Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Systém pro automatickou kontrolu samostatných prací vytvořených v MS Access

Plzeň, 2018 Vojtěch Kinkor

Zadání

1) Seznamte se s formátem souboru MS Access .accdb a možnostmi jeho čtení.
2) Seznamte se validátorem na portálu ZČU.
3) Vytvořte konfigurovatelný systém pro kontrolu samostatných prací vytvořených v MS Access se zaměřením na splnění zadání a plagiarismus.
4) Část systému pro kontrolu splnění zadání adaptujte pro validátor.
5) Celý systém pečlivě otestujte.

< tato strana bude nahrazena originálním zadáním / kopií ! >

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 17. 5. 2018

Vojtěch Kinkor

Abstract

System for Automatic Checking of Student Works Created in MS Access

This thesis describes a design and implementation of a system for automatic checking of database files created in Microsoft Access. The goal is to automatically check structure and content of databases against rules specified by a user. In addition, the system is also able to detect plagiarism among databases. The presented solution consists of an application with a graphical user interface and a tool adapted for use in cooperation with a student works validator, which is part of ZČU portal. This system has the ambition to reduce an amount of time needed for manual checking of fulfilment of students' works objectives.

Abstrakt

Systém pro automatickou kontrolu samostatných prací vytvořených v MS Access

Cílem této práce je navrhnout a implementovat systém, který umožní automatické kontrolování databázových souborů vytvořených v aplikaci Microsoft Access. Podstatou kontroly je ověření existence a struktury objektů uložených v databázových souborech dle uživatelem zadaných kritérií, další částí pak je detekce plagiarismu na základě podobnosti databází. Vytvořené řešení se skládá z aplikace s grafickým rozhraním a části adaptované pro použití v rámci validátoru studentských prací na portálu ZČU. Výsledkem této práce je systém schopný automaticky vyhodnocovat samostatné práce s ohledem na splnění zadání a vyučujícím tak usnadnit jejich kontrolu.

Obsah

1 Úvod 1

2 Databázový software Microsoft Access 2

2.1 Základní informace 2

2.2 Objekty uložené v databázi 2

2.2.1 Tabulky 2

2.2.2 Dotazy 5

2.2.3 Formuláře 6

2.2.4 Sestavy 7

2.2.5 Skryté systémové tabulky 7

2.3 Metadata databázových souborů 7

2.4 Formát ACCDB a možnosti jeho čtení 8

2.4.1 ODBC 9

2.4.2 Microsoft Office Interoperability 9

2.4.3 MDB Tools 10

2.4.4 MDB Tools Java 10

2.4.5 Jackcess 11

2.4.6 JDBC 11

3 Portál ZČU 13

3.1 Základní informace 13

3.2 Validátor studentských prací 13

3.2.1 Validační servery 15

3.2.2 Validační domény 15

3.2.3 Princip validace práce 17

4 Analýza řešení kontroly prací 19

4.1 Požadavky na řešení 19

4.2 Případy užití 20

4.3 Metoda čtení databázových souborů 21

4.3.1 Kritéria pro výběr metody čtení 21

4.3.2 Výběr metody čtení 22

4.4 Validace databázových souborů 22

4.4.1 Definice základních validačních pravidel 23

4.4.2 Pravidla existence a počtu výskytů 24

4.5 Vyhodnocení plagiarismu ... 24
4.6 Grafické uživatelské rozhraní ... 25
4.7 Adaptace pro validátor studentských prací ... 26
4.8 Návrh struktury systému ... 26

5 Implementace systému pro automatickou kontrolu prací ... 28
5.1 Použité technologie ... 28
5.2 Struktura implementovaného systému ... 28
5.2.1 Modul validator ... 29
5.2.2 Modul configurator ... 31
5.3 Čtení databázových souborů ... 33
5.4 Validace databáze ... 35
5.4.1 Implementovaná validační pravidla ... 36
5.4.2 Ukládání nakonfigurovaných pravidel do souborů ... 37
5.5 Hledání podobností a detekce plagiarismu ... 38
5.6 Grafické uživatelské rozhraní ... 40
5.7 Adaptace pro validátor portálu ZČU ... 41
5.7.1 Konfigurace validátoru ... 43

6 Testování vytvořeného systému ... 45
6.1 Čtení databázových souborů ... 45
6.2 Validace a validační pravidla ... 46
6.3 Detekce plagiarismu ... 46
6.4 Grafické uživatelské rozhraní ... 47
6.5 Aplikace adaptovaná pro validátor portálu ZČU ... 48
6.6 Testování systému na množině reálných dat ... 48

7 Závěr ... 50

Reference ... 51

Přílohy ... 54
A Uživatelská příručka ... 54
Spuštění a kompilace nástroje ... 54
Obsluha nástroje ... 55
Použití systému v rámci validátoru studentských prací ... 56
B Obsah přiloženého média ... 58

1 Úvod

V rámci výuky práce s aplikací Microsoft Access na Západočeské univerzitě odevzdávají studenti samostatné práce prostřednictvím speciálního portletu na portálu ZČU. Vyučující pak musí všechny tyto práce ručně zkontrolovat, což bývá časově velmi náročné. Navíc v případě, že práce nesplňuje požadavky dané zadáním, vrací ji studentovi k opravě. Pro studenty to znamená, že odevzdáním začíná čekání na informaci, zda je jejich práce vyhovující.

Portlet pro odevzdávání samostatných prací již v současné době může být napojen na validátor studentských prací, který zajišťuje automatickou kontrolu odevzdaných souborů dle pravidel nastavených vyučujícím. Student se po nahrání své práce okamžitě dozví, zda je či není vyhovující. Validátor byl původně vyvíjen pro potřeby předmětu Počítače a programování 1 (PPA1), od té doby byl ale rozšířen a použit i v několika dalších předmětech, ale doposud neumožňuje kontrolu souborů vytvořených v aplikaci Microsoft Access.

Cílem této práce je prozkoumat možnosti programového čtení souborů ve formátu ACCDB, tedy formátu používaného právě v aplikaci Microsoft Access. Na základě získaných poznatků bude implementován systém pro automatické kontrolování těchto souborů dle pravidel určených uživatelem. Smyslem je automaticky vyhodnocovat, zda studentské práce splňují zadání, a tedy usnadnit jejich hodnocení. Tato část systému bude dále adaptována pro použití v rámci validátoru portálu ZČU, díky čemuž se k vyučujícím dostanou pouze práce na určité úrovni kvality. Kromě zmíněné kontroly samostatných prací bude s ohledem na zadání v systému implementována detekce plagiarismu mezi pracemi.

2 Databázový software Microsoft Access

2.1 Základní informace

Microsoft Access je nástroj řadící se mezi takzvané systémy řízení báze dat (SŘBD či DBMS – database management system). Jedná se o software, který umožňuje práci s relačními databázemi. Je součástí kancelářského balíku Microsoft Office, případně prodáván i samostatně. Pro vytváření a správu databáze nabízí uživatelům přehledné grafické rozhraní [1].

Aplikace používá pro ukládání dat technologii Microsoft Jet Database Engine, v novějších verzích poté nazývanou Access Database Engine. Jednotlivé databáze jsou typicky uloženy v jediném souboru ve formátu ACCDB, nebo MDB [2].

2.2 Objekty uložené v databázi

V následujících podkapitolách jsou uvedeny různé objekty, které mohou být uloženy v databázích vytvořených aplikací Microsoft Access.

2.2.1 Tabulky

Tabulky jsou stěžejní součástí každé databáze. Lze je definovat jako strukturovanou kolekci dat. Skládá se ze sloupců a řádků (též záznamů) a v rámci databáze má unikátní název [1 str. AC 4].

Sloupce tabulky

Struktura tabulky je definována sloupci, které mají specifikovaný název a datový typ. Microsoft Access[1] podporuje následující datové typy [1 str. AC 57, 3]:

- *Automatické číslo* – pro každý nový záznam se automaticky nastaví na následující hodnotu posloupnosti, nebo na náhodné číslo (dle nastavení). Typicky se využívá pro sloupec označený jako primární klíč (viz dále).

[1] Aktuálně ve verzi 2016.

- *Číslo* – rozsah a typ (celočíselné/s desetinnou čárkou) lze zvolit ve vlastnostech sloupce.
- *Krátký text* (dříve Text) – text do délky 255 znaků.
- *Dlouhý text* (dříve Memo) – text do velikosti 1 GB.
- *Datum a čas*.
- *Měna* – číselného datového typu s fixní desetinnou čárkou (uchovává 4 desetinná místa).
- *Ano/ne* – uchovává hodnotu `-1` (Ano) nebo `0` (Ne); v rámci Microsoft Access zobrazeno jako zaškrtávací pole (*checkbox*).
- *Hypertextový odkaz*.
- *Objekt OLE* – umožňuje vložit speciální objekty, například obrázek, jiný dokument, či odkaz na soubor.
- *Příloha* – umožňuje vložit libovolný soubor jako součást záznamu. Jedná se o univerzálnější možnost k předchozímu.
- *Počítané* – automatické vypočítání hodnoty na základě zadaného vzorce.

Každému sloupci lze dále nastavit různé vlastnosti dle vybraného datového typu – typicky se jedná o ověřovací pravidla (validace vstupu od uživatele ještě před přidáním záznamu do databáze), výchozí hodnotu a dále nastavení zobrazení v tabulce (formátování, zarovnání, titulek po najetí myší, atp.).

Primární klíč

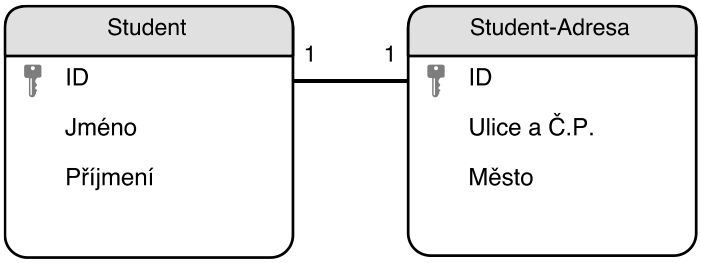
Tabulka může mít primární klíč – typicky se jedná o sloupec, jehož hodnoty jsou unikátní a vždy zadané (tzv. *not null*). V případě, že vytvoříme primární klíč pomocí více sloupců, nazýváme jej složeným primárním klíčem [1 str. AC A4].

Primární klíč slouží k jednoznačnému určení konkrétního záznamu v tabulce, čehož se využívá při vytváření dotazů nebo tvoření relačních vazeb mezi tabulkami. Pro vytváření primárních klíčů se obvykle používá datový typ Automatické číslo, který každému záznamu přiřadí unikátní celé číslo. Často bývá takový sloupec pojmenován „ID“ (*identification*) [1 str. AC A3–A4].

Relace mezi tabulkami a cizí klíče

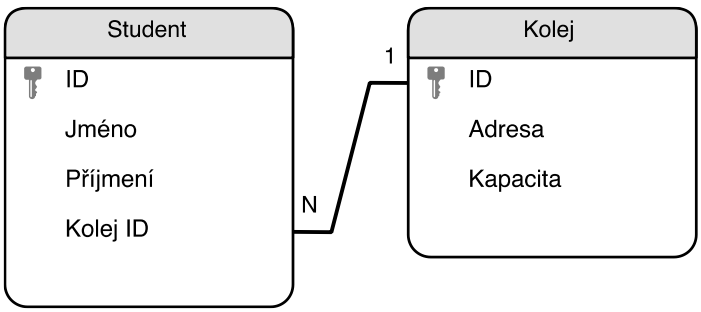
V případě, že chceme propojit více tabulek mezi sebou, využijeme tzv. relačních vazeb, zkráceně relací. Jedná se o situaci, kdy záznam v tabulce A odkazuje („má referenci“) na jeden konkrétní záznam v tabulce B. To lze obecně zajistit přidáním tzv. *cizího klíče* do tabulky A – sloupce, který bude obsahovat pouze hodnoty primárního klíče tabulky B (příp. skupiny sloupců, pokud se jedná o složený primární klíč). Rozlišují se tři typy relačních vazeb [1 str. AC A5–A8, 4 str. 419]:

- *Relace typu 1:1* – jednomu záznamu v tabulce A odpovídá žádný, či právě jeden záznam v tabulce B. Typicky je tato relace vhodná v situaci, kdy jen málo záznamů odkazuje do druhé tabulky. Pro cizí klíče zde platí, že jejich hodnoty jsou v rámci tabulky unikátní. Alternativně lze pro referencování použít primární klíče obou tabulek, viz obrázek 2.1 [4 str. 420].



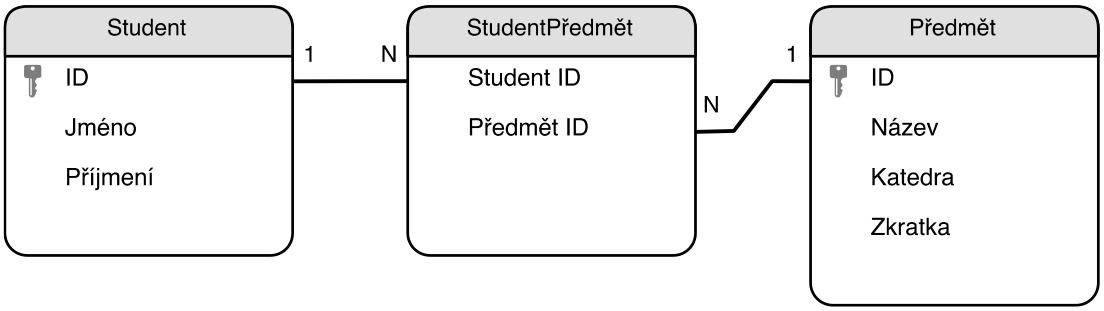
Obrázek 2.1 – model relace 1:1 využívající v obou tabulkách primární klíč.

- *Relace typu 1:N* – k více záznamům v tabulce A lze přiřadit jeden záznam z tabulky B. Tato vazba je vždy realizována pomocí již zmíněných cizích klíčů. Jedná se o nejčastěji využívanou vazbu [4 str. 421]. Příklad vazby 1:N vidíme na obrázku 2.2.



Obrázek 2.2 – model relace 1:N; cizí klíč *Kolej ID* v tabulce *Student* referencuje primární klíč tabulky *Kolej*.

- *Relace typu M:N* – k M záznamům v tabulce A lze přiřadit N záznamů z tabulky B. Relace se realizuje pomocí *spojové tabulky* (též rozkladové) a dvojice relací 1:N. Spojová tabulka obvykle obsahuje pouze sloupce cizích klíčů [4 str. 422]. Příklad je na obrázku 2.3.



Obrázek 2.3 – model relace M:N, která je tvořena dvěma relacemi 1:N na spojovou tabulku *StudentPředmět*.

Relace mezi tabulkami mohou zajišťovat *referenčním integritu*. Cílem je zabránit odkazování na neexistující záznam (a rovněž tedy vzniku osiřelých záznamů, na které byly všechny reference zrušeny). Integritní pravidlo může dále zajistit kaskádovou aktualizaci polí – pokud se změní hodnota primárního klíče, změní se automaticky hodnota u všech záznamů, které na záznam odkazují. Stejně tak může zajistit kaskádové odstranění souvisejících záznamů – v případě smazání záznamu budou smazány i všechny další, které na právě tento záznam odkazovaly [1 str. AC A11].

2.2.2 Dotazy

Dotazy slouží k získávání, přidávání, mazání či upravování dat v databázi. Microsoft Access umožňuje ukládání dotazů do databáze – lze tedy vytvořit dotazy pro usnadnění následné práce s daty [1 str. AC 124].

Dotazy mohou mít parametry, které lze využít např. pro filtrování záznamů v rámci tabulky nebo nastavení hodnoty při vkládání/upravování záznamů. Uživatel je pak při každém spuštění dotazu vyzván k zadání konkrétních hodnot parametrů prostřednictvím dialogových oken [1 str. AC 249].

Podporovány jsou následující druhy dotazů:

- *Výběrové (SELECT)* – jedná se o dotaz, jehož výsledkem je množina vybraných záznamů. Struktura je dána dotazem – jednotlivé sloupce mohou pocházet z různých tabulek, či být spočítané „za běhu“. Obecně lze považovat výběrový dotaz za analogii k databázovým pohledům [1 str. AC 124–128].
- *Vytvářecí (MAKE TABLE)* – pracuje na stejném principu jako výběrový, výsledek dotazu však není ihned zobrazen uživateli, ale uložen do nové tabulky [1 str. AC 500, 501].
- *Přidávací (APPEND)* – slouží pro vkládání nových záznamů do existujících tabulek [1 str. AC 500, 506].
- *Aktualizační (UPDATE)* – umožňuje úpravu hodnot již existujících záznamů v tabulkách [1 str. AC 500, 512].
- *Křížový (CROSSTAB)* – výsledkem dotazu je tzv. kontingenční tabulka zobrazující data v kompaktní podobě. Typicky se používá například pro sumarizaci hodnot, nalezení průměrů, maximálních hodnot, atp. [1 str. AC 254, 256].

2.2.3 Formuláře

Formuláře poskytují přívětivé rozhraní pro vkládání či editaci záznamů v tabulkách. Grafické rozhraní je plně konfigurovatelné a umožňuje tedy jednotlivá pole záznamů různě seskupovat, přidat popisky, či některá úplně skrýt. Formuláře jsou v databázi opět uloženy pod unikátním názvem [1 str. AC 179].

Microsoft Access umožňuje vytvořit formuláře různých druhů [1 str. AC 180–200]:

- Formuláře pro editaci jednotlivých záznamů (dále označované jako standardní).
- Formuláře zobrazující více položek (záznamů) najednou.
- Navigační formuláře, které poskytují možnost přepínání mezi různými formuláři a umožňují tak vytvořit komplexní rozhraní pro správu databáze.
- Datové listy, které vypadají podobně jako zobrazení tabulky (tedy tabulka, kde každý řádek odpovídá jednomu záznamu).

- Rozdělené formuláře, které jsou kombinaci standardních formulářů v jedné části a datového listu v druhé části obrazovky.
- Modální dialogová okna, která mají stejné možnosti jako standardní formuláře, ale zobrazují se v samostatném okně.

2.2.4 Sestavy

Sestavy slouží pro vytváření výpisů dat z databáze v přívětivé podobě, zobrazující typicky více záznamů na jedné straně, na rozdíl od formulářů ale neumožňuje editaci dat. Často se využívá pro následné vytisknutí. Při návrhu se definuje záhlaví a zápatí stránek a rozložení prvků pro každý záznam („řádek“ sestavy) [1 str. AC 208].

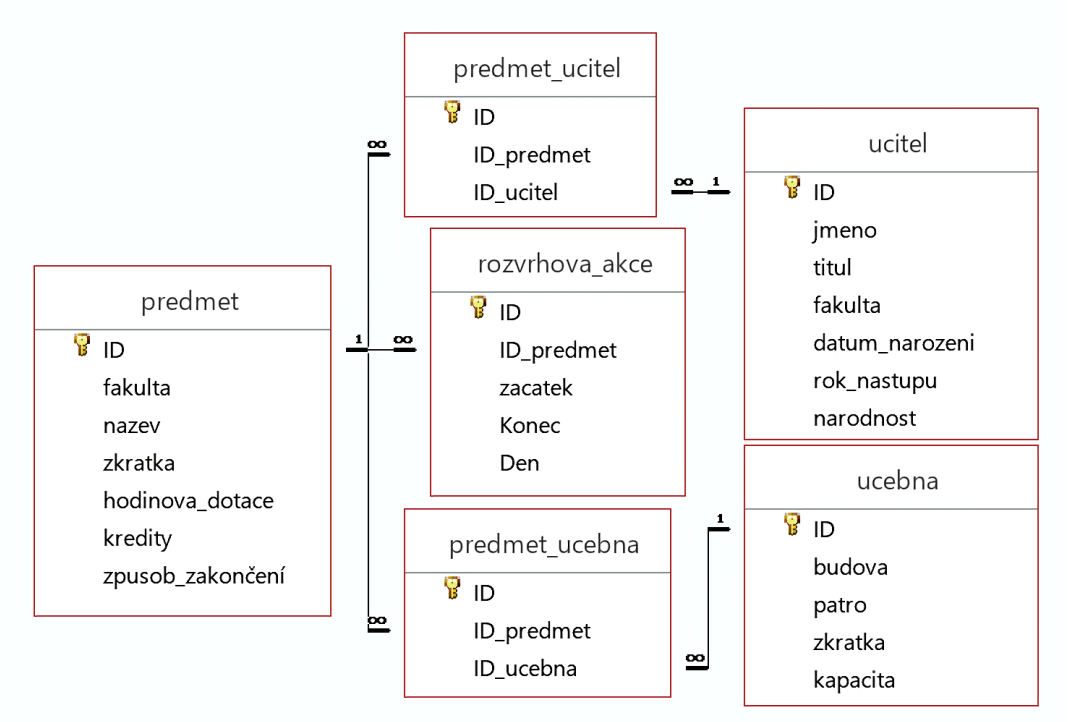
2.2.5 Skryté systémové tabulky

Databáze dále obsahují několik skrytých systémových tabulek, jejichž název vždy začíná prefixem `MSys`. Například v tabulce `MSysRelationships` jsou uloženy všechny relace mezi objekty (tabulkami a dotazy). Jednou ze zajímavějších tabulek je `MSysObjects` obsahující seznam všech objektů databáze a k nim i různé další údaje, čehož lze využít při hledání metadat (viz kapitola 2.3) [1 str. AC 582, 5].

2.3 Metadata databázových souborů

Kromě samotných uživatelských dat a databázových objektů lze v databázi nalézt i další doplňující údaje, obecně označované jako metadata [5, 6]:

- *Autor databáze* – jméno uživatele a název organizace.
- *Formát souboru* – verze aplikace, ve které byla databáze vytvořena, nastavení režimu kompatibility, atp.
- *Údaje o databázi a objektech* – datum vytvoření a poslední úpravy.
- *Uživatelské nastavení aplikace* – nastavení navigačního panelu (řazení/seskupení/šířka/…) a jiných prvků GUI až např. grafické rozvržení relací (viz obrázek 2.4).



Obrázek 2.4 – grafické rozvržení relací mezi tabulkami v aplikaci Microsoft Access 2016, které lze rovněž považovat za metadata uložená v databázi.

2.4 Formát ACCDB a možnosti jeho čtení

Nativním formátem pro ukládání Access databází je od verze 2007 ACCDB, v předchozích verzích byl hlavním formátem MDB. Oba jsou založeny na technologii Jet (u formátu ACCDB také označované jako Access Database Engine) a jsou si tedy technologicky podobné. Z uživatelského hlediska jsou rozdíly zejména v možnostech zabezpečení dat [2, 6, 7].

Jedná se o proprietární binární formáty vyvíjené společností Microsoft bez dostupné specifikace, avšak je zřejmé, že součástí databázových souborů ve formátu ACCDB, potažmo MDB, musí být uložené objekty zmíněné v kapitole 2.2 a metadata zmíněná v kapitole 2.3 [6].

Jediným oficiálním nástrojem pro správu je právě Microsoft Access, pro přístup k datům z jiných aplikací pak technologie ODBC a OLE DB. To velmi omezuje možnosti programového přístupu k databázím – pokud bychom vzali v potaz pouze oficiální nástroje, jsme limitováni na systémy s nainstalovanou aplikací Microsoft Access (a tím pádem i operačním systémem Microsoft Windows). V současné době jsou však

dostupné i nástroje vzniklé na základě reverzního inženýrství formátů ACCDB/MDB bez závislosti na programovém vybavení počítače [6].

V následujících podkapitolách jsou zmíněny všechny možnosti čtení souborů ACCDB včetně výhod a nevýhod, jaké přináší.

2.4.1 ODBC

ODBC (*Open Database Connectivity*) je standardizované API pro přístup k datům uloženým v databázích. Připojení ke konkrétním databázím je zajištěno speciálními ovladači, které lze do systému doinstalovat. Pro komunikaci skrze ODBC se typicky využívá jazyk SQL (*Structured Query Language*), ovladač poté zajistí vykonání příkazu nad konkrétní databází [8].

Pro přístup k ACCDB databázím v rámci OS Microsoft Windows se využívají ovladače Access Database Engine nainstalované spolu s aplikací Microsoft Access, případně ze samostatného distribučního balíku. Pro další platformy existují komerční Access ODBC ovladače[2]. Vzniká zde tedy závislost na dostupnosti ovladače, přičemž v určitých případech může být problém jej do systému doplnit [2].

Zásadní nevýhodou přístupu k datům přes ODBC API jsou omezení vyplývající z univerzálnosti metody. Jednoduše lze pracovat pouze s daty v tabulkách a není možné přímo přistupovat k dalším uloženým objektům. Jedinou možností je využít skryté systémové tabulky, pomocí kterých lze zjistit alespoň existenci objektů [8].

Novější obdobnou technologií je OLE DB (*Object Linking and Embedding, Database*), vyvinuté firmou Microsoft původně jako nástupce ODBC; a dále technologie ADO a ADO.NET stavící nad ODBC, resp. OLE DB [9]. Z hlediska způsobu použití a nabízených funkcí pro čtení souboru ACCDB jsou však všechny technologie shodné.

2.4.2 Microsoft Office Interoperability

Aplikace z balíku Microsoft Office lze programově ovládat pomocí technik obecně označovaných jako *interoperability* (zkráceně *interop*). Typicky se využívají proprietární technologie COM (*Common Object Model*) a OLE vyvinuté firmou Microsoft [9]. Dále jsou poskytovány *Primary Interop Assemblies* – knihovny určené

[2] Např. viz WWW: `https://www.easysoft.com/products/data_access/odbc-access-driver/`

pro použití na platformě .NET (tedy v tzv. řízeném kódu) obalující COM volání do objektového rozhraní. V současné době jsou tyto knihovny nejjednodušší cestou pro programové ovládání aplikací Microsoft Office (mj. se využívají i pro psaní doplňků, *plug-inů*, pro jednotlivé Office aplikace) [10, 11].

Tato technika oproti ODBC umožňuje kompletní správu databáze vč. všech dostupných objektů bez nutnosti analyzovat obsah systémových tabulek. Zůstává zde však omezení na systémy, kde je nainstalovaný Microsoft Access. Jde rovněž o poměrně pomalý přístup, jelikož „interop kód" de facto jen ovládá aplikaci Microsoft Access spuštěnou na pozadí.

2.4.3 MDB Tools

Jedná se o open-source sadu nástrojů pro práci se soubory Microsoft Access, respektive Jet databázemi ve formátu MDB, jejíž vývoj započal již v roce 2000. Vzhledem k uzavřenosti formátu vznikla většina nástrojů technikami reverzního inženýrství, není tedy zaručena stoprocentní funkčnost a kompatibilita [12].

Nástroje jsou napsány v jazyce C a mají konzolové rozhraní, dále existuje několik grafických nadstaveb pro prohlížení Access souborů. Součástí projektu je i dokument popisující strukturu a klíčové části Jet databází [5]. V posledních letech probíhá vývoj pomalým tempem a podpora novějších verzí Access databází včetně formátu ACCDB není zaručena [13]. Hlavní výhodou nástrojů je nezávislost na konkrétní platformě a externích knihovnách.

2.4.4 MDB Tools Java

V roce 2004 začala *portace* nástrojů MDB Tools pro jazyk Java, vývoj však již po roce ustal [14]. Následně vzniklo několik *forků* (projektů založených na kódu původního projektu), nejaktuálnější z nich lze nalézt pod názvem OME MDB Tools. Vývoj těchto projektů je ale spíše pomalý – poslední větší aktualizace proběhla v roce 2016 [15].

Oproti dále uvedené knihovně Jackcess nabízí méně možností a použití je značně komplikované [16]. Překážkou je chybějící dokumentace jak samotných nástrojů, tak i jejich programového kódu.

2.4.5 Jackcess

Jackcess je knihovna pro platformu Java poskytující čisté objektové rozhraní pro práci s Microsoft Access databázemi. Její vývoj započal v roce 2005 v rámci open-source projektu OpenHMS zaštítěného firmou Health Market Science, Inc a funguje na stejných principech jako nástroje MDB Tools, kterými se vývojáři na počátku inspirovali [17, 18].

Kromě čtení dat z tabulek umožňuje i základní editaci struktury databáze, výpis všech relací mezi tabulkami a výpis uložených dotazů. Díky přístupu ke skrytým systémovým tabulkám lze vyhledat i uložené formuláře a sestavy (reálně však lze zjistit pouze jejich existenci). Podporuje Access databáze ve verzích 2000 až 2016 (ve formátu ACCDB i MDB) a ve verzi 97 v režimu pro čtení. Knihovna neobsahuje rozhraní pro spouštění SQL dotazů, neumožňuje tedy ani vyhodnocení uložených dotazů [18].

Zásadní výhodou pro potřeby této práce je přenositelnost knihovny (nezávislost na platformě) a aktivní vývoj – tím pádem i podpora nejnovějších verzí Access databází. Vzhledem k distribuci v podobě samostatné Java knihovny (rovněž dostupné v Maven repozitářích) je její použití ve vlastní aplikaci jednoduché [19].

Pro knihovnu existuje rozšíření nazvané Jackcess Encrypt umožňující správu databází zašifrovaných heslem. Podporuje některé formy šifer aplikací Microsoft Access a Microsoft Money [20].

2.4.6 JDBC

Java Database Connectivity je API pro přístup k relačním databázím, a tedy obdobou technologie ODBC pro programovací jazyk Java. API je standardní součástí platformy Java SE. Připojení ke konkrétní databázi je opět zajištěno ovladači určenými pro konkrétní typ databází [21].

Microsoft neposkytuje vlastní JDBC ovladač pro práci s Access/Jet databázemi, existují ale speciální ovladače, tzv. *JDBC-ODBC bridge,* umožňující použití ODBC ovladačů. V rámci platformy Java byl takový ovladač standardní součástí až do verze 7; do novějších verzí jej lze překopírovat (jde ale o neoficiální postup bez záruky funkčnosti) nebo nahradit komerčními alternativami [8, 22].

Dále existuje několik JDBC „nativních“ ovladačů pro práci s Microsoft Access databázemi. Jde zejména o open-source projekt UCanAccess, který využívá již

zmíněnou knihovnu Jackcess [23]. K dispozici jsou rovněž komerční ovladače [3,4,5].

Stejně jako u technologie ODBC jsou největší nevýhodou omezení v důsledku univerzálnosti přístupu; jinými slovy – jednoduše pracovat je možné pouze s daty v tabulkách a k ostatním objektům databází je přístup obtížný či nemožný [21]. Výhodou je lepší přenositelnost na jiné platformy.

[3] Viz WWW: `https://www.easysoft.com/products/data_access/jdbc-access-gateway/`
[4] Viz WWW: `http://www.hxtt.com/access.html`
[5] Viz WWW: `http://sesamesoftware.com/relational-junction/jdbc-database-drivers-products/relational-junction-mdb-jdbc-driver/`

3 Portál ZČU

Obecně termínem portál označujeme webovou aplikaci, která uživateli poskytuje jednotným způsobem a centralizovaně informace z různých zdrojů, které uživatele zajímají nebo se ho nějakým způsobem týkají [24]. Následující podkapitoly se věnují portálu Západočeské univerzity v Plzni (dále jen ZČU) a validátoru studentských prací, který je s portálem úzce spjatý.

3.1 Základní informace

Portál ZČU[6] je informační zdroj pro studijní účely používaný studenty i zaměstnanci Západočeské Univerzity. Zastřešuje různé klíčové aplikace univerzity – z pohledu studentů se jedná zejména o *IS/STAG* (Informační systém studijní agendy, umožňující zápis předmětů, zkoušek, prohlížení rozvrhů, hodnocení kvality výuky, atp.) a *Courseware* (místo s materiály používanými v rámci výuky předmětů) [24].

Jednotlivé stránky portálu se skládají z více či méně nezávislých částí, které se nazývají portlety. Jedním z nich je *aplikace pro správu semestrálních prací, jejich odevzdávání a hodnocení*. Vyučujícím slouží k vypisování témat semestrálních prací, do nichž se studenti posléze mohou přihlašovat a odevzdávat své práce. Vyučující si může práce jednotlivě či hromadně stáhnout, schvalovat je nebo vracet k přepracování a zanechávat studentům poznámky a hodnocení [24, 25].

3.2 Validátor studentských prací

Studenti často po odevzdání své práce čekají delší dobu na zkontrolování a ohodnocení vyučujícím. Pokud práci shledá nevyhovující, vrací ji studentovi k přepracování. Tento proces se navíc může i několikrát opakovat, což znamená pro studenta čekání v nejistotě a pro vyučujícího velkou časovou zátěž.

Cílem validátoru studentských prací (který je propojen s již zmíněnou aplikací na

[6] Dostupný na adrese `https://portal.zcu.cz/`

portálu ZČU) je eliminovat tento proces a automaticky kontrolovat, zda je práce „vyhovující“ – to může mít mnoho podob, například [26]:

- **Práce splňuje formální náležitosti:**
  - soubor má požadovaný formát,
  - soubor je vhodně pojmenován,
  - v odevzdaném archivu jsou požadované soubory ve správné adresářové struktuře,
  - v nahraném archivu nejsou žádné nadbytečné soubory.
- **Práce je vyhovující kvalitativně:**
  - dokument splňuje stanovený rozsah (typicky minimální počet slov či stran),
  - zdrojový kód lze zkompilovat,
  - program má totožný výstup, jako referenční program připravený vyučujícím,
  - odevzdaný soubor obsahuje všechny prvky stanovené zadáním.

Automatizace této kontroly poté přináší mnoho výhod [26]:

- Student se okamžitě dozví, zda jeho práce splňuje základní kritéria stanovená učitelem.
- Protože je nevyhovující práce okamžitě odmítnuta, nemůže student zneužít nahrání takové práce k získání dalšího času.
- Pokud není omezen počet odevzdání, může student u určitých úloh zkoušet i různá „inovativní“ řešení, u kterých si není jist správností.
- Vyučující není zatěžován mechanickým kontrolováním formální správnosti.
- K hodnocení se dostanou jen práce s určitou minimální mírou kvality.

Mezi nevýhody naopak patří [26]:

- Časová náročnost přípravy a udržování konfigurace validátoru (v případě úprav zadání).

- Striktnost v hodnocení – označení práce jako nevyhovující kvůli zanedbatelnému problému (např. chybějící mezeře ve výstupu programu). Lze předejít podrobným zadáním a poskytováním informací o úskalích studentům.
- Zvýšení rizika plagiarismu, pokud vyučující hodnocení prací plně ponechá na validátoru. Lze vyřešit na straně portálu napojením aplikace pro odevzdávání na systém pro odhalování plagiátů.

3.2.1 Validační servery

Validátor je realizován jako služba běžící na samostatném serveru, se kterou aplikace pro odevzdávání semestrálních prací komunikuje. Příprava validačních pravidel a různých nástrojů pro kontrolu prací tedy probíhá odděleně na tomto serveru, který kromě konfigurace přes terminál poskytuje i jednoduché webové rozhraní[7] umožňující správu pravidel a následné testování nad vzorovými pracemi. Souběžně s tímto serverem je dostupný testovací validační server[8] určený pro vývoj nových funkcí a přípravu validačních pravidel, kde nehrozí nebezpečí poškození aktuálně používaných dat, a je tedy vhodný i pro využití v rámci realizace této práce. Služba validátoru studentských prací je napsána v jazyce Java a je spouštěna ve webovém kontejneru Apache Tomcat na serveru s OS Linux [26, 27].

3.2.2 Validační domény

Konkrétní postup pro vyhodnocení, zda je práce „vyhovující“, je určen tzv. *validační doménou*. Ta se skládá z libovolného počtu kroků, jimiž se zajistí validace odevzdané práce, a tedy určení, zda bude práce přijata, či vrácena studentovi. Domény jsou uloženy na validačním serveru, kde je má vyučující možnost libovolně přidávat, upravovat a odebírat. Každá je uložena pod vlastním názvem – pomocí něj ji lze následně nastavit v rámci portletu pro odevzdávání prací, a tím zajistit validaci odevzdaných prací. Na serveru je pro každou doménu vytvořen adresář s konfigurací, kam lze umístit další soubory potřebné pro validaci (např. referenční řešení, pomocné programy, atp.) [26, 28].

---

[7] Viz WWW: `https://validator.zcu.cz/`
[8] Viz WWW: `https://validator-test.zcu.cz/`

Jednotlivé kroky validace představují konkrétní akce, které se mají provést. V rámci domény mají unikátní název a volitelný popis. Pro každý krok lze určit podmínku, za které se má akce provést [28]:

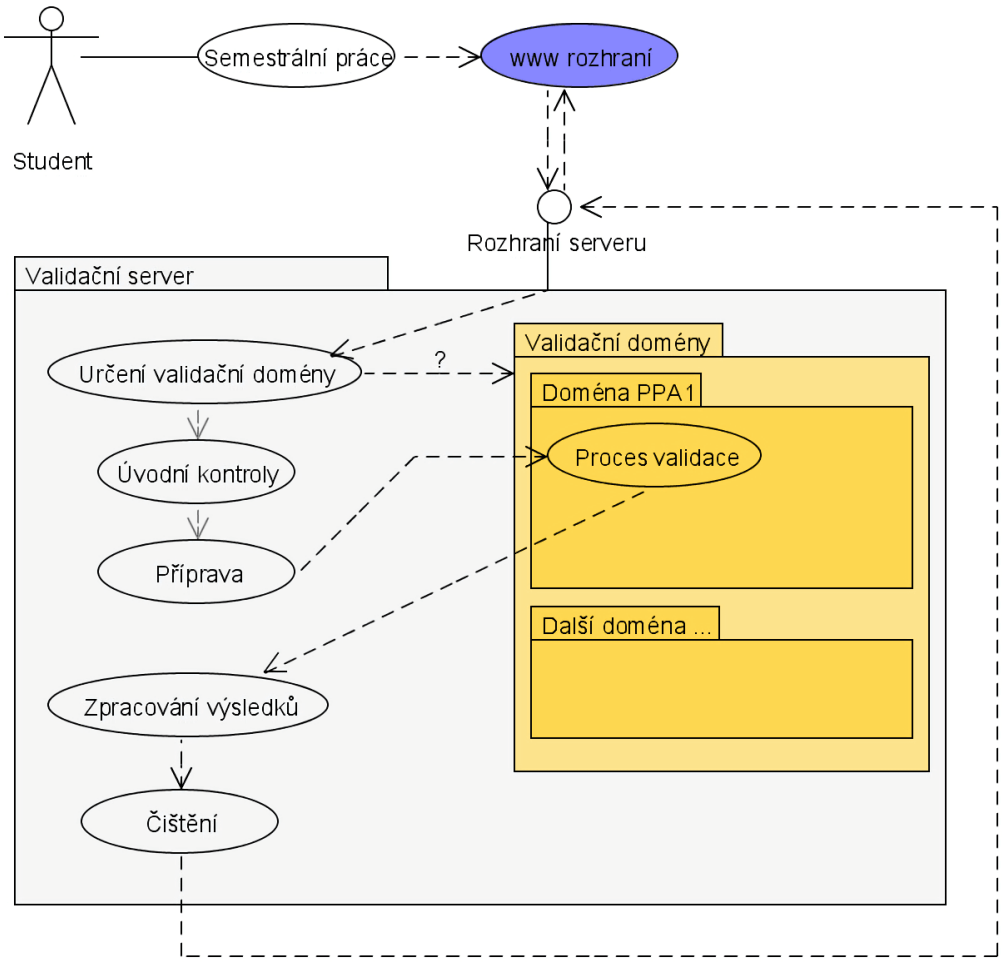
- *Vždy*.
- *Nikdy*.
- *Validace ještě neobsahuje chybu* – neboli žádný předchozí krok neskončil chybou.
- *Validace již obsahuje chybu* – alespoň jeden předchozí krok skončil chybou.
- *Vlastní skript* – lze napsat vlastní skript v jazyce JavaScript, který například vyhodnotí výsledek předchozí akce a určí, zda se má krok provést.

Akce lze rozdělit do několika kategorií [29]:

- *Textové výpisy* – pro zlepšení orientace je vhodné vypisovat relevantní informace k právě probíhané kontrole a jejím výsledkům (tedy např. *„Kontrola počtu stran“* nebo *„Chyba – očekáváno alespoň 20 stran, odevzdaný soubor obsahuje 17“*). Výpisy mohou být informativní, varovné či chybové.
- *Spouštění vlastních akcí* – jedná se o předem definované komplexnější validace, které lze ovládat pomocí parametrů. V současné době je dostupných 31 akcí rozdělených do několika skupin, např.: kompilace souboru, rozbalení ZIP souboru, spočítání slov/stran dokumentu, spuštění Java programu, kontrola názvu souboru, nalezení souboru v adresáři, porovnání PNG souborů, … Validátor lze rozšířit o další vlastní akce naprogramováním a nahráním tzv. *validačních modulů* na validační server [27, 30].
- *Vlastní skript* – opět lze napsat vlastní skript v jazyce JavaScript. Pro spouštění skriptů se používá knihovna Rhino, která umožňuje přistupovat i k běžným funkcím platformy Java [27, 28].
- *Skok na jiný krok validace* a *ukončení validace* – ve spojení s podmínkami lze takto řídit průběh validace.

3.2.3 Princip validace práce

Postup validace práce odevzdané studentem zahrnuje mnoho dílčích úkonů, jak znázorňuje diagram na obrázku 3.1.



Obrázek 3.1 – diagram znázorňující proces validace práce odevzdané studentem. Převzato z [27].

Stručně lze proces popsat následovně [27]:

1. Validační server práci přijme a načte přiřazenou validační doménu.
2. Proběhne úvodní kontrola dle konfigurace domény.
3. Příprava na validaci – vytvoření pracovního adresáře, do kterého je práce nahrána a ve které probíhají akce definované doménou (např. rozbalení archivu, atp.).

4. Spuštění validačního procesu validační domény – tedy zpracování všech definovaných kroků. Průběžně se generuje výsledek validace, který může obsahovat informativní nebo chybová hlášení.
5. Ukončení validačního procesu, vyčištění pracovního adresáře (závisí na nastavení validační domény).
6. Výsledek validace je vrácen volající aplikaci – obvykle tedy aplikaci pro odevzdávání studentských prací, který jej zobrazí studentovi. Ten má v tu chvíli možnost zjistit, zda byla jeho práce vyhovující, či nikoliv.

4 Analýza řešení kontroly prací

Cílem této práce je vytvoření systému pro kontrolu samostatných prací vytvořených v MS Access, konkrétně pro jejich validaci a rozpoznání plagiarismu; v návaznosti poté systém adaptovat pro validátor studentských prací. Následující podkapitoly se zabývají požadavky, úvodní analýzou a návrhem řešení systému.

4.1 Požadavky na řešení

Požadavky na řešení systému byly stanoveny s ohledem na využití vyučujícími, s důrazem na intuitivní použití výsledného systému a snadnou konfiguraci ve spojení s validátorem studentských prací.

Hlavní části požadovaného systému je aplikace umožňující *validaci databázových souborů vytvořených v aplikaci Microsoft Access* – neboli kontrolu databází s ohledem na splnění zadání. Vzhledem k různorodosti jednotlivých zadání musí aplikace umožnit pohodlnou konfiguraci této kontroly. To bude spočívat v možnosti zahrnout do kontroly různá pravidla testující obsah databází (dále též *validační pravidla*). Výsledkem kontroly bude označení zadaného databázového souboru jako *vyhovujícího* či *nevyhovujícího*; v druhém případě by měla aplikace zároveň poskytnout informaci o pravidle, které „selhalo“ (a zapříčinilo tedy vyhodnocení databáze jako nevyhovující).

Pravidla musí umožňovat kontrolu existence tabulek, relací mezi tabulkami různých typů, případně i dalších objektů uložených v databázi. Dále musí být možné prověřit strukturu tabulek – existenci sloupců konkrétního názvu či datového typu, atp. Bude možné i ověřit počet řádků v tabulce či tabulkách (zejména v podobě kontroly, zda všechny tabulky obsahují alespoň určitý počet řádků).

Další částí aplikace bude detekce plagiarismu, respektive vyhledávání podobností mezi databázovými soubory. Pro databáze může být obtížné definovat, kdy se již jedná o plagiát – například dvě databáze obsahující stejně pojmenované tabulky mohou, ale nemusí být plagiátem. V takovém případě může aplikace pouze upozorňovat na podobnost; pokud se ale podaří na základě dalších informací jednoznačně určit, že se o plagiát jedná, budou tak takové soubory označeny.

Aplikace bude uživatelům nabízet přívětivé grafické rozhraní, pomocí kterého budou

moci spravovat aktuální konfiguraci validace – tedy nastavení jednotlivých pravidel. Validaci musí být možné spustit hromadně nad více databázemi, výsledek poté bude pro každou z nich určen zvlášť.

Součástí systému bude dále aplikace přizpůsobená pro použití v rámci validátoru studentských prací – konkrétním řešením může být konzolová aplikace, případně nový validační modul. S ohledem na jednoduchou konfiguraci je nutné, aby hlavní část systému (tedy již popsaná aplikace s grafickým rozhraním) umožňovala export konfigurace (pravidel validace) pro použití v části adaptované pro validátor. Výstupem exportu může být konfigurační soubor nebo textový řetězec.

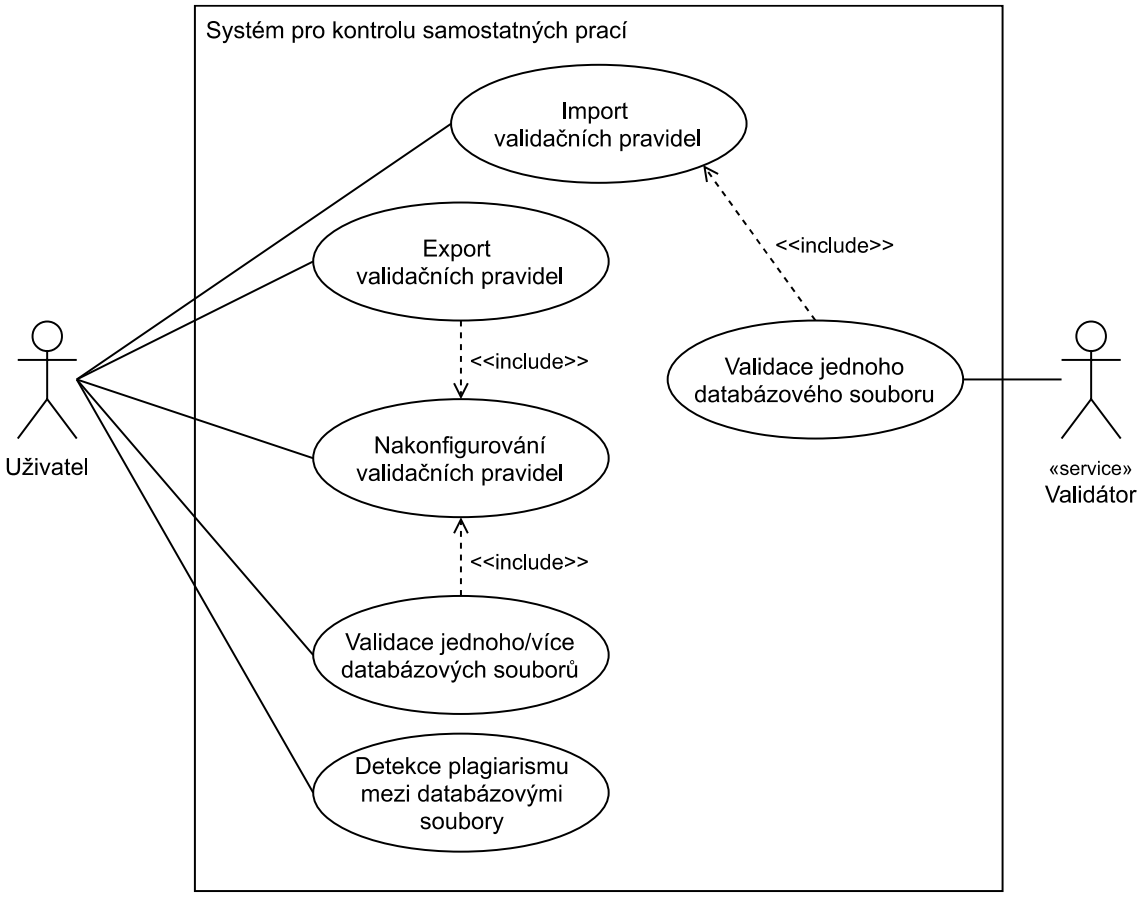
4.2 Případy užití

Diagram na obrázku 4.1 znázorňuje případy užití celého systému z pohledu uživatele i validátoru studentských prací.

Popis diagramu:

- *Import validačních pravidel* – uživatel může načíst dříve vytvořenou sadu pravidel pro validaci databázových souborů ve formátu ACCDB. Stejnou funkcionalitu využívá i část adaptovaná pro validátor studentských prací, které jsou automaticky předávány odevzdané studentské práce.
- *Export validačních pravidel* – uživatel může exportovat vytvořenou konfiguraci validace, tedy sadu aktivních validačních pravidel.
- *Nakonfigurování validačních pravidel* – uživatel může vytvořit sadu pravidel ověřující, zda jsou zadané databáze vyhovující. Bližší popis konfigurace je uveden v kapitole 4.4.
- *Validace jednoho či více databázových souborů* – uživatel může pomocí aplikace provést validaci databázových souborů. Výstupem akce je informace, zda jednotlivé soubory vyhovují aktuálně nakonfigurovaným pravidlům.
- *Detekce plagiarismu mezi datovými soubory* – uživatel může při zadání více databázových souborů spustit vyhledání plagiátů.
- *Validace jednoho databázového souboru* – komunikace se systémem ze strany validátoru studentských prací; do systému je předán databázový

soubor a konfigurace validačních pravidel. Výstupem je opět informace, zda je soubor vyhovující pravidlům.



Obrázek 4.1 – diagram případů užití systému pro kontrolu samostatných prací.

4.3 Metoda čtení databázových souborů

Jak již bylo probráno v kapitole 2.4, existuje několik možností pro čtení databázových souborů ve formátu ACCDB.

4.3.1 Kritéria pro výběr metody čtení

Na základě požadavků na řešení systému lze stanovit kritéria pro výběr metody čtení souborů v podobě následujících vyžadovaných schopností:

- Vypsání názvů tabulek v databázi.

- Vypsání struktury tabulky – zjištění názvů a datových typů sloupců, informace o primárních klíčích.
- Zjištění počtu řádků v tabulce, případně získání všech dat z tabulky.
- Nalezení relací mezi tabulkami v podobě, aby bylo možné určit typ.
- Vypsání dotazů uložených v databázi včetně druhu.

Dalšími omezujícími kritérii jsou funkční požadavky na výsledný systém:

- Spustitelnost na osobních počítačích s OS Microsoft Windows.
- Spustitelnost části adaptované pro validátor na validačním serveru, tj. kompatibilita se službou napsanou v jazyce Java na serveru s OS Linux.

4.3.2 Výběr metody čtení

Některé nalezené metody pro čtení souborů ve formátu ACCDB je nutné na základě stanovených kritérií vyřadit – konkrétně se jedná o ODBC a Microsoft Office Interop. Nesplňují zejména poslední zmíněný bod, tedy spustitelnost na serveru s OS Linux (na kterém není dostupný Microsoft Access ani ODBC ovladače). U metod přístupu prostřednictvím API ODBC a JDBC je navíc problematické pracovat s relacemi a dotazy uloženými v databázích.

Ze zbývajících možností je ideální volbou knihovna *Jackcess* pro platformu Java a bude tedy použita pro implementaci systému. Vývoj v jazyce Java přinese výhodu v multiplatformnosti a bude tak možné vyvíjet obě části (hlavní aplikaci a část adaptovanou pro validátor) nad společnými základy. Oproti ostatním metodám navíc knihovna Jackcess poskytuje nejucelenější přístup k databázím, podporu i nejnovějších verzí díky aktivnímu vývoji a podrobnou programovou i uživatelskou dokumentaci.

4.4 Validace databázových souborů

Validace databází bude spočívat ve vyhodnocení sady nakonfigurovaných pravidel kontrolujících obsah a strukturu. Pokud budou všechna pravidla splněna, označí se databáze jako *vyhovující*, v opačném případě jako *nevyhovující*.

4.4.1 Definice základních validačních pravidel

Na základě požadavků na řešení lze nyní definovat základní validační pravidla potřebná pro kontrolu prací se zaměřením na splnění zadání a možný princip jejich funkčnosti (zohledňující možnosti knihovny Jackcess):

- *Kontrola počtu tabulek v databázi* – získání seznamu všech tabulek v databázi (například jejich názvů) a ověření počtu vůči požadovanému.
- *Kontrola existence tabulky dle názvu* – získání názvů všech tabulek v databázi a ověření, že je mezi nimi hledaná tabulka.
- *Kontrola počtu řádků v každé tabulce* – získání seznamu všech tabulek v databázi a pro každou z nich ověření počtu řádků vůči požadovanému.
- *Kontrola počtu sloupců v každé tabulce* – získání seznamu všech tabulek v databázi, pro každou z nich získání seznamu sloupců; následně ověření počtu vůči požadovanému.
- *Kontrola existence sloupce dle zadaných kritérií v každé tabulce (kritériem může být název, datový typ nebo příslušnost k primárnímu klíči)* – získání seznamu všech tabulek v databázi, pro každou z nich získání seznamu sloupců; následně ověření existence sloupce dle zadaných kritérií.
- *Kontrola počtu sloupců dle zadaných kritérií v každé tabulce* – získání seznamu všech tabulek v databázi, pro každou z nich získání seznamu sloupců; následně ověření počtu sloupců vyhovujících zadaným kritériím. Jedná se pouze o zobecnění předchozího pravidla.
- *Kontrola počtu tabulek obsahujících sloupec dle zadaných kritérií* – získání seznamu všech tabulek v databázi, pro každou z nich získání seznamu sloupců; odebrání tabulek neobsahujících sloupec dle zadaných kritérií; ověření počtu zbylých tabulek v seznamu vůči požadovanému počtu.
- *Kontrola počtu relací typu 1:1* – získání seznamu všech relací mezi tabulkami v databázi, vyfiltrování relací označených jako 1:1; ověření počtu vůči požadovanému.
- *Kontrola počtu relací typu 1:N* – získání všech relací mezi tabulkami v databázi, odebrání relací označených jako 1:1; ověření počtu vůči požado-

vanému. Tento postup zahrne do počtu i všechny relace s rozkladovými tabulkami, které v důsledku tvoří relace M:N.

- *Kontrola počtu relací typu M:N* – získání seznamu všech tabulek v databázi, vyfiltrování rozkladových tabulek; ověření počtu zbylých tabulek vůči požadovanému. Postup pro nalezení rozkladových tabulek může spočívat v hledání tabulek, které mají cizí klíče odkazující se na právě dvě další tabulky.
- *Kontrola počtu dotazů dle zadaného druhu* – získání seznamu všech dotazů v databázi, vyfiltrování dle druhu a ověření počtu dle požadovaného.

4.4.2 Pravidla existence a počtu výskytů

Kontroly na existenci či počty objektů v databázi lze zobecnit na hledání počtu pomocí porovnávacích operátorů (tedy „rovná se“, „větší než“ a „menší než“). Uživatel by měl možnost nakonfigurovat pravidla například v následující podobě:

- Databáze obsahuje právě jednu tabulku s názvem „články“.
- Databáze obsahuje méně než dvě tabulky obsahující méně než 5 řádků.
- Databáze obsahuje nula relací typu 1:1.
- Databáze obsahuje více než jednu relaci typu 1:N.

4.5 Vyhodnocení plagiarismu

Za plagiátorství lze označit úmyslné kopírování nebo celkové napodobování prací jiných autorů a vydávání za vlastní. To platí i v případě samostatných prací vytvořených v aplikaci Microsoft Access. Vzhledem k automatizaci kontroly je však nutné najít spolehlivý a důvěryhodný postup pro označování prací jako plagiátů.

Samotné vyhodnocení plagiarismu mezi několika databázovými soubory může spočívat v hledání různých společných „prvků podobnosti“, přičemž každý takovýto prvek může přispívat určitou „vahou“ k označení dvou souborů jako plagiátů (konkrétní rozlišení originálu a plagiátu je ponecháno na posouzení uživatele). Příkladem takových prvků jsou:

- Názvy a struktura tabulek (názvy sloupců, datové typy, …).

- Názvy a typy dotazů uložených v databázi.
- Názvy dalších objektů (formulářů a sestav).
- Metadata databázového souboru:
    - Údaje o autorovi databáze (jméno a název organizace).
    - Datum a čas vytvoření/poslední úpravy jednotlivých objektů.
    - Datum a čas vytvoření/poslední úpravy databázového souboru.
    - Nastavení uživatelského rozhraní.
    - Grafické rozvržení relací mezi tabulkami (viz obrázek 2.4).

Z těchto prvků lze pro jednoznačné určení plagiátu využít zejména grafické rozvržení relací mezi tabulkami, které je typicky pro každou databázi unikátní.

Knihovna Jackcess podporuje získání některých metadat pomocí vlastních funkcí v podobě hashovacích tabulek (jedná se právě o různé údaje typu autor a název databáze, nastavení uživatelského rozhraní, informace o formátu), další údaje je nutné získat ze skrytých systémových tabulek, ke kterým knihovna umožňuje přistupovat stejně jako k uživatelským tabulkám. Již zmíněné grafické rozvržení relací je uloženo v systémové tabulce `MSysObjects` (záznam s typem `-32758`, sloupec `LvExtra`) v binární podobě, díky reverznímu inženýrství je znám i formát těchto dat[9].

4.6 Grafické uživatelské rozhraní

Grafické rozhraní hlavní aplikace bude rozděleno do několika částí, přičemž všechny mohou být součástí jednoho okna:

- Seznam dostupných validačních pravidel.
- Seznam právě aktivních konfigurovatelných pravidel.
- Konfigurační panel právě vybraného pravidla.
- Seznam databázových souborů ke kontrole.
- Nástrojová lišta s tlačítky pro ovládání programu.

[9] Viz kód nástroje z webu `http://www.lebans.com/saverelationshipview.htm`

Databázové soubory ke kontrole mohou být zobrazeny ve stromové struktuře, přičemž v první úrovni stromu budou zobrazeny názvy souborů a ve druhé informace o výsledcích kontroly. Jednotlivé položky stromu je rovněž vhodné rozlišit pomocí různých ikon pro vyhovující a nevyhovující databáze a dále takové, které byly vyhodnoceny jako plagiáty.

4.7 Adaptace pro validátor studentských prací

Patrně nejjednodušší způsob adaptace vytvářeného systému pro validátor studentských prací spočívá ve využití již existující „vlastní akce“ validátoru pro spuštění Java programu zabaleného do souboru formátu JAR. K tomu je zapotřebí vytvořit aplikaci s konzolovým rozhraním, která bude nahrána na validační server do složky příslušné validační domény.

Aplikace se bude ovládat pouze pomocí parametrů – jedním z nich bude název odevzdaného souboru (tj. studentské práce nahrané na portál), druhým pak název souboru s validačními pravidly pro kontrolu správnosti zadání. Výstupem aplikace bude textová informace o výsledku validace, v případě neúspěchu dále vypíše podrobnosti o příčinách (neboli popis validačního pravidla, které selhalo) na standardní chybový výstup – ten je následně zobrazen studentům, kteří tak získají informaci, z jakého důvodu je odevzdaná práce nevyhovující.

4.8 Návrh struktury systému

Systém pro kontrolu samostatných prací bude napsán v programovacím jazyce Java – důvodem je volba knihovny Jackcess pro čtení souborů ve formátu ACCDB (která je vytvořena právě na platformě Java), výhodou pak je výsledná multiplatformnost a snadné začlenění do validátoru studentských prací (který rovněž využívá platformu Java). Jazyk podporuje rozdělení kódu do oddělených jmenných prostorů nazývaných *balíky*, případně vytvoření oddělených projektů. Lze tak dosáhnout rozdělení kódu na volně spojené zapouzdřené části či komponenty.

Navrhovaný systém lze rozdělit do několika vzájemně propojených částí:

- *Práce s daty obsaženými v databázi* – zprostředkovává rozhraní pro získávání údajů o tabulkách, relacích, atd. prostřednictvím knihovny Jackcess.

- *Validace databáze* – obsahuje seznam dostupných validačních pravidel a poskytuje rozhraní k nástroji na validaci databázových souborů.
- *Kontrola plagiarismu* – poskytuje rozhraní k nástroji na vyhodnocení plagiarismu mezi více databázovými soubory.
- *Grafické rozhraní* – slouží pro konfiguraci validačních pravidel, jejich export, spouštění validace a kontroly plagiarismu nad zadanými databázovými soubory.
- *Konzolové rozhraní* – umožňuje spustit validaci jedné databáze pomocí validačních pravidel uložených v souboru.

Výsledný systém bude rozdělen do projektů tak, aby aplikace používaná na validačním serveru obsahovala minimum dalších závislostí. Možným způsobem je rozdělení na projekt obsahující tzv. aplikační logiku systému (tedy vše potřebné pro kontrolu prací) spolu s konzolovým rozhraním a projekt obsahující pouze grafické rozhraní určené pro obsluhu uživatelem.

5 Implementace systému pro automatickou kontrolu prací

Na základě předchozí analýzy jsou známy požadavky na vyvíjený systém, je navržena základní struktura i funkčnost výsledných aplikací. Tato kapitola se zaměřuje na samotnou implementaci. Popisuje, jak je výsledný systém rozdělený do částí a komponent a zaměřuje se i na algoritmy použité pro zajištění požadované funkčnosti.

5.1 Použité technologie

Výsledný systém je naprogramován v jazyce Java 8. Grafické rozhraní hlavní aplikace je postaveno na platformě JavaFX 8, dále se využívá knihovna ControlsFX[10] obsahující různé předpřipravené prvky grafického uživatelského rozhraní. Pro čtení databázových souborů vytvořených v aplikaci Microsoft Access se používá open-source knihovna Jackcess ve verzi 2.1.10. Pro testování jsou využity nástroje JUnit 5[11] a Mockito 2[12]. Celý projekt je spravován nástrojem Apache Maven[13] a vyvíjen byl ve vývojovém prostředí JetBrains IntelliJ IDEA[14].

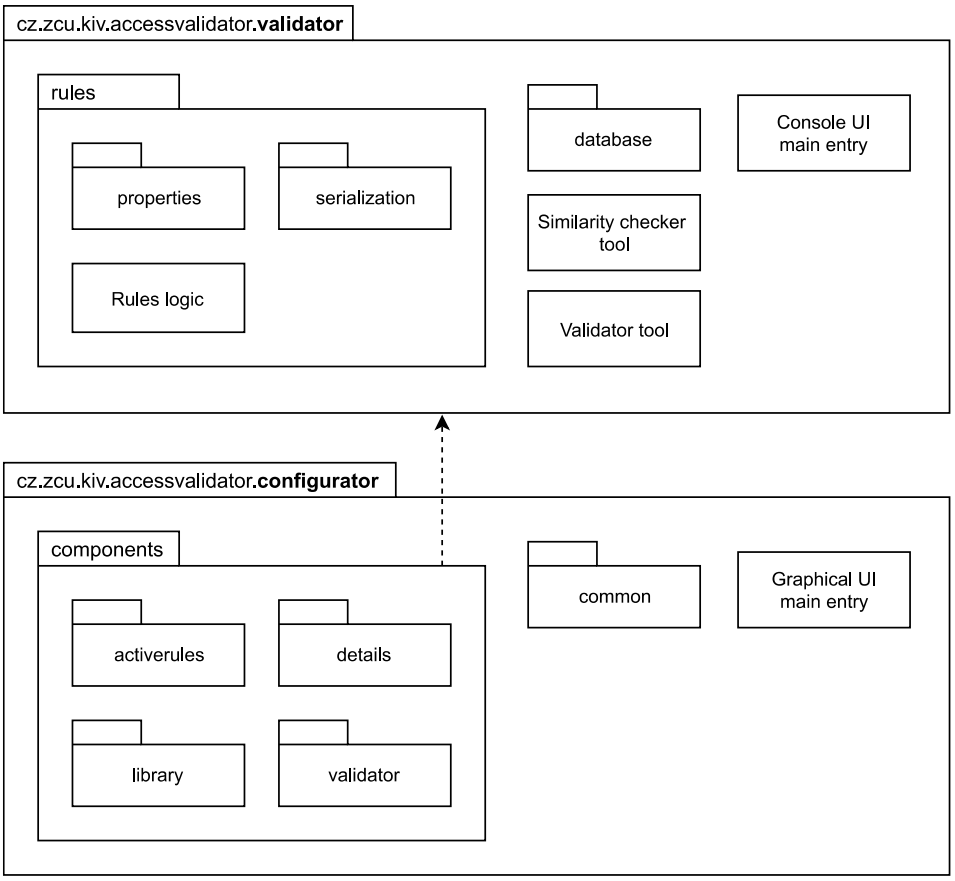
5.2 Struktura implementovaného systému

Systém je rozdělen do dvou projektů či *modulů* (v terminologie nástroje Apache Maven) dle návrhu z kapitoly 4.8. Rozvržení je rovněž vidět na obrázku 5.1, který zachycuje moduly a vnořené balíky.

Pozn.: Konkrétní názvy balíků a tříd v následujících diagramech jsou uvedeny bez nadřazeného balíku `cz.zcu.kiv.accessvalidator`.

---

[10] Viz web knihovny ControlsFX: `http://fxexperience.com/controlsfx/`
[11] Viz web knihovny JUnit 5: `https://junit.org/junit5/`
[12] Viz web knihovny Mockito: `http://site.mockito.org/`
[13] Viz web nástroje Apache Maven: `https://maven.apache.org/`
[14] Viz web vývojového prostředí IntelliJ IDEA: `https://www.jetbrains.com/idea/`



Obrázek 5.1 – diagram modulů a balíku implementovaného systému.

5.2.1 Modul validator

První modul nazvaný *validator* obsahuje tzv. aplikační logiku – neboli vše potřebné pro kontrolu samostatných prací. Dále je do tohoto modulu začleněno konzolové rozhraní aplikace (viz kapitola 5.7). Modul je členěn do 28 tříd rozdělených do 5 balíků, dále následuje popis nejdůležitějších z nich. Rozvržení je vidět na obrázku 5.2.

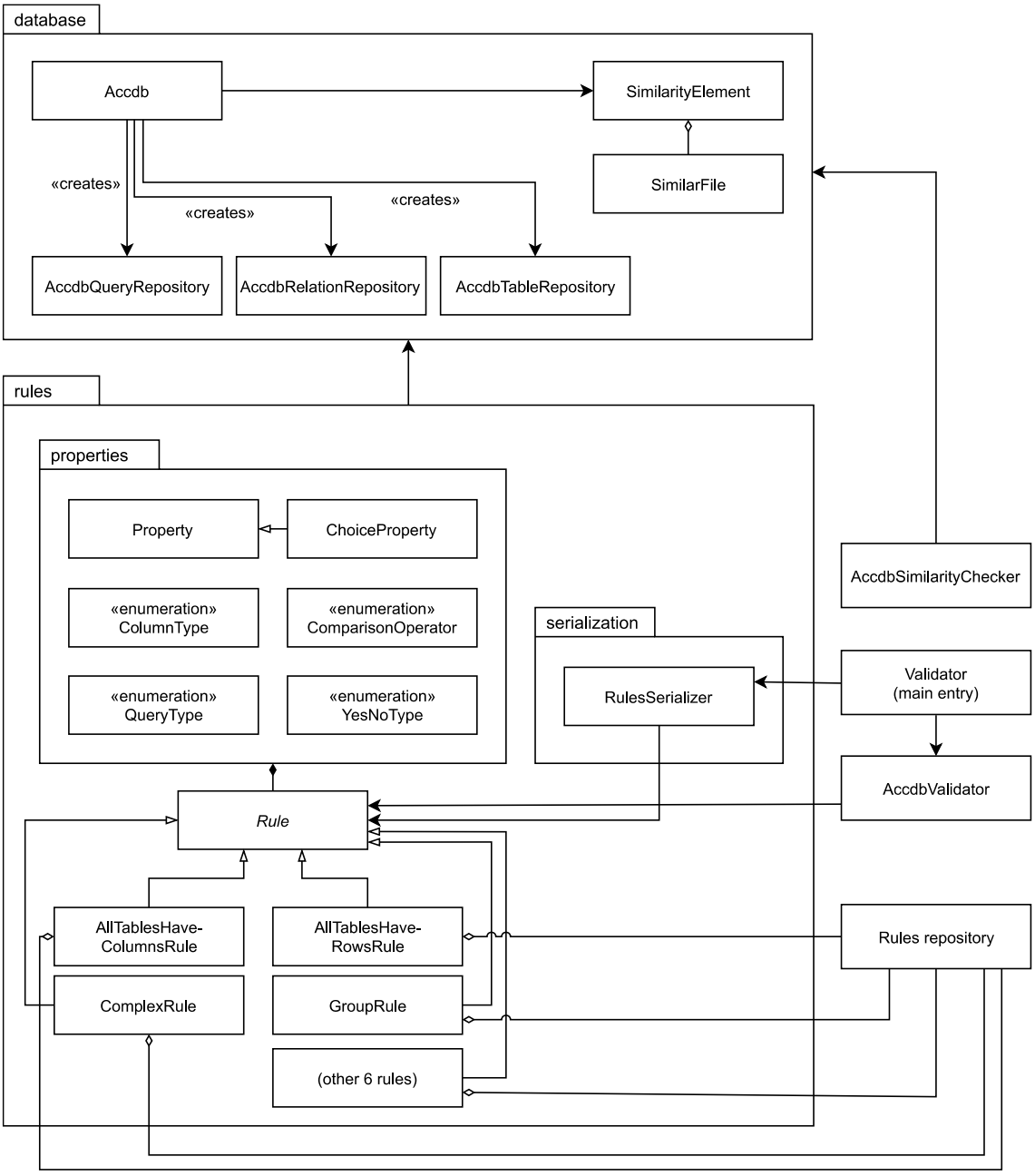
Balík `validator`

Hlavní balík `validator` obsahuje dvě třídy reprezentující klíčové nástroje pro kontrolu samostatných prací vytvořených v aplikaci Microsoft Access:

- Třída `AccdbValidator` – slouží pro validaci databázových souborů neboli kontrolu se zaměřením na splnění zadání. Podrobnosti implementace jsou uvedeny v kapitole 5.4.

- Třída `AccdbSimilarityChecker` – slouží pro kontrolu samostatných prací se zaměřením na plagiarismus. Podrobnosti implementace jsou uvedeny v kapitole 5.5.

Balík dále obsahuje třídu s konzolovým rozhraním aplikace adaptovaným pro validátor portálu ZČU (podrobnosti viz kapitola 5.7) a třídu poskytující seznam se všemi validačními pravidly.



Obrázek 5.2 – zjednodušený diagram tříd v modulu validator.

Balík `database`

Třídy obsažené v balíku `database` poskytují funkce pro práci s databázovými soubory. Pro svoji funkčnost využívají knihovnu Jackcess a oddělují tak použití této knihovny od zbytku systému.

Balík `rules`

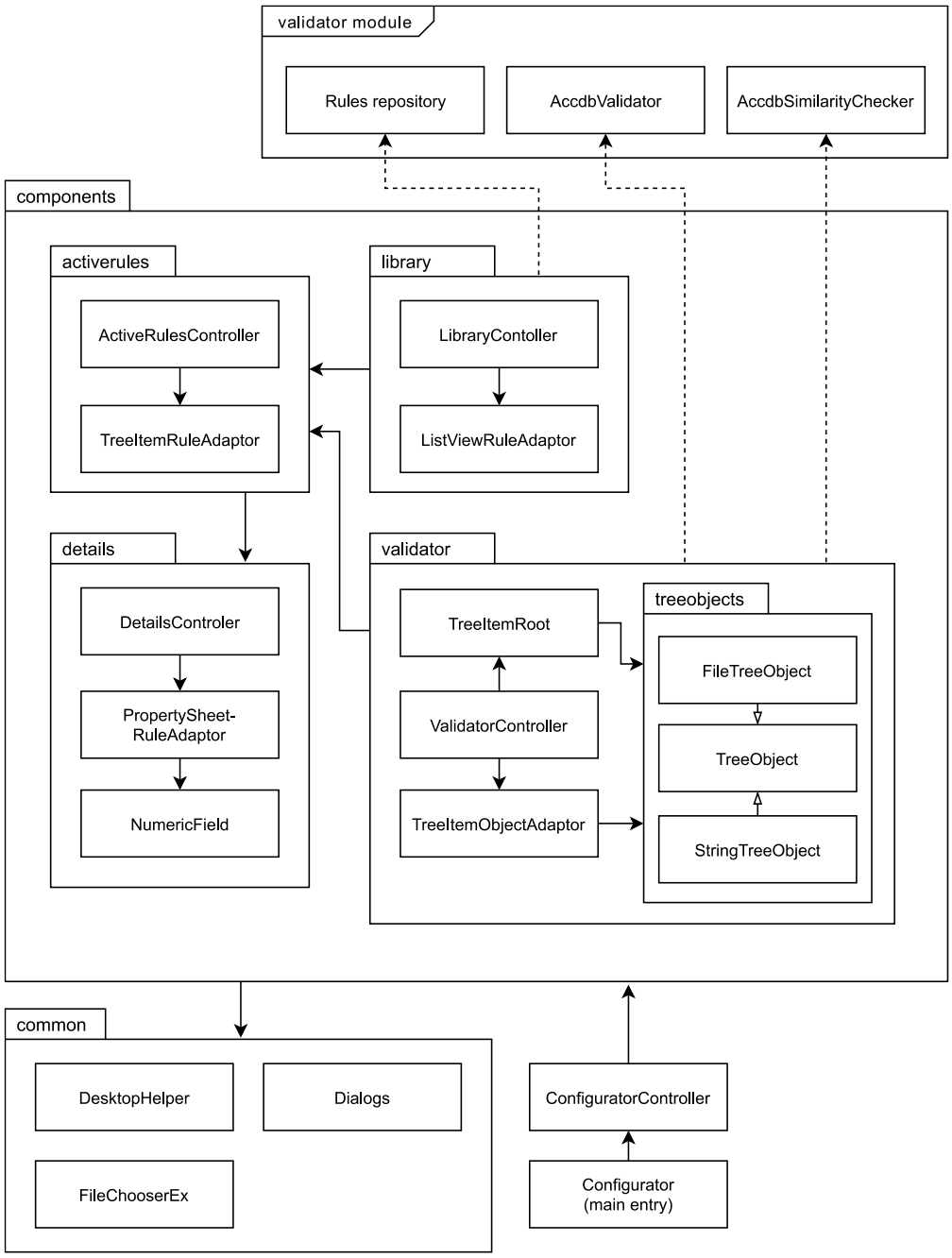
Součástí balíku `rules` je deset tříd reprezentujících různá validační pravidla. Každé pravidlo má definovaný obecný název (např. „Existence tabulky dle názvu“), metodu vracející popis aktuální konfigurace („Existence tabulky s názvem ‚studenti‘“) a v neposlední řadě kolekci různých vlastností, prostřednictvím nichž lze pravidlo nakonfigurovat (pro zmíněné pravidlo půjde o vlastnost „Název tabulky“). Podrobnosti o jednotlivých implementovaných pravidlech jsou uvedeny v kapitole 5.4.1.

Vnořený balík `properties` obsahuje právě třídy používané pro realizaci vlastností validačních pravidel. Připraveny jsou dva typy vlastností – jeden umožňuje uživatelům zadání libovolné hodnoty, druhý pak pouze výběr jedné možnosti z několika předem definovaných. Součástí balíku je několik výčtových tříd obsahujících např. seznam datových typů sloupců používaných v databázích aplikace Microsoft Access. Tyto třídy se používají právě ve zmíněném druhém typu vlastností.

Druhý vnořený balík `serialization` obsahuje jedinou třídu, která, jak již název balíku napovídá, slouží k serializaci a deserializaci nakonfigurovaných pravidel do souborů. Pro realizaci byl použit formát XML.

5.2.2 Modul configurator

Modul s názvem *configurator* obsahuje aplikaci s grafickým uživatelským rozhraním, pomocí kterého mohou uživatelé spouštět a konfigurovat kontrolu samostatných prací a prohlížet výsledky těchto kontrol. Dále umožňuje exportovat (ukládat) a opět načítat konfiguraci validačních pravidel ze souborů. Podrobnosti o implementaci rozhraní jsou uvedeny v kapitole 5.6. Na obrázku 5.3 je opět znázorněno rozvržení tříd v rámci modulu.



Obrázek 5.3 – zjednodušený diagram tříd v modulu configurator.

Balík `configurator`

V balíku `configurator` je umístěna třída s hlavní metodou programu, která spouští zmíněné grafické uživatelské rozhraní. Cílem této metody je pouze inicializovat JavaFX aplikaci, jejíž obsluhu poté zajišťuje druhá třída tohoto balíku, tzv. *controller*.

Balík `common`

Součástí balíku `common` jsou tři pomocné třídy. První z nich slouží k otevírání souborů v asociovaném programu, respektive nadřazené složky v průzkumníku souborů. Druhá třída obsahuje metodu pro zobrazení chybového dialogového okna. Poslední třída obaluje funkčnost třídy `FileChooser` sloužící pro výběr souborů pomocí dialogového okna a přidává schopnost pamatovat si poslední použitý adresář i po restartu aplikace.

Balík `components`

Balík `components` je rozdělen na čtyři další balíky, z nichž každý představuje jednu část uživatelského rozhraní zaměřenou na specifickou oblast výsledné funkčnosti. Jedná se o panel se seznamem dostupných validačních pravidel, panel s právě aktivní sadou validačních pravidel, panel sloužící ke konfiguraci vybraného pravidla a panel s databázovými soubory určenými k validaci. Každá část má svůj vlastní controller, který zajišťuje funkčnost a definuje veřejné akce pro ovládání daného panelu. Ty se využívají jako reakce na stisknutí tlačítka v grafickém rozhraní, případně se přímo volají z jiných panelů.

5.3 Čtení databázových souborů

Čtení databázových souborů vytvořených v aplikaci Microsoft Access je zajištěno knihovnou Jackcess. Pro oddělení závislosti na této knihovně (a usnadnění případné výměny za jinou metodu čtení) bylo její použití „zabaleno“ do čtyř tříd. Vytvořený systém umožňuje čtení souborů ve formátech ACCDB a MDB.

Knihovna poskytuje plně objektové rozhraní pro práci s databázemi. Instance třídy `Database` představují jednotlivé otevřené databázové soubory, veškeré operace nad nimi, jako například získání seznamu tabulek či relací, pak probíhají pomocí metod.

Třída reprezentující databázový soubor

Každý načtený databázový soubor je v rámci systému reprezentován instancí třídy `Accdb`. Ta slouží zejména jako tzv. *továrna* pro následující tři třídy, které zprostředkovávají přístup k objektům uloženým v databázi. Dále třída obsahuje metodu pro porovnání s druhou databází, která se využívá pro vyhodnocení podobnosti a detekci plagiarismu mezi databázemi.

Repozitář tabulek uložených v databázi

Repozitáře v rámci programů obecně slouží pro oddělení přístupu k datům od vlastní logiky aplikace. Obvykle poskytují rozhraní pro získávání dat včetně možnosti vyhledávání konkrétním záznamů (resp. filtrování všech dostupných) a rozhraní pro ukládání nových či aktualizovaných záznamů.

Implementovaná třída `AccdbTableRepository` slouží jako repozitář zprostředkovávající uživatelské tabulky uložené v databázi. Při prvotní inicializaci repozitáře se z databáze získá seznam názvů všech tabulek, který je následně možné zavoláním příslušných metod filtrovat. Výsledný seznam lze získat metodou `getTables`. Implementovány jsou například následující možnosti filtrování:

- *Dle názvu tabulky*. Ponechána je pouze tabulka, jejíž název se přesně shoduje s parametrem filtru. V případě, že hledaná tabulka neexistuje, zůstane seznam nalezených tabulek prázdný.
- *Dle počtu sloupců vyhovujících parametrům*. Parametry jsou porovnávací operátor[15], požadovaný počet sloupců, název sloupce, datový typ a údaj, zda sloupec má či nemá být součástí primárního klíče. V případě, že je parametr názvu prázdný, pak se pro filtrování neuplatňuje; obdobně lze vynechat i další dva parametry.

  Interně tento filtr rovněž funguje jako malý repozitář. Pro každou tabulku v seznamu jsou nejprve nalezeny údaje o všech sloupcích, následně se provede filtrování dle názvu, poté dle datového typu a nakonec dle příslušnosti k primárnímu klíči. Pokud zbylý počet sloupců odpovídá požadovanému, je tabulka v seznamu ponechána. V opačném případně dojde k odstranění.
- *Nalezení rozkladových tabulek pro relace M:N*. Princip činnosti je podobný předchozímu filtru, pouze místo sloupců se pracuje s relacemi tabulky. Vyfiltrování rozkladových tabulek spočívá v ověření, že má daná tabulka cizí klíče tvořící vazbu 1:N s dalšími dvěma tabulkami (princip byl již zmíněn v analýze, viz kapitola 4.4).

[15] Podporované operátory jsou „rovná se“, „větší nebo rovno“ a „menší nebo rovno“.

Repozitář relací mezi tabulkami

Druhý repozitář, implementovaný třídou `AccdbRelationRepository`, slouží k získávání informací o relacích mezi tabulkami uloženými v databázi. Funguje na stejném principu jako repozitář tabulek, neboli získání všech relací při inicializaci a poté možnost jejich filtrování. K dispozici jsou následující filtry:

- *Vyhledání relací typu 1:1*. Ponechány jsou pouze relace označené knihovnou Jackcess jako typ 1:1. Knihovna interně využívá příznak uložený v objektu relace.
- *Vyhledání relací typu 1:N*. Funguje na stejném principu jako předchozí filtr, pouze v seznamu zachovává takové relace, které zmíněný příznak nemají. Do výsledného seznamu jsou zahrnuty i relace s rozkladovými tabulkami, které ve výsledku tvoří relace typu M:N, neboť *de facto* jsou relacemi typu 1:N.

Filtrování relací typu M:N není do tohoto repozitáře zahrnuto, neboť se jedná o vazbu, která vzniká až v důsledku aplikace jiných vazeb (konkrétně dvou relací 1:N). Pokud chceme takové relace nalézt, lze použít repozitář tabulek a filtr pro nalezení rozkladových tabulek, které právě relaci M:N zajišťují.

Repozitář dotazů uložených v databázi

Poslední implementovaný repozitář ve třídě `AccdbQueryRepository` slouží pro vyhledávání dotazů uložených v databázi. Opět je postaven na stejném principu jako předchozí dva repozitáře, ale je z nich nejjednodušší, neboť poskytuje pouze jednu „filtrační metodu“:

- *Vyhledání dotazu dle typu*. Ve výsledném seznamu jsou zachovány pouze ty dotazy, které přesně odpovídají typu zadanému parametrem. Informace o typu je opět zprostředkována knihovnou Jackcess.

5.4 Validace databáze

Kontrola databází s ohledem na splnění zadání, neboli *validace databází*, je realizována prostřednictvím různých validačních pravidel. Každé pravidlo obsahuje metodu `check`, která přejímá referenci na objekt reprezentující databází a vyhodnocuje, zda je pro danou databázi splněno. Konkrétní implementace této metody se tedy samozřejmě

u každého pravidla liší, navíc je výsledek ovlivněn i aktuální konfigurací pravidla. Ta je dána nastavením *vlastností*, které jsou součástí pravidla.

Jedním z implementovaných pravidel je *skupinové pravidlo*, do kterého lze „vnořit“ další pravidla. Pro vyhodnocení pravidla se poté ověřují i všechna vnořená. Pravidla tak ve výsledku tvoří stromovou strukturu a validace databáze ve výsledku spočívá ve spuštění kontroly v kořenovém pravidle.

Nástroj pro validaci poskytuje seznam pravidel, která nebyla během validace splněna. Validace databáze ovšem obvykle běží pouze do prvního pravidla, které „selže“, výstupem tedy bývá pouze toto jedno pravidlo.

5.4.1 Implementovaná validační pravidla

Implementováno bylo celkem deset různých validačních pravidel, přičemž všechny jsou potomky abstraktní třídy `Rule`, která definuje jejich základní podobu. Většina z nich byla navrhnuta v rámci analýzy řešení (viz kapitola 4.4), omezíme se proto jen na jejich výčet a možnosti konfigurace:

- *Skupinové pravidlo* umožňující zanoření dalších pravidel. Vyhodnocení pravidla je závislé na režimu, který lze zvolit: AND – je nutné splnit všechna vnořená pravidla. OR – je nutné splnit alespoň jedno vnořené pravidlo.
- *Kontrola existence tabulky dle názvu*. Nakonfigurovat lze název tabulky.
- *Kontrola počtu sloupců v každé tabulce*. Nakonfigurovat lze porovnávací operátor, požadovaný počet sloupců, filtr dle názvu sloupce, filtr dle datového typu a filtr dle příslušnosti sloupce k primárnímu klíči.
- *Kontrola počtu řádků v každé tabulce*. Nakonfigurovat lze porovnávací operátor a požadovaný počet řádků.
- *Kontrola počtu tabulek, které obsahují sloupec dle zadaných kritérií*. Nakonfigurovat lze porovnávací operátor, požadovaný počet tabulek, datový typ sloupce, název sloupce a příslušnost k primárnímu klíči.
- *Komplexní kontrola počtu tabulek*. Nakonfigurovat lze porovnávací operátor, požadovaný počet tabulek, filtr dle názvu sloupce, filtr dle počtu sloupců (opět lze nastavit porovnávací operátor a požadovaný počet) a filtr dle existence sloupce (dle názvu, datového typu a příslušnosti k primárnímu klíči).

Pravidlo umožňuje ověřit, že v databázi existuje požadovaný počet tabulek, které mají určitý počet sloupců (libovolných vlastností) a které zároveň obsahují sloupec požadovaných vlastností.

- *Kontrola počtu dotazů dle zadaného druhu*. Nakonfigurovat lze porovnávací operátor, požadovaný počet dotazů a druh dotazů. Lze vybrat jeden z devíti druhů.
- *Kontrola počtu relací typu 1:1*. Nakonfigurovat lze porovnávací operátor a požadovaný počet relací.
- *Kontrola počtu relací typu 1:N*. Nakonfigurovat lze porovnávací operátor a požadovaný počet relací.
- *Kontrola počtu relací typu M:N*. Nakonfigurovat lze porovnávací operátor a požadovaný počet relací.

Validační pravidla využívají implementovaných repozitářů, uplatňuje se zde tedy stejné chování jako u filtračních metod – tj. v případě nevyplnění vlastností *název*, *datový typ*, *typ dotazu*, apod. nebudou tyto vlastnosti pro filtrování použity.

5.4.2 Ukládání nakonfigurovaných pravidel do souborů

Systém uživatelům nabízí možnost ukládání a načítání nakonfigurovaných validačních pravidel z/do souborů. Tím je uživatelům umožněno zálohovat si jednotlivé konfigurace validace a používat je opakovaně. S ohledem na využití dané konfigurace aplikací adaptovanou pro validátor portálu ZČU mluvíme též o *exportování pravidel*.

Ukládání pravidel je implementováno jako tzv. *serializace* pravidel do souborů. Serializace obecně představuje proces převodu objektů do podoby, kterou lze uložit do paměti, souboru, nebo např. poslat po síti – striktně vzato lze říci, že se jedná o převod na posloupnost bytů. Opačným procesem je pak deserializace, jejímž cílem je zpětně rekonstruovat původní objekt (resp. vytvořit jeho identickou kopii).

V rámci realizovaného systému se pro serializaci využívá formát XML. Každé pravidlo je ve výstupním souboru zastoupenou jedním elementem, jehož atributy představují vlastnosti daného pravidla. Třída zajišťující serializaci přejímá pouze jedno pravidlo, pro uložení více pravidel se předpokládá využití skupinového pravidla.

Vnořená pravidla se zpracovávají rekurzivně a výsledný dokument tedy tvoří stromovou strukturu ve stejné podobě, jako jsou validační pravidla.

Uložení konfigurace validátoru spočívá v serializaci kořenového skupinového pravidla. Příklad konfigurace serializované do formátu XML je uveden v ukázce 5.1.

```
<?xml version="1.0" ?>
<GroupRule mode="AND">
    <AllTablesHaveRowsRule count_op="GTE" count="5" />
    <AllTablesHaveColumnsRule count_op="EQ" count="1" column_type="AUTO_NUMBER"
                              column_name="" column_primary="_ANY" />
    <CountRelations11Rule count_op="EQ" count="0" />
    <ExistsTableByNameRule name="student" />
</GroupRule>
```

Ukázka 5.1 – pět pravidel serializovaných do formátu XML.

5.5 Hledání podobností a detekce plagiarismu

Další důležitou částí systému je detekce plagiarismu mezi databázovými soubory vytvořenými v aplikaci Microsoft Access. Způsob detekce implementovaný v systému je založen na hledání stejných či podobných částí mezi všemi zadanými databázemi.

Třída zodpovědná za detekci plagiarismu přejímá seznam souborů určených ke kontrole. Samotná kontrola probíhá v několika krocích:

1. Zpracování všech databází – zavolání implementované metody, která vrací množinu „prvků podobnosti“.
2. Zařazení souborů do skupin dle nalezených podobností. V každé skupině tak je nejprve jeden soubor a postupně do ní mohou být zařazovány i další. Jakmile jsou ve skupině alespoň dva soubory, lze je označit jako vzájemně podobné.
3. Získání výsledků ve dvou různých podobách:
    - Mapa, kde klíčem je databázový soubor (dále „hlavní“) a hodnotou je třída spravující seznam jemu podobných souborů. Pro každý podobný soubor je dále k dispozici seznam podobností s hlavním souborem.
    - Mapa, kde klíčem je databázový soubor a hodnotou kolekce „prvků podobnosti“. Každý tento prvek pak v sobě nese seznam dalších souborů majících tento prvek.

Mezi databázemi se vyhledávají podobnosti pouze v rámci následujících metadat:

- Datum a čas vytvoření systémové tabulky `MSysDb`. Odpovídá datu vytvoření samotného databázového souboru.
- Datum a čas poslední úpravy systémové tabulky `MSysDb`. Odpovídá datu posledního uložení databáze.
- Datum a čas vytvoření systémové tabulky `Admin`. Odpovídá datu vytvoření relací mezi tabulkami v databázi.
- Datum a čas poslední úpravy systémové tabulky `Admin`. Odpovídá datu poslední úpravy relací mezi tabulkami v databázi.
- Údaje o autorovi databáze (jméno a název organizace). Vynechávají se údaje, které jsou v systémech často jako výchozí – např. databáze s názvem „Database“, jména autorů obsahující slovo „Windows“ nebo jména organizací obsahující „Microsoft“ (přidáno na základě testování).
- Grafické rozvržení relací mezi tabulkami. Způsob získání byl popsán v závěru kapitoly 4.5.

Na základě prvotního otestování bylo potvrzeno, že nalezení stejného grafického rozvržení relací mezi tabulkami je dostatečně průkazný prvek pro označení databáze jako plagiátu. Všechna ostatní pravidla způsobí pouze upozornění uživatele na podobnost mezi databázemi, má tedy možnost dodatečně zkontrolovat i další „podezřelé“ práce. Interně je rozlišení mezi plagiátem a podobným souborem docíleno ohodnocením nalezených prvků podobnosti, přičemž již zmíněné grafické rozvržení relací je ohodnoceno číslem 100, zbylé číslem 1. Aplikace s grafickým rozhraním pak označuje za plagiáty takové soubory, jejichž suma podobnosti je větší nebo rovna hodnotě 100.

V případě testování většího množství databází se může objevit některý prvek podobnosti, který způsobuje označení databáze jako podobné nebo plagiátu, ale může být očekávaný a není tedy žádoucí jej vyhodnocovat. Typicky může jít o databáze vytvořené v rámci jedné organizace, a tedy mající shodné údaje o autorovi. Dalším příkladem mohou být samostatné práce v rámci výuky, u nichž je součástí zadání předpřipravený databázový soubor. V tomto případě systém poskytuje možnost do

nástroje pro kontrolu předat seznam prvků podobnosti, které se mají ignorovat. Při dalších kontrolách pak již nejsou součástí výsledků.

5.6 Grafické uživatelské rozhraní

Pro hlavní aplikaci systému bylo vytvořeno grafického uživatelské rozhraní na platformě JavaFX. Bylo implementováno dle návrhu připraveného v rámci analýzy řešení (viz kapitola 4.6), tj. je rozděleno do čtyř částí:

- Panel *Knihovna pravidel* obsahuje seznam všech dostupných validačních pravidel. Je umístěn v levé části okna a zobrazuje pravidla automaticky načtená ze třídy `RulesRepository`.
- Panel *Aktivní pravidla* se všemi validačními pravidly, které jsou součástí aktuální konfigurace validátoru databází (dále označeny jako *aktivní pravidla*). Pravidla se zobrazují ve stromové struktuře zajištěné *skupinovými pravidly*. Ve výchozím stavu obsahuje panel jen jedno skupinové pravidlo nastavené do režimu AND.
- Panel *Detaily pravidla* zobrazuje všechny konfigurovatelné vlastnosti právě označeného aktivního pravidla.
- Panel *Databáze ke kontrole* obsahuje uživatelem přidané databáze určené ke kontrole s ohledem na splnění zadání a detekci plagiarismu. Každý přidaný soubor tvoří kořen stromu, vnořené listy pak představují různé dodatečné informace.

Pro kontrolu aplikace dále slouží nástrojová lišta a menu. Panel s aktivními pravidly lze ovládat tlačítky a akcemi pro načtení konfigurace ze souboru, uložení aktuální konfigurace do souboru a vytvoření nového souboru (neboli odstranění všech aktuálně aktivních pravidel).

Pokud chce uživatel přidat validační pravidlo z knihovny pravidel mezi aktivní, může tak učinit dvojklikem, nebo označením a stisknutím tlačítka na liště. Pokud chce naopak aktivní pravidlo odstranit, opět jej jen označí a stiskne příslušné tlačítko na liště. Aktivní pravidla mají v rámci panelu popisky skládající se z názvu a zjednodušeného popisu jejich konfigurace, uživatel tak vždy vidí, jak pravidla nakonfiguroval.

Po označení aktivního pravidla jsou zobrazeny všechny dostupné vlastnosti, které má uživatel možnost upravit. Vlastnosti jsou pro některá pravidla seskupena do souvisejících bloků, panel ale umožňuje toto seskupení vypnout a zobrazit vše v „ploché“ struktuře.

Přidat či odebrat databázové soubory ke kontrole lze opět tlačítky na liště či v menu. Přidání souborů je rovněž možné přetažením příslušných souborů z průzkumníka souborů na panel se soubory ke kontrole. Přidány jsou pouze podporované soubory ve formátech ACCDB a MDB. Po přidání do panelu má uživatel možnost otevřít kontextovou nabídku s možnostmi pro otevření souboru v asociované aplikaci (tedy typicky v aplikaci Microsoft Access) a pro otevření nadřazeného adresáře v průzkumníku souborů (na platformě Microsoft Windows bude navíc příslušný soubor zvýrazněn).

Kontrolu databázových souborů může uživatel spustit tlačítkem *Otestovat databáze*. Ve výchozím stavu zahrnuje jak validaci, tak detekci plagiarismu – tu může uživatel pomocí položky v menu vypnout či opět zapnout. Výsledkem kontroly je označení každého databázového souboru v panelu ikonou dle toho, zda databáze vyhovuje aktivním pravidlům, či nikoliv. V případě detekování plagiarismu je daný soubor označen červenou barvou, ikonou a textovým popiskem; jak již bylo uvedeno, systém může najít pouhou podobnost mezi databázemi, kterou ale nelze jednoznačně vyhodnotit jako plagiarismus – takové soubory jsou pak označeny pouze ikonou.

Souborům, které nesplňují aktivní pravidla nebo u nich byla detekována podobnost s jinými soubory (tj. i plagiarismus), jsou po kontrole přidány vnořené položky s informacemi o příčinách daného stavu. Tj. pravidlo, které při validaci „selhalo“, respektive seznam podobných databází a prvky podobnosti, které sdílejí. Zde má uživatel možnost konkrétní podobnosti pro následující kontroly skrýt prostřednictvím kontextového menu.

Pokud se během kontroly vyskytne chyba (například z důvodu poškozeného souboru), je uživateli upozorněn dialogovým oknem s popisem problému.

5.7 Adaptace pro validátor portálu ZČU

Cílem vytvářeného systému je i použití v rámci validátoru studentských prací portálu ZČU, který je v některých vyučovaných předmětech využívaný pro odevzdávání

samostatných prací vytvořených v aplikaci Microsoft Access. Smyslem je zajistit prvotní základní kontrolu „správnosti“ databázového souboru s ohledem na zadání práce, zejména ověřit formální správnost struktury databáze.

Jako způsob propojení validátoru s vytvářeným systémem byla vybrána již existující možnost spouštět vlastní Java programy v souborech formátu JAR přímo z validační domény. Pro tento účel byla vytvořena druhá aplikace mající pouze konzolové rozhraní. Aplikace je ovládána prostřednictvím dvou parametrů:

- Cesta k souboru s exportovanou konfigurací validačních pravidel ve formátu XML. Jedná se o soubor vzniklý uložením aktivních pravidel v hlavní aplikaci s grafickým rozhraním.
- Cesta k souboru ve formátu ACCDB nebo MDB. Typicky se jedná o soubor odevzdaný studentem prostřednictvím portálu ZČU.

Výstup programu je různý v závislosti na situaci:

- Na standardní výstup je vypsáno slovo `VALID` v případě, že validace proběhla úspěšně a zadaný databázový soubor *vyhovuje* validačním pravidlům. Návratovým kódem aplikace je 0.
- Na chybový výstup je vypsáno slovo `INVALID` – validace databázového souboru proběhla, nicméně zadaný soubor *nevyhovuje* validačním pravidlům. Dále je na chybový výstup vypsán důvod negativního vyhodnocení databáze oddělený jedním prázdným řádkem, viz ukázka 5.2, a aplikace použije návratový kód 100.
- Na chybový výstup je vypsáno slovo `ERROR`, pokud nastane během otevírání souborů nebo procesu validace chyba (je „vyhozena výjimka“). Může se tak stát v případě poškozených nebo nepodporovaných souborů. Na chybový výstup je na dalších řádcích vypsán tzv. *stack trace* odchycené výjimky a návratovým kódem je hodnota 101.

Validátor portálu ZČU soubor přijme v případě, že není nic vypsáno na chybový výstup a zároveň je návratovým kódem hodnota 0. V opačném případě je odevzdaná práce odmítnuta a v záznamu validace (který je dostupný studentům) je uveden výstup aplikace [29].

```
1 INVALID
2
3 Databáze nesplňuje pravidlo: Počet řádků v každé tabulce >= 5
```

Ukázka 5.2 – výstup konzolové aplikace v případě kontroly databáze, která nevyhovuje validačním pravidlům.

5.7.1 Konfigurace validátoru

Aby bylo možné v rámci validátoru studentských prací kontrolovat práce vytvořené v aplikaci Microsoft Access, je nutné jej nakonfigurovat tak, aby využíval připravenou aplikaci s konzolovým rozhraním.

Konfiguraci je nutné provést pro každé zadání samostatné práce[16], které chceme nechat v rámci odevzdávání prací na portálu ZČU automaticky kontrolovat. Navržený postup konfigurace lze shrnout do následujících kroků, předpokladem je již vytvořený soubor s validačními pravidly:

1. Vytvoření nové validační domény v rámci validátoru studentských prací.
2. Zkopírování aplikace (v archivu formátu JAR) a souboru s validačními pravidly do adresáře validační domény.
3. Vytvoření kroku validace, který zkopíruje tyto dva soubory do pracovního adresáře.
4. Vytvoření kroku validace, který spustí Java aplikaci ze souboru formátu JAR a jako parametry použije soubor s validačními pravidly a aktuálně studentem odevzdávaný soubor.
5. V detailním nastavení validační domény upravení hodnoty `max_file_size`, která určujte maximální povolenou velikost kontrolovaného souboru v kB[17].
6. Zvolení vytvořené validační domény v rámci *aplikace pro správu semestrálních prací, jejich odevzdávání a hodnocení* na portálu ZČU.

[16] V terminologii portálu ZČU se jedná o *okruhy* a *témata* prací.

[17] Testované práce vytvořené v rámci výuky mají v převážné většině velikost do 2 MB, s jediným extrémem 4,2 MB. Lze tedy předpokládat, že nastavení limitu na 10 MB bude dostačující.

Alternativně lze vytvořenou validační doménu nakonfigurovat přímou úpravu konfiguračních souboru umístěných v adresáři validační domény na validačním serveru:

- Soubor `domain.xml` obsahuje detailní nastavení včetně hodnoty `max_file_size` (viz krok 5 předchozího postupu).
- Soubor `webmodule.xml` obsahuje uživatelské jméno správce validační domény, její popis a dále definici všech kroků validace.

Tímto postupem lze urychlit konfiguraci nových validačních domén, neboť všechny obsahují totožné kroky validace a konfigurační soubory tak lze z velké části kopírovat. Stále je však nutné provést kroky 1 a 2, tedy vytvoření nové validační domény a zkopírování potřebných souborů na validační server.

Podoba souboru `webmodule.xml` domény nakonfigurované dle zmíněného postupu je vidět v rámci ukázky 5.3.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<domena>
    <userName>vkinkor</userName>
    <popis>Domena accessvalidator</popis>
    <krok nazev="_krok_1">
        <popis></popis>
        <podminka>
            <typ>vzdy</typ>
        </podminka>
        <akce>
            <typ>vlastniakce</typ>
            <param key="nazevakce">KopirovatSlozkyDoWorkdir</param>
            <param key="PARAM_FOLDER_NAMES">validator;</param>
        </akce>
    </krok>
    <krok nazev="_krok_2">
        <popis></popis>
        <podminka>
            <typ>vzdy</typ>
        </podminka>
        <akce>
            <typ>vlastniakce</typ>
            <param key="nazevakce">SpusteniJARProgramu</param>
            <param key="PARAM_NAME">validator/validator.jar</param>
            <param key="PARAM_ARGUMENTS">validator/rules.xml $inputFile</param>
            <param key="PARAM_CHECKBOX_OUTPUT">checked</param>
            <param key="PARAM_CHECKBOX_ERRPUT">checked</param>
        </akce>
    </krok>
</domena>
```

Ukázka 5.3 – soubor `webmodule.xml` validační domény nakonfigurované pro validaci prací vytvořených v aplikaci Microsoft Access.

6 Testování vytvořeného systému

Úkolem testování je ověřit robustnost a správnou funkčnost vyvíjeného systému. Tato kapitola se zabývá testováním vytvořeného systému skládajícího se z několika částí. Testování bylo prováděno převážně automatickými jednotkovými testy a doplněno ručním testování.

Pro realizaci jednotkových testů byl využit nástroj JUnit 5 včetně možnosti tvořit parametrizované testy. Dále byla využita knihovna Mockito 2 umožňující tvorbu tzv. *mock objektů*, které simulují chování reálného objektu, na kterém právě testovaná část závisí. Všechny jednotkové testy byly pojmenovávány a psány sebevysvětlujícím způsobem, komentáře tak byly připsány pouze v komplikovanějších úsecích kódu.

V rámci implementovaného systému je tzv. aplikační logika oddělena do samostatného modulu nazvaného *validator*. Po vyhotovení všech testů bylo pokryto 100 % řádků tohoto modulu. Popis způsobu a výsledky testování této i dalších částí systému je obsažen v následujících podkapitolách.

6.1 Čtení databázových souborů

„Základním kamenem“ celého systému je schopnost číst databázové soubory vytvořené v aplikaci Microsoft Access. K tomu je využita knihovna Jackcess v rámci čtyř implementovaných tříd.

Vývoj samotné knihovny je velmi dobře dokumentovaný a její kód je z velké části pokryt testy[18], není tedy nutné provádět detailní testování schopnosti číst data z databázových souborů.

Zmíněné čtyři třídy se podílejí na reprezentaci zpracovávaných databází v rámci systému, respektive na získávání informací a dat z databáze. Pro ně bylo vytvořeno celkem 52 jednotkových testů pokrývajících kód daných tříd. Otestována je jak základní funkčnost kódu, tak integrace s knihovnou Jackcess a je tedy v důsledku testována i základní funkčnost této knihovny na reálných databázových souborech.

[18] Viz WWW: `http://jackcess.sourceforge.net/jacoco/index.html`

6.2 Validace a validační pravidla

Jak již bylo v této práci mnohokrát uvedeno, validace či kontrola databázových souborů spočívá ve vyhodnocení validačních pravidel definovaných uživatelem, které ověřují strukturu a obsah dané databáze. Jedná se o nejdůležitější část systému.

V rámci práce se čistě na validaci podílí 19 různých tříd (nástroje obsluhující validaci, validační pravidel, vlastností pravidel a nástroj pro serializaci konfigurace). Pro ně bylo vytvořeno celkem 153 jednotkových testů, z nichž 7 komplexně testuje validaci pomocí reálného databázového souboru. Funkčnost validace byla rovněž testována na množině reálných dat, viz kapitola 6.6.

6.3 Detekce plagiarismu

Další důležitou částí testování je detekce plagiarismu. Zde je nutné testovat zvlášť správnost algoritmů, které slouží k hledání podobností – tedy zda systém dokáže nalézt prvky podobností mezi databázemi a správně vytvořit množiny podobných souborů. Za tímto účelem bylo vytvořeno 19 jednotkových testů zaměřených na algoritmickou správnost příslušných tříd.

Druhá část testování má za cíl ověřit, zda vybraný způsob detekce plagiarismu je spolehlivý a dostačující. Zde bylo přistoupeno k ručnímu testování v následující podobě:

1. Použití aplikace k otestování množiny databázových souborů, mezi nimiž bylo několik předem známých plagiátů (různé podobnosti).
2. Ověření, že byly plagiáty správně označeny.
3. Posouzení, zda případné další soubory označené jako plagiáty (předem neznámé) splňují kritéria pro dané označení – neboli že kopírují částečně či zcela jinou práci.

Pro testování byla použita sada deseti ručně připravených souborů a poté množina reálných dat. Testování potvrdilo, že vybraná metoda detekce plagiarismu je zcela vyhovující a rozdělení souborů na „plagiáty“ a „podobné“ je rozumný kompromis z hlediska spolehlivosti. Zjevné plagiáty (kopie souborů s jen málo pozměněným obsahem) byly správně označeny. Pokud došlo i k základním úpravám struktury

databáze (přejmenování tabulek, pozměnění relací), jsou takové soubory označeny jako podobné.

6.4 Grafické uživatelské rozhraní

Pro hlavní část systému bylo vytvořeno grafické uživatelské rozhraní na platformě JavaFX. Pro aplikace vytvořené na této platformě je možné vytvořit jednotkové testy za pomoci knihovny TestFX[19]. Během tvorby testovacích metod se však objevily problémy s dialogovými okny pro výběr souborů (bylo je možné pouze otevřít a zavřít) a zejména nepředvídatelnost chování panelu pro konfiguraci vlastností pravidel (náhodné přepínání kategorií, nemožnost úpravy hodnot). Od této knihovny bylo nakonec upuštěno.

Grafické rozhraní bylo ručně testováno pomocí připravených scénářů použití, během nichž se ověřovalo chování aplikace. Scénáře jsou koncipovány způsobem, kdy jakýkoliv rozpor mezi popsaným očekávaným chováním a realitou znamená chybu v aplikaci. Příklad scénáře pro otestování přidávání validačních pravidel mezi aktivní (upraveno do podoby srozumitelné pro čtenáře):

1. Spusťte aplikaci.
2. V části „Knihovna pravidel“ jsou vidět pojmenovaná pravidla. V části „Aktivní pravidla“ je umístěno pouze jedno pravidlo.
3. Dvojklikem na pravidlo v knihovně pravidel je pravidlo přidáno mezi aktivní.
4. Označením pravidla v knihovně a stisknutím tlačítka „Přidat pravidlo“ je vybrané pravidlo přidáno mezi aktivní.
5. Přidejte mezi aktivní pravidla 3 skupinová pravidla.
6. Označte prostřední přidané aktivní pravidlo.
7. Přidání dalšího libovolného pravidla nyní způsobí jeho zařazení „dovnitř“ skupinového pravidla.

[19] Viz WWW: `https://github.com/TestFX/TestFX/wiki`

6.5 Aplikace adaptovaná pro validátor portálu ZČU

Testování funkčnosti samotné aplikace s konzolovým rozhraním probíhalo nejprve lokálně, tedy spuštěním s různými databázovými soubory a soubory s pravidly a ověřování správnosti výsledků.

Následně bylo prováděno i testování v rámci testovacího validátoru studentských prací[20], který byl nakonfigurován dle postupu uvedeného v uživatelské příručce. Validátor poskytuje v rámci webového rozhraní možnost jednoduše testovat vytvořenou validační doménu nahráním testovacího souboru. Tímto způsobem byla ověřena funkčnost i v rámci validátoru.

Pro otestování v rámci portálu ZČU je nutné validační doménu přesunout z testovacího validačního server na tzv. ostrý server. To nebylo v rámci práce bohužel možné realizovat, funkčnost tedy není ověřena. Není nicméně znám žádný důvod, který by vedl k nefunkčnosti systému.

6.6 Testování systému na množině reálných dat

Funkčnost celého systému na kontrolu samostatných prací bylo možné vyzkoušet na množině reálných dat. K dispozici bylo 36 samostatných prací vytvořených studenty v rámci předmětu vyučovaného na Západočeské univerzitě v Plzni[21].

Práce byly použity pro ruční otestování systému zejména prostřednictvím vytvořené aplikace s grafickým rozhraním. Systém v průběhu závěrečného testování nevykazoval žádné problémy ani při použití vyššího počtu validačních pravidel (v řádu desítek) a rychlost reakcí na prováděné akce lze považovat za vyhovující.

Na základě profilování kódu bylo zjištěno největší zdržení v metodě hledající podobnosti mezi databázemi, které mělo dle původního návrhu kvadratickou asymptotickou složitost. Úpravou metody se podařilo snížit složitost algoritmu na lineární. Přesto i po optimalizaci je tato část časově nejnáročnější. Po vypnutí detekce plagiarismu způsobuje největší prodlevy samotné čtení databázových souborů.

Bylo provedeno měření výkonnosti při kontrole 1, 36 a 300 databázových souborů (množina byla vytvořena duplikací původních 36 souborů) se zapnoutou i vypnoutou

[20] Testovací validátor je dostupný na adrese `https://validator-test.zcu.cz/`
[21] Soubory studentských prací byly anonymizovány.

detekcí plagiarismu, výsledky jsou uvedeny v tabulce 6.1. Experimentálně bylo vyzkoušeno, že počet validačních pravidel na výsledný čas má spíše malý vliv. Zároveň není ani předpoklad, že by se výsledný systém používal s mnoha pravidly najednou, proto byla pro tento test použita jen sada 10 různých validačních pravidel. Měření probíhalo v rámci grafického uživatelského rozhraní, do výsledných časů je tedy zahrnuta i režie zpracování výsledků kontroly (tj. aktualizace prvků uživatelského rozhraní). Rychlý nárůst času při zapnuté detekci v rámci 300 souborů je způsoben právě touto režií z důvodu mnoha nalezených duplicit.

Počet souborů	**1**	**36**	**300**
Včetně detekce plagiarismu	0,01 s	0,11 s	1,22 s
Bez detekce plagiarismu	< 0,01 s	0,07 s	0,33 s

Tabulka 6.1 – výsledky měření výkonnosti systému během kontroly databázových souborů.

7 Závěr

Cílem této práce bylo navrhnout a implementovat systém pro kontrolu samostatných prací vytvořených v aplikaci Microsoft Access, tj. databázových souborů ve formátu ACCDB, a následně jej adaptovat pro použití v rámci validátoru studentských prací na portálu ZČU. Cílem kontroly je zejména formální ověření struktury databází s ohledem na splnění zadání, druhou částí pak je detekce plagiarismu mezi více pracemi.

Pro realizaci tohoto systému se bylo potřeba nejprve seznámit s formátem souborů ACCDB a možnostmi jeho čtení. Bylo nalezeno celkem šest různých metod, které byly porovnány a z nichž byla jedna vybrána pro následné využití v rámci systému. Dále byly prozkoumány možnosti validátoru na portálu ZČU a nalezena možnost propojení s vytvářeným systémem. Na základě zjištěných poznatků byl proveden návrh a implementace.

Představený systém se skládá z aplikace s přívětivým uživatelským rozhraním, kterou mohou uživatelé využít pro konfigurování kontroly samostatných prací a rovněž kontrolování samotné. Další částí systému je konzolová aplikace připravená pro použití v rámci validátoru portálu ZČU. Obě tyto části byly pečlivě otestovány a optimalizovány z hlediska výkonnosti. Během vývoje byl kladen důraz na jednoduchou možnost budoucího rozšíření o nová pravidla kontroly.

Aplikace byly vytvářeny s cílem poskytnout zejména vyučujícím jednoduchý způsob automatizace kontrol studentských prací. V případě napojení na validátor bude vytvořený systém přínosem i pro studenty, kteří se tak během elektronického odevzdávání svých prací okamžitě dozví, zda splnili všechny požadované náležitosti.

Zadání tedy bylo splněno v celém rozsahu a vytvořený systém je připravený pro použití v rámci výuky s ambicí zpříjemnit vyučujícím i studentům kontrolování resp. odevzdávání samostatných prací.

Reference

[1] ADAMSKI, Joseph J.; FINNEGAN, Kathy T. ; SCOLLARD, Sharon. *New perspectives on Microsoft Access 2013: comprehensive*. Stamford, CT: Cengage Learning, 2014. ISBN 978-1-285-09920-0.

[2] Introduction to the Access 2007 file format. *Microsoft Office help and training - Office Support*. [Online] [Citace: 20. 3. 2018]. Dostupné z: `https://support.office.com/en-us/article/Introduction-to-the-Access-2007-file-format-8cf93630-0b68-4a40-a13c-7528b9f074b6`

[3] Data types for Access desktop databases. *Microsoft Office help and training - Office Support*. [Online] [Citace: 22. 3. 2018]. Dostupné z: `https://support.office.com/en-us/article/data-types-for-access-desktop-databases-df2b83ba-cef6-436d-b679-3418f622e482`

[4] CONNOLLY, Thomas; BEGG, Carolyn. *Database Systems: A Practical Approach to Design, Implementation, and Management*. 6. Harlow: Pearson Education Limited, 2014. ISBN 978-1-292-06118-4.

[5] BRUNS, Brian. HACKING. *MDB Tools repository*. [Online] [Citace: 20. 4. 2018]. Dostupné z: `https://github.com/brianb/mdbtools/blob/master/HACKING`

[6] Microsoft Access ACCDB File Format Family. *Digital Preservation at the Library of Congress*. [Online] [Citace: 20. 3. 2018]. Dostupné z: `https://www.loc.gov/preservation/digital/formats/fdd/fdd000462.shtml`

[7] Which Access file format should I use? *Microsoft Office help and training - Office Support*. [Online] [Citace: 20. 3. 2018]. Dostupné z: `https://support.office.com/en-us/article/which-access-file-format-should-i-use-012d9ab3-d14c-479e-b617-be66f9070b41`

[8] KYLE, Geiger. *Inside ODBC*. Redmond, WA: Microsoft Press, 1995. ISBN 978-1556158155.

[9] ROFF, Jason T. *ADO: ActiveX Data Objects*. místo neznámé: O'Reilly Media, 2001. ISBN 9781491935576.

[10] Office Primary Interop Assemblies. *Microsoft Developer Network*. [Online] [Citace: 02. 04. 2017]. Dostupné z: `https://msdn.microsoft.com/en-us/library/15s06t57.aspx`

[11] WHITECHAPEL, Andrew. *Microsoft .NET Development for Microsoft Office*. Redmond, WA: Microsoft Press, 2005. ISBN 0-7356-2132-2.

[12] BRUNS, Brian. *MDB Tools repository*. [Online] [Citace: 20. 4. 2018]. Dostupné z: `https://github.com/brianb/mdbtools/`

[13] —. Access 2013 support. *MDB Tools repository*. [Online] [Citace: 20. 4. 2018]. Dostupné z: https://github.com/brianb/mdbtools/issues/77

[14] SMITH, Calvin R. mdbtools is being ported to java. *MDB Tools Discussion*. [Online] 2. 5. 2004 [Citace: 20. 4. 2018]. Dostupné z: https://sourceforge.net/p/mdbtools/discussion/6688/thread/a543445a/

[15] Open Microscopy Environment. *OME MDB Tools*. [Online] [Citace: 20. 4. 2018]. Dostupné z: https://github.com/ome/ome-mdbtools

[16] —. ColumnTest source code (ukázka použití). *OME MDB Tools*. [Online] [Citace: 20. 4. 2018]. Dostupné z: https://github.com/ome/ome-mdbtools/blob/master/src/main/java/mdbtools/tests/ColumnTest.java

[17] *Jackcess*. [Online] Health Market Science, 31. 3. 2018 [Citace: 20. 4. 2018]. Dostupné z: http://jackcess.sourceforge.net/

[18] Frequently Asked Questions. *Jackcess*. [Online] Health Market Science, 31. 3. 2018 [Citace: 20. 4. 2018]. Dostupné z: http://jackcess.sourceforge.net/faq.html

[19] Cookbook. *Jackcess*. [Online] Health Market Science, 31. 3. 2018 [Citace: 20. 4. 2018]. Dostupné z: http://jackcess.sourceforge.net/cookbook.html

[20] *Jackcess Encrypt*. [Online] Health Market Science, 9. 10. 2017 [Citace: 20. 4. 2018]. Dostupné z: http://jackcessencrypt.sourceforge.net/

[21] MAYDENE FISHER, Jon Ellis, Jonathan Bruce. *JDBC™ API Tutorial and Reference*. Boston, MA: Addison Wesley, 2003. ISBN 0-321-17384-8.

[22] ORACLE. JDBC-ODBC Bridge. *Java SE Documentation*. [Online] [Citace: 20. 4. 2018]. Dostupné z: https://docs.oracle.com/javase/7/docs/technotes/guides/jdbc/bridge.html

[23] AMADEI, Marco. *UCanAccess*. [Online] [Citace: 20. 4. 2018]. Dostupné z: http://ucanaccess.sourceforge.net/site.html

[24] Centrum informatizace a výpočetní techniky. *Referenční příručka portálového rozhraní IS/STAG*. Plzeň: Západočeská univerzita, 2009. ISBN 978-80-7043-807-7.

[25] —. Aplikace pro správu semestrálních prací, jejich odevzdávání a hodnocení. *IS/STAG - Helpcentrum*. [Online] Západočeská univerzita [Citace: 23. 4. 2018]. Dostupné z: https://is-stag.zcu.cz/napoveda/stag-v-portalu/spnew-studium_odevzdavani-praci.html

[26] HEROUT, Pavel. *Validační server pro studentské projekty*. [Online] [Interní dokument] [Citace: 20. 4. 2018]. Dostupné z: https://validator-test.zcu.cz/vs/auth/doc/doc/validacni-server-uzivatelsky-popis-2.pdf

[27] VALENTA, Lukáš; DUDOVÁ, Veronika. *Validační server - manuál*. [Online] [Citace: 20. 4. 2018]. Dostupné z: https://validator-test.zcu.cz/vs/auth/doc/index.html

[28] DUDOVÁ, Veronika. *Webová konfigurace validačního serveru*. Plzeň, 2010. Bakalářská

práce. Západočeská univerzita. Fakulta aplikovaných věd. Katedra informatiky a výpočetní techniky. Vedoucí práce Pavel HEROUT.

[29] *Testovací validační server pro studentské projekty*. [Online] [Citace: 20. 4. 2018]. Dostupné z: `https://validator-test.zcu.cz/`

[30] *Wiki - Validační server a jeho moduly - Redmine KIV ZČU*. [Online] [Citace: 20. 4. 2018]. Dostupné z: `https://students.kiv.zcu.cz:3443/projects/validator/wiki`

Přílohy

A Uživatelská příručka

Spuštění nástroje

Vytvořený nástroj, pojmenovaný *AccessValidator*, lze spustit pomocí souboru `configurator.jar`, například z příkazové řádky následujícím příkazem:

```
java -jar configurator.jar
```

Aplikace ke svému běhu vyžaduje JRE (Java SE Runtime Environment) ve verzi 8. Vzhledem k nekompatibilitě použité knihovny ControlsFX s novějšími verzemi JRE, byla připravena aplikace využívající novější verzi knihovny a lze ji spustit pomocí souboru `configurator_jre9.jar`. Tato verze vyžaduje JRE ve verzi 9 nebo vyšší.

Kompilace nástroje

Projekt obsahující nástroj byl spravován systémem Maven, který lze použít pro kompilaci. Je vyžadováno JDK (Java Development Kit) ve verzi 8 nebo vyšší. Pro zkompilovaný nástroj platí stejná omezení zmíněná v předchozím odstavci, tj. při použití JDK 8 lze nástroj spustit pouze v prostředí JRE 8. Kompilaci lze spustit následujícím příkazem:

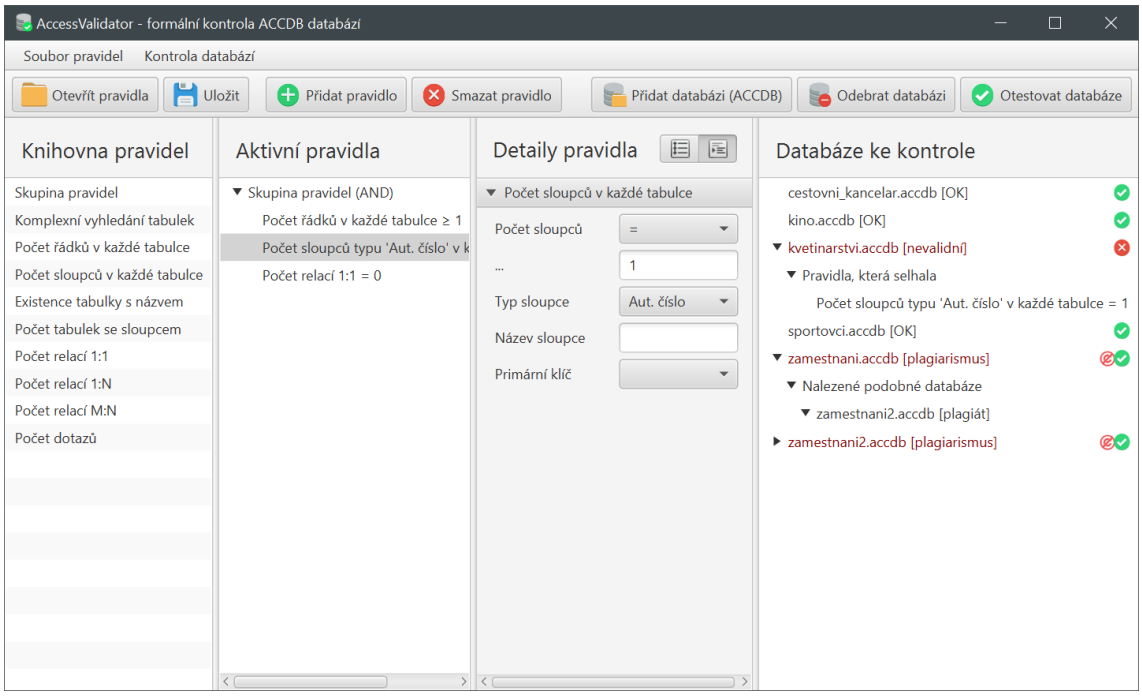
```
mvn package
```

Soubor JAR bude uložen v adresáři `validator/target/`. Současně bude vygenerován i JAR soubor s aplikací pro použití v rámci validátoru studentských prací do adresáře `configurator/target/`. Upozornění: v současné době je na validačním serveru dostupné JRE ve verzi 8, je tedy nutné aplikaci pro validátor kompilovat při použití JDK 8.

Obsluha nástroje

Po spuštění nástroje se uživateli zobrazí jednoduché okno rozdělené na čtyři části, viz obrázek A.1:

- *Knihovna pravidel* – obsahuje seznam validačních pravidel použitelných pro kontrolu databázových souborů.
- *Aktivní pravidla* – obsahuje seznam pravidel, která jsou aktuálně používány pro kontrolu databázových souborů.
- *Detaily pravidla* – slouží ke konfiguraci právě označeného aktivního pravidla.
- *Databáze ke kontrole* – obsahuje databázové soubory ve formátu ACCDB nebo MDB, které si uživatel přeje zkontrolovat.



Obrázek A.1 – náhled uživatelského rozhraní.

Typický postup práce s nástrojem je následující:

1. Přidání pravidel z knihovny mezi aktivní. Lze tak učinit tlačítkem na liště nebo dvojklikem na pravidlo.

2. Nakonfigurování všech aktivních pravidel, tj. označení a poté úprava parametrů v panelu *Detaily pravidla*.
3. Přidání databázových souborů ACCDB ke kontrole. Lze tak opět učitin tlačítkem na liště nebo přetažením příslušných souborů z průzkumníka souborů na panel *Databáze ke kontrole*.
4. Spuštění kontroly tlačítkem na liště *Otestovat databáze*.
5. Prozkoumání výsledků v panelu *Databáze ke kontrole*. Jednotlivé soubory jsou označeny ikonami (zelená pro vyhovující soubory, červený křížek pro nevyhovující, přeskrtnuté *C* pro detekované plagiáty; podrobnosti o kontrole jsou případně uvedeny jako podpoložky souboru).

Použití systému v rámci validátoru studentských prací

Aplikace adaptovaná pro použití v rámci validátoru je umístěna v souboru `validator.jar`. Jedná se o aplikace s konzolovým rozhraním spustitelnou obvyklým způsobem.

Postup konfigurace může být následující:

1. Vytvoření nové validační domény v rámci validátoru studentských prací, typicky prostřednictvím webového rozhraní.
2. Vytvoření a exportování validačních pravidel do souboru prostřednictvím hlavní aplikace s grafickým rozhraním. Soubor můžeme pojmenovat např. `rules.xml`.
3. Vytvoření adresáře `validator` v rámci adresáře vytvořené validační domény na validačním serveru.
4. Zkopírování vytvořené aplikace (`validator.jar`) a exportovaných pravidel (`rules.xml`) do adresáře vytvořeného v kroku 4.
5. Vytvoření dvou kroků validace v rámci validační domény:
   - Krok 1 nastavený následovně:
     Podmínka: *vždy*
     Vlastní akce: *Kopírovat složky do workdir*
     Seznam složek ke zkopírování: `validator;`

- Krok 2 nastavený následovně:
  Podmínka: *vždy*
  Vlastní akce: *Spustit JAR program*
  Spustitelný JAR soubor: `validator/validator.jar`
  Argumenty: `validator/rules.xml $inputFile`

6. V detailním nastavení validační domény upravení hodnoty `max_file_size`, která určujte maximální povolenou velikost kontrolovaného souboru v kB[22].
7. Zvolení vytvořené validační domény v rámci *aplikace pro správu semestrálních prací, jejich odevzdávání a hodnocení* na portálu ZČU.

Konfigurace validační domény je rovněž možná pomocí souboru `domain.xml` a `webmodule.xml`, viz kapitola 5.7.1 na straně 43.

Aplikaci `validator.jar` lze zkompilovat stejným způsobem, jako vytvořený nástroj, viz kapitola Kompilace této příručky.

[22] Testované práce vytvořené v rámci výuky mají v převážné většině velikost do 2 MB, s jediným extrémem 4,2 MB. Lze tedy předpokládat, že nastavení limitu na 10 MB bude dostačující.

B Obsah přiloženého média

Součástí práce je přiložené paměťové médium (DVD) obsahující tyto adresáře a soubory:

- `Binaries/` – adresář obsahující soubor `configurator.jar` a `configurator_jre9.jar` s implementovaným nástrojem (vyžaduje JRE 8, resp. JRE 9 nebo novější) a soubor `validator.jar` pro použití v rámci validátoru portálu studentských prací portálu ZČU.
- `Poster/` – adresář obsahující *poster* ve formátu PDF a PUB,
- `Project/` – adresář obsahující projekt vytvořeného systému,
- `Resources/` – adresář obsahující další podpůrné soubory (podklady vývoje, testovací sada souborů, text této práce v editovatelné podobě),
- `Kinkor_A16N0040P_DP.pdf` – text této práce ve formátu PDF,
- `readme.txt` – textový soubor obsahující popis struktury DVD.

Obsah přiloženého média (včetně aktuální verze nástroje) je možné najít též v repozitáři projektu v rámci služby GitHub na adrese:
`https://github.com/ikeblaster/access-validator/`