FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Správa číselníků v IT systémech
Student:	Přemysl Dědic
Vedoucí:	Ing. Jiří Mlejnek
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

Cílem práce je navrhnout a implementovat systém správy a distribuce číselníků.
1) Analyzujte požadavky pro správu číselníků. Na základě těchto požadavků navrhněte aplikaci podporující evidenci IT systémů, jejich číselníků a uživatelů, kteří je mohou editovat. Systém musí podporovat verzování struktur i hodnot jednotlivých číselníků.
2) Navrhněte rozhraní pro komunikaci jednotlivých systémů s touto aplikací, které bude použito pro distribuci číselníků. Výběr zvoleného řešení zdůvodněte.
3) Navrhované řešení implementujte, důkladně otestujte a zdokumentujte.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

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

V Praze dne 28. ledna 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Správa číselníků v IT systémech

Přemysl Dědic

Katedra softwarového inženýrství)
Vedoucí práce: Ing. Jiří Mlejnek

28. června 2019

Poděkování

Chci poděkovat širší rodině i kamarádům za svatou trpělivost, kterou se mnou měli během dálkového studia. Mnohdy jsem pro sebe uzurpoval čas, jež jsem měl věnovat jim. V neposlední řadě též vedoucímu práce, jehož trpělivost byla prověřena též.

Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, 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 autorského zákona.

V Praze dne 28. června 2019 ……

České vysoké učení technické v Praze
Fakulta informačních technologií

Odkaz na tuto práci

Dědic, Přemysl. *Správa číselníků v IT systémech*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019.

Abstrakt

Práce se věnuje číselníkům v informačním systému. Definuje jejich strukturu, zkoumá možnosti jejich uložení přímo v prostředí objektového rozšíření relační databáze Oracle, které demonstračně využije pro vývoj serverové části REST rozhraní pro publikaci číselníků. Součástí je vytvoření GUI klienta rozhraní, který umožňuje verzovanou editaci jak struktury, tak i obsahu číselníků.

Klíčová slova číselníky, verze, správa číselníků, objektové typy Oracle, systémová integrace, analýza požadavků

Abstract

The work deals with code lists. It defines their structure, examines the possibilities of storing them directly in the environment of the Object extension of the Oracle relational database. Finally we use the Oracle to develop the server part of the REST interface for publishing them. It also includes the creation of a GUI client interface that allows versioned editing of both the structure and the contents of the code lists.

Keywords code lists, versioning, managing code lists, Oracle objects, system integration, requirement analysis

Obsah

Seznam obrázků

Úvod

Informační systémy obsahují sady údajů, které lze při různých stupních exaktnosti označit za číselníky. Ze své povahy nejde o data, která jsou zásadními doménovými daty informačních systémů, jsou však potřebná pro jejich segmentaci při analýzách, výměně dat mezi nimi a při prezentaci dat uživatelům. Práce spočívá v analýze, implementaci a nasazení samostatné komponenty, která číselníky udržuje, vydává je komunikujícím systémům a prostřednictvím grafické nadstavby umožňuje jejich správu, změny a opravy. Změny definic číselníků i jejich obsah je verzován, veřejně publikované verze jsou odděleny od verzí pracovních.

Práce vychází z komerčního zadání privátní společnosti. Nad uživatelským zadáním jsem nejprve provedl analýzu požadavků. Rozsah byl zúžen na požadavky, realizované přímo v rámci této práce, a na ostatní, které budou realizovány později jako další rozvoj zpracované základní verze. Vyčleněný rozsah byl dále analyzován, identifikovány případy užití a zpracován doménový model vytvářené komponenty. S ohledem na podmínky nasazení byly zvoleny vhodné technologie, model rozvržen do vrstev a zvolen způsob realizace. Navržené řešení bylo poté implementováno. Výsledkem je provozuschopná, otestovaná a zdokumentovaná aplikace.

Kapitola 1

Cíl práce

Cílem práce je analyzovat, navrhnout, implementovat a otestovat systém pro správu a publikaci číselníků, dále označovaný jako *CC2*. Je určen k nasazení jako samostatná komponenta informačního systému konkrétní privátní společnosti, jejíž zadání, zúžené na rozumný rozsah pro prvotní verzi, se stalo základem této práce. Motivací pro práci je vědomí, že osvědčí-li se CC2 ve svém základu, bude dále dopracován podle potřeb zadavatele a postupně propojen s desítkami interagujících systémů, ve kterých nebude nutno číselníky udržovat izolovaně, ale jednotně prostřednictvím nové komponenty.

Vedlejším cílem práce je bližší seznámení s prostředky softwarového inženýrství praktickou aplikací znalostí nabytých během studia tohoto předmětu a také hlubší prozkoumání možností objektových rozšíření prostředí Oracle, které jde nad rámec použití běžných prostředí RDBMS a skrývá možný potenciál přímého zapouzdření práce s fyzickými daty. Typy aplikací, které by mohly objektové prostředí RDBMS efektivně využít, by profitovaly ze zvýšení bezpečnosti uložených dat a efektivity přístupu k nim vložením vrstvy základní logiky s definovaným rozhraním namísto přímé expozice dat ORM vrstvám vyšších vrstev aplikací.

Poptávka samostatné komponenty spravující číselníky a umožňující náhrady mnohdy komplexních informací jednoduchými zástupnými identifikátory vznikla z praxe. Informační systém, ve kterém má být nasazena, představuje distribuované prostředí desítek interních i externích komponent komunikujících s různě velkou latencí. V současné komunikaci jsou přítomny redundantní, avšak poměrně samostatné struktury, povahy pouze popisné a nikoliv hlavních doménových dat. Nasazením komponenty pro jejich správu se očekává zlepšení přesnosti těchto údajů, zlepšení organizace práce s nimi a zvýšení efektivity komunikace, neboť většina obchodních procesů s těmito strukturami nepracuje přímo a mnohdy jsou v plné šíři potřebné pouze pro zobrazení uživateli nebo tisk.

Kapitola 2

Analýza zadání

Existence problému údržby číselníků je vnímána obecně, v mém případě zejména díky desítkám let praxe. Dostatečně velké informační systémy sady údajů číselníkové povahy obsahují. Jsou zpravidla popisné povahy, vysoké redundance a užívány mimo hlavní obchodní logiku systému. Jejich relevance pro obchodní logiku je rozmanitá, je jim však společné, že nebývají součástí klíčů instancí základních entit, odtud též jejich popisná povaha. Společná je jim též skutečnost, že stávají mimo hlavní pozornost a rozvoj systémů až do doby, kdy původně marginální oblast začne spotřebovávat zdroje nutné k jejich konsolidaci, potřebnou pro interoperabilitu systémů. Toto uvědomění si potřeby konsolidace je předstupněm rozvoje procesů MDM, jež jsou mnohdy však příliš robustní a finančně náročné vzhledem k řešené problémové doméně.

Údaje číselníkové povahy se informační systémy snaží evolučně, nebo řízeně, vytěsnit do samostatných celků, na které se pouze odkazují. Taková strukturální změna zvýší výkon systémů a umožní samostatnou správu číselníků, která též zahrnuje jejich očištění od zjevných chyb, formálních duplicit a balastu. Ve struktuře dat, vyměňovaných v distribuovaném prostředí zadavatele a jeho partnerů, tvořeném desítkami interních i externích komponent komunikujících s různě velkou latencí, je uvedený trend velmi zřetelný.

Subsystém CC2 je určen ke skladování číselníků, které chápeme jako množinu záznamů silně souvisejících údajů:

- referenčních, tvořících zpravidla část klíčových doménových dat; jedná se většinou o faktorizaci popisných údajů umožňující udržet ostatní data ve 3. normální formě (názvy, značky, jména, výrobci vozidel, aj.),
- často a opakovaně používaných v různých částech informačního systému,
- v rámci informačního systému téměř statických (vzhledem k rychlosti změn ostatních dat),

- značně samostatných a nezávislých na jiných skupinách doménových dat; zpravidla jsou podstatné jen pro určité úkony, doplňují se pouze při zobrazování v UI, pro tisk, apod.,
- vhodně seskupitelných do sad, skupin, podle místa výskytu, způsobu použití,
- globálního charakteru pro spolupracující subsystémy partnerů (jsou společné nebo externě dané).

Záznamy číselníku vyjadřují popis variant vlastností instancí jedné entity. Popis budeme rozumět hodnotu n-tice atributů, která je referencována nějakým rozlišovacím kódem, který je jedním z atributů. Jako vhodnou definici lze též užít státní správu [1]. Záznamy v číselníku mají zpravidla omezenou platnost, jsou platné pro určité časové období. V této práci období platnosti nerozlišujeme od jiných atributů a jejich interpretace je dána až konkrétním použitím.

2.1 Analýza požadavků

Úspěšnost realizace softwarového díla je od počátku závislá na míře vzájemného porozumění zadavatele a zhotovitele díla. Inženýrství požadavků je pak činností, který k akceptovatelné shodě vede. Jejím výsledkem je strukturovaný dokument „Specifikace požadavků“, popisující vlastnosti a požadované chování díla, akceptovatelný jak zadavatelem (často technickým laikem) i zhotovitelem (týmem vývojářů). Náprava nedorozumění, vzniklých při definici požadavků, avšak odhalených v pozdějších fázích tvorby díla, je vždy časově a tedy i finančně náročnější a je odpovědností projektového vedoucího dojednat a udržet akceptovatelný kompromis mezi časovou dotací činnosti a přesností a úplností jejího výstupu. Požadavky se dělí na dvě zásadní skupiny:

- **funkční**
  skutečně vykonávané aktivity, požadované činnosti a chování,
- **obecné (nefunkční)**
  omezení a podmínky kaldené na prostředí, výkon, nástroje, postupy.

Zadání, tak jak bylo sepsáno, s nepřesnostmi vyjadřování, neujasněnými pojmy, atd., jsem přepsal do dokumentu „Analýza požavdavků“. Při transkripci jsem označováním zpracovaného původního textu zajistil, že každá věta a pojem má svůj přepsaný obraz. Výsledný dokument byl znovu předán zadavateli k připomínkám a odsouhlasení, tedy potvrzení skutečnosti, že svou představu vidí nejen ve svém zadání, ale též ve zmíněném analytickém dokumentu. Tento proces zpravidla nebývá přímočarý, úkolem analytika, zpravidla ve spolupráci s projektovým manažerem, je udržet počet iterací na akceptovatelné úrovni (1-3) a, v případě nezdaru, eliminovat sporná místa. Možnou prvou aproximací je zúžit rozsah a dohodnout dodání bezesporné makety nebo ideálně prvé verze (součást agilních technik, např. [2]).

2.1.1 Funkční požadavky

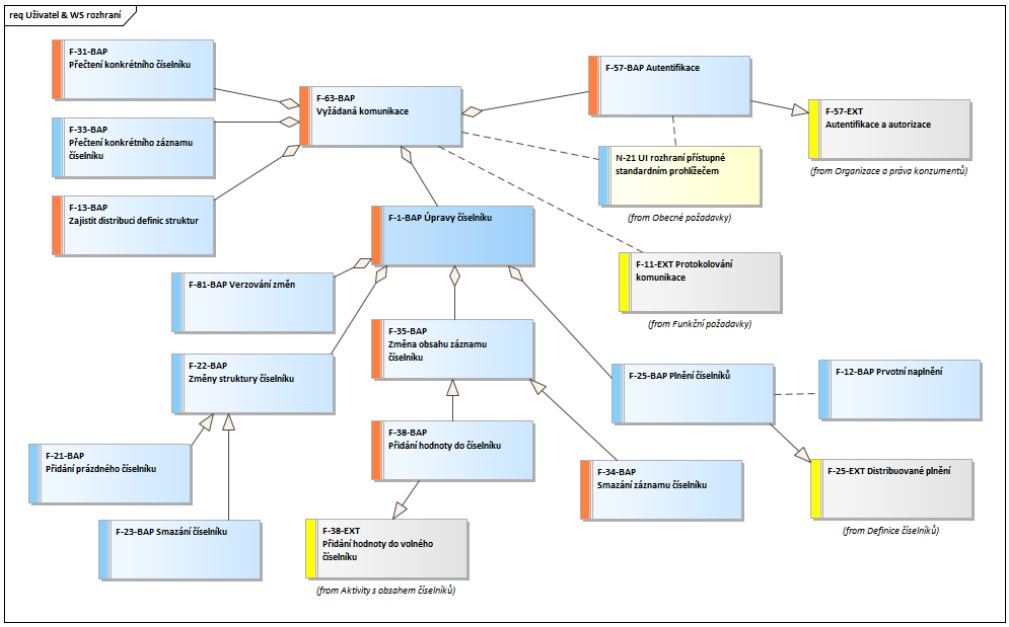
Požadované funkce a chování systému je vhodné zadavatele nechat formulovat v přirozeném jazyce a ideálně písemně, z případných jednání pořídit zvukový záznam. Je třeba, aby sám zadavatel svým formulacím rozuměl v tom smyslu, že podle něj dostatečně popisují jeho představu. Při následné analýze požadavků dochází k překladu volných formulací do kompaktnějších tak, aby každý požadavek byl:

- **jednoznačný**
  formulace je dostatečná, nepřipouští varianty výkladu,
- **splnitelný**
  požadovaná funkčnost je reálná, nepodmíněná, dosažitelná,
- **oveřitelný**
  je zřejmý nebo definovaný způsob, jak ověřit jeho naplnění a odsouhlasit akceptaci zadavatelem.

Výsledek slouží zpravidla jako příloha uzavíraného kontraktu. Je výhodné, pokud již prvotní formulace zadavatele alespoň částečně uvedené principy naplňují. Je vhodné jej k tomu vést, nikoliv však za cenu násilím prosazených formulací, které bývají zdrojem nedorozumění.

Pro účely této práce byl celkový rozsah požadavků zúžen tak (viz obr. 2.1), aby bylo možno reálně nasadit kompaktní a funkční prvou verzi komponenty. Úplný rozsah požadavků je zachycen v příloze (viz obr. C.1). Požadavky 25, 38 a 57 jsou zúžením, specializací obecněji formulovaných požadavků, pro tuto práci.

- **Vyžádaná komunikace** (F-63)
  Práci s číselníky zprostředkovává standardizované rozhraní. Poskytuje služby pro práci se strukturou i obsahem číselníků, čtení i úpravy. Rozhraní akceptuje pouze autentifikované požadavky/uživatele.
- **Změny obsahu číselníku** (F-34,35,38)
  Obsah číselníku musí být možné měnit. Vytvořit nový záznam (ve struktuře, kterou číselník má), smazat záznam a změnit jeho obsah. Tyto funkce poskytne primární rozhraní. Aplikace pro správce, jako nadstavba nad primárním rozhraním, uživateli tyto funkce vhodně zprostředkuje – například vypíše stránkovaný seznam záznamů a u každého prezentuje ovládací prvek kterým lze záznam editovat nebo smazat. Vykreslí ovládací prvek pro přidání nového záznamu. Zobrazí dialog pro nový záznam nebo jeho změnu ve struktuře daného číselníku. Prováděné změny přiměřeně kontroluje. Provedení aktivity nechá uživatele potvrdit. V případě, že aktivitu nelze provést, zobrazí chybovou zprávu svou nebo vrácenou primárním rozhraním.
- **Změny celého číselníku** (F-21,22,23)
  Číselník jako celek musí být možno měnit. Vytvořit nový, smazat existující, změnit jeho strukturu změnou atributu, přidáním a smazáním atributu. Tyto funkce



Obrázek 2.1: Funkční požadavky

poskytuje primární rozhraní. Aplikace pro správce tyto činnosti zprostředkuje – vypisuje seznamy číselníků a prezentuje ovládací prvky pro změnu struktury, přidání a smazání číselníku. Vykomunikuje s uživatelem změny a prostřednictvím primárního rozhraní je provede. Komunikuje chyby a hlášení primárního rozhraním.

- **Čtení obsahu číselníků** (F-31,33)
  Musí být zajištěno čtení číselníku jako celku, volitelně též po částech nebo konkrétních hodnot. Čtení zařizuje primární rozhraní, aplikace pro uživatele komunikaci s rozhraním zprostředkovává a vhodně prezentuje a stránkuje přečtený obsah.

- **Čtení definice struktury číselníku** (F-13)
  Musí být možno získat definici struktury určeného číselníku. Čtení definice zařizuje primární rozhraní. Uživatelská aplikace přečtenou definici vhodně zobrazí.

- **Plnění číselníků** (F-25,12)
  Číselník musí být možno globálně naplnit. Plnění zařizuje primární rozhraní a volitelně jej může zprostředkovat i uživatelská aplikace.

- **Autentifikace** (F-57)
  V cílové infrastruktuře bude aplikace napojena na konkrétní autentifikační modul. Pro účely první verze a této práce je třeba autentifikaci provést, ale jiným způsobem. Použít mockup nebo veřejnou službu. Systém není otevřený, jeho účastníci musí být potvrzeni správcem.

- **Protokolování komunikace** (F-11-EXT)
  Rozšiřující požadavek pro bezprostřední následný rozvoj. Nutno vést protokol o příjmu/výdeji dat a zúčastněných komunikujících stran. Vyvíjené řešení jej musí umožnit realizovat bez zásadního přepracovávání. Musí být dohledatelná identifikace komunikujícího partnera a uživatele, identifikován vstupní dotaz a jeho vyřízení. Identifikací se rozumí vhodné značení na rozhraní, umožňující dohledání komunikace v provozních protokolech nejméně po určenou dobu (cca 6 měsíců).

2.1.2 Obecné požadavky

Obecné požadavky neříkají, co má systém dělat, ale jak to má dělat, za jakých podmínek, v jakém prostředí, při jakých omezeních. Měly by být opět jednoznačné, pokud takové prvotní formulace zadavatele nejsou, nutno je při analýze požadavků zpřesnit do té míry, aby nevyvolávaly spory v popisu procesu akceptace. Například obecný požadavek „Spustitelný na pracovní stanici uživatele“ je nutno zpřesnit doplněním „nejméně v prostředí Windows 10 Redstone 4“. Obecné požadavky, které je z celkové množiny možno realizovat již v rámci této práce jsou následující, popis je pro ilustraci uveden jen u vybraných:

Obrázek 2.2: Obecné požadavky

- **N-01 Výkonnost při změnách**
  Počty vkládaných či modifikovaných záznamů budou v řádu jednotek za hodinu. Vkládat či modifikovat údaje lze pouze prostřednictvím interaktivní aplikace (GUI).
- **N-02 Zachovat možnost rozšíření o importy ze souborů**
  Předpokladem je budoucí rozšíření o možnost plnění formou importních souborů ve formátech XML a JSON/CSV/TXT. Řešení je nutno z hlediska architektury

připravovat s ohledem na toto možné rozšíření. Import dat nesmí mít dopad na dostupnost subsystému, chová se jako uživatel, který by postupně provedl editaci definic a dat na importované hodnoty.

- **N-03 Dostupnost GUI aplikace**
  Pro interaktivní práci, tedy referenční aplikaci správce, je požadována dostupnost 12x5 (cca 7-19h v pracovní dny). Provozní a technologické výpadky aplikace jsou tedy možné např. večer a v noci. Součástí řešení není zohlednění klientů CC2, implementovaných v systémech partnerů.

- **N-04 Dostupnost webových služeb rozhraní,**
  N-05 Omezení servisních odstávek
  Dostupnost online služeb 24x7. Mimořádné servisní odstávky v délce nejdéle 2 dnů a nejvíce 2x v jednom kalendářním měsíci, vždy pouze mimo pracovní dny.

- **N-07 Odolnost vůči výpadkům a haváriím**
  N-06 Časová flexibilita procesů
  Systém se musí umět zotavit po vlastní havárii nebo výpadku okolí (okolní systémy, dodávka el. energie, apod.). Zotavení musí proběhnout bez ztráty dat a narušení konzistence procesů probíhajících před havárií/výpadkem, nesmí dojít k vypuštění probíhajícího procesu / jeho části, nebo naopak k jeho duplicitnímu běhu či zpracování. Případné interní procesy nesmí být vázány na konkrétní čas, je třeba je definovat způsobem cca 2x do hodiny, cca 3x denně, apod.

- **N-11 Typ backend databáze**
  Backend – databáze Oracle Enterprise 11g R2, provozována na OS HP-UX 11.31.

- **N-12 Typ aplikačního serveru**
  Virtualizovaný OS pro aplikační server v prostředí VMware, minimálně Windows 2008 Server (64bit) nebo Red Hat Enterprise Linux Server release 6.4 (64bit). Jako vlastní aplikační server preferován GlassFish Server Open Source Edition 3.1.2.2 (build 5). Případný HTTP server Apache verze 2.4.32 a vyšší.

- **N-15 Členění vrstev aplikační logiky**
  Členit aplikační logiku do vrstev, dělit mezi backend databázi, samostatné komponenty aplikačního serveru a vlastní aplikaci.

- **N-16 Použití produktů třetích stran**
  Použití produktů třetích stran (např. knihoven) je možné. Musí být dokumentován přesný rozsah jejich použití, způsob získání a instalace. Musí být zajištěna možnost jejich užití (licence) a nutné aktualizace.

- **N-17 Standardy**
  Maximální používání veřejných standardů, zejména na rozhraních systému. Součástí řešení není zohlednění napojení konkrétních systémů partnerů, ale obecné, standardizované rozhraní.

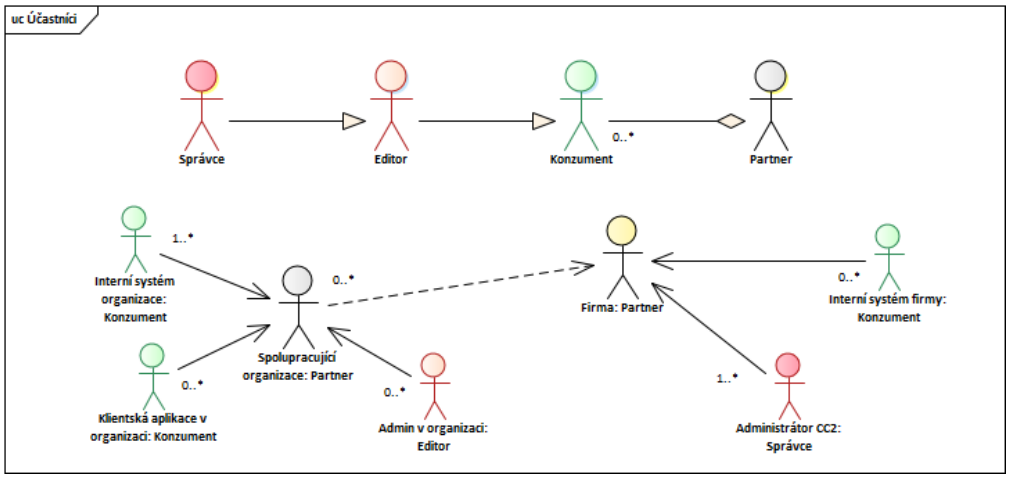
- **N-21 GUI rozhraní přístupné standardním prohlížeče**
  Nad standardizovaným rozhraním bude postavena grafická nadstavba, určená pro uživatele – správce, která bude vhodným způsobem funkčnost primárního rozhraní zprostředkovávat. (Musí být postavena způsobem, který dovoluje spuštění prostřednictím běžného webového prohlížeče.)

- **N-22 Realizační horizont**
  Systém CC2 implementovat do konce roku 2019, nejméně v rozsahu možnosti získat číselníky přes veřejná rozhraní publikovaná ČKP, a možnosti je spravovat ze strany ČKP. Rozšíření v podobě distribuované správy a publikování listenerů pro registraci klientů, kteří mají být automaticky notifikováni, bude řešeno následně jako aktualizace. Předpokládaná technologická životnost systému je plánována na pět let.

- **N-32 Znakové sady**
  Používaná znaková sada je „utf-8“, případné další podporované jsou „windows-1250“, „iso-8859-2“. Tato tři kódování jsou používána v systémech partnerů a nesmí se předpokládat připravenost je překódovat na jejich straně.

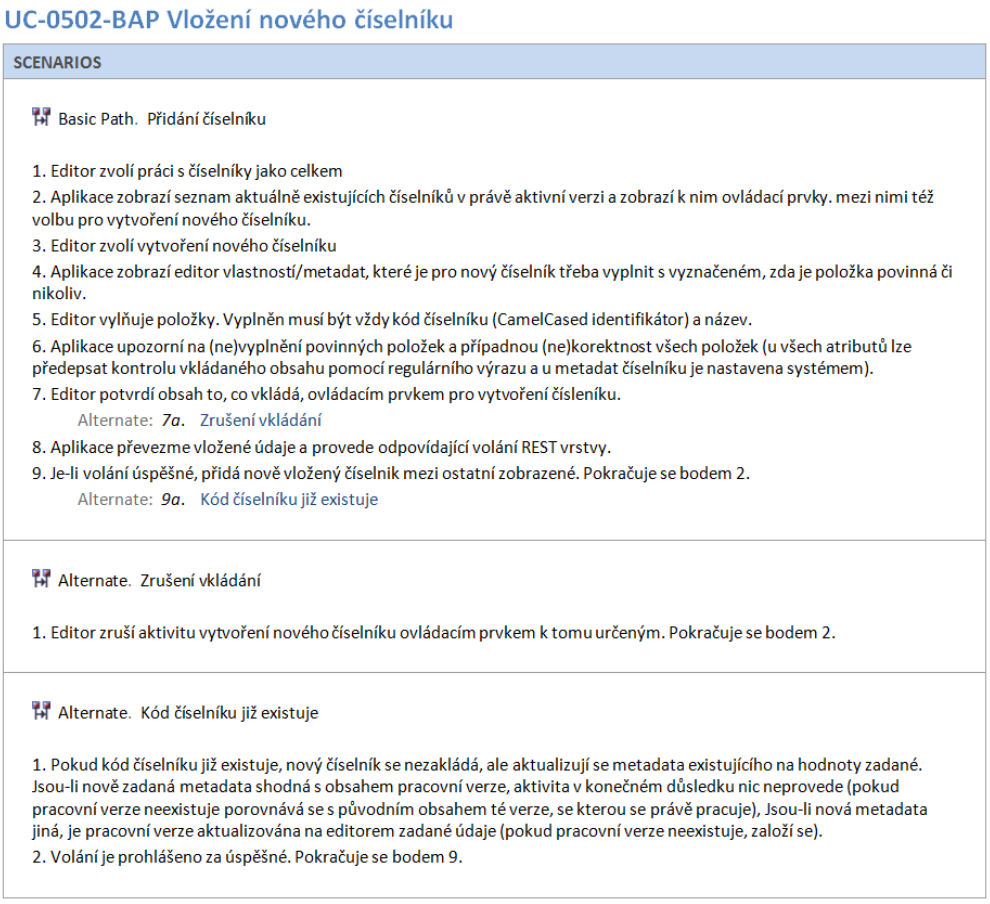
2.1.3 Řešené případy užití

Případy užití znázorňují funkční požadavky zápisem interakce zúčastněných účastníků v procesu jejich naplňování prostřednictvím scénářů [3], [4]. Scénář představuje seznam kroků, které vymezují činnosti účastníků (jeden krok, jeden účastník). Tento typ zápisu dovoluje dobrou prezentaci dílčích činností a je mediátorem řešení neupřesněných situací mezi zadavatelem a analytikem už pouze tím, že „udělá jeden další krok“ v členění procesů.



Obrázek 2.3: Účastníci – hierarchie a užití

Neexistuje vyvolený standardizovaný způsob zápisu případů užití, nicméně je třeba dodržet určité zásady. Případ užití musí mít vhodný popisný název, hrubý slovní popis se zjevným cílem či výsledkem, zřejmé zúčastněné aktéry, vymezené podmínky spuštění a, zejména kvůli častých odkazů, má být opatřen identifikátorem (očíslován). Vždy se zapisuje hlavní scénář jako pozitivní průchod systémem. Doplní se alternativní průchody v místech zásadních rozhodnutí, větvení procesů. Přidají se známé výjimky vyžadující ošetření formou dalších alternativ. Analytické nástroje poskytují podporu zápisům scénářů, zpracování v nástroji Enterprise Architect demonstruje obrázek (viz obr. 2.4).


UC-0502-BAP Vložení nového číselníku

SCENARIOS

Basic Path. Přidání číselníku

1. Editor zvolí práci s číselníky jako celkem
2. Aplikace zobrazí seznam aktuálně existujících číselníků v právě aktivní verzi a zobrazí k nim ovládací prvky. mezi nimi též volbu pro vytvoření nového číselníku.
3. Editor zvolí vytvoření nového číselníku
4. Aplikace zobrazí editor vlastností/metadat, které je pro nový číselník třeba vyplnit s vyznačeném, zda je položka povinná či nikoliv.
5. Editor vylňuje položky. Vyplněn musí být vždy kód číselníku (CamelCased identifikátor) a název.
6. Aplikace upozorní na (ne)vyplnění povinných položek a případnou (ne)korektnost všech položek (u všech atributů lze předepsat kontrolu vkládaného obsahu pomocí regulárního výrazu a u metadat číselníku je nastavena systémem).
7. Editor potvrdí obsah to, co vkládá, ovládacím prvkem pro vytvoření čísleníku.
   Alternate: *7a*. Zrušení vkládání
8. Aplikace převezme vložené údaje a provede odpovídající volání REST vrstvy.
9. Je-li volání úspěšné, přidá nově vložený číselnik mezi ostatní zobrazené. Pokračuje se bodem 2.
   Alternate: *9a*. Kód číselníku již existuje

Alternate. Zrušení vkládání

1. Editor zruší aktivitu vytvoření nového číselníku ovládacím prvkem k tomu určeným. Pokračuje se bodem 2.

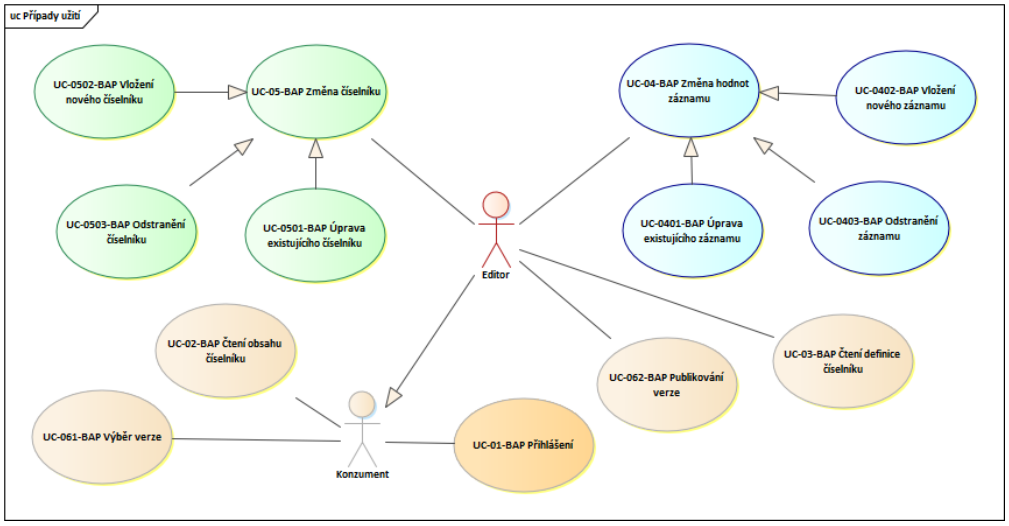
Alternate. Kód číselníku již existuje

1. Pokud kód číselníku již existuje, nový číselník se nezakládá, ale aktualizují se metadata existujícího na hodnoty zadané. Jsou-li nově zadaná metadata shodná s obsahem pracovní verze, aktivita v konečném důsledku nic neprovede (pokud pracovní verze neexistuje porovnává se s původním obsahem té verze, se kterou se právě pracuje), Jsou-li nová metadata jiná, je pracovní verze aktualizována na editorem zadané údaje (pokud pracovní verze neexistuje, založí se).
2. Volání je prohlášeno za úspěšné. Pokračuje se bodem 9.

Obrázek 2.4: Ukázka scénáře ve zpracování EA

Uvedené obecné principy různé notace a metodiky více rozpracovávají a formalizují. Identifikované případy užití řešeného systému jsou zachyceny na obrázku (viz obr. 2.5) v notaci UML.

- **UC-01 Přihlášení**
  Smyslem je autentifikace přistupujícího účastníka komunikace, protože řešený systém nemá být plně otevřen. Všichni autentifikovaní účastníci mohou obsah čísel-



Obrázek 2.5: Případy užití

níků číst, ale pro další činnosti je třeba ověřit též jejich autorizaci pro úpravy číselníků či celkovou správu (potvrzování účastníků a jejich autorizace). Týká se každého účastníka, výsledkem je jeho odmítnutí jako neautentifikovaného, nebo stupeň jeho autorizace.

- **UC-02 Čtení obsahu číselníku**
  Přímočaré přečtení číselníku z primárního rozhraní. V požadavku, který klient rozhraní zadává, specifikuje požadovanou výstupní serializaci JSON nebo XML a také znakovou sadu. Rozhraní vydá číselník v uživatelském tvaru, tedy záznamy, ve kterých jsou hodnoty atributů zapsány podle definice. Seřazení záznamů není definováno. Čtení může probíhat prostřednictvím uživatelské aplikace, která čtení zprostředkuje a výsledek zobrazí v GUI.

- **UC-03 Čtení definice číselníku**
  Varianta případu užití UC-02, avšak nedojde k přečtení obsahu číselníku, ale jeho definice, tedy seznamu atributu číselníků a jejich metadat. Čtení může rovněž probíhat prostřednictvím uživatelské aplikace.

- **UC-04xx Změna hodnot záznamu**
  Smyslem je upravit jeden záznam jednoho konkrétního číselníku. Úpravou se rozumí jeho přidání, odstranění jako celku nebo změna jedné či více hodnot atributů v jeho právě platné definici. Aktivitu může provést pouze účastník s autorizací „editor“ nebo „správce“. Vykonání aktivity nemá specifické podmínky, avšak číselník musí existovat a mít alespoň jeden definovaný atribut. Vkládané hodnoty jsou ověřovány na splnění kontrolní podmínky, zadané pro daný atribut. Změny mohou být prováděny prostřednictvím uživatelské aplikace, která aktivity prováděné prostřednictvím GUI transformuje na volání rozhraní nižší vrstvy.

- **UC-05xx Změna číselníku**
  Cílem je úprava vlastního číselníku, tedy přidání nového, odstranění existujícího jako celku nebo změna jeho definice – přidání, odstranění nebo modifikace atributů. Zápis dílčího scénáře pro přidání nového číselníku ilustruje obrázek (viz obr. 2.4). Aktivitu může provést pouze účastník s autorizací uveditor nebo „správce". Vykonání aktivity nemá specifické podmínky, avšak číselník musí existovat a mít alespoň jeden definovaný atribut. Obsah číselníku se přizpůsobí nové struktuře tak, aby byla dodržena integrita dat. Omezení a kontroly na vyšší úrovni (například splnění změněné kontrolní podmínky na vložené hodnoty) se na již uložených datech neprovádí a jsou na odpovědnosti editora. Změny mohou být prováděny prostřednictvím uživatelské aplikace, která aktivity prováděné prostřednictvím GUI transformuje na volání rozhraní nižší vrstvy.

2.1.4 Vztah případů užití a požadavků

	UC-01-BAP Přihlášení	UC-02-BAP Čtení obsahu číselníku	UC-03-BAP Čtení definice číselníku	UC-04-BAP Změna hodnot záznamu	UC-0401-BAP Úprava existujícího záznamu	UC-0402-BAP Vložení nového záznamu	UC-0403-BAP Odstranění záznamu	UC-05-BAP Změna číselníku	UC-0501-BAP Úprava existujícího číselníku	UC-0502-BAP Vložení nového číselníku	UC-0503-BAP Odstranění číselníku	UC-061-BAP Výběr verze	UC-062-BAP Publikování verze
F-1-BAP Úpravy číselníku				←				←					
F-12-BAP Prvotní naplnění													
F-13-BAP Zajistit distribuci definic struktur			←										
F-21-BAP Přidání prázdného číselníku										←			
F-22-BAP Změny struktury číselníku									←				
F-23-BAP Smazání číselníku											←		
F-25-BAP Plnění číselníků				←				←					←
F-31-BAP Přečtení konkrétního číselníku		←											
F-33-BAP Přečtení konkrétního záznamu číselníku		←											
F-34-BAP Smazání záznamu číselníku							←						
F-35-BAP Změna obsahu záznamu číselníku					←								
F-38-BAP Přidání hodnoty do číselníku						←							
F-57-BAP Autentifikace	←												
F-63-BAP Vyžádaná komunikace		←	←										
F-81-BAP Verzování změn												←	←
F-82-BAP GUI nadstavba		←	←	←				←				←	←

Obrázek 2.6: Mapování požadavků do scénářů případů užití

Funkčnost, požadovaná zadavatelem, a identifikované scénáře případů užití, se musejí

potkat v tom smyslu, že každý scénář, musí realizovat nějaký požadavek. V opačném případě je zbytečný, nesmí dojít k jeho dalšímu zpracování a realizaci. A naopak – každý funkční požadavek musí být realizován alespoň jedním scénářem, v opačném případě se jedná o požadavek nepodchycený do vývoje systému, anebo požadavek zbytečný a má být ze seznamu požadavků vypuštěn, anebo diskutován zvlášť, mimo realizaci. Mapování požadavků a případů užití identifikovaných pro tuto práci ukazuje obrázek (viz obr. 2.6). Vidíme, že požadavek *F-12* se nerealizuje v žádném scénáři. Pokud zadavatel trvá na jeho ponechání, jako v tomto případě, je třeba přesto zajistit jeho realizaci. V tomto případě je *F-12 Prvotní naplnění* specializací požadavku *F-25*, jsou identické s tím, že *F-12* je rovno *F-25* pro počáteční start systému.

2.2 Východiska řešení

Zpracováním analýzy požadavků, po rozboru zadání, jeho vymezením a ujasněním potřebné základní funkčnosti existuje dostatek informací pro návrh základního rámce řešení:

- přizpůsobit existující řešení,
- akceptovat robustnější, širší a dražší existující řešení,
- vyvinout vlastní řešení,
- navrhnout podržet existující stav a řešení na čas odložit.

Ve firemní infrastruktuře zadávající organizace sice aktuálně správa číselníků existuje, avšak jedná se o komponentu technologicky zastaralou, stále větším problémem je zajistit běžný provoz a údržbu, natož rozvoj. S uvážením potřeby mít referenční zdroj číselníků trvale přístupný online, je poslední varianta, tedy odložení řešení, neakceptovatelná. Vývoj vlastního řešení je obchodně vždy nejméně druhým v pořadí, nejlépe je vždy využít řešení existující, a to i při vyšší ceně. Je vyvážena existencí dostupné servisní podpory, kterou by bylo třeba při vlastním vývoji automaticky zajistit navíc, nejméně v rámci záruky. Existujícím řešením jsou věnovány následující kapitoly, avšak plyne z nich neefektivnost těchto možností. Pro dané zadání je efektivní vytvořit vlastní řešení, integrované do firemního prostředí, a po konci jeho technologické životnosti znovu zvážit dále popisované možnosti.

2.3 Stav a možná existující řešení

Poptávka byla zadavatelem formulována kvůli absenci dostatečně samostatného známého řešení. Číselníky jsou součástí všech provozních systémů a mnohdy tvoří značnou část jejich provozní správy (například personalistika, stavebnictví a další výrobní software, systémy státní správy). Rešerše byla v rámci této práce provedena v březnu 2019 a bohužel potvrdila očekávané – správa číselníků je většinou řešena individuálně

v každém jednotlivém systému. O jejich údržbu a aktualizace se většinou stará výrobce systému, případně systém správců, kteří reagují na měnící se podmínky, zákony, normy a také potřeby svých klientů maximálně možností automatizovaně prováděných importů, zpravidla formou datových dávek. Relativně samostatným řešením je *Jednotná správa číselníků*, pro správu několika set číselníků [5]. Zaměřením a obsahem je cílena na oblast, v níž je integrována a externí využití není možné. Nepochybně však vykazuje podobu s hledaným řešením, i důvody jsou obdobné:

> *Vzniku aplikace Jednotná správa číselníků, předchází správa číselníků v předchozím systému. Avšak neustále narůstající počet registrů, systémů a organizací, zapojených do projektu, postupné zastarávání, a nevyhovující způsob přístupu a správy číselníků byl podnětem pro požadavek na nový systém správy číselníků, který by reflektoval požadavky uživatelů, jeho vlastnosti a funkce byly přidávány in-house a bez zbytečného prodlení. Zastřešoval by nejen číselníky a potřeby ÚZIS, ale i ostatních organizací zapojených do projektu NZIS, jak i subdodavatelů dodávajících systémy a registry pro projekt NZIS.*

Aplikace je však opět integrována do většího informačního systému, tedy přímo nepřenosná a vzdálená „krabicovému“ řešení. Sklad číselníků, dostupný jak uživatelsky, jak i pro strojové čtení, nalezneme také u Celní správy [6]. Také jejich řešení nelze přímo převzít, pouze se inspirovat.

2.4 Akceptace robustnějšího řešení

Řešení, která problematiku zpracovávají šířeji, lze rozdělit na dvě hlavní oblasti:

- související s MDM (Master Data Management),
- související se standardem Otevřená data a RDF.

Správa číselníků svou povahou souvisí s oblastí čistoty dat, hraničí tedy s technikami MDM. Existenci vhodných menších řešení se hledáním během řešerší nepovedlo potvrdit. Existují jako části CRM systémů nebo jako platformy (viz např. [7]) pro vývoj, což je však šíří záběru zcela mimo rozsah původního požadavku zadavatele.

Projekty a řešení, zaměřené na „Otevřená data“, mají širší základ, cíl dále v budoucnosti a staví na modernějších technologiích – dotazovacím jazyce SPARQL [8] a daty publikovanými ve standardu RDF [9]. Souvisí s tvorbou rozsáhlých ontologických slovníků, postupně pokrývajících celý svět, všechny existující pojmy [10], [11]. Bude-li dostatečný čas na jejich rozvoj, bude kterýkoliv objekt či pojem namapován na nějaký model, abstraktní entitu, a číselníky se globálně stanou pouhým dotazem, výběrem vlastností příslušných entit. Zde je třeba zmínit projekt státní správy „Otevřená data“ a související otevírání veřejných údajů, v prvé řadě číselníků (viz [12], [13]). Obdobně existuje projekt otevřených dat Evropské unie (viz [14]). Ač věřím v budoucnost těchto postupů, z krátkodobého horizontu nejsou akceptovatelné, zejména pro náročnost na zdroje i čas.

Kapitola 3

Návrh řešení

Rešerše možných řešení vedla k závěru, že při aktuálním zadání a formulovaném krátkodobém horizontu realizace, je efektivní vyvinout řešení vlastní s vědomím, že jde o řešení co do rozsahu nasazení i technologické životnosti pouze lokální. Při překročení či rozšíření formulovaných obecných požadavků na dostupnost a robustnost, nebo plánované živostnosti, může aktualizované zadání znovu uplatněno a výroba systému znovu poptána na nové technologické úrovni.

Zdůrazňovaným požadavkem je standardizace exportovaného rozhraní připravovaného systému. Vzhledem k povaze řešené problematiky se navrhuje implementovat rozhraní typu REST. Cílová firemní infrastruktura sice běžně pracuje se SOAP WS, která se však pro řešenou problematiku nejeví jako vhodná pro svou režii a také vyšší počáteční časovou investici při vývoji řešení na straně partnerů. Očekává se, že absolutní většina komunikace na rozhraní bude spočívat v pouhém vydávání kompletních číselníků nebo jejich indiviudálních záznamů. Zatímco pro SOAP WS je nutno připravit alespoň určitá základní prostředí, což vyžaduje, byť minimální, edukaci na straně partnerů, volání rozhraní typu REST je možno realizovat triviálně a funkčnost lze ověřit i běžným prohlížečem.

Obecné požadavky definují jako uložiště dat prostředí Oracle. Jako aplikační platformu připouští jiné možnosti, z preferencí aplikačního serveru GlassFish vyplývá možnost použít platformu Java(EE). Další současné možnosti (např. Python, C++, .NET) obecné požadavky vylučují, byť je vždy možnost konzultovat úpravu zadání se zadavatelem. Vzhledem k menšímu rozsahu řešené problematiky je vývoj proveditelný na libovolné platformě a omezení na prostředí Oracle a Java, dané obecným požadavkem, lze bez dalšího akceptovat. Při vědomí nevelého rozsahu se nabízí možnost omezit se na pouze prostředí jediné, což zjednoduší správu a nasazení, a navrhnout řešení nativně realizované v prostředí Oracle a jeho objektového rozšíření. Oracle dokumentace uvádí možnost pracovat s objektovými typy běžným způsobem [15], známým z jiných objektově zaměřených jazyků.

Každý systém, aplikace, zahrnuje v různém poměru práci s daty, základní a obchodní logiku. V případě, že logika tvoří podstatné části, jde více o aplikaci v užším slova smyslu,

převažuje-li práce s daty, pak spíše jako reportovací nástroj nad daty. V případě zde řešeného systému CC2 není obchodní majoritní částí logika, ale zejména práce s uloženými daty. Jde o příležitost vyzkoušet praktickou užitečnost objektového rozšíření Oracle a implementovat podstatné části logiky přímo na straně databázového serveru. Sleduji tím maximalistickou aplikaci principu zapouzdření práce s daty v Oracle prostředí do té míry, že bude přímo poskytovat výstupy pro REST rozhraní. Z hlediska integrity a konsistence je to u systému, který má přesně vymezenou a omezenou funkčnost ideální, neboť neumožňuje s daty zacházet jinak, než definovaným způsobem. Nasazení CC2 se předpokládá do prostředí Oracle 11g či 12c, tedy s objektovým rozšířením již dostatečně vyvinutým, aby bylo možno se o uvedené úspěšně pokusit. Mnohé části textu, týkající se práce s prostředím Oracle, nejsou citovány a vycházejí z mé vlastní, více než dvacetileté praktické zkušenosti s verzemi od 8.1.5 po 11g.

Aplikaci pro správce, formulovaná v obecných požadavcích, bude vhodné vytvořit jako další vrstvu nad REST rozhraním systému. Půjde tedy o referenčního klienta, vizuálně prezentujícího jeho funkce a čtená data. Bude postačovat, aby aplikace rozuměla JSON serializaci. Musí umět prezentovat definice číselníků i jejich obsah, který bude nutno stránkovat. Nabízí se možnost využít některý široce užívaný framework, kterým je například Bootstrap [16]. Aplikaci bude možno provozovat prostřednictvím běžného webového prohlížeče. Komunikaci s podkladovým REST rozhraním bude zajišťovat aktivní skriptování prostřednictvím technologie AJAX.

3.1 Číselníky

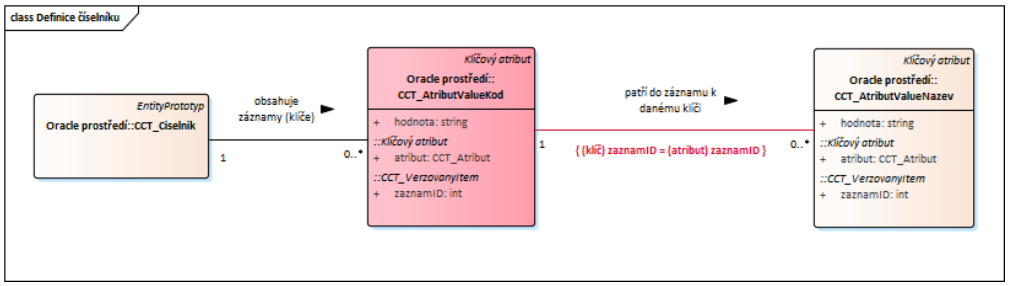
Pro účely této práce považujeme číselník za určený svou *definicí* a *obsahem*. Definicí (číselníku) rozumíme jeho název spolu se seznamem názvů a typů jeho *atributů*. Obsahem pak rozumíme sadu záznamů hodnot jeho atributů. Změny definice i obsahu mohou být prováděny prostřednictvím rozhraní správy číselníků. Čtení lze provádět tímtéž rozhraním, nebo jednodušším rozhraním číselníků (určeno pouze pro čtení).

Atributem rozumíme pojmenovaný typ údaje, společný pro všechny záznamy číselníku. Jeho hodnota nese nějakou vlastnost, stav nebo jinou charakteristiku a může obecně existovat jen pro některé záznamy a pro jiné existovat nemusí. Vlastními atributy rozumíme atributy přiřazené číselníku prostřednictvím rozhraní správy číselníků. Každému číselníku připojí subsytém CC2 automaticky navíc své *technické atributy*, důležité pro interní činnost CC2.

CC2 bude obsahovat několik technických číselníků, interních provozních struktur, uživatelsky přímo nepřístupných. Jedním z technických číselníků je seznam vlastností atributů, neboť, čistě technicky, je definice číselníku sama krátkým číselníkem, jehož záznamy tvoří specifikace jeho atributů. Druhým podstatným interním číselníkem je seznam číselníků obsažených v CC2.

Číselník má vždy právě jediný *klíč*, tvořený podmnožinu svých *atributů*, který svými hodnotami určuje jednoznačný záznam z obsahu číselníku. Hodnota klíče představuje persistentní *veřejný identifikátor* záznamu číselníku, vhodný pro použití v datech vyměňovaných přes různá rozhraní. V každé jednotlivé verzi určuje jednoznačně nejvýše

jediný jeho záznam. Klient, který data zpracovává, si může zpracovávaná data obsahující rozšířit o kompletní sadu atributů prostřednictvím veřejných dentifikátorů v datech obsažených. CC2 si doplňuje do každého číselníku jeden interní *technický klíč*, sestavený výhradně z technických atributů, potřebný pro interní práci systému. Aktuálně implementovaná prvá verze systému definuje klíč napevno. Tvoří jej jediný atribut, pojmenovaný „Kod“. V případě, že se ukáže potřeba definovat složené klíče, může být takový požadavek dodatečně implementován.



Obrázek 3.1: Identifikace záznamu číselníku

Na číselník lze pohlížet jako na prostou množinu hodnot jeho atributů. Seznam hodnot klíče určuje seznam záznamů, pokud budeme umět ke každé hodnotě klíče najít hodnoty ostatních atributů číselníku. To však lze, neboť všechny takové hodnoty mají identickou hodnotu *zaznamID* jako příslušná hodnota klíče. V prvém kroku vybereme pro číselník hodnoty klíče a získáme množinu identifikací záznamů *zaznamID*. Ve druhém vybereme pro všechny atributy jejich hodnoty se zvoleným *zaznamID* a tyto tvoří jeden záznam číselníku (viz (viz obr. 3.1)).

3.2 Verze

Verze je pojmenovaný a adresovatelný (přístupný) obsah CC2, tedy logicky konsistentní souhrn definic a obsahu číselníků. Při provádění změn jsou prováděné změny skryty, nejsou veřejně přístupné, existují pouze v dočasné pracovní oblasti. Po ukončení, kontrole, a explictním potvrzení (publikování) provedených změn je aktuální stav pojmenován a zpřístupněn - vzniká nová verze, adresovatelná a využitelná.

Elementární změny, které lze v CC2 provádět, jsou:

- změna definice (atributu, číselníku),
- změna obsahu – hodnoty jakéhokoliv atributu.

Změny lze provádět pouze prostřednictvím rozhraním správy (číselníků). Druhé, aplikační rozhraní, je určeno pouze ke čtení dat. Obsah CC2 po aplikaci potvrzených změn nazýváme verzí. Verze má dvě základní vlastnosti –- viditelnost a přístupnost. Z hlediska viditelnosti může být:

- privátní; její obsah je viditelný všem klientům u stejného partnera jako klient, který změny provedl,
- veřejná; obsah je viditelný všem (všem klientům u všech partnerů).

Z hlediska přístupnosti může být Verze:

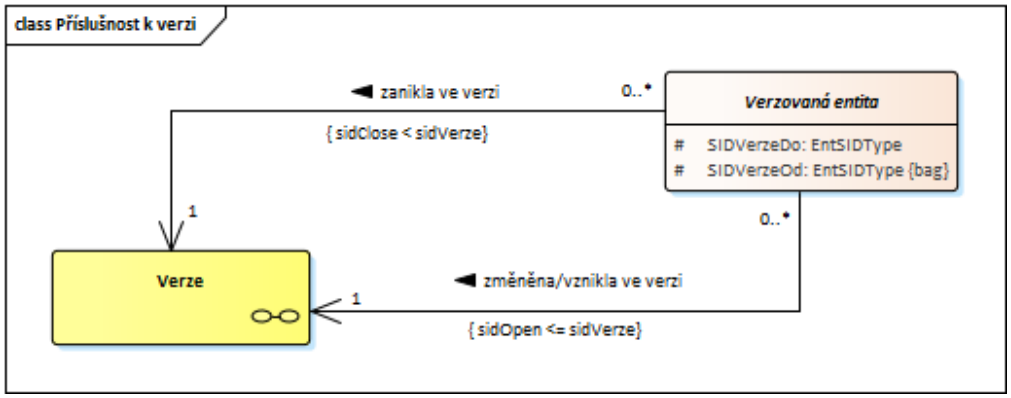
- pracovní; nepřístupná veřejně, přístupná výhradně rozhraním správy (číselníků) a pouze ze strany partnera, který ji založil, je možné v ní provádět elementární změny (na straně jednoho Partnera existuje nejvýše jediná),
- publikovaná; nelze v ní provádět změny, je přístupná i prostřednictvím aplikačního rozhraní, má další dva podstavy:
  - archívní; jiná než naposledy publikovaná verze (už byla nahrazena mladší publikovanou verzí),
  - zbytečná; archivní verze, která byla partnerem označena jako nepodstatná, chybná, existuje její náhrada.

Ve vztahu k provádění změn se verze chová jako transakce -– všechny dílčí prováděné elementární změny ze strany stejného partnera se akumulují do pracovní verze. Potvrzení provedených změn je provedeno publikováním dosud pracovní verze, která se tím stane publikovanou a zároveň privátně dostupnou. Zároveň se založí nová, prázdná, verze pracovní. Publikovanou verzi není možné „vzít zpět“. Je však možno ji nahradit, opravit, publikováním verze další. Novým publikováním se předchozí publikovaná verze stane archivní. V případě zjištění chyby nebo jiných důvodných situacích lze archivní verzi označit jako zbytečnou a napovědět tak klientům, že ji mají přeskočit či urychleně aktualizovat na verzi další.

V CC2 existuje role správce (číselníků), která je specializací partnera. Správci jsou dostupné všechny publikované verze bez ohledu na jejich viditelnost. Úkolem správce je publikované změny konsolidovat, ověřovat a, pokud je vyhodnotí jako rozumné, v maximální míře je publikovat veřejně. Všechny publikované verze jsou podkladem pro práci správce, který z nich může klonovat své pracovní verze, jakkoliv pak do nich zasáhnout a provést další úpravy dle své úvahy. Následně je publikuje a poté je může zveřejnit. Každá veřejná verze je vždy viditelná všemi klienty všech partnerů a vždy byla publikovaná správcem.

3.3 Entita, instance entity, aktivita

Entitou rozumíme učitý typ informace v CC2 (číselník, atribut, hodnota). Instancí entity rozumíme jednu vyplněnou a uloženou informaci takového typu (konkrétní název číselníku, konkrétní názvy a typy atributů konkrétního číselníku). Každá instance má přiřazen začátek a konec své aktivity . Začátek aktivity je určen verzí, ve které byla instance vytvořena (či změněna do dané podoby). Konec aktivity je určen verzí, v níž instance poprvé neexistuje. Interval mezi těmito verzemi nazýváme intervalem aktivity



Obrázek 3.2: Příslušnost k verzi

(instance) a říkáme, že instance je aktivní od verze X do verze Y, pokud byla ve verzi X vytvořena (změněna) a verze Y je prvou verzí v hierarchii verzí, ve které již neexistuje. Instance mimo svůj interval aktivity nejsou pro běžné čtení dostupné. CC2 si, čistě formálně, definuje i „budoucí“ verzi , v níž přestávají existovat všechny zbylé instance. Instance, které mají konec aktivity nejpozději s aktuálně publikovanou verzí , nazýváme archivní a celý jejich soubor archivem .

Interval aktivity je určen dvojicí údajů *sidOpen, sidClose* (viz obr. 3.2). Každá aktualizace nastaví hodnotu *sidOpen* na novou, odebranou ze vzestupné sekvence. Při publikování nové verze se pouze další hodnota sekvence zapíše do tabulky verzí a opatří se kódem a názvem verze. Instance, přístupné v dané verzi, se velmi dobře poznají, platí pro ně jednoduchá podmínka $sidOpen <= sidVerze < sidClose$.

Provedení změny neznamená nikdy přímou úpravu instance jakékoliv entity. Změna je provedena v těchto krocích:

- původní instanci je nastaven konec aktivity na hodnotu pracovní verze (tím tato instance formálně zaniká),
- je vytvořen klon instance původní,
- potřebné změny se aplikují na vytvořený klon; vznikne tak formálně nová instance odrážející provedené změny,
- začátek aktivity je nastaven na konec aktivity instance původní (tedy an hodnotu pracovní verze), konec aktivity formálně na „budoucí“ verzi.

Vkládá-li se instance jako nová, přeskočí se prvé tři kroky. Zaniká-li instance (byla smazána), provede se pouze první krok. CC2 uchovává instance dvojího typu:

- základní; nesou popisné údaje a interní identifikátor (sid) spojený se vznikem instance,

- úplné; nesou navíc údaje o vymezení intervalu aktivity a link na případnou svou náhradu, pokud zanikly (příprava pro další vývoj komponenty).

Mezi základní patří například verze, můžeme pro ně formálně definovat interval jejich aktivity jako „stále aktivní“. Úplnými jsou číselník, atribut, hodnota atributu.

3.4 Oracle Objects

Prostředí Oracle nabízí již od verze 8 objektové typy jako objektové rozšíření RDBMS [15], [23]. Ve svém raném věku doplňovaly strukturované typy, které bylo možno používat programovém prostředí PL/SQL, avšak na rozdíl od nich je bylo též možno ukládat do relačních databázových tabulek, které ovšem Oracle sám spravoval. Objektové typy se své podstatě chovaly jako čisté datové objekty, měly pouze jediný konstruktor a jejich metody i objekty samotné podléhaly, na rozdíl od objektových jazyků jako C++ nebo Java, mnohým omezením. Oracle postupně práci s objekty zdokonaloval až do současných verzí 18, 19. Verze 9i je vybavila konstruktory, dědičností a polymorfismem, ve verzi 11.1 doplnil Oracle možnost volat v rámci redefinované metody potomka původní metodu předka (do té doby bylo nutno kód metody předka zahrnout do její redefinice v potomkovi). Zápis objektů odpovídá jiným jazykům a existuje též možnost jej dokumentovat ve stylu javadoc (viz výpis C.5).

3.5 Oracle PL/SQL

PL/SQL je silně typový procedurální jazyk navržený pro prostředí Oracle a práci se SQL příkazy (viz [24]). Programové bloky PL/SQL Oracle serverem překládány a ukládány v databázi spolu s daty. Přímý SQL přístup k datům je obalen aplikační obchodní logikou, naprogramovanou prostřednitvím PL/SQL, a před vnějším světem skryt. Tento přístup se obecně nazývá paradigma tlusté databáze. Znamená, že k fyzickým datům není z vnějšího světa umožněn přímý přístup, ale lze k nim přistupovat a měnit je pouze určeným konkrétním způsobem, daným obchodní logikou. Efektivně se zajistí jejich konsistence i bezpečnost, protože SQL příkazy i PL/SQL kód běží společně ve stejném procesu na straně serveru pod určenými právy a využívá se tím robustnost Oracle databáze.

3.6 REST

Zkratka z anglického „Representational State Transfer“ – architektura rozhraní, navržená pro distribuovaná prostředí [18], [19]. Na rozdíl od rozhraní XML-RPC či SOAP, je orientována datově, nikoli procedurálně. Datové objekty jsou v REST terminologii nazývány zdroji. Zdroje mají vlastní identifikátor, kterým je URI. Pro přístup k nim definuje REST čtyři základní metody (create, read, update, delete), realizované nad HTTP protokolem jeho čtyřmi metodami, „slovesy“:

- **POST** – vytvoří nový zdroj v zadané kolekci zdrojů a vrátí jeho identifikaci v hlavičce uvLocation, případně uvContent-Location,
- **GET** – přečte individuální zdroj, identifikovaný URI, nebo kolekci zdrojů,
- **PUT** – aktualizuje zdroj, identifikovaný URI, nebo kolekci zdrojů, případně vytvoří nový zdroj, je-li jeho identifikátor znám dopředu,
- **DELETE** – odstraní či, v našem případě, zneplatní zdroj identifikovaný URI.

Praktické informace jsou dobře popsány například v práci Todda Fredricha [20]. Tamtéž je k nalezení popis stavových kódů, které má vracet HTTP server jako reakce na realizovaná volání.

3.7 Serializace na rozhraní

Je třeba rozhodnout, v jakém tvaru budou zdroje – číselníky – na REST rozhraní serializovány. S ohledem na formulované obecné požadavky a určitou flexibilitu zvolíme:

- **XML**
  ověřený standard, může být přísně typový, rozumí mu široké spektrum nástrojů [21],
- **JSON**
  jednoduchá flexibilní struktura, volně typovaná, rozlišuje pouze čísla, texty, kolekce a objekty, jejichž obsahem je rekurzívně totéž, široce adoptováno a podporováno zejména ve skritovaných aplikacích [22].

Obě serializace jsou lidsky čitelné, což je důležité při řešení problémů. Mají přibližně stejnou režii a, jsou na sebe, v řešeném rozsahu, transformovatelné. Obě podporují použití jmenných prostorů a tím odlišení informací různé povahy. Použití těchto dvou serializací by mělo být dostatečně flexibilní pro implementace klientů na straně partnerů.

3.8 Bootstrap a FooTable

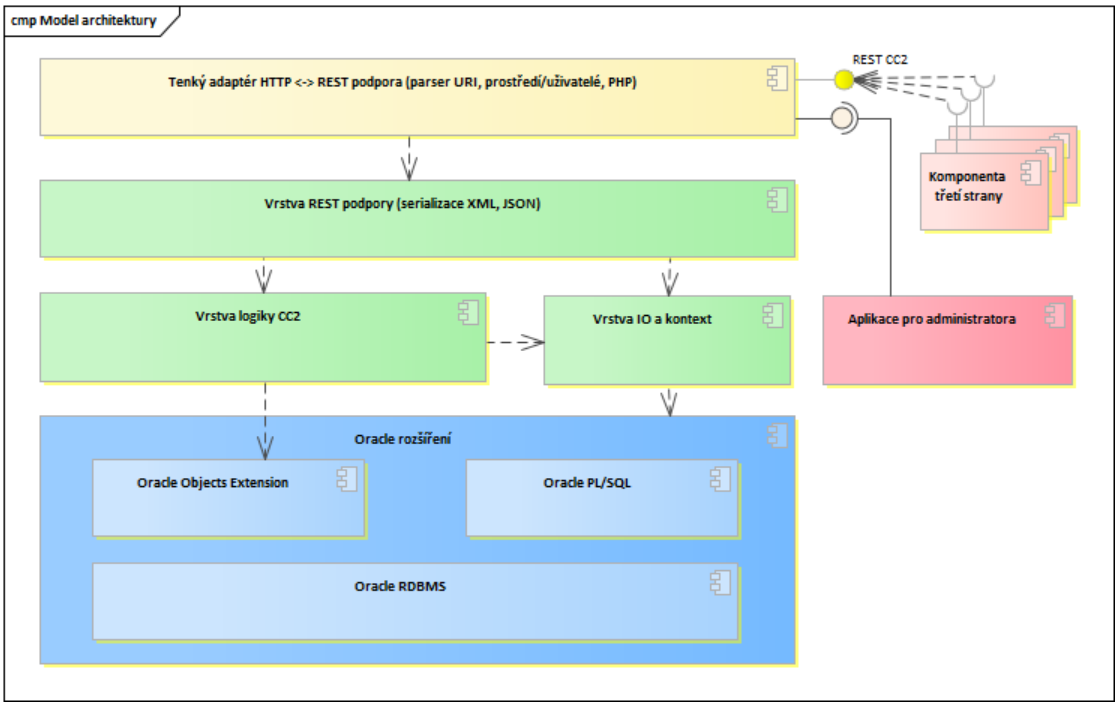
Open source prostředí pro vývoj postavený na standardech HTML, CCS a JS. Jednoduchá sada nástrojů pro tvorbu webu a webových aplikací. Obsahuje šablony založené na HTML a CSS, umožňuje úpravy typografie, formulářů, tlačítek, navigace a dalších komponent vizuálního rozhraní [16]. Jeho použití předpokládá znalosti HTML a CSS. Předpokládá se také znalost skriptování, které je zpravidla nezbytné. Pro efektivní práci s tabulkami se ukázala vhodnou komponenta *FooTable* [17] pod GPLv3 licencí, která je na rozdíl od známějšího a rozsáhlejšího frameworku *DataTables* jednodušší a tím snáze pro tuto práci přizpůsobitelná.

Kapitola 4

Realizace

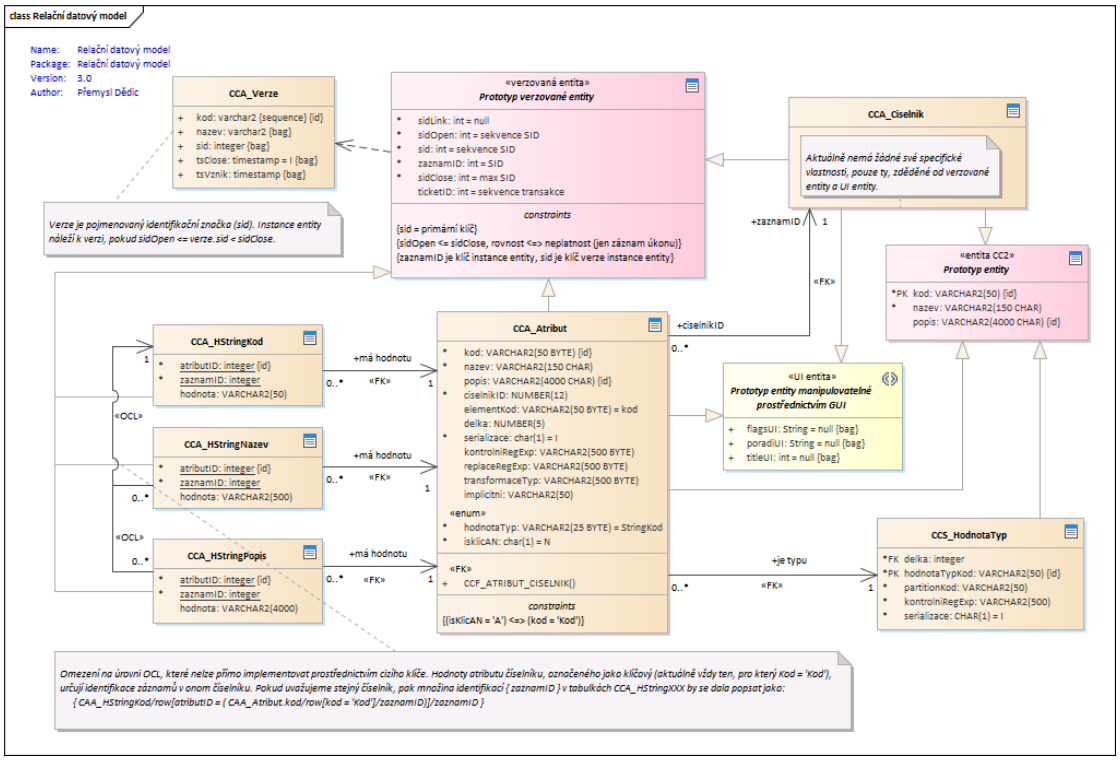
4.1 Celková architektura

Systém CC2 byl vytvořen jako klasická třívrstvá aplikace. Reálně je však členěna do šesti logických vrstev. S ohledem na budoucí nasazení CC2 byla fyzická vrstva v uvedeném logickém členění realizována v prostředí Oracle 11g.



Obrázek 4.1: Celková architektura

- **fyzická vrstva** (tlustá):
  - úložiště vlastních dat,
  - základní logika (integrita, verzování, zapouzdření do základních objektů),
  - protokolování přístupů a změn,
  - podpora REST služeb,
- **aplikační server** (tenká):
  - REST rozhraní,
  - WEB aplikace,
- **klient** (statický):
  - aktivně skriptovaný klient,



Obrázek 4.2: Relační model

Fyzické uložení dat do databázových tabulek ukazuje obrázek (viz obr. 4.2). Ukládání hodnot je segmentované – velikost hodnoty je lze přidělit ve třech úrovních a každá z nich je ukládána samostatně. Způsob implementace fyzické vrstvy také definuje omezení na

velikost hodnot atributů číselníků na cca 4 000 znaků (základní datový typ Oracle do verze 11), případně 32 kB (možnost od verze 12). V případě potřeby ukládat jako hodnoty atributů číselníku specifické údaje, například zvukové záznamy, obrázky, prostorové definice, apod. je možné segmentaci ukládaných dat adaptovat tak, aby pokryl potřebné typy údajů. Na fyzické vrstvě jsou prostřednictvím integritních omezení realizovány i základní vazby entit.

Základní logika je implementována prostřednictvím databázových pohledů a triggerů. Zajišťuje přidělování technických klíčů a příslušnost prováděných změn k verzím číselníků. Pohledy tvoří vůči vyšším vrstvám proxy pro čtení obsahu číselníků a dovolují skrýt vnitřní segmentaci uložení údajů. Prvý z pohledů čte skutečně v definici uložené údaje, druhý, pro reálný provoz navíc substituuje případně nastavené implicitní hodnoty.

Protokolování přístupů a změn je samostatnou a oddělenou částí základní logiky. Pracuje s tikety, objekty nesoucí identifikaci přistupujícího partnera, konkrétního uživatele, času vzniku a doby trvání. Mohou spolupracovat s vstrvou REST rozhraní a svázat se s konverzačním tokenem HTTP komunikace, například se session identifikátorem, takže má HTTP konverzace přímý odlesk v protokolování přístupů k fyzické vrstvě. Jakákoliv činnost, vykonávaná prostřednictvím vrstvy logiky ve fyzických datech do nich propisuje také identifikaci tiketu. Každá provedená změna má tedy u sebe zaznamenána svého autora.

Vrstvu aplikační logiky jsem vytvořil, opět s ohledem na nasazení CC2, v prostředí Oracle PL/SQL. Je realizováno třídou *CCT_CiselnikREST*. Obsahuje skupiny metod pro podporu REST rozhraní:

- output_XXX – připraví serializované fragmenty číselníku:
  - DefDefinice – připraví definici číselníku, tedy jeho metadata a strukturu, tvořenou definicemi atributů,
  - DefObsah – připraví serializovanou kolekci záznamů číselníku, neaplikuje případné implicitní hodnoty,
  - Obsah – připraví pouhé čtení číselníku, aplikuje případné implicitní hodnoty.
- read_XXX – serializovaný celý číselník:
  - DefDefinice – definice číselníku, tedy jeho metadata a struktura, tvořená definicemi atributů,
  - DefObsah – obsah, tedy kolekce záznamů číselníku, neaplikuje případné implicitní hodnoty,
  - Obsah – obsah, tedy kolekce záznamů číselníku, aplikuje případné implicitní hodnoty,
  - DefKomplet – kompletní definice číselníku, metada, struktura i obsah, přitom plikuje případné implicitní hodnoty.
- create_XXX – přijme serializovaná data číselníku a vytvoří z nich:

  - DefDefinice – definici číselníku, tedy jeho metadata a strukturu, každý obsažený atribut je přidán do definice číselníku, pokud již existoval, je aktualizován,
  - DefObsah – obsah, tedy každý obsažený záznam je přidán do obsahu číselníku, pokud již existuje, je aktualizován,
  - DefKomplet – aktualizuje jak strukturu, tak i obsah.
- delete_AtributANY – odstraní z definice číselníku atribut s uvedeným klíčem,
- smaz – odstraní kompeltní číselník.
- publishWorkVersion – vypublikuje pracovní verzi, pokud obsahuje změny, hierarchicky volá publikační metodu na své struktuře a záznamech.

Volání výkonných metoda podpůrného objektu je řízeno PL/SQL balíčkem *CC2B_WS_A9*, který obsahuje implementaci CRUD operací a nastavení případné konkrétní verze (implicitně aktuální). Je volán tenkým PHP skriptem, posazeným na aplikačním serveru do prostředí Apache. Konfigurace a bližší popis je uveden v příloze.

4.2 Vyjednání obsahu

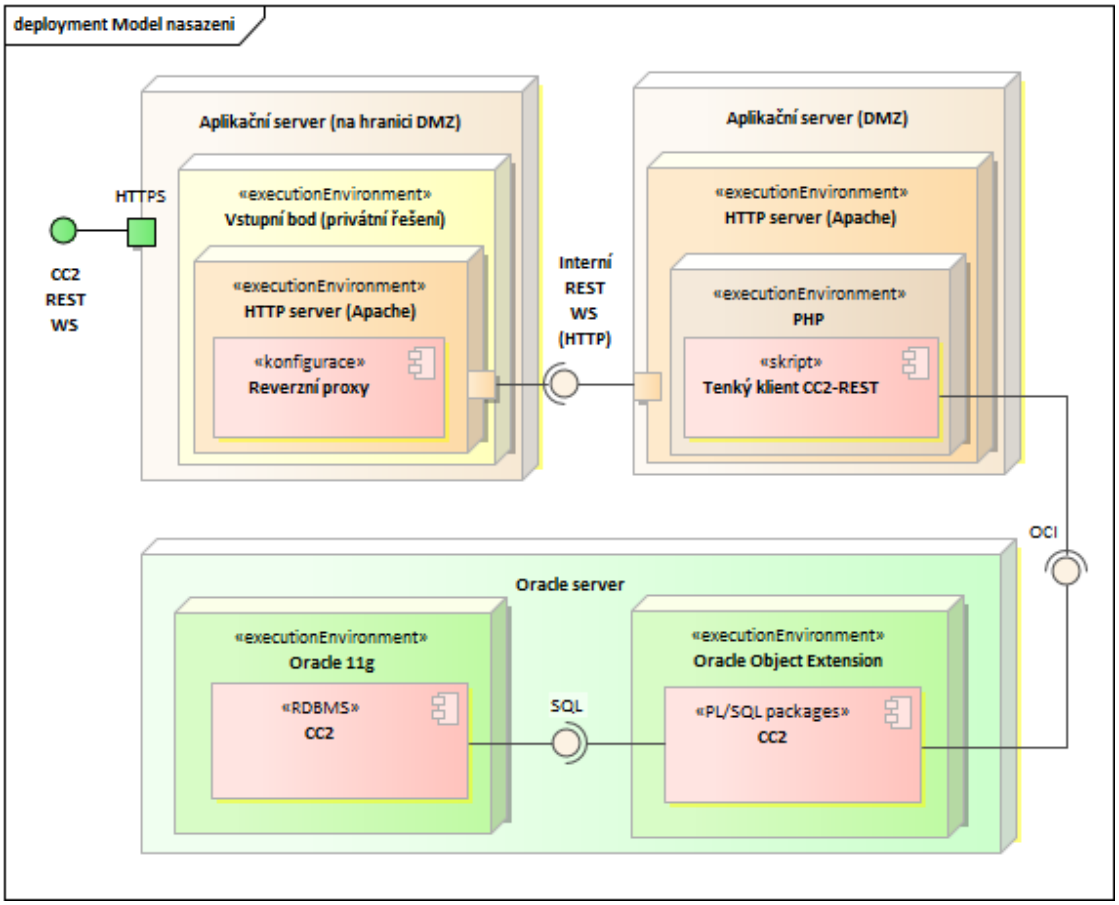
Klient si může od serveru vyžádat způsob serializace odpovědi prostřednictvím standardních HTTP hlaviček:

- **Accept**
  Hodnotou je povolený typ média. Hlavičku odesílá klient serveru a jejím prostřednictvím navrhuje způsob serialilzace. Povolenými hodnotami jsou *application/json*, *application/xml* a *text/xml*. Server odpoví odpovídajícím způsobem. Pokud hlavička chybí, prozkoumá server, zda od klienta dorazila hlavička *Content-Type*. Pokud ano a obsahuje uvedená média, odešle odpověď ve stejném tvaru. Ve všech jiných situacích vrací odpověď ve tvaru XML dokumentu.
- **Accept-Charset**
  Hodnotou je povolená znaková sada. Hlavičku odesílá klient serveru a jejím prostřednictvím navrhuje znakovou sadu, ve které požaduje odpověď. Povolenými hodnotami jsou *utf-8*,*windows-1250* a *iso-5589-2*. Pokud hlavička chybí, prozkoumá server, zda od klienta dorazila hlavička *Content-Type*. Pokud ano a obsahuje některou z povolených znakových sad, odešle odpověď ve stejném tvaru. Ve všech jiných situacích vrací odpověď v *utf-8*.

4.3 Zabezpečení

Ačkoliv není komponenta pro správu číselníků zásadní z hlediska bezpečnosti, bude součástí interní firemní infrastruktury a neměla by být jejím slabým článkem. je též prav-

děpodobné, že v případě úspěšného nasazení mohou být formálně mezi číselníky adoptovány některé seznamy na pomezí neveřejných informací nebo osobních údajů a vhodné zabezpečení služeb, komponnetou poskytovaných, pokládám za nezbytné.



Obrázek 4.3: Nasazení do IT infrastruktury

Komponentou publikované REST rozhraní je vystaveno na veřejném vstupním uzlu. Nicméně přístup cizích komponent k REST rozhraní bude ve firemním prostředí zabezpečen prostředky, standardními pro toto prostředí. Pro práci s číselníky, neobsahujícími kritická data, jsou více než dostatečné:

- **zabezpečený protokol**
  https protokol zabezpečený nejméně kryptografickým protokolem TLS 1.2, což znemožňuje odposlouchávání komunikace a vyhovuje aktuálně používaným standardům [25],
- **monitorovaný vstupní bod**

vstupní uzel je monitorovaný firemní implementací SIEM [26], která může, v případě vyhodnocení incidentu nebo útoku, přístup zablokovat,

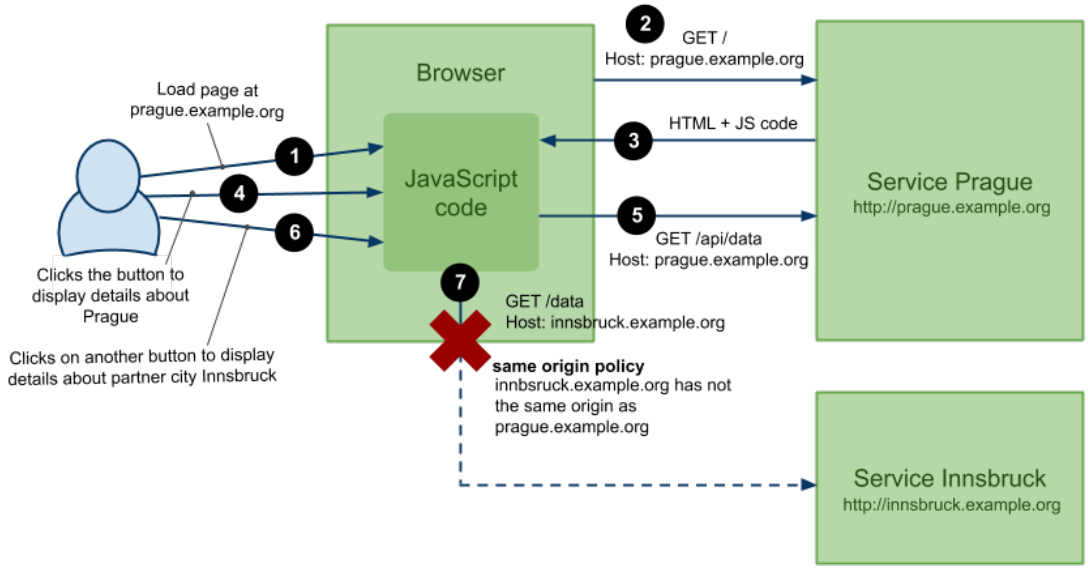
- **nutnost použít certifikát**
  komunikující komponenta se musí představit platným certifikátem, vydaným důvěryhodnou certifikační autoritou nebo firemní certifikační autoritou,
- **mapování na interní server**
  klientské požadavky nejsou do vnitřní sítě směrovány přímo, ale až na úrovni aplikačního protokolu http, vstupní bod je zprostředkovává v režimu reverzního proxy serveru [27],
- **síťovým propojením ve vyhrazené VLAN**
  mapovaná komunikace je na linkové vrstvě oddělena od ostatního provozu směrováním do vyhrazené VLAN [28],
- **oddělením provozního databázového účtu**
  provozní účet, kterým tenká vrstva REST služeb přistupuje k nižším vrstvám v prostředí Oracle, je samostatným účtem, který má silně omezená práva, nemá přístup k žádným datům ani nástrojům a umí pouze volat delegované funkce.

4.4 Autentifikace a autorizace

Autentifikace komponenty, přistupující přímo k REST rozhraní, je dána použitým certifikátem. Certifikát je vydáván a evidován na *partnera*, případně, je-li to účelné, na přistupující komponentu partnera nebo konkrétního uživatele. Vnitřní infrastruktura, ve které bude číselníková komponenta nasazena, již obsahuje komponentu správy certifikátů, která k nim zároveň eviduje přidělená práva a je specifickou nadstavbou LDAP. Autorizace bude v budoucnu prováděna právě jejím prostřednictvím. Dočasně, pro účely vývoje a zkoušení a hodnocení číselníkové komponenty, byl vydán testovací certifikát na fiktivního partnera, který je možno použít pro přístup ke komponentě reálně nasazené v privátní síti.

Aplikace pro správu číselníků, tedy grafická webová nadstavba REST rozhraní, provádí autentifikaci a autorizaci obdobně jako přímé REST rozhraní, protože využívá shodného vstupního bodu a jeho infrastruktury. Pro účely vývoje a zkoušení byla doplněna další možností autentifikace – má implementováno Google Sign-In rozhraní [29], postavené nad standardem Auth 2.0, takže vlastník Google účtu se může do aplikace pro správu číselníků přihlásit do testovacího prostředí jako administrátor.

Robustnější konstrukce, než výše popsaná, nebyla shledána efektivní vzhledem ke skutečnosti, že se jedná rozhraní mezi konečným, malým a dopředu ověřeným počtem přistupujících klientů, a nikoliv o rozhraní veřejné (neznámý, potenciálně značný počet nedeterministicky se chovajících klientů). Je možné, že v budoucnu dojde k omezení práv přístupu a konkrétní číselníky budou přístupny pouze konkrétním uživatelům. počáteční verze je v tomto směru otevřená, protože je jejím smyslem publikovat zejména číselníky veřejné mezi komunikujícími částmi systémů partnerů.



Obrázek 4.4: CORS komunikace

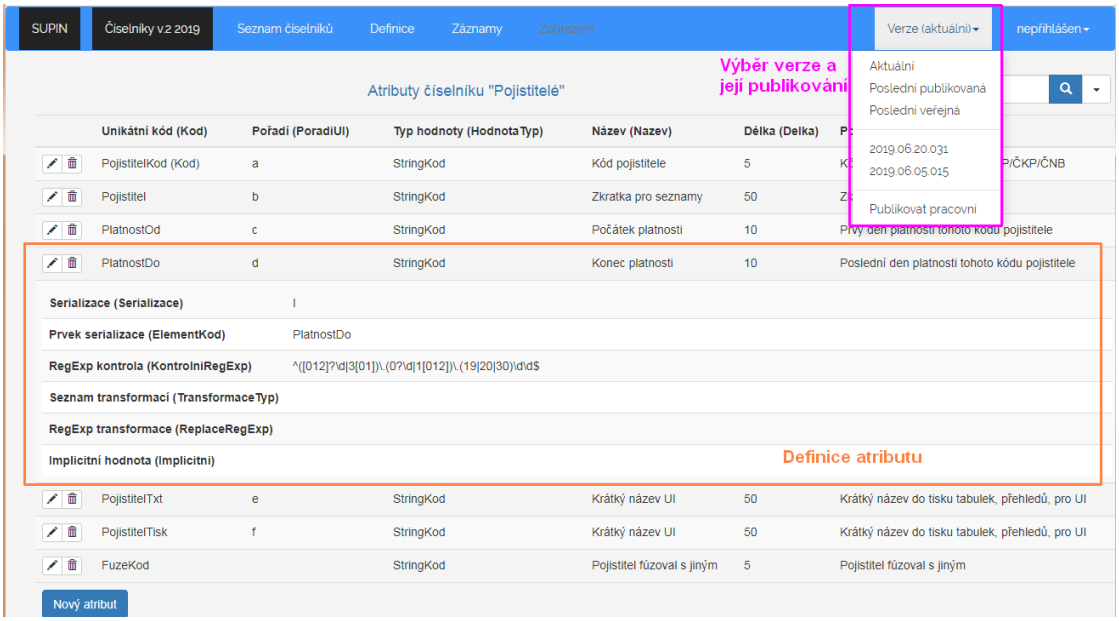
4.5 CORS

Moderní prohlížeče implementují nepodmíněně bezpečnostní politiku známou jako Same Origin Policy – skript ve webové stránce může získat data pouze z lokace se shodným původem, pro jiné lokace prohlížeč pro skript zablokuje přečtení obsahu odpovědi. Původ dvou lokací se považuje za shodný, pokud se liší pouze v cestě, shodná musí být trojice [protokol, host, port]. Pro úspěšnou komunikaci skriptů na straně klienta s publikovaným rozhraním číselníků je nutno implementovat CORS protokol (viz obr. 4.4). Představuje techniku, jak umožnit skriptu webové stránky přečíst data z webové stránky, umístěné v jiné lokaci. Aplikace, vyvinutá jako grafická nadstavba rozhraní k číselníkům obsahuje skripty, které užitím AJAX s tímto rozhraním komunikují a které ne nutně komunikují se stejným vstupním bodem – záleží na umístění samotné aplikace a vstupního bodu rozhraní.

4.6 Aplikace pro správu číselníků

Aplikace pro práci s úložištěm číselníků je statická HTML stránka, obsahující aktivní skriptování. Statická stránka je postavena na frameworku Bootstrap, obsahuje menu a definované oblasti, do kterých skripty umisťují jednotlivé zobrazované komponenty. Prostřednictvím skriptů a technologie AJAX srtánka komunikuje s REST rozhraním pro práci s číselníky. Implementuje všechny CRUD operace, tedy vytvoření nové instance (viz obr. C.3), čtení stávajícího stavu, aktualizaci instance (viz obr. C.4) i její smazání (viz obr. C.5). Kromě čtení probíhají všechny operace v pracovní verzi čísel-

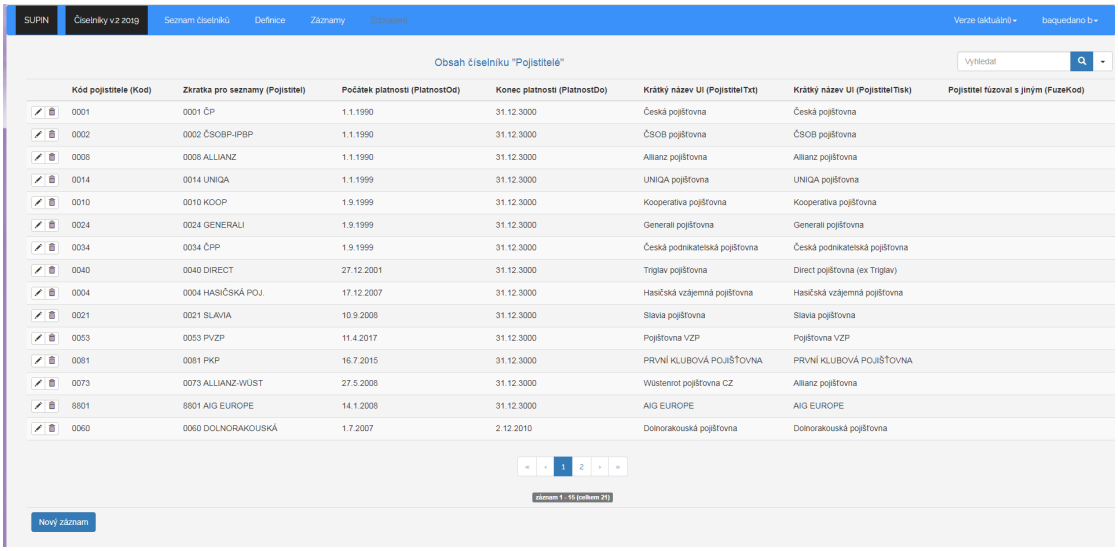
níků. Pracovní verze do sebe akumuluje všechny prováděné změny a zůstává pracovní až do jejího explicitního publikování správcem. Při publikování vzniká nová veřejná verze. Zvolenou veřejnou verzi lze prostřednictvím menu (viz obr. 4.5) nastavit jako aktivní. Procházený obsah číselníků i jejich struktura v takovém případě odpovídá stavu v okamžuku jejího publikování. Vydání dat konkrétní verze zařídí přímo rozhraní číselníků, verze je součástí identifikátoru zdroje, tedy URI. Není-li verze specifikována, používá se pro čtení aktuální, tj. poslední veřejně publikovaná, a pro úpravy verze pracovní.



Obrázek 4.5: Zobrazení definice číselníku

Obrázek (viz obr. 4.5) ukazuje, jakým způsobem je vizuálně prezentována struktura číselníku. Strukturou se rozumí seznam atributů, každému z nich je nutno nastavit jeho charakteristiky. Na řádku jsou uvedeny ty z nich, které jsou pro uživatele dostatečně významné a přehledné, ostatní jsou zobrazeny v detailním pohledu po kliknutí na konkrétní řádek. Charakteristiky jednoho atributu jsou zachyceny uvnitř oranžového rámečku. Správce má u každého řádku, tedy atributu, ikonky, jejichž prostřednictvím může daný atribut z číselníku odstranit, nebo změnit jeho charakteristiky. Klíčový atribut z číselníku odstranit nelze, je jeho integrální součástí a odstraní se případně spolu s číselníkem (každý číselník má tedy vždy alespoň jeden atribut a tím je jeho klíč).

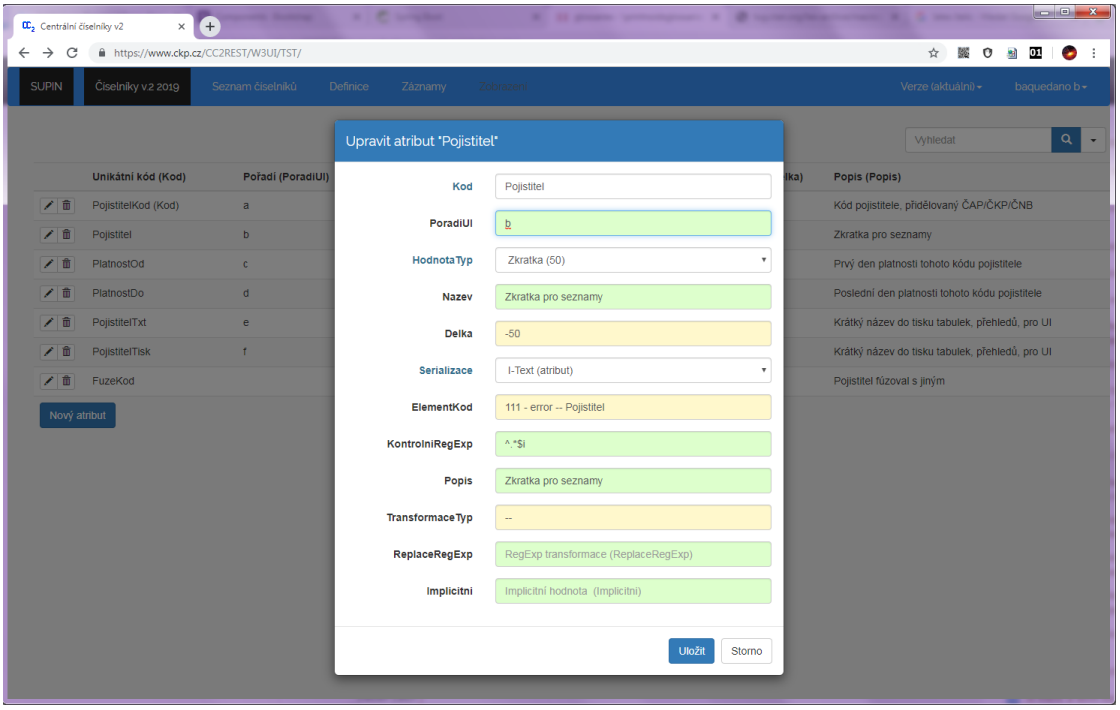
Obrázek (viz obr. 4.6) ukazuje, jak aplikace vizuálně zobrazí obsah číselníku, tedy jeho záznamy. Opět platí – co řádek to jeden záznam, podstatné atributy jsou zobrazeny přímo na řádku, ostatní se zobrazí po kliknutí na řádek daného záznamu. Aplikace rozpozná menší šíři zobrazovaného okna, například na tabletu, a může přesunout některé atributy záznamu z řádku do zobrazení detailu řádku. Seznamy, které jsou delší než nakonfigurovaný počet najednou zobrazených se stránkují. V hodnotách atributů lze



Obrázek 4.6: Výpis záznamů číselníku

vyhledávat. Správce má na začátku řádku k dispozici ikonu, jejímž prostřednictvím může záznam upravit (viz obr. C.4).

- **Kod** – Povinný. Krátký kód, identifikace atributu, pro strojové zpracování. Pokud není zadána hodnota ElementKod, pak také název atributu/elementu/klíče, do kterého se jeho hodnoty serializují. Aktuálně implicitní klíč (pro rozlišení atributů v rámci číselníku).
- **Poradi UI** –
- **HodnotaTyp** – Povinný. Základní typ hodnoty, kterých může atribut nabývat. Může být dalším vývojem rozšiřováno podle potřeby. Aktuálně:
  - StringKod – krátké kódy a názvy do 50 znaků,
  - StringNazev – delší názvy do 500 znaků,
  - StringPopis – dlouhé texty do 4000 znaků (akceptováno implementační omezení, může být v dalším vývoji nebo v jiném prostředí odstraněno),
  - VyberMoznosti – výběr z možností; vlastní hodnota je v tomto případě vždy *StringKod*; seznam voleb je uložen v tabulce CCA_AtributOption pro veřejný identifikátor atributu; aktuálně není uživatelsky dostupné, volby nutno zapsat přímo do relační tabulky.
- **Delka** – Nepovinný, má implicitní hodnotu odvozenou z typu. Maximální délka obsahu jako textu. U čísel počet znaků, kterými lze číslo zapsat. Implicitní hod-



Obrázek 4.7: Definice číselníku – zadání atributu a kontrola

nota je dána typem dat *HodnotaTyp*, která je jako konfigurace uložena v tabulce CCS_HodnotaTyp a není uživatelsky přístupná.

- **Nazev** – Povinný. Deskriptivní a krátký název atributu pro nadpisy a uživatelská rozhraní.
- **ElementKod** – Nepovinný, má implicitní hodnotu rovnou *Kod*. Název XML elementu, XML atributu nebo JSON klíče, do kterého se hodnoty serializují.
- **Serializace** – Nepovinný, má implicitní hodnotu „I“. Určuje typ serializace hodnot:
  - pro XML:
    - **I** – jako hodnota atributu,
    - **T** – jako hodnota elementu,
    - **C** – jako CDATA hodnota elementu,
    - **B** – jako hodnota elementu v kódování base64,
  - pro JSON:
    - **I, T, C** – jako hodnota klíče,
    - **B** – jako hodnota klíče v kódování base64.

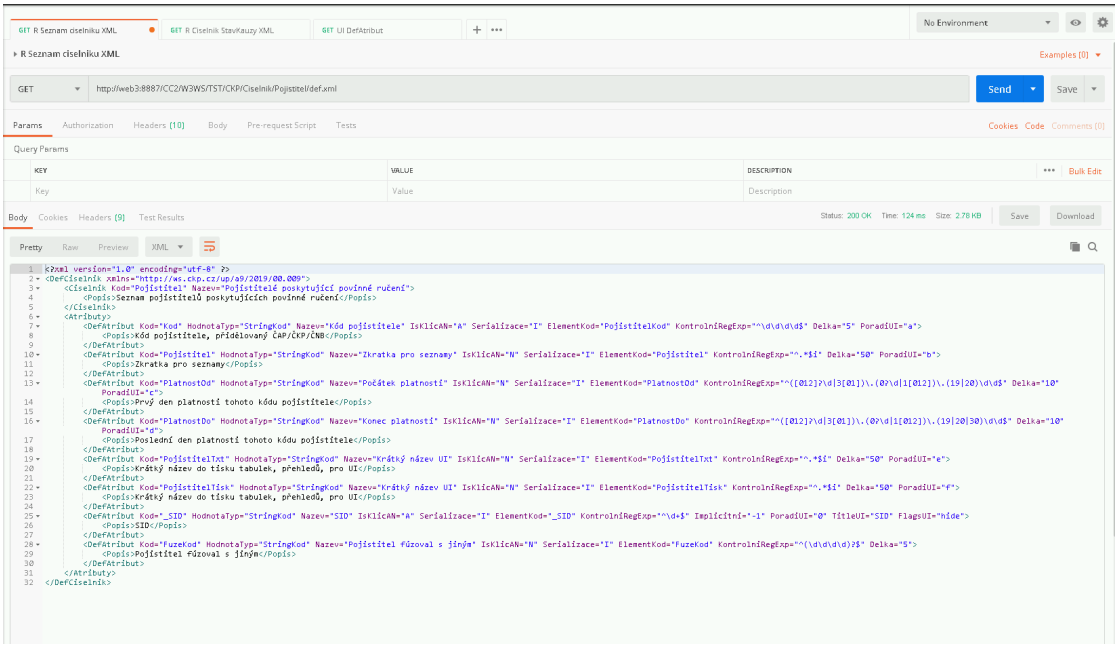
- **Popis** – Nepovinný, má implicitní hodnotu rovnou *Nazev*. Popis atributu. Měl by srozumitelně a výstižně vyjadřovat jeho užití, aby se dal jako referenční popis používat v dokumentech, které se jej týkají.
- **KontrolniRegEx** – Uživatelský kontrolní mechanismus. Regulární výraz, který, je-li uveden, se na konci zpracování vstupních dat aplikuje na ukládanou hodnotu (tedy až po případných transformacích a substitucích). Pokud hodnota nevyhoví, kontroly havarují a hodnota se neuloží. Používá se i na ošetření null / not null situací. Implicitní hodnota je dána typem dat *HodnotaTyp*, která je jako konfigurace uložena v tabulce CCS_HodnotaTyp a není uživatelsky přístupná.
- **ReplaceRegEx** – Určeno pro budoucí rozšíření. Regulární výraz, kterým lze uživatelsky transformovat vstupní data. Souvisí s atributem *TransformaceTyp* – Interně zná subsystém určité typizované transformace – CamelCase, lower, UPPER, sigle_spaced, IČO, RČ, apod. Hodnota určuje seznam transformací, které se mají na vstupní hodnotu aplikovat před tím, než je připravena uložení, ve fázi „kontrol“.
- **Implicitni** – Implicitní hodnota, alikuje se, pokud je hodnota atributu NULL.
- **HintUI** – Seznam hintů pro UI, oddělený mezerou. Implementačně závislé. Aktuálně používá UI pro zobrazení komponentu FooTables, která hinty určitým způsobem implementuje:
  - **nosort** – podle tohoto sloupce výpisu se neřadí,
  - **toggle** – ne vypisuje se do řádku, ale až do detailu řádku,
  - **wide** – vypisuje se do řádku pro široká UI, jinak do detailu řádku,
  - **hide** – je skrytý, nevypisuje se.

Při úpravách zobrazený dialog průběžně kontroluje korektnost vložených hodnot a podbarví pole, podle výsledku kontroly. Povinné atributy mají názvy odlišeny barvou. Povinnost se odvíjí od obsahu *KontrolniRegEx*. Pokud nepovoluje nevyplněnou hodnotu, je chápán jako povinný.

4.7 Testování a dokumentace

Testování během vývoje v prostředí Oracle je omezené. Neposkytuje komfort unit testů a je třeba mít neustále připraveny skripty, které ověřují konkrétní funkčnosti. Vývoj probíhal inkrementálně po jednotlivých objektech a každý inkrement byl odzkoušen. Pojmem „skript“ se rozumí zpravidla účelově vytvořená procedura, která vyplní relační tabulky vhodnými testovacími daty a poté další procedura provede ověření seznamu konkrétních činností (viz výpis C.6). Tento postup je sice přímočarý, ale chce pečlivost, netoleruje nepozornostní chyby, protože při tomto způsobu testování se chyby v samotném testování hůře odhalují. Pokud se však vždy každý další jednotlivý krok rozmyslí, nepůsobí tento relativně prehistorický postup problémy. Jak již bylo dříve řečeno, řešená problematika

není rozsáhlá – to jest lze si udělat a udržet představu najednou o celém celku, proto způsob testování dostačuje, byť je tento proces v moderních jazycích elegantnější.



Obrázek 4.8: Testování prostřednictvím Postman

Pro testování vlastního REST rozhraní se nabízí nástroj Postman, základní popis práce s ním je popsán například v seriálu [30]. Nástroj umožňuje ukládání HTTP dotazů do kolekcí a poté je lze hromadně spouštět. Umí vytvořit vzory volání pro řadu běžných prostředí (shell, CURL, Java, Javascript, a mnohé další). Zvládne vygenerovat též dokumentaci testů, což bývá podstatné pro protokolární technické převzetí výsledků zakázkového vývoje.

Dokumentaci PL/SQL kódu lze řešit běžným formátem javadoc, neboť existuje kompilátor [31] pod *GNU LGPL* licencí, který z adresáře se soubory obsahujícími definice Oracle struktur klasickou javadoc dokumentaci sestaví. Vzhledem k tomu, že exportované definice lze za pomoci menšího skriptování pohodlně umístit do vhodné adresářové struktury obdobně jako je uspořán zdrojový kód jazyka Java, je práce s tímto dokumentačním nástrojem pohodlná, vše lze navíc triviálně umístit například do GiT repozitáře.

Dokumentace je v příloze (CD disk) a sestává z:

- dokumentace REST rozhraní; adresace zdrojů – struktura URI, popis možných volání, příklady dotazů a reakcí,
- dokumentace webové aplikace,
- vygenerovaná dokumentace pro Oracle prostředí.

Závěr

Cíle práce byly splněny. Na zadání softwarového díla byly aplikovány postupy softwarového inženýrství, provedena analýza požadavků, navržen vhodný verzujícího systému správy číselníků. Po výběru vhodných technologií byl navržený model ve své prvé verzi implementován, v cílovém prostředí také nasazen a otestován. Na reálném prostředí byla také doplněna dokumentace o reálné ukázky obrazovek.

Během práce jsem se detailně seznámil s objektovým prostředím Oracle. Ačkoliv je v rozsahu této práce bylo možno použít jako ekvivalent objektů prostředí Java, neposkytuje pracovní komfort. Na úrovni Oracle prostředí by pomohlo odlišení transientních objektů, čistě programových, od persistentních. Aktuálně jsou rovnocenné, Oracle pak neumožní některé operace – překlady předků, volání konstruktoru předka. Konstruktory je nutno zapisovat ve všech objektech zvlášť, nedědí se.

Na druhou stranu objektové prostředí umožňuje polymorfismus a zapouzdření dat na stejné fyzické úrovni, ve stejním procesu, na které pracují SQL dotazy. Implicitně je tak zajištěna konsistence i efektivita. Nástroje na serializaci XML (nativní) i JSON (mnou napsané) jsou nve verzi 11 dostatečné. Verze 12 má již nativní podporu i pro JSON.

Jsem spokojen s implementací verzovacího mechanismu, který nemusí při změně jednoho atributu naklonovat celý číselníkový záznam. Seznámil jsem se více s technologiemi Bootstrap, XHR, jQuery, AJAX a Google Sign-In, které běžně nepoužívám. Spolu s komponentou FooTable se mi povedlo realizovat sice jednoduchou, ale pro řešenou problematiku funkční a dostatečnou klientskou aplikaci.

V krátkodobém horizontu bude v rámci rozvoje komponenty potřeba pokrýt kompletní nezúžené zadání a doplnit specifický autorizační modul pro konkrétní cílové prostředí.

Ve střednědobém horizontu budu řešit napojení jednotek vybraných interních systémů na novou číselníkovou komponentu a provádět dohled nad jejich integrovaným provozem. Po jeho zhodnocení provedu se zadavatelem revizi původních požadavků a případné úpravy a rozšíření vytvořené komponenty.

V dlouhodobějším výhledu bych rád prosadil otevření dalšího typu rozhraní ve formátu RDF dle standardu Otevřená data [12], [9].

Literatura

[1] Úřad vlády České republiky. *Datové prvky a číselníky ve správě Úřadu vlády ČR* [online]. Vláda ČR, c2009-2019 [cit. 2019-06-14]. Dostupné z: https://www.vlada.cz/cz/urad-vlady/datove-prvky-a-ciselniky-ve-sprave-uradu-vlady-cr-117680/

[2] KNESL, Jiří. Agilní vývoj – Úvod. V: *zdroják.cz* [online]. 11. prosince 2009 [cit 2019-06-17]. Dostupné z: https://www.zdrojak.cz/clanky/agilni-vyvoj-uvod/

[3] COCBURN, Alistair. *Use Cases: Jak efektivně modelovat aplikace*. 2. vydání. Brno: CP Books, a.s., 2005. ISBN 80-251-0721-3.

[4] ARLOW, Jim a NEUSTADT, Ila. *UML 2 a unifikovaný proces vývoje aplikací*. Brno: Computer Press, 2007. Část 2-5. ISBN: 978-80-251-1503.

[5] Aplikace pro správu číselníků. V: *Centrum pro rozvoj technologické platformy registrů Národního zdravotnického informačního systému, modernizace vytěžování jejich obsahu a rozšíření jejich informační kapacity*. [cit. 2019-06-25]. Dostupné z: https://https://rozvojnzis.uzis.cz/index.php?pg=vystupy-projektu--podpurne-aplikace

[6] Číselníky. V: *Celní správa České republiky* [online]. [cit. 201-06-26]. Dostupné z: https://www.celnisprava.cz/cz/aplikace/Stranky/ciselniky.aspx

[7] *Oracle Master Data Management* [online]. Oracle, [2013] [cit. 2019-03-09]. Dostupné z: http://www.oracle.com/us/products/applications/master-data-management/overview/index.html

[8] WORLD WIDE WEB CONSORCIUM (W3C). SPARQL Query Language for RDF [online]. Poslední změna 26. března 2013 [cit. 2019-06-20]. Dostupné z: https://www.w3.org/TR/rdf-sparql-query/

[9] WORLD WIDE WEB CONSORCIUM (W3C). Resource Description Framework (RDF). V: *W3C Semantic Web* [online]. Poslední změna 15. března 2014 21:35 [cit. 2019-06-17]. Dostupné z: https://www.w3.org/RDF/

[10] WORLD WIDE WEB CONSORCIUM (W3C). Vocabularies. V: *W3C Semantic Web* [online]. c2015 [cit. 2019-06-20]. Dostupné z: https://www.w3.org/standards/semanticweb/ontology.html

[11] *Linked Open Vocabularies* [online]. [cit. 2019-06-20]. Dostupné z: https://lov.linkeddata.es/

[12] MINISTERSTVO VNITRA ČR. Jak na číselníky. V: *Otevřená data* [online]. Poslední změna 26. června 2018 00:00. [cit. 2019-06-17]. Dostupné z: https://opendata.gov.cz/draft:%C4%8D%C3%ADseln%C3%ADky

[13] MINISTRERSTVO FINANCÍ ČR. Generální finanční ředitelství. *Open Data CEDR III* [online]. [cit. 2018-12-21]. Dostupné z: http://cedr.mfcr.cz/cedr3internetv419/OpenData/DocumentationPage.aspx#zalozka1

[14] *EU Open Data Portal* [online]. [cit. 2018-12-21]. ISSN 2315-3091. Dostupné z: https://data.europa.eu/euodp/

[15] Database Object-Relational Developer's Guide. V: *Oracle Database Online Documentation* [online]. Oracle, c2019 [cit. 2019-06-14]. Dostupné z: https://docs.oracle.com/cd/B28359_01/appdev.111/b28371/adobjint.htm#ADOBJ001

[16] *Bootstrap* [online]. [cit. 2019-06-20]. Dostupné z: https://getbootstrap.com/

[17] *FooTable: A responsive table plugin built on jQuery and made for Bootstrap* [onlilne]. c2016 [cit. 2019-05-15]. Dostupné z: https://fooplugins.github.io/FooTable/

[18] Representational state transfer. V: *Wikipedia, The Free Encyclopedia* [online]. Poslední změna 8. června 2019 [cit. 2019-06-14]. Dostupné z: https://en.wikipedia.org/wiki/Representational_state_transfer

[19] FIELDING, Roy Thomas. *Architectural Styles and the Design of Network-based Software Architectures* [online]. University of Califormia Irvine, 2000 [cit. 2019-06-14]. Dostupné z: https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf

[20] FREDRICH, Todd. *RESTful Service Best Practices: Recommendations for Creating Web Services* [online]. Pearson eCollege, 2. srpna 2013 [cit. 2019-06-20]. Dostupné z: www.RestApiTutorial.com

[21] KOSEK, Jiří. *XML eXtensible Markup Language* [online]. 1999, poslední změna 28. května 1999 [cit. 2019-06-18]. Dostupné z: https://www.kosek.cz/clanky/xml/xml-uvod.html

[22] Introducing JSON [online]. [cit. 201-06-14]. Dostupné z: https://www.json.org/

[23] BILLINGTON, Adrian. Type enhancements in oracle 9i. V: *oracle-developer.net* [online]. Prosinec 2005. Aktualizace červenec 2007 [cit.2019-03-15]. Dostupné z: http://www.oracle-developer.net/display.php?id=219

[24] *Oracle: Oracle Database PL/SQL* [online]. [cit. 2019-06-14]. Dostupné z: https://www.oracle.com/technetwork/database/features/plsql/index.html

[25] LOKHANDE, Bhushan. SSL Labs Grading 2018. V: *Qualys Community* [online]. Verze 3. 15. ledna 2018, poslední změna 28.2.2018 [cit. 2019-06-17]. Dostupné z: https://community.qualys.com/docs/DOC-6321-ssl-labs-grading-2018

[26] VOREL, David. Bezpečnostní monitoring – SIEM. V: *CZ.NIC – Konference Internet a Technologie 14* [online]. [Květen 2014] [cit. 2019-06-14]. Dostupné z: https://www.nic.cz/public_media/IT14/prezentace/David_Vorel.pdf

[27] Reverse Proxy Guide. V: *Apache HTTP Server Project* [online]. The Apache Software Foundation, c2019. [cit. 2019-06-17]. Dostupné z: https://httpd.apache.org/docs/2.4/howto/reverse_proxy.html

[28] How To Configure VLANs On the Catalyst Switches. V: *Cisco Community* [online]. 22. června 2009, poslední změna 3. ledna 2019 [cit. 2019-06-20]. Dostupné z: https://www.cisco.com/c/en/us/support/docs/lan-switching/vlan/10023-3.html

[29] Google Sign-In V: *Google Identity Platform* [online]. [cit. 2019-05-28]. Dostupné z: https://firebase.google.com/docs/auth/web/google-signin

[30] KUTÁČ, Pavel. *Postman, 1. část* [online]. Poslední změna 27. září 2018 [cit. 2019-06-25]. Dostupné z: https://www.kutac.cz/blog/weby-a-vse-okolo/postman-1-cast/

[31] *PLDoc Project* [online]. Poslední změna 23. listopadu 2016 [cit. 2019-06-25]. Dostupné z: http://pldoc.sourceforge.net/maven-site/

[32] Grafické uživatelské rozhraní. V: *Wikipedia, The Free Encyclopedia* [online]. [cit. 2019-06-14]. Dostupné z: https://cs.wikipedia.org/wiki/Grafick%C3%A9_u%C5%BEivatelsk%C3%A9_rozhran%C3%AD

[33] KUCHAŘ, Jaroslav: MI-W20 Web 2.0. V: *FIT CTU COURSE PAGES* [online]. [2019], poslední změna 14. února 2019 [cit. 2019-06-19]. Dostupné z: https://courses.fit.cvut.cz/MI-W20/lectures/index.html

[34] WORLD WIDE WEB CONSORCIUM (W3C). *Cascading Style Sheets home page* [online]. c1994-2019, poslední změna 27. června 2019 [cit. 2019-06-27]. Dostupné z: https://www.w3.org/Style/CSS/

[35] Refsnes Data. *HTML: The language for building web pages* [online]. c1999-2019 [cit. 2019-06-20]. Dostupné z: https://www.w3schools.com/

[36] element61 N.V. – Moore Stephens. *Master Data Management (MDM): Architecture & Technology* [online]. [cit. 2019-06-14]. Dostupné z: https://www.element61.be/en/resource/master-data-management-mdm-architecture-technology

Příloha A

Seznam použitých zkratek

GUI Graphical user interface. uživatelské rozhraní, umožňující ovládat počítač či software prostřednictvím grafických ovládacích prvků [32].

CORS Cross-origin Resource Sharing Protocol. Protokol implementovaný pomocí hlaviček HTTP protokolu, umožňující cross-site dotazy v prostředí aktivní bezpečnostní politiky Same Origin Policy. [33].

CRM Customer relationship management, řízení vztahů se zákazníky.

CSS Cascading Style Sheets. Jednoduchý standard, jak ovlivnit vzhled, tedy písma, barvy, umístění prvků na webových stránkách [34].

JSON JavaScript Object Notation. Flexibilní a jednoduchý tvar serializace dat, užitečný při komunikaci systémů. Master Data Management. Též EDM – Enterprise data management. Obecné označení procesů a nástrojů pro údžbu dat, směřující k jejich ověření, odstranění redundancí, nepřesností a celkové kultivaci pro použití v návazných procesech (viz např. [22]).

HTTP Hypertext Transfer Protocol. Aplikační protokol pro distribuované sdílené multimediálních informací, základní protokol služby World Wide Web.

HTML Hypertext Markup Language. Značkovací jazyk pro tvorbu webových stránek, propojených hypertextovými odkazy [35].

MDM Master Data Management. Též EDM – Enterprise data management. Obecné označení procesů a nástrojů pro údžbu dat, směřující k jejich ověření, odstranění redundancí, nepřesností a celkové kultivaci pro použití v návazných procesech (viz např. [36]).

ORM Relational Database Management System. Softwarový systém spravující data uložená ve formě relační databáze, například Oracle.

REST Representational State Transfer. Architektura rozhraní, navržená pro distribuovaná prostředí [18], [19].

RDBMS Relational Database Management System. Softwarový systém spravující data uložená ve formě relační databáze, například Oracle.

PL/SQL Procedural Language/Structured Query Language. Procedurální nadstavba jazyka SQL firmy Oracle, založená na programovacím jazyku Ada, převzatá též do jiných databázových prostředí, v nichž se samostaně vyvíjela (Transact-SQL, PL/pgSQL) [24].

XML Extensible markup language. Značkovací jazyk standardizovaný konsorciem W3C, který se vyvinul z obecnějšího jazyka SGML a používá se jako forma serializace dat při jejich výměně mezi různými systémy. Výhodou je jeho čitelnost, jedná se o obyčený textový soubor s pevně popsanou strukturou [21].

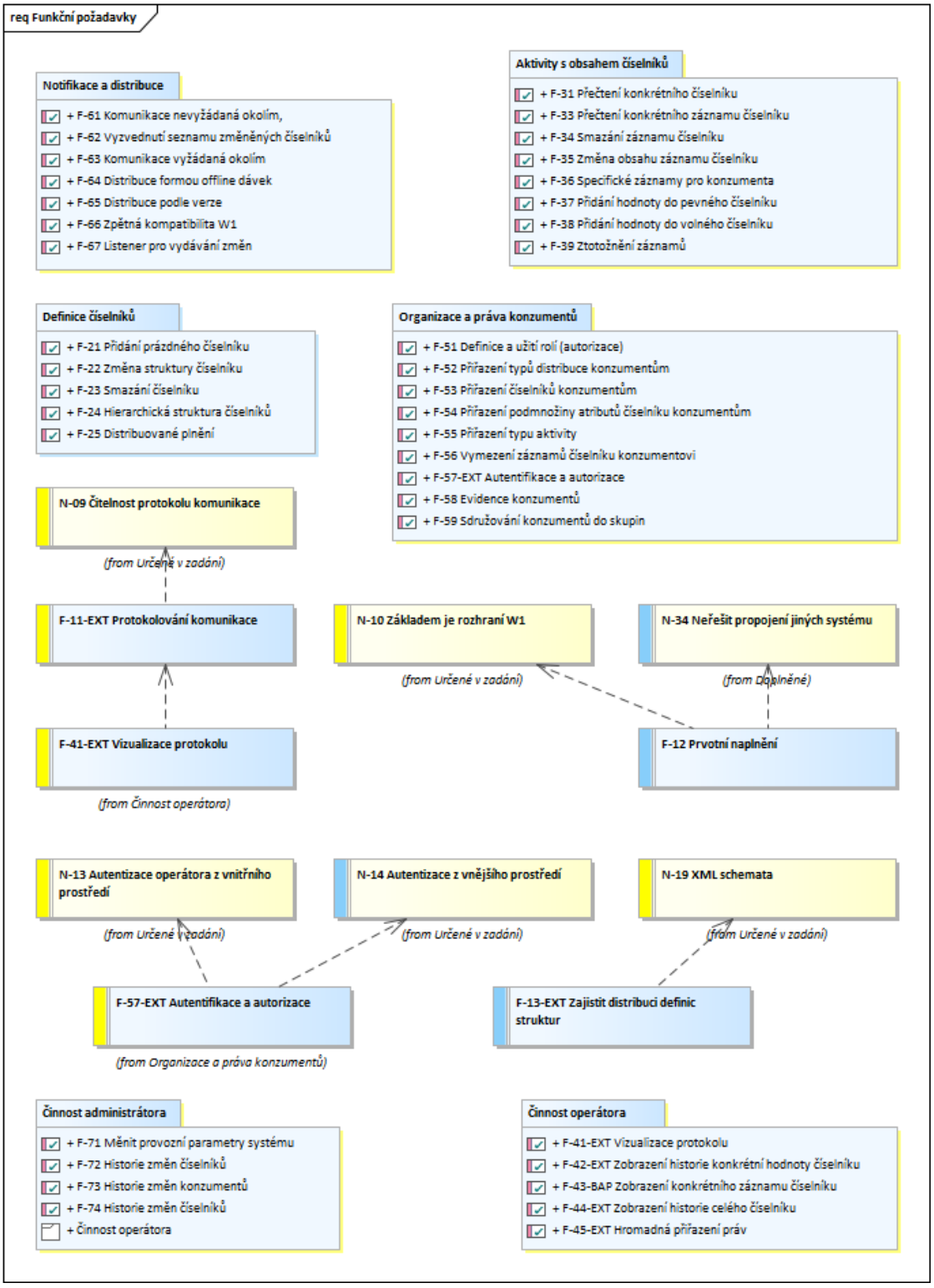
Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `ora` ..... zdrojové skripty pro Oracle
    - `doc` ..... vygenerovaná dokumentace
  - `http` ..... tenký klient - PHP skript a konfigurace
    - `doc` ..... popis konfigurace
  - `webapp` ..... webová aplikace - HTML/CSS/JS
    - `doc` ..... dokumentace
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX

Příloha C

Přílohy



Obrázek C.1: Hlavní členění funkčních požadavků

```
{
    "Zaznam": [
    {
        "Kriterium": "Objem",
        "DruhPalivaKod": "BASM",
        "DruhPalivaTxt": "Benzín"
    },
    {
        "Kriterium": "Objem",
        "DruhPalivaKod": "BA",
        "DruhPalivaTxt": "Benzín"
    },
    {
        "Kriterium": "Objem",
        "DruhPalivaKod": "BIONM",
        "DruhPalivaTxt": "Motorová nafta s podílem rostlinné složky"
    },
    ...
    {
        "Kriterium": "Vykon",
        "DruhPalivaKod": "EL",
        "DruhPalivaTxt": "Elektřina"
    },
    {
        "Kriterium": "Objem",
        "DruhPalivaKod": "NM",
        "DruhPalivaTxt": "Motorová nafta"
    }
    ],
    "Kod": "DruhPalivaPGF",
    "Nazev": "Druh paliva",
    "Popis": "Druh paliva (pro účely §4, zák.168/1999 Sb.)"
}
```

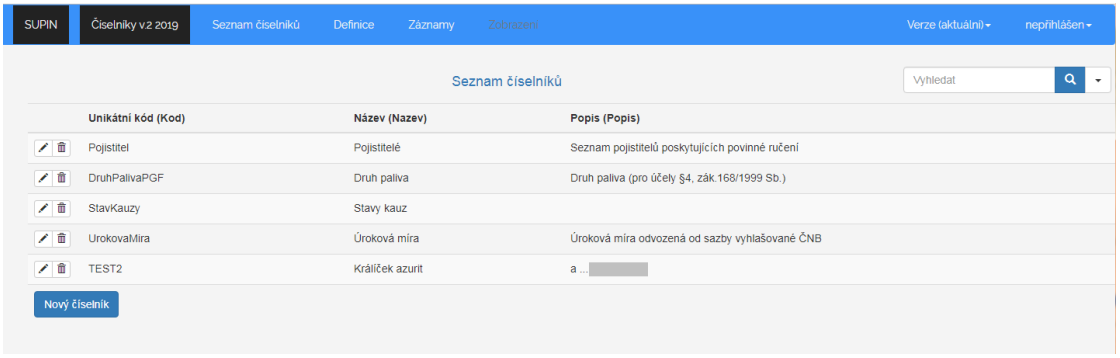
Výpis kódu C.1: Přečtený číselník v uživatelském tvaru, JSON serializace

```
<?xml version="1.0" encoding="utf-8" ?>
<Ciselnik xmlns="http://ws.ckp.cz/up/a9/2019/00.009" Kod="DruhPalivaPGF" Nazev="Druh pali
  <Popis>Druh paliva (pro účely §4, zák.168/1999 Sb.)</Popis>
  <Zaznam DruhPalivaKod="BASM" DruhPalivaTxt="Benzín" Kriterium="Objem"/>
  <Zaznam DruhPalivaKod="BA" DruhPalivaTxt="Benzín" Kriterium="Objem"/>
  ...
  <Zaznam DruhPalivaKod="VODIK" DruhPalivaTxt="Vodík (zkapalněný)" Kriterium="Objem"/>
  <Zaznam DruhPalivaKod="NM" DruhPalivaTxt="Motorová nafta" Kriterium="Objem"/>
</Ciselnik>
```

Výpis kódu C.2: Přečtený číselník v uživatelském tvaru, XML serializace

```xml
<?xml version="1.0" encoding="utf-8" ?>
<DefCiselnik xmlns="http://ws.ckp.cz/up/a9/2019/00.009">
  <Ciselnik Kod="DruhPalivaPGF" Nazev="Druh paliva">
    <Popis>Druh paliva (pro účely §4, zák.168/1999 Sb.)</Popis>
  </Ciselnik>
  <Atributy>
    <DefAtribut Kod="_SID" HodnotaTyp="StringKod" Nazev="SID" IsKlicAN="A" Serializace="I"
                ElementKod="_SID" KontrolniRegExp="^\d+$" Implicitni="-1" PoradiUI="0"
                TitleUI="SID" FlagsUI="hide">
               <Popis>SID</Popis>
    </DefAtribut>
    <DefAtribut Kod="Kod" HodnotaTyp="StringKod" Nazev="Unikátní kód" IsKlicAN="A"
                Serializace="I" ElementKod="DruhPalivaKod"
                KontrolniRegExp="^#|[A-Z]\w{1,9}$" Delka="10" PoradiUI="1">
               <Popis>Alfanumerický unikátní kód atributu v rámci číselníku</Popis>
    </DefAtribut>
    <DefAtribut Kod="DruhPalivaTxt" HodnotaTyp="StringNazev" Nazev="Druh paliva" IsKlicAN="N"
                Serializace="I" ElementKod="DruhPalivaTxt"
                KontrolniRegExp="^([A-Z].{0,499})?$i" Delka="150" PoradiUI="2">
               <Popis>Druh paliva</Popis>
    </DefAtribut>
    <DefAtribut Kod="Kriterium" HodnotaTyp="VyberMoznosti" Nazev="Kritérium posuzování"
                IsKlicAN="N" Serializace="I" ElementKod="Kriterium"
                KontrolniRegExp="^([A-Z]\w{0,49})?$i" Delka="25" PoradiUI="3">
               <Popis>Kritérium, které se uplatňuje při určení druhu vozidla. Je to buď
               zdvihový objem, anebo celková hmotnost (určeno vyhláškou).</Popis>
    </DefAtribut>
  </Atributy>
</DefCiselnik>
```

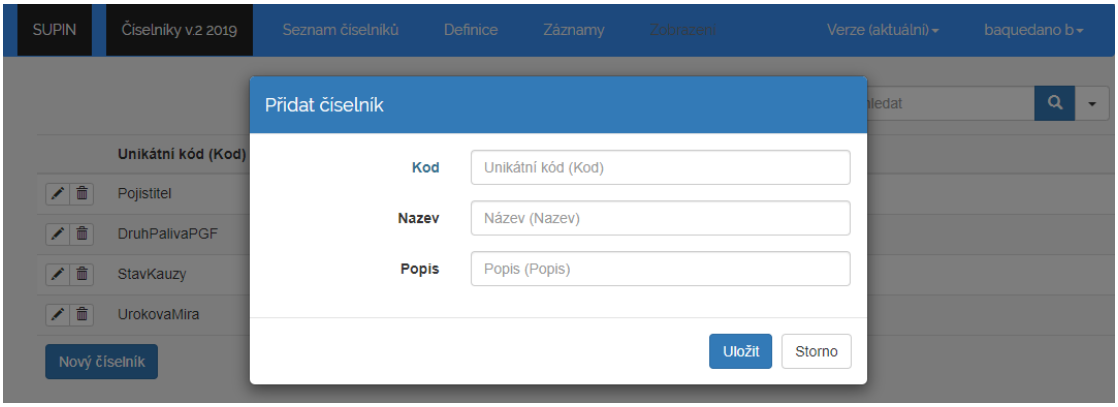
Výpis kódu C.3: Přečtená definice číselníku v XML serializaci



Obrázek C.2: Výpis seznamu číselníků

```xml
<?xml version="1.0" encoding="utf-8" ?>
<DefCiselnik xmlns="http://ws.ckp.cz/up/a9/2019/00.009">
 <Ciselnik Kod="DruhPalivaPGF" Nazev="Druh paliva">
  <Popis>Druh paliva (pro účely §4, zák.168/1999 Sb.)</Popis>
 </Ciselnik>
 <Obsah>
  <DefZaznam Kod="BASM" DruhPalivaTxt="Benzín" Kriterium="Objem"/>
  <DefZaznam Kod="BA" DruhPalivaTxt="Benzín" Kriterium="Objem"/>
  ...
  <DefZaznam Kod="EL+BA" DruhPalivaTxt="Benzínový hybrid" Kriterium="Vykon"/>
  <DefZaznam Kod="EL+NM" DruhPalivaTxt="Naftový hybrid" Kriterium="Vykon"/>
  <DefZaznam Kod="LNG" DruhPalivaTxt="Zkapalněný zemní plyn" Kriterium="Objem"/>
  <DefZaznam Kod="METAN" DruhPalivaTxt="BIO metan" Kriterium="Objem"/>
  <DefZaznam Kod="NG" DruhPalivaTxt="Stlačený zemní plyn" Kriterium="Objem"/>
  <DefZaznam Kod="VODIK" DruhPalivaTxt="Vodík (zkapalněný)" Kriterium="Objem"/>
  <DefZaznam Kod="NM" DruhPalivaTxt="Motorová nafta" Kriterium="Objem"/>
 </Obsah>
</DefCiselnik>
```

Výpis kódu C.4: Přečtený číselník v definičním tvaru pro editaci v aplikaci



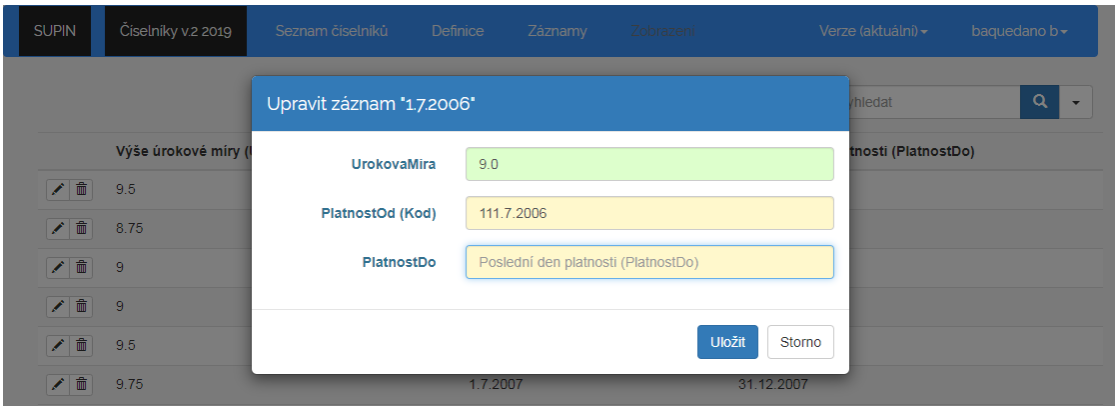
Obrázek C.3: Přidání číselníku

```
create or replace type VRT_PersistentItem under VRT_Item
(
/**
* Systém:       CC2
* Modul:        VR
* Popis:        Abstrakt pro entity (samostatné, rozhodované i primo zapisovane)
* Vytvoreno:    10.5.2019
* Autor:        Premysl Dedic
* Vztah k DB:   Abstrakt, pracuje s DB
*
* Modifikace:   <br/>
*    10.5.2019  vznik
*
* Použití:      <br/>
* Prototyp, pouzity pri konstrukci skeletu domenovych objektu. Realne typy objektu,
* které slouzi jako stavebni pruky projektu, si musí doimplementovat vhodneho potomka.
*/

constructor function VRT_PersistentItem  return self as result,

/**
* Vrací true, pokud je vyplnenost objekt dostatecna na to, aby jej bylo mozno nahrat z databaze.
* Implicitne vraci true, pokud je ID objektu > 0, ale ceka se redefinic napriklad bude-li pouzit
* jiny nez technicky primarni klic. Je na potomcich, aby si metodu pripadne smysluplneji
* predefinovaly a upravily si isLoadable() a loadItem().
* <pre>Visibility: public</pre>
*/
member function  isLoadable        return boolean,

/**
* Vrací true, pokud je objekt persistentni s databazi. Implicitne vraci true, pokud je
* ID objektu > 0, tedy predpoklada se, ze ulozeno. Je na potomcich, aby si metodu pripadne
* smysluplneji predefinovaly.
* <pre>Visibility: public</pre>
*/
member function  isSaved           return boolean,

/**
* Provede veskere konverze z puvodnich dat na data zapisovana, pripadne kontroly business logiky atd.
* Low level/hardcoded veci nemusi - treba ukladani do smetist se dela az pri zapisu a pripadne
* modifikace odtud pramenici (triggery) se do objektu jeste promitnou az v ramci zapisu. Spravne by
* mel persistent objekt umet absorbovat string udaje i pro datum, etc. a konverze by probihala pri
* kontrole. Nicmene pak by byl problematicky pouzitelny, takze rozumnou miru ponechavam na programatorovi.
* <pre>Visibility: public</pre>
*/
member procedure zkontroluj,

/**
* Deserializace PRIMO UKLADA objekt. Predpoklada zakladni torzo objektu, ktere neni uplne zadefinovano
* a lisi se po skupinach/vrstvach objektu (tvrzeni, a ostatni) a ma za ukol jej dovyplnit ze
* serializovanych dat a pote zapersistentnit. Uklada se primo, protoze zaserializovana je obvykle
* nejaka hierarchie, kvuli ktere musi byt prvek vyse ulozen, jinak nebude existovat vazba. Muze se to,
* pochopitelne, cela na konci abortnout. Nutno redefinovat, default hlasi vyjimku - neznama serializace.
* <pre>Visibility: public</pre>
*/
member procedure storeFrom_JSON(pClobData in varchar2, pDatumCasCiselnik in date default null),
member procedure storeFrom_XML (pXmlData  in xmlType,  pDatumCasCiselnik in date default null),

...

) not instantiable not final
```

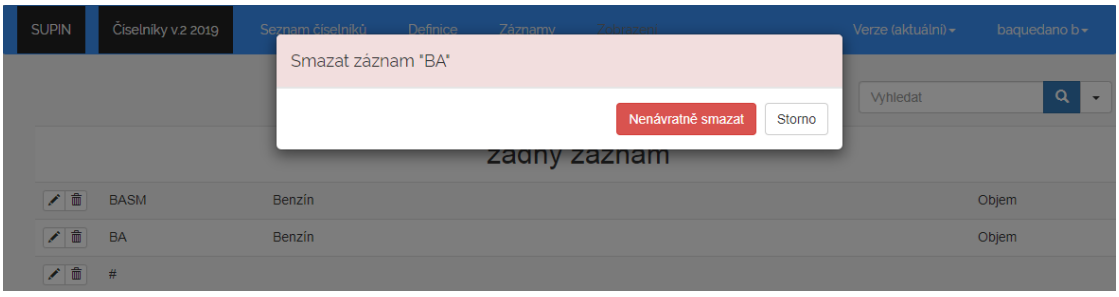
Výpis kódu C.5: Ukázka definice objektového typu v Oracle prostředí

```
create or replace procedure TST_Atribut_01( pIsDebug in integer default null )
is
CA_IS_DEBUG               integer := 1 ;
xCiselnikID               integer ;
xAtributID                integer ;
xZaznamID                 integer ;
m                         integer ;
xSidOpen                  integer := 1000000 ;
xSidClose                 integer := 999999999999 ;
begin
--
delete from CCA_Ciselnik c
where c.kod       = 'TEST'
;
m := DBG_Commit( pIsDebug => nvl(pIsDebug,CA_IS_DEBUG) ) ;
--
insert into CCA_Ciselnik (Sidopen, Sidclose, Kod, Nazev, Popis)
values ( xSidOpen, xSidClose, 'TEST', 'Testovaci', 'Testovací číselník')
returning zaznamID into xCiselnikID
;
m := DBG_Commit( pIsDebug => nvl(pIsDebug,CA_IS_DEBUG) ) ;
--
insert into CCA_Atribut  (Sidopen, Sidclose, CiselnikID, Isklican, Kod, Hodnotatyp, Nazev)
values ( xSidOpen, xSidClose, xCiselnikID, 'A', 'AKlic', 'StringKod', 'Testovací atributíček' )
returning zaznamID into xAtributID
;
m := DBG_Commit( pIsDebug => nvl(pIsDebug,CA_IS_DEBUG) ) ;
--
insert into CCA_HSTRINGKOD ( Sidopen, Sidclose, ATRIBUTID, HODNOTA )
values ( xSidOpen, xSidClose, xAtributID, 'Klic' )
returning zaznamID into xZaznamID
;
m := DBG_Commit( pIsDebug => nvl(pIsDebug,CA_IS_DEBUG) ) ;
--
insert into CCA_Atribut  (Sidopen, Sidclose, CiselnikID, Isklican, Kod, Hodnotatyp, Nazev)
values ( xSidOpen, xSidClose, xCiselnikID, 'N', 'ABubak', 'StringKod', 'Bubákova nálada' )
returning zaznamID into xAtributID
;
m := DBG_Commit( pIsDebug => nvl(pIsDebug,CA_IS_DEBUG) ) ;
--
insert into CCA_HSTRINGKOD ( Sidopen, Sidclose, ZAZNAMID, ATRIBUTID, HODNOTA )
values ( xSidOpen, xSidClose, xZaznamID, xAtributID, 'Bubakovská' )
;
m := DBG_Commit( pIsDebug => nvl(pIsDebug,CA_IS_DEBUG) ) ;
--
-- commit ;
--
end ;
```

Výpis kódu C.6: Ukázka skriptu pro testování během vývoje v Oracle



Obrázek C.4: Přidání záznamu – kontrola



Obrázek C.5: Smazání záznamu číselníku