOT [XML] (
    pivot_clause
    pivot_for_clause
    pivot_in_clause
);
```

Byl proveden průzkum na internetu, kde se problém, jak získat klauzuli `PIVOT` s dynamickým vstupem, řeší poměrně často. Mezi hlavní tři nalezené způsoby řešení patří:

1. Jednou z možností je XML varianta klauzuli `PIVOT`. V tom případě ale

není výstupem tabulka s požadovanými názvy sloupců, nýbrž záznamy v XML formátu.

2. Dalším způsobem je funkce `pivot` vytvořená Antonem Shefferem[14]. Vstupem této funkce je řetězec obsahující dotaz na potřebné sloupce. Funkce jinak funguje stejně jako klauzule `PIVOT`.

3. Následující varianta je z rad z diskuzního fóra[15]. Tato možnost spočívá ve vytvoření řetězce obsahujícího klauzuli `PIVOT` s dynamickým vstupem. Tento příkaz ve formě řetězce poté vykoná pomocí příkazu `EXECUTE IMMEDIATE`.

### 3.3.2 Řešení

Ze tří popsaných variant způsobů řešní byl vybrán ten poslední, z diskuzního fóra[15], protože je pro potřeby implementace nejvhodnější. XML varianta nebyla využita kvůli formátu výstupu a řešení Antona Sheffera[14] je pro potřeby řešení problému vytvářeného nástroje příliš komplexní.

Protože se předpokládá, že ke změně struktury číselníku bude docházet jen málo často, bylo rozhodnuto pro každý číselník vytvořit `VIEW`, které bude představovat číselník v uživatelském formátu, tedy ve formě tabulky. Jelikož se můžou názvy sloupců, neboli atributy číselníků, opakovat, v případě že jeden bude smazán a poté vytvořen nový se stejným názvem, budou pro názvy sloupců ve `VIEW` použity identifikátory atributů namísto jejich názvů. Názvy atributů budou doplněny až v uživatelském rozhraní.

Výsledný kód 3.2 se skládá z několika kroků. Prvním krokem je vytvoření znakového řetězce, který obsahuje seznam identifikátorů atributů, které mají být ve výsledném `VIEW`, oddělené čárkou. Dále je vytvořen řetězec, který bude představovat výběr sloupců výsledného `VIEW`. V následujícím kroku je vytvořen řetězec obsahující klauzuli `PIVOT`, do které je doplněn první vytvořený řetězec. Následně je vytvořen a proveden příkaz pro vytvoření požadovaného `VIEW`.

Zdrojový kód 3.2: Ukázka implemtenace sestavení číselníku

```
PROCEDURE CREATE_DATASET_VIEW (
  p_dataset_id                    IN NUMBER
) AS
  attribute_ids                   CLOB;
  attribute_ids_max               CLOB;
  create_pivot                    CLOB;
  create_view                     CLOB;
BEGIN

  SELECT LISTAGG (attribute_id, ' ,')
    WITHIN GROUP (ORDER BY attribute_id) INTO attribute_ids
    FROM (SELECT DISTINCT attribute_id
```

```
            FROM df_dataset_attribute
            WHERE dataset_id = p_dataset_id);

  SELECT LISTAGG (', MAX("' || attribute_id || '") ' || ' AS "
    ↪ ' || attribute_id || '"')
    WITHIN GROUP (ORDER BY attribute_id)
      INTO attribute_ids_max
    FROM (SELECT DISTINCT attribute_id
            FROM df_dataset_attribute
            WHERE dataset_id = p_dataset_id);


  create_pivot := 'SELECT * FROM df_value PIVOT ( MAX(value)
    ↪ FOR attribute_id IN (' || attribute_ids || '))';

  create_view := 'CREATE OR REPLACE VIEW dataset_view_' ||
    ↪ p_dataset_id || ' AS SELECT dataset_row_id, MAX(
    ↪ record_id) AS record_id' || attribute_ids_max || ',
    ↪ MAX(author) AS author, MAX(updater) AS updater, MAX(
    ↪ physical_from) AS physical_from, MAX(physical_to) AS
    ↪ physical_to, MAX(state) AS state FROM (' ||
    ↪ create_pivot || ') WHERE dataset_id = ' ||
    ↪ p_dataset_id || ' GROUP BY dataset_row_id';

  EXECUTE IMMEDIATE create_view;
END CREATE_DATASET_VIEW;
```

## 3.4 APEX

Komponenta pro správu číselníků je tvořena několika obrazovkymi vytvořenými v nástroji APEX. Jejich kompletní seznam je uveden v tabulce 3.2. Obrazovka `LOGIN_PAGE` slouží k přihlášení uživatele do aplikace DATA_FRAME Application a je shodná se stejnými obrazovkami v ostatních komponentách. Obrazovka `HOME` představuje domovskou stránku komponenty. V případě komponenty pro správu číselníků obsahuje pouze přesměrování na obrazovku se seznamem číselníků.

V následující části práce jsou popsány některé zajímavé konstrukce, které bylo nutné v nástroji APEX vytvořit, aby bylo možné splnit požadavky na novou komponentu.

### 3.4.1 Dynamická tabulka

Základní obrazovkou nové komponenty, kterou můžete vidět na obrázku B.3, je detail číselníku. Tato obrazovka je pro všechny číselníky shodná, až na samotná data číselníku. Na obrazovce tedy musí být tabulka, která má dynamicky se měnící sloupce a data, podle toho, který číselník je zrovna zobrazen.

Tabulka 3.2: Seznam obrazovek komponenty

| ID | Název | Titulek | Popis |
|---|---|---|---|
| 100 | LOGIN_PAGE | LOGIN_PAGE | Obrzovka pro přihlášení užívatele |
| 200 | HOME | HOME | Domovská obrazovka komponenty |
| 3100 | DATASET_DETAIL | DATASET_DETAIL | Obrazovka detailu složky a číselníku |
| 3105 | DATASET_EDIT | Dataset Detail | Dialog pro editaci struktury / vytvoření číselníku |
| 3107 | DATASET_RENAME | Rename Dataset | Dialog pro přejmenování číselníku |
| 3110 | DATASET_PERMISSION | Edit permissions | Dialog pro změnu oprávnění pro číselník |
| 3115 | RECORD_EDIT | Edit Record | Dialog pro editaci/vytvoření záznamu |
| 3140 | FOLDER_EDIT | Edit Folder | Dialog pro editaci/vytvoření složky |
| 3142 | FOLDER_RENAME | Rename Folder | Dialog pro přejmenování složky |
| 3155 | RECORD_HISTORY | RECORD_HISTORY | Obrazovka historie úprav číselníků, atributů a záznamů |
| 3200 | ROLE_LIST | ROLE_LIST | Obrazovka se seznamem uživatelských rolí komponenty |
| 3205 | ROLE_EDIT | Edit Role | Dialog pro editaci/vytvoření uživatelské role |
| 3207 | ROLE_RENAME | Rename Role | Dialog pro přejmenování uživatelské role |
| 3210 | USER_LIST | List of users | Dialog se seznamem uživatelů aplikace |
| 3215 | USER_ROLE_EDIT | Users roles | Dialog pro přiřazení rolí uživateli |

Dynamickou tabulku bohužel není možné v nástroji APEX vytvořit jako základní komponentu. Všechny tabulky musejí mít daný počet sloupců a jejich názvy při kompilaci stránky. Z tohoto důvodu muselo být zobrazení detailu číselníku vyřešeno trochu oklikou.

Byl využit způsob, který už se používá v jiných komponentách aplikace DATA_FRAME Application. Na stránce je vytvořena prázdná tabulka představující šablonu. Dále jsou vytvořeny skryté položky představující názvy jednotlivých sloupců a jednu položku, do které bude následně uložen skript pro načtení číselníku. Nakonec je vytvořen proces, který je spuštěn po načtení hlavičky stránky. Tento proces spustí databázovou funkci, která pomocí funkce rozhraní APEX `APEX_UTIL.set_session_state`[16] vyplní položky pro názvy sloupců a skript pro získání hodnot informacemi o zobrazovaném číselníku. Vytvořený skript je `SELECT` nad `VIEW` daného číselníku, který je upraven tak, aby splňoval strukturu šablony tabulky na stránce. Skript je poté zobrazen na výsledné obrazovce místo prázdné tabulky.

Nevýhodou tohoto řešení je, že je omezený počet atributů, který může číselník mít. Maximální počet atributů je závislý na počtu v šabloně vytvořených sloupců. Jelikož jsou všechny číselníky zobrazovány jako varianta jedné a té samé tabulky pouze se změněným obsahem, je další nevýhodou to, že při uživatelském nastavení jednoho číselníku (například seřazení záznamů podle hodnot v konkrétním sloupci nebo zvýraznění hodnot) je nastavení aplikováno na všechny číselníky.

Stejný způsob dynamické tabulky je využit na obrazovce historie záznamů číselníku, kterou můžete vidět na obrázku B.6.

### 3.4.2 Aktualizace adresářové struktury

Editace detailu číselníků a složek probíhá pomocí dialogových oken. Po zavření takového okna je potřeba aktualizovat jednotlivé prvky obrazovky, na kterou je uživatel vrácen, pro případ, že by došlo ke změně zobrazovaných dat. V nástroji APEX je tento problém běžně řešen pomocí dynamické akce, která reaguje na uzavření dialogového okna. Tato dynamická akce iniciuje aktualizaci regionu.

Běžný postu popsaný v předchozím odstavci bohužel nefunguje na region `Tree`, který slouží k zobrazení adresářové struktury číselníků. Místo aktualizace regionu je potřeba znovu načíst celou stránku a tak i aktualizovaná data. Tento proces se na rozdíl od popisovaného v předchozím odstavci týká dialogového okna pro editaci namísto cílové obrazovky. V dialogovém okně je místo uzavření okna použito přesměrování na danou obrazovku. Přesměrováním je docíleno toho, že se obrazovka načte celá znovu. Všechny parametry obrazovky určující její stav, jako například identifikátor zobrazovaného číselníku, je potřeba přeposlat do dialogu a po jeho uzavření s jejich hodnotami iniciovat načítanou obrazovku. Tak je zajištěno, že se obrazovka detailu načte ve stejném stavu, ve kterém byla, když byl otevřen dialog.

### 3.4.3 Editace záznamu

Podobným problémem, jako je tabulka s dynamickou strukturou, je vytvoření dialogu pro editaci záznamu. APEX totiž stejně jako u tabulky potřebuje mít při kompilaci formuláře definované jeho hodnoty. Jelikož má každý číselník jinou strukturu záznamu / jiné atributy, není jejich dopředná definice možná.

Způsobů řešení tohoto problému je více, zde budou představeny dva z nich. U každého z nich je potřeba vytvořit databázovou proceduru, podobně jako u dynamické tabulky, která bude určovat, které atributy danému číselníku přísluší.

#### 3.4.3.1 Dynamický formulář

První způsob spočívá ve vytvoření celého formuláře dynamicky. V databázové proceduře jsou pomocí funkcí APEX API z balíčku `APEX_ITEM` vytvořena jednotlivá pole formuláře. Tento způsob umožňuje vytvořit různé typy polí pro různé atributy, například rozbalovací seznam pro cizí klíče nebo výběr data z kalendáře pro datumový datový typ.

Nevýhodou tohoto způsobu je, že vytvořená pole nejsou nijak stylizována. Styl je jednotlivým polím potřeba přidat pomocí HTML kódu, který se vloží v podobě znakového řetězce do databázové procedury, který formulář vytváří. Aby byl formulář vizuálně shodný s ostatními v aplikaci, HTML kód v databázové metodě by musel vypadat podobně jako v ukázce 3.3. Tento kód představuje jedno pole formuláře, v tomto případě textové. Také není možné pro takto vytvořená pole v aplikaci APEX vytvořit validace. Validace by musely být řešeny v databázi, stejně ta i jejich zobrazení u jednotlivých polí.

Zdrojový kód 3.3: HTML kód pro zobrazení pole formuláře

```
<div class="t-Form-fieldContainer t-Form-fieldContainer--
    ↪ floatingLabel is-required apex-item-wrapper apex-item-
    ↪ wrapper--text-field js-show-label">
    <div class="t-Form-labelContainer">
        <label class="t-Form-label">
            attribute_name
        </label>
    </div>
    <div class="t-Form-inputContainer">
        <div class="t-Form-itemWrapper">
            <input required="is_required" class="text_field
                ↪ apex-item-text" value="attribute_name" size
                ↪ ="60" maxlength="1020">
        </div>
    </div>
</div>
```

Protože kontaminace datové vrstvy prvky z vrstvy prezentační není z pohledu architektury aplikace správně, nebyl tento způsob vytvoření formuláře použit.

#### 3.4.3.2 Statický formulář

Druhá možnost je ekvivalentní způsobu, jakým byla vytvořena dynamická tabulka. Formulář je vytvořen jako šablona s prázdnými poli. Těmto polím jsou pomocí funkce APEX API `APEX_UTIL.set_session_state` nastaveny v databázové proceduře hodnoty. Pro názvy polí jsou vytvořeny položky obrazovky, jejichž hodnoty jsou nastaveny stejným způsobem.

Hlavní nevýhodou tohoto způsobu je, že všechny položky seznamu jsou vždy textová pole. Typ vstupu musí být určen při kompilaci a v tu dobu není možné zjistit, který atribut je jakého typu a jaký typ pole by pro něj měl být vytvořen. Také je, stejně jako u dynamické tabulky, omezen počet polí formuláře, tedy atributů záznamu.

Bohužel ani tato varianta není ideální, ale i přesto byla vybrána jako ta méně špatná, protože do datové vrstvy není nutné vkládat žádný kód vrstvy prezentační, jak tomu je u první varianty. Různé varianty datových typů hodnot budou řešeny validacemi nad jednotlivými poli. Omezení počtu polí formuláře nepředstavuje v této situaci velký problém, protože k omezení dochází již u dynamické tabulky. Vzhled výsledného formuláře i s příkladem hlášky nevalidní hodnoty je na obrázku B.4.

### 3.4.4 Schválení/zamítnutí změny záznamu

V tabulce s neschválenými změnami číselníku s procesem se schvalováním má být podle návrhu na každém řádku se změnou tlačítko pro schválení a zamítnutí. V základu bohužel tabulka neumožňuje vytvořit tlačítko jako obsah buňky. Řešení tohoto problému bylo provedeno podle návodu na blogu Jackeiho Macilroye[17].

Řešení se skládá ze dvou částí. Nejdříve je vytvořeno tlačítko a poté je mu nastavena dynamická akce reagující na kliknutí. Tlačítko je vytvořeno jako sloupec tabulky typu `Link`, kterému jsou přidány atributy CSS definující vzhled tlačítka. Také je mu nastavena unikátní CSS třída. Tato třída je využita pro druhý krok – vytvoření dynamické akce schválení/zamítnutí změny. Dynamická akce má dva kroky. Nejdříve je pomocí javascriptu nastavena hodnota jedné z položek obrazovky na identifikátor záznamu, nad kterým je akce provedena. Následně je zavolána databázová funkce pro schválení/zamítnutí změny. Výsledná obrazovka je vidět na obrázku B.5.

### 3.4.5 Autorizace

Jelikož jsou součástí řešení i různé přístupové role pro jednotlivé číselníky, bylo potřeba vyřešit autorizaci. Autorizace probíhá na dvou úrovních, jednou

jsou zmíněné role pro přístup k číselníkům, druhou jsou role na úrovni celé aplikace DATA_FRAME Application. Tato část je zaměřena na přístupové role pro číselníky, autorizace rolí celé aplikace se vztahují pouze na přístup ke správě rolím a historii.

#### 3.4.5.1 Autorizační schémata

Nová komponenta umožňuje nastavení různých práv rolí pro různé číselníky. Oprávnění jsou: čtení, editace záznamů, schvalování změn, správa struktury. Jelikož jsou oprávnění takto stupňována, je potřeba nastavit různým prvkům obrazovek různá pravidla pro zobrazení/použití.

Autorizace je jednou z funkcí, které nabízí nástroje APEX. Jako návod, jak s touto funkcí pracovat, byl použit článek na blogu Oracle APEX[18]. Dle tohoto návodu bylo vytvořeno několik autorizačních schémat. Pro většinu typů oprávnění bylo vytvořeno schéma, které povoluje přístup exkluzivně danému oprávnění, a schéma, které umožňuje přístup danému typu i všem typům nadřazeným. Každé ze schémat je řízeno obdobnou SQL funkcí určující, jestli má přihlášený uživatel požadované oprávnění. Příklad této funkce pro případ prvku přístupnému pouze uživateli s oprávněním `ADMIN` je uveden v ukázce 3.4. K určení, zda je uživatel k dané akci oprávněn je vždy potřeba identifikátor číselníku, se kterým pracuje. Jelikož se schéma používá na více stránkách, je potřeba kontrolovat identifikátor číselníku na dané stránce. K získání identifikačního čísla stránky je použita metoda `v('APP_PAGE_ID')`. Pro správnou funkčnost autorizace je potřeba dodržet stejný název identifikátoru číselníku na všech stránkách, kde je autorizační schéma používáno.

Zdrojový kód 3.4: Příklad SQL funkce autorizačního shcématu

```
DECLARE
  l_role_name        df_rd_user_role.role%TYPE;
BEGIN
  SELECT MAX(ur.role) INTO l_role_name
    FROM df_dataset_permission dp
      JOIN df_rd_user_role ur ON (ur.role = dp.role)
    WHERE ur.user_id = :APP_USER
      AND dataset_id = v('P' || v('APP_PAGE_ID') || '
          ↪ _DATASET_ID')
      AND permission = 'ADMIN';

  RETURN (l_role_name IS NOT NULL);
END;
```

Použitý způsob umožňuje, po vytvoření autorizačních schémat, jednoduše je nastavit jednotlivým prvkům stránek tak, jak je potřeba.

#### 3.4.5.2 Formulář pouze pro čtení

Výše uvedený způsob bohužel nelze použít pro všechny potřebná omezení v komponentě. Případem, kdy to použít nelze, je dialog pro editaci struktury číselníku. Ten obsahuje formulář a tabulku s atributy. Ty musí být editovatelná pro administrátora, ale pouze ke čtení pro ostatní uživatele, kteří mají k číselníku přístup. Verze pro čtení je potřeba, aby měli uživatelé přístup k informacím o struktuře číselníku.

Řešení pro tento problém bylo nalezeno na diskuzním fóru. Podle rady byl vytvořena položka obrazovky určující, zda má být stránka editovatelná. Tato položka je před načtením lavičky stánky nastavena na příslušnou hodnotu procesem, jehož spuštění podléhá autorizačnímu schématu. Hodnota je nastavena tedy jen tehdy, nemá-li přihlášený uživatel práva na editaci struktury číselníku. Výseldný dialog pro uživatele s obrávnění je na obrázku B.2, verze pro neoprávněného uživatele je na obrázku B.1.

Kapitola 4

# Testování

Otestování vytvořeného produktu je nedílnou součástí softwarového procesu, a proto je této oblasti věnována celá poslední kapitola. K otestování komponenty byly využity dva typy testů, a to jednotkové testy pro testování PL/SQL kódu a scénářové testy pro testování uživatelského rozhraní. Způsob jakým byly testy vytvářeny a jak byly provedeny je hlavním obsahem této kapitoly.

## 4.1 Jednotkové testy

Prvním způsobem, jakým byla vytvořená aplikace testována, jsou jednotkové testy. Protože veškerá logika komponenty je implementována v PL/SQL, jedná se o jednotkové testy PL/SQL funkcí a procedur. Pro realizaci jednotkových testů bylo nejdříve potřeba vybrat nástroj, ve kterém budou testy vytvářeny, poté bylo nutné vytvořit testovací verzi databáze s testovacími daty a následně ve vybraném nástroji vytvořit samotné testy.

### 4.1.1 Výběr nástroje

Jak už bylo zmíněno výše, nejdříve je nutné vybrat nástroj pro tvorbu jednotkových testů, který umožní jejich alespoň částečnou automatizaci. Nástrojů pro vytváření a automatizaci jednotkových testů pro PL/SQL existuje veliké množství, zde jsou představeny tři z nich.

#### 4.1.1.1 utPLSQL

UtPLSQL je open-source nástroj pro testování PL/SQL a SQL kódu. Umožňuje automatické testování balíčků, funkcí, procedur, triggerů, view a čehokoliv jiného, co může být provedeno čí sledováno pomocí PL/SQL.

Nástroj existuje již od roku 1999, původní verze byla ale zcela přepsána členy komunity v roce 2016. K vytvoření současné verze byly využity objek-

tově orientované schopnosti databáze Oracle, aby byla konzistentnější s jinými nástroji pro jednotkové testy, jako je například JUnit pro Javu.[19]

UtPLSQL nabízí uživateli absolutní kontrolu na kódem testu a to hlavně kvůli tomu, že si uživatel musí napsat a spravovat veškerý kód sám. Nástroj umožňuje pomocí API spustit všechny testy jedním příkazem a zobrazí, jestli testy úspěšně prošly či ne.[20]

#### 4.1.1.2 Quest Code Tester for Oracle

Quest Code Tester pro Oracle automatizuje proces testování PL/SQL programů, což umožňuje rychle identifikovat chyby a ověřit správnost programu. Nástroj je placený, ale existuje i jeho třicetidenní zkušební verze zdarma.

Nástroj nabízí grafické rozhraní, pomocí něhož uživatel popíše předpokládané chování testovaného programu. Code Tester poté vygeneruje potřebný PL/SQL kód testu a příkazem skrz rozhraní čí příkazovou řádku testy provede. Nástroj zobrazí výsledky testů zvýrazněné barvami. Také je možné v nástroji vytvářet zálohy definic testů a sdílet je mezi vývojáři pomocí importu exportu.[21]

#### 4.1.1.3 SQL Developer Unit Testing

Poslední zde rozebíranou možností je nástroj na jednotkové testy, který je součástí SQL Developeru. Tento nástroj poskytuje možnost testování PL/SQL objektů, jako jsou funkce a procedury a monitorování výsledků v průběhu času.

Podobně jako Quest Code Tester for Oracle nabízí tento nástroj grafické rozhraní pro vytváření testů. Uživatel zadá pouze informace o tom, co bude testováno a jaký je požadovaný výsledek. Implementace jednotkových testů v nástroji SQL Developer je postavena na klasické a dobře známé kolekci frameworků jednotkových testů xUnit.[22]

#### 4.1.1.4 Výběr

K vytvoření jednotkových testů pro tuto práci byl vybrán nástroj SQL Developer Unit Testing. Hlavním důvodem tohoto výběru je to, že se k implementaci SQL Developer již využívá. Navíc nástroj nabízí veškeré potřebné funkce.

### 4.1.2 Testovací data

Po výběru nástroje pro práci s jednotkovými testy bylo potřeba vytvořit testovací schéma databáze a to naplnit testovacími daty. Bylo vytvořeno několik skriptů, které vytvoří testovací data. Testovací data byla volena tak, aby pokryla co možná největší počet možných stavů komponenty v průběhu jejího používání.

Pro účely testování byly vytvořeny tři číselníky. Dva z nich validní, každý s jiným typem schvalovacího cyklu, a jeden číselník, který byl v minulosti

smazán. Každému z číselníků byly vytvořeny atributy představující primární klíče, jeden z číselníků má primární klíč tvořený více atributy. Dále bylo vytvořeno několik obyčejných atributů, cizí klíč a odstraněný atribut. Atributy jsou různě rozděleny mezi dva validní číselníky. Všechny číselníky byly naplněny několika záznamy. Pro číselník se schvalováním změn byly vytvořeny záznamy ve všech možných stavech schvalovacího procesu.

Kromě číselníků byly vytvořeny také testovací uživatelské role, které byly přiřazeny základním uživatelům, kteří jsou v aplikaci vytvořeni. Každá z rolí dostala různá oprávnění pro práci s různými číselníky. Také bylo vytvořeno několik složek pro testování práce s adresářovou strukturou.

### 4.1.3 Tvorba testů

Posledním krokem je samotné vytvoření jednotkových testů. Jak bylo určeno výše, byl k tomu použit nástroj SQL Developer Unit Testing. Pro každou vytvořenou funkci a proceduru byl vytvořen jeden jednotkový test s různými implementacemi pro různá vstupní data. Po provedení testu se provedou ještě kódy kontrolující správný stav databáze – pokud je to nutné a následně se databáze vrátí do původního stavu před testem. Jednotlivé testy jsou rozděleny do souborů podle toho, kterou část systému testují (číselníky, atributy, role a další).

Pro podrobnější náhled do vytvořených testů je v následující části popsán test funkce pro vytvoření číselníku. Jsou pro něj vytvořeny čtyři různé implementace/varianty:

- Vytvoření číselníku s procesem bez schvalování

  V této variantě je testováno vytvoření číselníku s procesem bez schvalování, kdy jsou všechny vstupní parametry funkce validní – všechny hodnoty jsou vyplněny a název číselníku je unikátní. Při očekávaném chování funkce je do tabulek `DF_DATASET` a `DF_DATASET_HISTORY` přidán řádek se zadanými hodnotami. Funkce by měla projít bez vyhození výjimky.

- Vytvoření číselníku s procesem se schvalováním

  V druhé variantě se jedná o stejný případ jako v prvním případě, jen je číselníku nastaven jiný typ schvalovacího procesu.

- Vytvoření číselníku s názvem odstraněného číselníku

  Další implementace testuje, zda se vytvoří číselník s názvem, který se shoduje s číselníkem, který byl někdy v minulosti odstraněn. I když je odstraněný číselník stále v databázi a je možné ho dohledat v historii změn, mělo by vytvoření nového číselníku proběhnout. Název číselníku totiž musí být unikátní pouze mezi aktuálně existujícími číselníky. Očekávaným výsledkem je tedy, stejně jako v předchozích případech, přidání

řádku do tabulky `DF_DATASET` a `DF_DATASET_HISTORY` se zadanými hodnotami. Funkce by měla projít bez vyhození výjimky.

- Vytvoření číselníku s názvem existujícího číselníku

  Poslední variantou testu je použití názvu číselníku, který existuje. Tato akce je nevalidní a test by proto měl skončit zachycením výjimky způsobené neunikátním názvem číselníku. Po provedení této varianty je zkontrolováno, že se záznamy číselníků nijak nezměnily.

U každé varianty je vždy po zavolání testované funkce zkontrolován stav databáze pomocí několika PL/SQL skriptů. Skripty kontrolují řádky tabulek `DF_DATASET`, `DF_DATASET_HISTORY` a nastavená oprávnění pro přístup k danému číselníku v tabulce `DF_DATASET_PERMISSION`. Příklad skriptu je uveden v ukázce 4.1, tento se týká konkrétně kontroly vytvoření nového řádku v tabulce `DF_DATASET`.

Zdrojový kód 4.1: Příklad kontrolního testu v rámci jednotkových testů

```
DECLARE
    l_count NUMBER;
    wrong_count EXCEPTION;
BEGIN
    SELECT count(*) INTO l_count
        FROM df_dataset
        WHERE business_name = 'test_create_dataset';
    IF l_count <> 1
    THEN
        RAISE wrong_count;
    END IF;
END;
```

Všechny implementace testu mají navíc jeden společný skript, kterým se po provedení testu a kontrolních skriptů vrátí databáze do původního stavu. To je potřeba k tomu, aby mohly být jednotlivé jednotkové testy spouštěny nezávisle na sobě a navzájem se neovlivňovaly.

## 4.2 Testy uživatelského rozhraní

Dalším způsobem testování komponenty jsou testy uživatelského rozhraní. Bylo vytvořeno několik scénářů, pomocí kterých jsou otestovány všechny obrazovky a jejich funkcionalita. Testy nejsou automatizovány, je potřeba je provádět ručně průchodem aplikací.

Každý test obsahuje krátký popis toho, co je předmětem daného testu a jaký je jeho cíl. Každý test má také uvedeno, jaký je potřebný stav systému před začátkem testování. To obnáší stav testovacích dat, zda má být uživatel přihlášen, na jaké obrazovce test začíná a další. Následuje scénář testu, který

je jeho hlavní částí. Scénář obsahuje jednotlivé akce, které musí tester vykonat, a odpovídající reakce systému na ně. Pokud se reálná reakce systému liší od reakce popisované v testu, je test vyhodnocen jako nesplněný.

Aplikace byla otestována všemi vytvořenými testy uživatelského rozhraní a funguje v souladu s jejich scénáři. Všechny chyby, které byly při testování nalezeny, byly opraveny.

# Závěr

Cílem této diplomové práce bylo analyzovat, navrhnout a implementovat komponentu pro práci s číselníky v rámci nástroje DATA_FRAME Application. Konkrétněji bylo úkolem definovat funkční a nefunkční požadavky na komponentu a její interface na ostatní komponenty, navrhnout datový model, podle kterého následně implementovat funkční prototyp, který je zdokumentován a řádně otestován.

V rámci analýzy práce bylo provedeno několik kroků. Prvním krokem bylo seznámení se se stávající verzí nástroje DATAF_RAME Application a jeho komponent. Poté byla provedena rešerše několika existujících řešení na správu referenčních dat, podle které byly definovány funkční požadavky na novou komponentu. Dále byly definovány nefunkční požadavky podle požadavků zadavatele. Poslední částí analýzy bylo vytvoření případů užití nové komponenty.

Na základě provedené analýzy byl vytvořen návrh komponenty. Byl definován způsob, jakým bude reprezentován číselník v datovém modelu a jakým způsobem bude řešena jeho historizace. Dále byl vytvořen kompletní datový model pro celou komponentu a byl definován interface na ostatní komponenty. Také byly vytvořeny návrhy obrazovek výsledného řešení. Obrazovky byly navrhovány s ohledem na nefunkční požadavek použití nástroje APEX pro vytvoření uživatelského rozhraní.

Podle vytvořeného návrhu byla provedena implementace prototypu komponenty. Jak bylo definováno v nefunkčních požadavcích, kompletní logika aplikace byla implementována pomocí PL/SQL a uživatelské rozhraní bylo vytvořeno v nástroji APEX. Komponenta byla vytvořena jako součást nástroje DATA_FRAME Application. Prototyp byl řádně otestován pomocí jednotkových a uživatelských testů.

Provedením všech výše popsaných kroků bylo splněno zadání diplomové práce. Výsledkem je funkční a otestovaný prototyp komponenty pro správu číselníků nástroje DATA_FRAME Application.

## Další vývoj komponenty

Vytvořený prototyp splňuje všechny definované požadavky, a tak umožňuje uživateli nástroje DATA_FRAME Application spravovat i číselníky. Komponenta nabízí grafické uživatelské rozhraní, které umožňuje spravovat číselníky i uživateli bez znalosti databázových programovacích jazyků. Navíc je možné v komponentě vytvořit role a jim nastavit různá přístupová práva pro různé číselníky, čímž je možné zvoleným uživatelským rolím zamezit přístupu k některým funkcím nástroje. Možnost zobrazení veškerých změn provedených na číselnících umožňuje dohledat kdy a kdo provedl jaké úpravy. Výsledek práce bude začleněn do aplikace DATA_FRAME Application a dále rozvíjen v rámci této aplikace. Možností, jak zefektivnit a zpříjemnit uživatelům práci s číselníky, je mnoho a tak se nabízí i velký počet způsobů, jak vytvořený prototyp v budoucnosti ještě vylepšit.

Jelikož v rámci této práce byl kladen jen minimální důraz na rychlost a efektivitu řešení, je optimalizace určitě jedním z vhodných kroků rozvoje prototypu. Například by bylo vhodné do některých tabulek přidat indexy pro rychlejší vyhledávání.

Jistě užitečnou novou funkcí komponenty by byl import a export z a do různých tabulkových formátů, jako je například CSV nebo XML. Tato funkce by zpříjemnila uživateli přenos již existujících číselníků, které nejsou zavedeny v nástroji DATA_FRAME Application, ale jsou ukládány například v exceovských tabulkách.

Možností, jak komponentu rozvinout, je přidat další varianty schvalovacích procesů. Tím je například proces dvojího schválení, který je i součástí nástroje od společnosti IBM[10]. Více procesů v aplikaci by uživatelům umožnilo větší variabilitu kontroly nad prováděnými změnami v číselnících.

# Literatura


[1] Oracle Tutorial: *Oracle PIVOT*. [cit. 02. 04. 2020]. Dostupné z: https://www.oracletutorial.com/oracle-basics/oracle-pivot/

[2] ManagementMania.com: Kmenová data (Master Data). *V: ManagementMania.com [online]*, červen 2018, [cit. 17. 3. 2020]. Dostupné z: https://managementmania.com/cs/kmenova-data-master-data

[3] Chisholm, M.: *The Foundations of Successful Reference Data Management*. TopQuadrant, [cit. 17. 3. 2020]. Dostupné z: https://www.topquadrant.com/docs/whitepapers/TopBraid_ReferenceDataManagementWhitepaper-3-18-15.pdf

[4] Narasimharajan, M.: Reference Data Management and Master Data: Are they Related ? *V: Oracle Blogs [online]*, prosinec 2012, [cit. 17. 3. 2020]. Dostupné z: https://web.archive.org/web/20151011124443/https://blogs.oracle.com/mdm/entry/reference_data_management_and_master

[5] Nicolas, F.: Back to Basics: Transactional, Master, Golden and Reference Data explained. *V: Semarchy [online]*, říjen 2018, [cit. 19. 3. 2020]. Dostupné z: https://blog.semarchy.com/backtobasics_data_classification

[6] Rouse, M.: What is data governance and why does it matter? *V: TechTarget [online]*, únor 2020, [cit. 23. 3. 2020]. Dostupné z: https://searchdatamanagement.techtarget.com/definition/data-governance

[7] International, D.: *The DAMA Guide to the Data Management Body of Knowledge*. Technics Publications, duben 2009, ISBN 9781935504009.

[8] Firican, G.: 5 best practices for managing reference data. *V: Lightsondata consulting & training [online]*, červenec 2018, [cit. 17. 3. 2020].

Dostupné z: https://www.lightsondata.com/5-best-practices-for-managing-reference-data/

[9] Zýka, O.: *DATA_FRAME Architecture*. Profinit EU, s.r.o., čtvrté vydání, 2017, [cit. 5. 3. 2020].

[10] IBM® Information Management Software: *A Practical Guide to Managing Reference Data with IBM InfoSphere Master Data Management Reference Data Management Hub*. 10 vydání, 2013, [cit. 2. 3. 2020]. Dostupné z: http://www.redbooks.ibm.com/redbooks/pdfs/sg248084.pdf

[11] Informatica: *Informatica Master Data Management Cloud MDM - Reference 360*. 2019, [cit. 2. 3. 2020]. Dostupné z: https://docs.informatica.com/master-data-management-cloud/reference-360/current-version/mdm---reference-360/preface.html

[12] International, D.: *The DAMA Dictionary of Data Management*. Technics Publications, duben 2011, ISBN 9781935504139.

[13] Margy Ross, R. K.: *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*. Wiley, třetí vydání, červenec 2013, ISBN 9781118530801.

[14] Jellema, L.: Dynamic SQL Pivoting – Stealing Anton's Thunder. *V: AMIS Technology blog [online]*, květen 2006, [cit. 02. 04. 2020]. Dostupné z: https://technology.amis.nl/2006/05/24/dynamic-sql-pivoting-stealing-antons-thunder/

[15] user2179919: Dynamic pivot in oracle sql. *V: Stack Overflow [online]*, květen 2016, [cit. 14. 12. 2019]. Dostupné z: https://stackoverflow.com/questions/15491661/dynamic-pivot-in-oracle-sql

[16] Oracle: *Application Express API Reference*. [cit. 04. 04. 2020]. Dostupné z: https://docs.oracle.com/cd/E37097_01/doc.42/e35127/index.html

[17] Mcilroy, J.: Delete a Row of a Report with a Dynamic Action. *V: Jackie McIlroy - Oracle & APEX [online]*, březen 2018, [cit. 05. 01. 2020]. Dostupné z: https://jackiemcilroy.blogspot.com/2018/03/delete-row-of-report-with-dynamic-action.html

[18] Gault, D.: Custom Authentication and Authorization using built in APEX Access Control - A How To. *V: Oracle [online]*, září 2019, [cit. 10. 02. 2020]. Dostupné z: https://blogs.oracle.com/apex/custom-authentication-and-authorization-using-built-in-apex-access-control-a-how-to

[19] utPLSQL: *About*. [cit. 27. 11. 2019]. Dostupné z: `http://utplsql.org/about/`

[20] Feuerstein, S.: Recommendations for unit testing PL/SQL programs. *V: Obsessed with Oracle PL/SQL [online]*, březen 2015, [cit. 27. 11. 2019]. Dostupné z: `http://stevenfeuersteinonplsql.blogspot.com/2015/03/recommendations-for-unit-testing-plsql.html`

[21] Quest Software Inc.: *Quest Code Tester for Oracle 3.1 Installation and Configuration Guide*. [cit. 27. 11. 2019].

[22] Oracle: *Unit Testing with SQL Developer*. [cit. 27. 11. 2019]. Dostupné z: `https://docs.oracle.com/cd/E15846_01/doc.21/e15222/unit_testing.htm#RPTUG45072`

Příloha **A**

# Seznam použitých zkratek

**ACORD** Association for Cooperative Operations Research and Development

**APEX** Oracle Application Express

**API** Application Programming Interface

**BI** Business Intelligence

**CSS** Cascading Style Sheets

**CSV** Comma-separated Values

**DG** Data Governance

**DW** Data Warehouse

**GUI** Graphical User Interface

**HTML** Hypertext Markup Language

**ISO** International Organization for Standardization

**MDM** Master Data Management

**OLTP** Online Transaction Processing

**PL/SLQ** Procedural Language/Structured Query Language

**RDM** Reference Data Management

**REST** Representational State Transfer

**SCD** Slowly Changing Dimensions

**SQL** Structured Query language

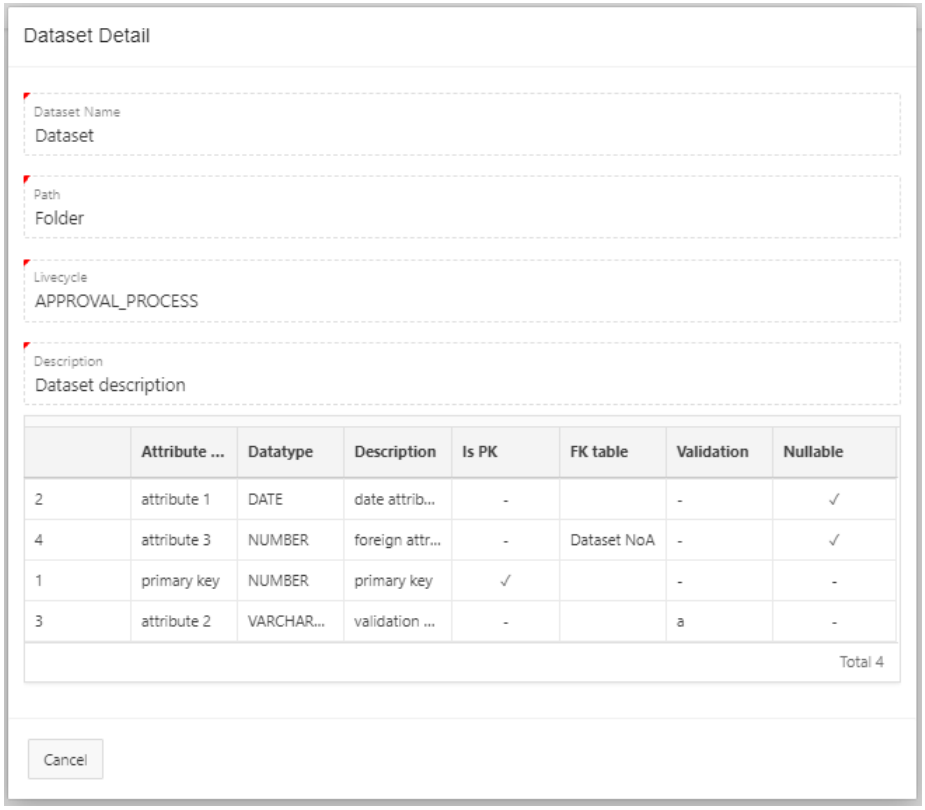
**SWIFT** Society for Worldwide Interbank Financial Telecommunication

**UML** Unified Modeling Language

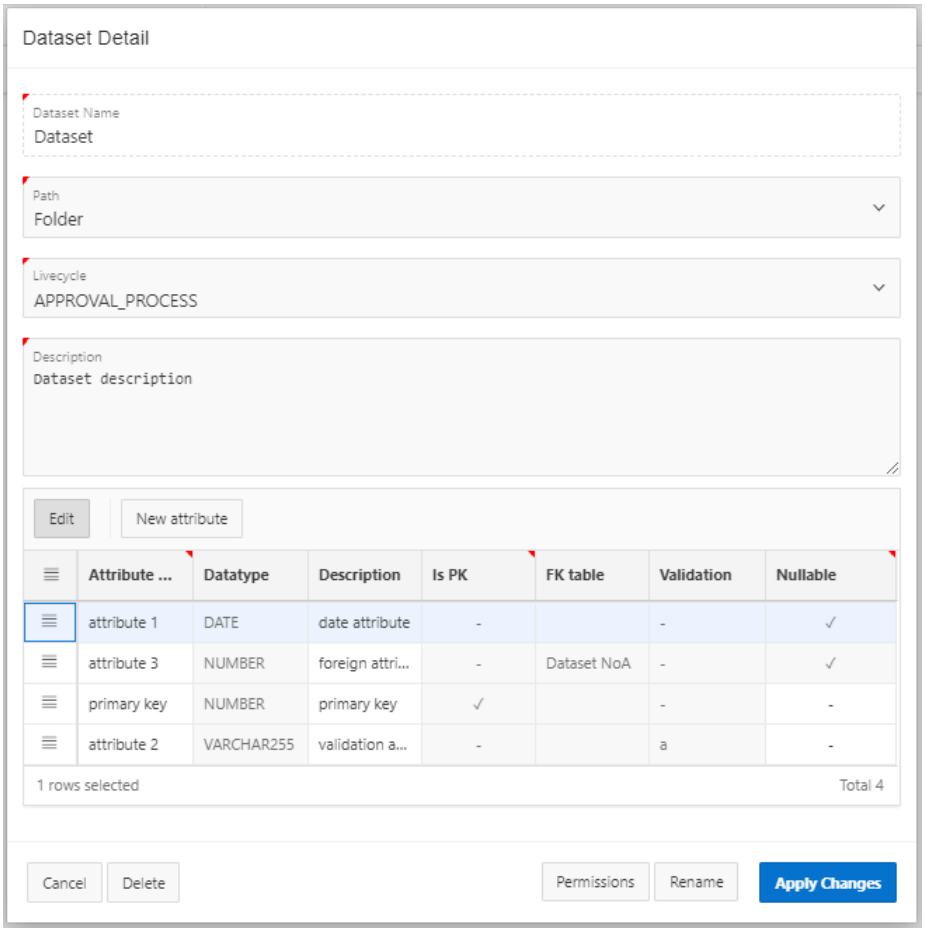
**XML** Extensible markup language

Příloha B

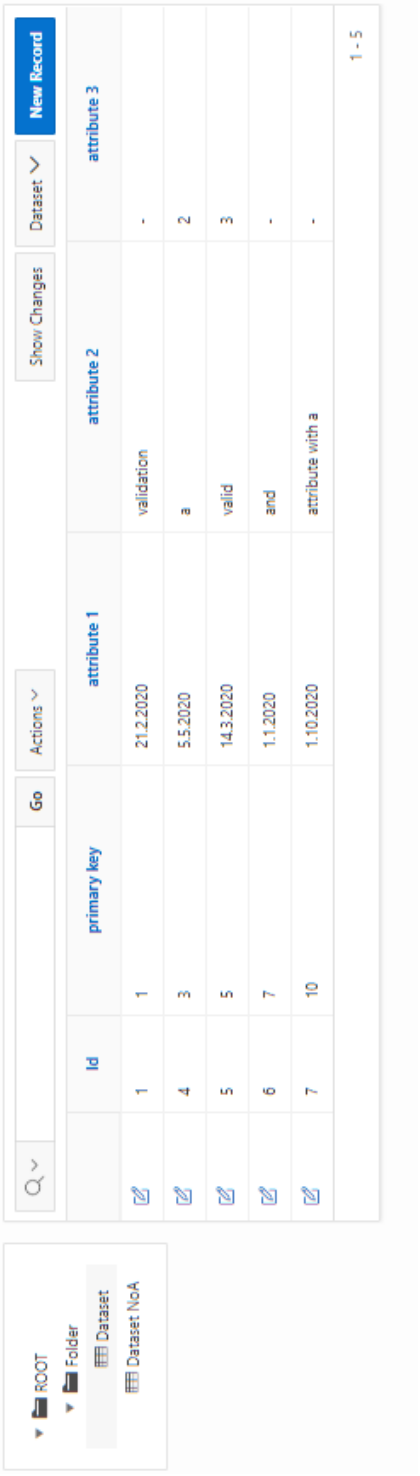
# Obrazovky aplikace



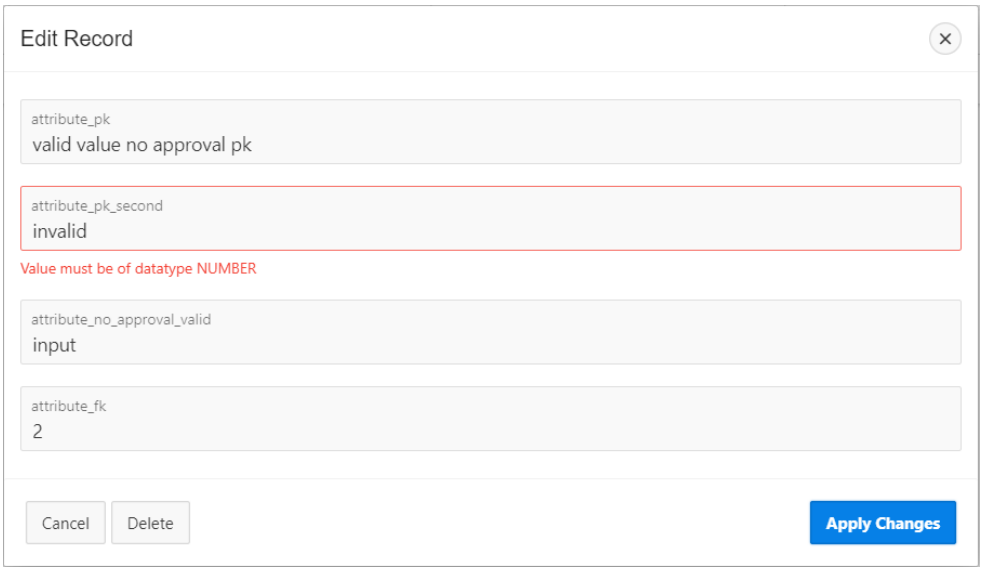

Obrázek B.1: Dialog pro editaci struktury číselníku bez oprávnění



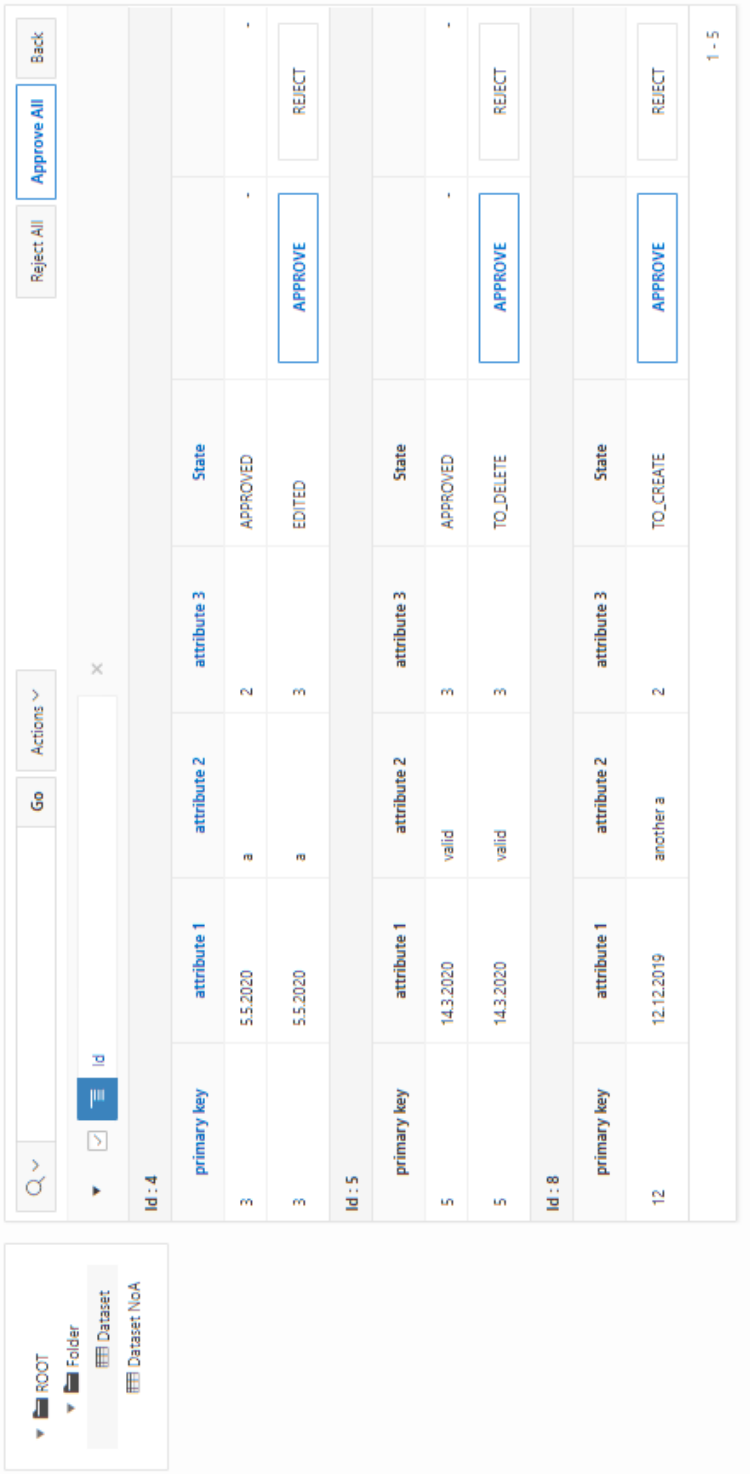

Obrázek B.2: Dialog pro editaci struktury číselníku s oprávněním




Obrázek B.3: Seznam aktálních záznamů číselníku




Obrázek B.4: Dialog pro editaci záznamu




Obrázek B.5: Neschválené změny v číselníku

Obrázek B.6: Historie změn záznamů číselníku

Příloha C

# Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu digitálního média
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `attachments`
  - `GUI_tests` ..... testy uživatelského rozhraní
  - `wireframes.pdf` ..... návrhy obrazovek
- `text` ..... text práce
  - `DP_Weberova_Sarka_2020.pdf` ..... text práce ve formátu PDF