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

Diplomová práce

Aplikace s možností injekce chyb pro ověřování kvality testů

Plzeň 2018 Jiří Matyáš

Místo této strany bude zadání práce.

Prohlášení

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

V Plzni dne 16. května 2018

Jiří Matyáš

Abstract

There are currently some methods and software applications to check the quality of test methods. However, these are predominantly programs of relatively small size which are in some cases insufficient. The goal of this thesis is to design and implement a relatively sofisticate application that uses DBMS, which will simulate the functions of a real university information system. Furthermore, the way for purposely inserting of entities with error functionality into the application will be designed and implemented. In this thesis, several entities will be created with an error function to demonstrate the functionality of the solution. The application will be used for controlled experiments, verification of a test methods quality and educational purposes in the university environment.

Abstrakt

V současné době existují určité metody a softwarové aplikace pro ověřování kvality testovacích metod. Nicméně se většinou jedná o programy relativně malé velikosti a složitosti, které jsou v některých případech neprůkazné. Cílem této práce je návrh a implementace přiměřeně složité aplikace využívající SŘBD, která bude simulovat funkce reálného univerzitního informačního systému. Dále bude navržen a implementován způsob pro úmyslné vkládání entit s chybovou funkcionalitou do aplikace. Součástí práce bude vytvoření několika entit s chybovou funkcionalitou pro demonstraci funkčnosti řešení. Aplikace bude použita k řízeným experimentům, ověřování kvality testovacích metod a k edukačním účelům v univerzitním prostředí.

Poděkování

Rád bych poděkoval Doc. Ing. Pavlu Heroutovi, PhD. za odborné rady a cenné připomínky, které mi pomohly tuto diplomovou práci vypracovat.

Obsah

1 Úvod

V současné době se pro ověřování správnosti a kvality testovacích metod používají pokusné softwarové aplikace, ve kterých se nacházejí předem známé chyby. Na těchto aplikacích je možné ověřovat existující a případně nově vyvíjené testovací metody. Většinou se jedná o programy menší velikosti a často i triviální funkčnosti, které demonstrují jeden konkrétní typ chyby. Dále existují nástroje, které mohou záměrně vkládat uměle vytvořené chyby do hotových softwarových aplikací. Bohužel tyto nástroje provádějí většinou pouze drobnou syntaktickou změnu programu bez dalších modifikací. Proto by bylo vhodné vytvořit složitější aplikaci, do které by bylo možné úmyslně vkládat libovolné chyby.

Hlavním cílem práce je navrhnout a implementovat přiměřeně složitou softwarovou aplikaci se (semi)realistickým použitím, která bude sloužit pro ověřování správnosti a kvality testovacích metod jako SUT (System Under Test). Aplikace bude používat SŘBD jako hlavní zdroj dat. Jako datové vstupy a výstupy budou použity soubory v různých formátech.

Dále bude navržen a implementován způsob pro úmyslné vkládání entit s chybovou funkcionalitou do vytvořené aplikace. Z tohoto důvodu bude aplikaci možné sestavit z dílčích entit, které budou implementovat konkrétní funkce aplikace. Pro vytvořenou aplikaci bude existovat kolekce entit s korektní implementací, aby bylo možné sestavit výslednou aplikaci bez chyb.

Jako součást řešení bude připraveno několik vybraných chybových entit pro demonstraci funkčnosti implementovaného řešení. Uživatel si bude moci jednoduše vybrat, ze kterých entit bude chtít výslednou aplikaci sestavit. Implementace dalších chybových entit dle budoucích potřeb by měla být co nejsnazší. Součástí práce bude také vytvoření postupů, které budou implementované chybové entity ve výsledné aplikaci jednoznačně identifikovat.

Výsledná aplikace bude sloužit pro řízené experimenty, ověřování kvality testovacích metod a edukační účely v univerzitním prostředí. Také se očekává její budoucí rozšiřování o nové korektní a chybové funkce.

2 Způsoby ověřování kvality testovacích metod

2.1 Mutační testování

Mutační testování neboli mutační analýza je typ white-box testování software, který mění (mutuje) původní testovaný program tzv. System Under Test (SUT) nebo jeho části a uměle tak do jeho kódu zanáší chyby za účelem zjištění, zda testovací metoda chyby odhalí. Tento typ testování se nejvíce používá v kombinaci s jednotkovým testováním. Mutace (změny) v testovaných programech by neměly být rozsáhlé, aby příliš neovlivnily výslednou činnost celé aplikace [1, 2].

Mutační testování ověřuje zejména kvalitu testovacích metod a ne přímo testovaného softwarového produktu. Testovací metody by měly být dostatečně robustní, aby byly schopné odhalit mutací vložené drobné chyby [2, 3].

Postup mutačního testování je jednoduchý. Nejprve jsou z původního testovaného programu vytvořené různé verze s různými chybami (mutacemi) tzv. mutanti. Každý mutant by měl obsahovat jednu chybu (mutaci), kterou by testovací metody měly odhalit a demonstrovat tak svoji efektivitu a robustnost. Po provedeném testování se mohou porovnat výsledky testování původního programu a všech mutantů. V případě, že testovací metoda vrací odlišný výsledek pro původní program a pro mutanta, můžeme prohlásit, že testovací metoda uspěla a tzv. „zabila“ mutanta. V opačném případě mutant „přežil“ a je nutné testovací metodu vylepšit nebo upravit [1, 3].

Mutace je drobná syntaktická změna kódu, která je provedená v jednom příkazu testovaného programu. Mutace mohou být např. změny logických operátorů v podmínkách, záměna operandů příkazu za konstanty, záměna dvou operandů v podmínce nebo odstranění else větve podmínky. Každý vytvořený mutant by se měl odlišovat od původního programu právě jednou mutací [1, 3].

Pro ověřování kvality testovacích metod lze u mutačního testování využít mutačního skóre pro ohodnocení testovacích metod. Mutační skóre je definované jako procentuální úspěšnost zabitých mutantů z celkového počtu mutantů [3].

$$MutacniSkore = (pocetZabitychMutantu / celkovyPocetMutantu) \times 100$$

Výhoda mutačního testování spočívá v tom, že se jedná o robustní tech-

niku pro vytvoření kvalitních testovacích metod, které mají vysoké pokrytí různých chybových stavů testovaných programů. Mutační testování v testovaných programech odhaluje různé nejasnosti a neošetřené vstupy. Díky tomu je výsledný testovaný program spolehlivější a stabilnější [2, 3].

Na druhou stranu je mutační testování velice časově náročné. Pro kvalitní ověření testovacích metod je nutné vytvořit značné množství mutantů. Z tohoto důvodu existují nástroje pro automatizaci, které např. vytvářejí mutanty z původního testovaného programu. Vybrané nástroje pro automatizaci jsou popsané v následujících kapitolách. Z povahy mutačního testování, není možné tuto techniku použít pro black-box testování [2, 3].

2.1.1 Mutator

Mutator je vícejazyčný komerční nástroj, který provádí mutační analýzu pro programy napsané v jazycích Java, Ruby, JavaScript, PHP a dalších. Mutator je založený na technice selektivní mutace, díky které vytváří pouze mutace, které simulují skutečné chyby v softwarových aplikacích. Tyto chyby jsou vybrány zkušenými vývojáři softwaru. Spolupracuje s testovacími frameworky jako jsou JUnit v3 a v4, PHPUnit nebo Karma. Mutator provádí mutační testování také pro vícevláknový běh v jazyce Java. Je schopný detekovat problém uvíznutí (deadlock) a další chyby spojené s vícevláknovým během [4].

2.1.2 MuJava

μJava (MuJava, Mutation System for Java) je mutační systém pro programy napsané v jazyce Java. Tento nástroj je schopný automaticky generovat mutanty pro jednotlivé Java třídy nebo Java balíčky tříd. Používá dva typy mutačních operátorů. První typ obsahuje 24 mutačních operátorů na úrovni Java tříd, které se specializují na objektově orientované chyby v Java programech. Druhý typ obsahuje mutační operátory na úrovní metod v Java třídách, které jsou založené na selektivní mutaci připravené autory μJava systému. Po vytvoření mutantů je μJava schopná spustit množinu testovacích metod a vyhodnotit procento zabitých mutantů [5].

2.1.3 Bacterio

BACTERIO Mutation Test System je nástroj pro testování programů napsaných v jazyce Java založený na technice mutační analýzy. Bacterio umožňuje do jednoho mutanta vložit jednu nebo i více chyb. Nástroj také umožňuje

spustit testovací metody a určit procento zabitých mutantů (mutační skóre) a ověřit tak kvalitu testovacích metod. Bacterio vytváří různé verze, na kterých spouští a ověřuje testovací metody. Verze obsahuje jednotlivé mutanty (jeden mutant v jedné verzi) a také jejich kombinace (kombinace dvou mutantů v jedné verzi) [6].

2.2 Repositář SIR

SIR (Software-artifact Infrastructure Repository) je repositář software artefaktů, které slouží pro řízené experimenty určené pro programovou analýzu, ověřování technik testování software a vzdělávání v těchto oblastech [7].

Repositář obsahuje různé programy v různých verzích napsané v programovacích jazycích Java, C, C++ a C#. Součástí jsou také testovací sady, chybová vstupní data, podpůrné skripty a dokumentace popisující, jak objekty repositáře používat v experimentech. Dokumentace také obsahuje informace o procesech výběru, uspořádání a vylepšování artefaktů nacházejících se v repositáři [7].

Artefakty repositáře většinou reprezentují programy menších velikostí nebo knihovny, které demonstrují nějakou konkrétní chybu nebo množinu chyb [7].

2.3 Shrnutí

Ačkoliv je z předchozích částí zřejmé, že existují různé SUT se zavedenou úmyslnou chybovou funkcionalitou nepodařilo se nalézt ani jeden SUT (volně použitelný), který by splňoval požadavky kladené v této diplomové práci.

3 Typické funkční chyby

V následujících kapitolách je uveden výběr typických funkčních a bezpečnostních chyb (včetně popisů), které jsou součástí seznamu CWE (Common Weakness Enumeration) [8]. CWE je seznam typů slabých míst softwarových aplikací, které mohou vzniknout v architekturách, při návrhu nebo při tvorbě kódu. Tento seznam slouží pro popis v současné době 716 známých slabin v softwaru, dělených do mnoha kategorií. Jednotlivé slabiny jsou ohodnocené podle známého postupu, který je přístupný k nahlédnutí na stránkách CWE. Tato ohodnocení mohou sloužit například k prioritizaci řešených chyb při vývoji nebo údržbě software [8].

CWE vytváří komunita specialistů a výzkumných pracovníků z různých organizací, kteří se zaměřují na oblast zajištění kvality software. Tito lidé využívají své praktické a odborné znalosti pro rozšiřování CWE seznamu slabých míst a chyb v software. V tomto seznamu vytvářejí definice jednotlivým prvkům a doplňují ho o prvky nové [8].

Většině těchto chyb v softwaru lze předejít kvalitním testováním, které by mělo zajišťovat a zároveň ověřovat kvalitu vyvíjeného softwaru. Chyby ve vytvářených softwarech by měli být odstraňovány již ve fázi jejich návrhu, kdy je cena na odstranění chyby menší, než např. ve fázi vydávání. Existuje mnoho technik testování softwaru, které se používají v závislosti na subjektu, který software testuje, typu testovaného softwaru a fázi vývoje, ve kterém se testovaný software nachází [9].

Ve fázi implementace softwaru je většinou nejvíce používána technika jednotkového testování, které testuje dílčí části (jednotky) aplikace (např. metody v Java třídách). Jednotkový test by měl testovat pouze jednu konkrétní funkční část softwaru a být nezávislý na ostatních testech. Tyto testy odhalují podstatnou část chyb, které se nacházejí v kódu výsledného softwarového produktu [10].

3.1 Chyby zpracování dat

Chyby v této kapitole se typicky vyskytují ve funkcionalitách zpracovávání různých dat.

Citlivá data v chybových výpisech (CWE-209)

Tato chyba se projevuje v softwarových aplikacích, které generují chybové výpisy včetně citlivých dat. Tyto výpisy se mohou vyskytovat v uživatelském rozhraní nebo také v logu aplikace. Citlivá data se mohou týkat prostředí, ve kterém aplikace běží, nebo uživatelů, kteří aplikaci používají (např. hesla).

Procházení bufferu se špatnou délkou (CWE-805)

Chyba může nastat při sekvenčním procházení bufferu se špatně zadaným parametrem celkové velikosti bufferu. Tato chyba způsobí pokus o přístup do paměti mimo buffer. Může dojít k přetečení bufferu a tím k neočekávaným výsledkům nebo dokonce pádu programu.

Nevhodné kódování výstupu (CWE-838)

Jedna komponenta může generovat vstup druhé komponentě, ale používat jiné kódování než druhá komponenta očekává. Druhá komponenta tak může zpracovávat jiný obsah, než který první komponenta odesílala.

Přetečení bufferu (CWE-120)

Program kopíruje vstupní buffer do nového výstupního bufferu bez kontroly velikosti obou bufferů. V případě, že velikost vstupního bufferu je větší než velikost výstupního bufferu, dojde k přetečení bufferu. Tato chyba může také nastat v případě, že se program snaží do bufferu vložit více dat, než je schopný pojmout.

Používání externě řízeného formátu řetězce (CWE-134)

Program používá funkci, která přijímá formát řetězce jako vstup, ale formát řetězce pochází z externího zdroje. V případě, že metoda přijme řetězec s jiným formátem než očekává, může dojít k chybě (např. přetečení bufferu).

SQL Injection (CWE-89)

Při vytváření SQL dotazu na základě dat získaných na vstupu (např. z uživatelského rozhraní) může docházet ke špatnému nebo žádnému ošetřování speciálních elementů, které by mohly ovlivnit záměr SQL dotazu. Takto vytvořený SQL dotaz by mohl provést potenciálně škodlivou akci nad daty v databázi.

3.2 Bezpečnostní funkce

Chyby v této kategorii se týkají bezpečnosti software, které lze rozdělit na autentizaci, autorizaci, kryptografii a řízení práv.

Používání nedostatečně náhodných čísel (CWE-330)

Program používá nedostatečně náhodná čísla v bezpečnostních procesech, které jsou založené na nepredikovatelných číslech. V případě, že se v takových procesech používají predikovatelná čísla, mohou být bezpečnostní procesy útočníkem prolomeny, protože útočník může tato predikovatelná čísla uhodnout.

Chybějící autentizace pro kritické funkce (CWE-306)

Program neprovádí autentizaci u kritických funkcí, které vyžadují jednoznačnou identifikaci uživatele nebo spotřebovávají značné množství zdrojů.

Nesprávné omezení častých pokusů o autentizaci (CWE-307)

Software neošetřuje správně případy častých požadavků o autentizaci, které selhávají v krátkých časových intervalech. Software se tak stává velice náchylný na útoky hrubou silou.

Chybějící šifrování citlivých dat (CWE-311)

Program nešifruje citlivé nebo důležité informace při jejich přenosu nebo persistenci. Při přístupu útočníka k těmto datům mohou být tato data jednoduše zneužita.

Používání prolomeného nebo nebezpečného šifrovacího algoritmu (CWE-327)

Program používá prolomený nebo nebezpečný šifrovací algoritmus. Tím může dojít k neoprávněnému přístupu ke chráněným citlivým nebo důležitým informacím.

Použití jednosměrného hashe bez soli (CWE-759)

Program používá jednosměrnou šifrovací hash funkci např. pro uložení hesla v databázi bez použití soli jako doplňující vstup pro tuto funkci. Pro potenciálního útočníka je jednodušší predikovat hodnotu hashe hesla např. pomocí slovníkového útoku.

Uvedení pověřovací údaje (Credentials) v kódu (CWE-798)

Program obsahuje pověřovací údaje (např. přihlašovací jméno a heslo) „napevno“ v kódu. Tyto údaje používá pro svou vlastní vnitřní autentizaci, autentizaci při komunikaci s dalšími programy nebo šifrování interních dat. V případě, že se útočník dostane k těmto informacím, může je zneužít pro nekorektní činnost.

Chybějící autorizace (CWE-862)

V software chybí autorizační proces při pokusu uživatele o přístupu ke zdrojům nebo k provedení akce. Uživatel tak může přistupovat k datům nebo provádět akce, ke kterým nemusí mít oprávnění.

Nesprávná autorizace (CWE-863)

Software obsahuje autorizační proces při pokusu uživatele o přístup ke zdrojům nebo k provedení akce, ale tento proces je nesprávný nebo obsahuje chyby. Potenciální útočník může takový autorizační proces obejít.

Používání funkcionality z nedůvěryhodné řídící oblasti (CWE-829)

Software používá nedůvěryhodnou spustitelnou funkcionalitu, která může provádět škodlivou činnost (např. knihovny třetích stran). Tato chyba se vyskytuje např. ve webových aplikacích, ve kterých jsou použity JavaScriptové knihovny třetích stran. Tyto knihovny mohou vkládat malware nebo zneužívat přístup k informacím nebo k provádění neoprávněných operací (např. útok Cross-site Scripting pro odcizení uživatelských cookies nebo pro přesměrování na stránku s malwarem).

3.3 Webové Problémy

Chyby v této kategorii se týkají webových technologií.

Přesměrování na URL nedůvěryhodné stránky (CWE-601)

Web aplikace přijímá uživatelské vstupy pro určení URL odkazu k přesměrování uživatele. Přesměrování může vést na nedůvěryhodné stránky, které mohou sloužit například pro phishingové útoky.

Cross-site scripting – XSS (CWE-79)

Software neošetřuje uživatelské vstupy, které jsou použité pro výstupy k vytvoření webové stránky. Na tuto webovou stránku mají přístup ostatní uživatelé aplikace. Webová aplikace může např. vytvářet stránky na základě vstupních dat uživatele, které mohou obsahovat nedůvěryhodný obsah. Tento nedůvěryhodný obsah může být spuštěn v prohlížeči uživatele a způsobit nežádoucí akce (např. pomocí JavaScriptu).

Cross-Site Request Forgery – CSRF (CWE-352)

Tento útok se týká webových aplikací, které dostatečně neverifikují, zda korektně vytvořený požadavek na nějakou operace je skutečně požadován uživatelem, který požadavek odeslal. Útok využívá akcí uživatelů, kteří jsou přihlášeni k nějaké aplikaci, pro odeslání požadavků na tuto aplikaci nezamýšlených přihlášeným uživatelem.

3.4 Obsluhy chyb

V této kategorii se nachází chyba spojená s nesprávnou obsluhou chybových stavů v softwarových aplikacích.

Neošetřené nahrávání souborů nebezpečného typu (CWE-434)

Software umožňuje uživatelům nahrávat a odesílat soubory nebezpečného typu, které mohou provádět nebezpečnou činnost uvnitř softwaru. Tento soubor může být nahráván útočníkem, který chce v softwaru provést nějakou škodlivou akci.

3.5 Špatné praktiky psaní kódu

Tato kategorie obsahuje slabiny související s praktikami programování, které se považují za nebezpečné nebo zvyšují zranitelnost výsledných aplikací. Nejedná se přímo o chyby softwaru, ale o nedostatky vytvořené při nedbalém vývoji.

Použití potenciálně nebezpečné funkce (CWE-676)

Program bude volat potenciálně nebezpečnou funkci, která by mohla vést ke zranitelnosti celého programu (např. zastaralé funkce). Může se jednat také o špatně použitou bezpečnou funkci.

Chybějící inicializace proměnné (CWE-456)

Program neinicializuje důležité proměnné, které tak mohou obsahovat neočekávané hodnoty. Neinicializované proměnné mohou při běhu programu způsobovat neočekávané hodnoty výsledků při použití takových proměnných.

3.6 Problémy s ukazateli (pointery)

Chyby v této kategorii jsou spojené s nesprávným použití ukazatelů (pointerů).

Dereference NULL pointeru (CWE-476)

NULL pointer dereference nastává při pokusu dereference pointeru s hodnotou NULL, který program považuje za nenullový. Jedná se o velmi častou chybu, která může způsobovat pád programu.

Dereference nedůvěryhodného pointeru (CWE-822)

Program může získat nedůvěryhodnou hodnotu pointeru, kterou se může pokusit dereferencovat. Útočník se může pokusit přistoupit k paměti, ke které by neměl mít přístup, a provést zde nějakou škodlivou činnost. Může například přistoupit do paměti, kde je připravený škodlivý kód, který se bude útočník snažit spustit.

Dereference neplatného pointeru (CWE-825)

Program se pokouší dereferencovat pointer ukazující do paměti, která byla dříve validní, ale při dereferencování již není. Program může provádět několik kroků, při kterých pracuje s pointrem ukazujícím do paměti. V některém kroku může být paměť na kterou pointer ukazuje uvolněna. Při dalším pokusu pointer dereferencovat vzniká chyba nekorektního přístupu do paměti.

3.7 Numerické chyby

Chyby v této kapitole souvisí s nesprávným výpočtem nebo konverzí čísel.

Nesprávné ošetření indexu pole (CWE-129)

Program používá nevalidovaný nebo špatně validovaný vstup pro výpočet indexu nebo přímo jako index do pole. Výsledný index nemusí být validní

pro referencování v konkrétním poli. Může dojít k indexování pole mimo jeho hranice.

Nesprávná konverze mezi číselnými typy (CWE-681)

V případě konverze jednoho číselného datového typu na druhý (např. long na integer) může dojít k neočekávaným výsledkům konverze. V případě použití výsledků takovýchto operací v citlivých nebo důležitých výpočtech může dojít k chybným výsledkům.

Nesprávný výpočet velikosti bufferu (CWE-131)

Program nesprávně počítá velikost bufferu pro jeho alokaci v paměti. V tomto případě může následně dojít k přetečení bufferu.

Přetečení Integeru (CWE-190)

K přetečení celočíselného datového typu dochází při výpočtu, kdy se předpokládá vyšší hodnota výsledku než původní hodnoty, ze kterých se výpočet provádí. To může mít za důsledek další chyby v závislosti na kontextu, kde se výpočet provádí (např. při správě zdrojů nebo řízení toku programu).

3.8 Chyby obchodní logiky

V této kategorii jsou známé a dobře popsané chyby reprezentující základní problémy, které umožňují potenciálnímu útočníkovi napadat obchodní logiku (business logic), která může ovlivňovat celou aplikaci.

Přidělování zdrojů bez omezení (CWE-770)

Program přiděluje, na žádost uživatele, znovupoužitelné zdroje (např. socket při navazování spojení) bez použití omezení počtu přidělených zdrojů.

Nedodržení pořadí vykonávaných funkcí (CWE-696)

Software provádí více souvisejících funkcí pro provedení komplexnější operace, ale neošetřuje, zda jsou funkce provedeny v požadovaném pořadí. V případě provedení funkcí v nesprávném pořadí se software může dostat do chybového nebo nestabilního stavu.

4 Stavba aplikace z dílčích entit

Tato kapitola je zaměřena na stavbu obecné aplikace z dílčích entit, které v sobě zapouzdřují určitou funkcionalitu.

4.1 Komponenty

Softwarová komponenta je základní stavební jednotka komponentově orientovaného softwarového inženýrství (CBSE – Component-Based Software Engineering), která slouží zejména ke kompozici větších celků. Tyto celky mohou být komplexnější komponenty nebo funkční aplikace. Komponenty mohou také sloužit k vyšší abstrakci jednotlivých částí aplikace řešící určitý problém [11].

Použití komponent při návrhu architektury aplikace by mělo být v souladu s principem oddělení zodpovědností (Separation of Concerns – SoC), což znamená, že každá komponenta by měla implementovat pouze funkčnost, ke které je určena. Implementované funkčnosti by se měly v rámci více komponent co nejméně překrývat a být tudíž modulární [11].

Komponenta by měla zapouzdřovat svojí implementaci a pro komunikaci s okolím by měla poskytovat rozhraní s kompletně popsaným kontraktem. V tomto rozhraní definuje služby, které poskytuje a služby, které požaduje pro svojí korektní funkčnost. Kontrakt přesně specifikuje, jak se má komponenta používat [11].

Výhoda použití komponent spočívá zejména v jejich znovupoužitelnosti. Díky tomu je možné existující komponenty nezávisle použít vícekrát v různých aplikacích a skládat aplikace z hotových dílů. Tato výhoda značně urychluje vývoj a také samotné testování a ověřování kvality aplikací [11].

Za nevýhodu komponentového přístupu k tvorbě aplikací můžeme považovat vyšší režii při vývoji a údržbě. Pro snížení této režie je možné použít hotové frameworky, které podporují komponentově orientovaný přístup. Při implementaci nové funkčnosti je nutné zachovat starší v rámci zpětné kompatibility [11].

4.1.1 Spring framework

Spring framework je aktuálně nejpoužívanější open-source aplikační framework napsaný v jazyce Java, který slouží primárně pro vývoj J2EE aplikací. Celý framework je rozdělený do modulů, které jsou zaměřené na určitou oblast použití. Programátoři aplikací si tak mohou vybrat pouze moduly, které chtějí použít v jejich aplikacích [12]. Více o Springu viz kapitola 5.

Jádrem Spring frameworku je core container, který obsahuje konfigurační model a mechanismus pro vkládání závislostí (DI – dependency injection). Mechanismus DI je postaven na využití návrhového vzoru přenesení zodpovědnosti (IoC – Inversion of Control). V případě DI to znamená, že Spring převezme zodpovědnost za vytvoření a vložení závislostí, kde je programátor definuje. Tuto funkci obstarává Spring IoC container [12].

Uvedené funkce Spring frameworku umožňují vytvořit robustní komponentově orientovanou aplikaci. Spring zajistí vzájemné vazby mezi jednotlivými komponentami. Komponenty pouze definují své závislosti a Spring IoC container provede jejich inicializaci a vložení. To má za důsledek rozvolnění vztahů mezi komponentami [12].

4.1.2 OSGi

OSGi (Open Services Gateway initiative) je aplikační framework, který definuje specifikace dynamického komponentového systému pro aplikace napsané v jazyce Java. Tyto specifikace umožňují vytvářet modely aplikací, které jsou sestavené z komponent. Pro definování komponent je zavedený pojem balíček, tzv. *bundle*. Komponenty, které jsou v OSGi označovány jako moduly, jsou zabaleny do těchto balíčků a komunikovat mezi sebou mohou pomocí tzv. nano-služeb. Nano-služby jsou v podstatě objekty, které jsou sdílené mezi komponentami. Tato infrastruktura je součástí OSGi implementace [13].

OSGi umožňuje instalaci a odinstalaci těchto balíčků za běhu aplikace, což je jeho velká výhoda. Balíčky se vytvářejí jako běžné JAR soubory se speciální hlavičkou v souboru manifest. Skutečná implementace jednotlivých komponent je tedy skrytá a viditelné je tak pouze rozhraní. Rozhraní definuje nano-služby, které komponenta požaduje a poskytuje [13].

4.1.3 Enterprise JavaBeans

Enterprise JavaBeans (EJB) je technologie a zároveň specifikace, která umožňuje modulární konstrukci podnikových aplikací. Enterprise JavaBean je

také označení pro komponentu na straně serveru, která zapouzdřuje business logiku určité podnikové aplikace. EJB vychází z jednoduchých Java Beans a jejich vlastností a je zároveň podmnožinou Java EE specifikací. V současnosti je aktuální verzí EJB 3, která byla vydána v roce 2004 [14].

EJB komponenty se vytvářejí a udržují v EJB containeru, který poskytuje prostředí pro běh EJB. Tento kontejner se stará také o Dependenci injection v rámci jednotlivých EJB komponent uvnitř kontejneru. Kontejner také obstarává komunikaci se vzdáleným klientem a aplikací [14].

4.2 Maven moduly

Maven je nástroj pro správu a automatizací procesu sestavení (buildování) aplikací. Projekty těchto aplikací popisuje Maven pomocí souboru POM (Project Object Model) [15].

Velice rozsáhlé a složité Maven projekty je možné rozdělit do více menších modulů, které lze provázat závislostmi. Moduly jsou v podstatě další Maven projekty, které je možné nahrazovat jinými, verzovat a znovupoužívat v dalších projektech [15].

Existují dva způsoby jak tvořit hierarchickou strukturu modulů v rámci jednoho projektu. První možností je definovat rodiče v POM souboru potomka. Díky tomu je následně možné používat závislosti a pluginy rodiče v potomkovi. Druhou možností je agregace, což je opačný přístup oproti předchozímu. Místo určení rodiče v potomkovi, rodič určí jeho potomky ve svém POM souboru. V tomto případě musí rodičovský modul vědět, ze kterých potomků se skládá [15].

Pro Maven projekty se stejnou nebo podobnou konfigurací je možné použít první způsob, kdy v dalším nezávislém Maven projektu bude nastavena tato konfigurace. Projekty pak specifikují tento projekt jako svého rodiče a budou tímto způsobem sdílet stejnou konfiguraci. Druhý způsob může být použit v případě, že je nutné spouštět Maven příkazy pro celou skupinu Maven projektů. Tyto projekty budou specifikované v rodičovském projektu jako jeho moduly. V případě spuštění Maven příkazu pro rodiče, se tento příkaz vykoná i pro všechny jeho potomky [15].

Mechanismus, který spravuje a sestavuje projekt složený z více modulů, se nazývá Reactor. Ten shromažďuje všechny dostupné Maven moduly k výslednému sestavení projektu. Nalezené moduly řadí pro korektní provedení procesu sestavení, který nakonec sám provede. Jelikož moduly v rámci jednoho projektu mohou být závislé na ostatních, je důležité aby je Reactor správně seřadil pro zajištění bezproblémového sestavení projektu [15].

5 Spring Framework

Spring je open source aplikační framework vytvořený Rodem Johnsonem. Je zaměřený na vývoj širokého spektra různých Java EE (J2EE) aplikací, který je díky němu mnohem rychlejší a snadnější. Framework definuje další různé dílčí frameworky, komponenty a další nástroje, které se specializují na odlišné odvětví vývoje enterprise aplikací [16, 17].

5.1 Jádro

Java EE aplikace se skládají z částí o různých velikostech tvořených jednotlivými objekty, které navzájem spolupracují za účelem plnění funkcí aplikace. Každý objekt, který ke své činnosti potřebuje další objekty, je na těchto objektech závislý. Závislosti mohou být implementovány různě těsnými vazbami s tím, že je žádoucí použít co nejvolnější vazbu. A právě to je jeden z důležitých cílů Spring frameworku [16].

Jádro Spring frameworku je postaveno na návrhovém vzoru Inversion of Control (IoC, obrácené řízení), který slouží pro odstranění těsných vazeb jednotlivých objektů se závislostmi. Návrhový vzor je postaven na principu přesunutí zodpovědnosti v objektově orientovaném návrhu pro dosažení volnějších vazeb. IoC je ve Springu implementován jako důmyslný systém pro správu bean objektů v IoC kontejneru (více viz kapitola 5.2.1) [16, 17].

Dependency Injection (DI) je konkrétní technika IoC, která je zodpovědná za vkládání závislostí do jednotlivých komponent. V případě závislosti jedné komponenty na druhé, DI zajistí za běhu aplikace vytvoření instance druhé komponenty a vložení této závislosti v první komponentě [16, 17].

5.2 Moduly

Spring framework obsahuje různé funkce, které jsou organizované v modulech. Tyto moduly tvoří několik skupin, které pokrývají různé oblasti při tvorbě Java enterprise aplikací. Skupiny tvoří např. moduly implementující funkcionalitu Core (IoC) Containeru, přístupu k datům, integraci, webových aplikací, aspektově orientovaného programování, testování atd [16].

5.2.1 Core Container

Nejdůležitější částí této kategorie modulů je Spring Inversion of Control kontejner (Core Container), který implementuje návrhový vzor Inversion of Control, konkrétně techniku vkládání závislostí (DI). V procesu vkládání závislostí definují objekty jejich závislosti jako argumenty jejich konstruktoru, tovární metody nebo pouze jako jejich atributy, které jsou nastaveny až po inicializaci objektu (setter metodami). Závislosti jsou další objekty, které jsou nezbytné pro korektní funkčnost objektů definující závislosti. IoC kontejner vkládá tyto závislosti při procesu vytváření bean, což jsou objekty v kontejneru. O inicializaci bean a jejich závislostí se tak stará IoC kontejner namísto samotné beany (resp. programátora, který beanu programuje), proto byl zvolen název Inversion of Control (obrácené řízení) [16].

Důležitou částí IoC kontejneru je *BeanFactory* rozhraní továrny bean, která poskytuje konfigurační mechanismus a základní funkce pro správu všech typů bean. *BeanFactory* obsahuje metody pro přístup ke komponentám aplikace (beanám), které obsahuje IoC kontejner. Poskytuje základní klientský pohled bean kontejneru. Objekty, které implementují toto rozhraní udržují několik definicí bean, které jsou identifikovatelné podle jejich jména. Jméno bean musí být v rámci aplikace unikátní. V závislosti na definici bean budou implementace *BeanFactory* vracet buď nezávislé instance bean (návrhový vzor prototyp) nebo jednu sdílenou instanci beany (alternativa návrhového vzoru jedináček). Který typ instance se použije závisí na konfiguraci implementované továrny [16].

Rozhraní *BeanFactory* rozšiřuje *ApplicationContext* rozhraní, které přidává další funkce pro enterprise aplikace jako např. různé konfigurace používané v aplikaci. *ApplicationContext* poskytuje za běhu aplikace pouze read-only operace pro práci s IoC kontejnerem. Toto rozhraní umožňuje načíst celou konfiguraci včetně definice bean ze souboru. Také poskytuje schopnost zveřejňovat události (event) registrovaným posluchačům (listener) [16].

5.2.2 Data Access

Tato kategorie obsahuje moduly s funkcionalitou pro přístup k datům a interakci mezi datovou a business nebo servisní vrstvou. Jednotlivé moduly z této kategorie pokrývají funkcionalitu pro správy transakcí, podporu DAO (Data Access Object) objektů, přístup k datům pomocí JDBC, objektově relační mapování (ORM) nebo XML serializace [16].

Spring framework definuje abstrakci pro správu transakcí, která poskytuje model konzistentní s různými aplikačními rozhraními jako jsou např. Java Transaction API (JTA), JDBC, Hibernate a Java Persistence API

(JPA). Také umožňuje deklarativní správu transakcí pomocí AOP (Aspect-Oriented Programming) a nabízí jednoduché rozhraní pro správu transakcí. Model je také kompatibilní se Spring abstrakcí přístupu k datům [16].

Pro přístup k datům Spring implementuje podporu DAO, pro jednoduchý a konsistentní způsob spolupráce s technologiemi pro přístup k datům jako např. JDBC, Hibernate a JPA. Díky tomu je možné používat stejnou verzi DAO objektů pro přístup k datům pomocí různých technologií bez nutnosti měnit kód programu. Také není nutné ošetřovat různé stavy (zachytávat výjimky) specifické pro různé technologie [16].

Spring také poskytuje modul pro abstrakci práce s JDBC pro jednodušší práci s relačními databázemi. Tento modul nabízí funkcionality pro otevírání a zavírání databázového spojení, připravení a spuštění databázových příkazů a zpracovávání jejich výsledků, odchytávání chybových stavů (výjimek) a správu transakcí [16].

5.2.3 Integration

Tato kategorie modulů definuje funkcionality pro integraci řady různých Java EE technologií pro tvorbu rozsáhlých enterprise aplikací [16].

Mezi techniky Spring framework integrace patří např. vzdálené volání metod (Remove Method Invocation – RMI), Spring HTTP invoker pro vzdálenou komunikaci pomocí HTTP, webové služby (JAX-WS) nebo integrace pomocí JMS (Java Message Service) [16].

Kategorie také obsahuje pomocný modul pro posílání emailů, který tvoří abstrakci nad základním mail systémem a je zodpovědný za správu zdrojů pro mail klienta na nižší úrovni [16].

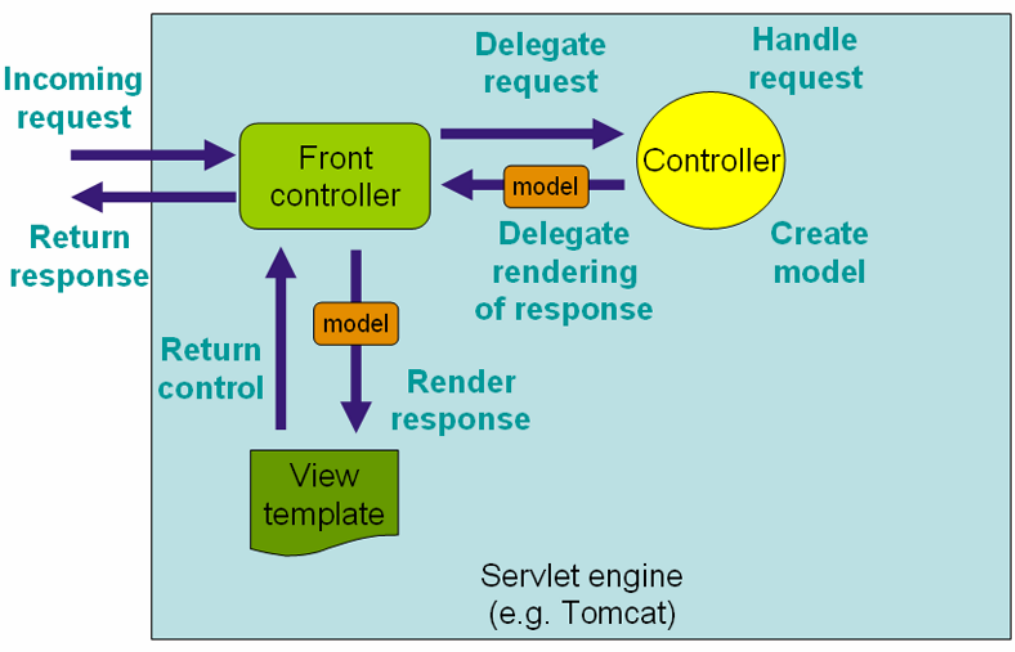
5.2.4 Web MVC

Spring Web MVC je webový framework založený na architektonickém vzoru Model-View-Controller, který umožňuje jednoduše integrovat tento architektonický vzor s dalšími komponentami aplikace používající nástroje Springu. Tento framework poskytuje základní funkce pro webové aplikace [17].

Spring Web MVC je řízen požadavky uživatelů, které jsou přijímány centrálním servletem. Tento centrální servlet (DispatcherServlet) přeposílá uživatelské požadavky dalším kontrolerům a poskytuje další užitečnou funkcionalitu pro usnadnění vývoje webových aplikací [16, 17].

DispatcherServlet implementuje návrhový vzor Front Controller viz obrázek 5.1. Tento kontroler přijímá všechny uživatelské požadavky webové aplikace. Přijaté požadavky jsou přeposílány dalším kontrolerům označe-

ným anotací *@Controller* a jsou mapovány na konkrétní metody anotacemi *@RequestMapping*. Od Spring frameworku verze 3.0 lze pomocí tohoto mechanizmu a přidané anotace *@PathVariable* vytvořit RESTful webové aplikace [16].



Obrázek 5.1: Zpracovávání požadavků v Spring Web MVC (převzato z [16])

Implementovaný controller je typicky zodpovědný za přípravu modelu s daty a výběr pohledu, který bude zobrazen uživateli. Model je mapa, kterou je možné použít v šablonovacích technologiích pro tvorbu webových stránek (např. JSP) nebo je možné z této mapy vytvořit přímo XML, JSON a další typy formátu pro obsah [16].

5.2.5 Test

Test modul poskytuje funkce pro podporu testování Spring aplikací pomocí frameworků JUnit nebo TestNG. Spring test umožňuje použití různých Spring komponent v testovacím prostředí. Nabízí možnosti mockování objektů, které se mohou používat při testování izolovaně. Spring test modul také umožňuje použití IoC principů pro jednotkové testování a podporu Spring frameworku pro integrační testování [16].

Java balíček *org.springframework.mock* obsahuje mock implementace různých objektů z Spring frameworku. Je zde například mock implementace

Servlet API, která obsahuje komplexní množinu Servlet API mockovaných objektů pro testování webového kontextu, kontrolerů a filterů. Tyto mockované objekty jsou určeny pro testování Spring Web MVC frameworku. Balíček také obsahuje mock objekty Spring Web Reactive, mezi které patří např. ServerHttpRequest a ServerHttpResponse pro testování WebFlux aplikací [16].

5.2.6 AOP

Spring AOP (Aspect-Oriented Programming) modul implementuje podporu pro aspektově orientované programování, ve kterém je klíčová jednotka modularity aspekt (např. aspekt logování). Tento modul poskytuje zejména deklarativní enterprise služby, které by měly nahradit deklarativní EJB služby v Java EE aplikacích. Spring AOP framework umožňuje programátorům implementovat různé aspekty a rozšířit tak objektově orientovanou aplikaci o AOP [16].

Spring AOP je implementován pouze v jazyce Java. To znamená, že není potřeba další zvláštní kompilační proces pro jeho používání. Přístup Springu k AOP se liší od většiny ostatních AOP frameworků, které se snaží nabídnout co nejúplnější podporu AOP přístupu. Cílem Spring AOP je spíše poskytnout co možná nejvhodnější integraci AOP do aplikací, které již využívají komponenty Spring frameworku a pomocí AOP vyřešit běžné problémy v enterprise aplikacích s možným použitím dalších komponent Spring frameworku [16].

5.3 Security

Spring Security je framework zaměřený na poskytování bezpečnostních služeb pro Java EE podnikové aplikace. Podobně jako u ostatních komponent Spring frameworku je kladen důraz na jednoduchou úpravu funkcionality pro různé případy použití a kompatibilitu s ostatními komponentami Spring frameworku [18].

Bezpečnostní služby tohoto frameworku pokrývají zejména dvě hlavní části bezpečnosti enterprise aplikací autentizaci a autorizaci. Také poskytuje ochranu proti různým útokům jako např session fixation, clickjacking a cross site request forgery. Spring Security je také možné integrovat přímo se Spring Web MVC frameworkem pro zabezpečení webových aplikací, ale není to nutností [18].

6 Persistence dat

6.1 Práce s databází v Java aplikacích

6.1.1 Java Persistence API

Java Persistence API je specifikace aplikačního rozhraní, které definuje jak lze provádět persistenci dat v jazyce Java. Největší položkou této specifikace je objektově relační mapování, avšak nabízí také řešení architektur pro škálovatelné podnikové aplikace s použitím Java EE (Java Enterprise Edition), ale také pro aplikace s použitím Java SE (Java Standard Edition). JPA specifikace je součástí již zmíněné specifikace EJB 3.0 jako součást JSR-220 (Java Specification Request) [19].

JPA definuje entity jako jednotky pro persistenci. Tyto entity jsou objekty jednoduché Java třídy (entitní třídy), která je označena anotacemi (`@Entity`) nebo je definovaná v konfiguračním XML souboru. Entitní třída musí mít implicitně nebo explicitně definovaný konstruktor bez parametrů pro svoji inicializaci. Objekty takové třídy jsou většinou ukládány do relační databáze jako řádky tabulky určené pro tyto entity. S entitami jsou spojena entitní metadata, která umožňují persistentní vrstvě aplikace správně rozpoznávat, interpretovat a řídit data. Tato metadata mohou definovat každý detail týkající se entity jako např. vztahy mezi entitami nebo definování klíčů entit v databázi. Metadata se mohou definovat dvěma způsoby podobně jako entity anotacemi, nebo v XML souboru. Oba způsoby jsou možné a jsou ekvivalentní [19].

O správu entit se stará Entity Manager, který zapouzdřuje většinu persistentní logiky. Funkce pro správu entit poskytuje pomocí jediného rozhraní *EntityManager*. Když Entity Manager objekt získá referenci na entitu, začne jím být entita řízena. Množina entit řízených Entity Managerem se nazývá persistentní kontext (*persistence context*). V tomto kontextu nesmí existovat více instancí entit se stejnou persistentní identifikací (*persistent identity*). V následující ukázce kódu je uveden postup vytvoření Entity Managera, ke kterému je potřeba *EntityManagerFactory* objekt. K jeho vytvoření je nutné uvést jméno Java balíčku s entitními třídami, nebo přímo konkrétní jméno entity pro kterou bude vytvořený Entity Manager spravovat [19].

```
EntityManagerFactory emf =
Persistence.createEntityManagerFactory("cesta.k.entitam");
EntityManager em = emf.createEntityManager();
```

JPA dále definuje dotazovací jazyk JPQL (Java Persistence Query Language). Tento jazyk je velice podobný dotazovacímu jazyku SQL a slouží pro dotazování dat entit uložených v databázi. Pro abstrakci databázového dotazu je vytvořen objekt *Query*, který je získán pomocí objektu *Entity Manager*. V dotazech se nevyskytují názvy konkrétních tabulek v databázi, ale názvy tříd a atributů entitních objektů. V dotazech je také možné využívat referencí entitních objektů. Dotazy jsou tedy tvořeny na entitní objekty uložené v databázi a ne přímo na konkrétní položky databázových tabulek, čímž je vrstva aplikace pro přístup k datům oddělena od databázové vrstvy. Výhodou těchto dotazů je, že nejsou závislé na žádné konkrétní databázové technologii. Můžeme je tedy použít ve více různých databázových systémech (např. Oracle, MySQL) [19, 20].

Alternativou JPQL je použití Criteria API, které je založeno na abstraktním schématu persistentních entit a jejich vztahů. Toto abstraktní schéma umožňuje pracovat s datovými entitami na úrovni volání funkcí Criteria API namísto vytváření JPQL dotazů. To může být vítanou výhodou například pro programátory, kteří nemají velké zkušenosti s vytvářením složitějších databázových dotazů. Criteria API a JPQL jsou úzce spojené a jsou navržené tak, aby poskytovaly stejné operace [20].

Pro tvorbu dotazů je v Criteria API použit objekt *CriteriaBuilder*, který je vytvořen pomocí objektu *Entity Manager*. Dotaz je vytvořen jako instance objektu, který implementuje rozhraní *CriteriaQuery*. Modifikací tohoto objektu se vytváří dotaz, který je následně zaslán do databáze. V následující ukázce kódu je uvedeno vytvoření objektu *CriteriaBuilder* pomocí Entity Managera. Také je zde uvedeno vytvoření dotazu (*CriteriaQuery*) pro entitní třídu *Entity.class* [20].

```
CriteriaBuilder cb = em.getCriteriaBuilder();
CriteriaQuery<Entity> cq = cb.createQuery(Entity.class);
```

6.1.2 Hibernate

Hibernate je jednou z implementací Java Persistence API specifikace. Díky tomu se jednoduše používá a může být jednoduše zaměnitelný za jinou implementaci splňující JPA. Hibernate umožňuje vyvíjet persistentní entity, které splňují aspekty objektově orientovaného přístupu jako např. dědičnost, polymorfismus, asociace mezi objekty, kompozice objektů a použití Java kolekcí. Hibernate používá pro optimalizaci výkonu např. odloženou inicializaci (*Lazy initialization*), různé strategie pro načítání dat a optimistické zamykání s automatickým verzováním a vytvářením časových značek. Pro zvýšení

výkonu je zde také snaha o vytvoření co nejvíce SQL dotazů při inicializaci aplikace místo za běhu aplikace [21].

Jednou z nejdůležitějších části Hibernate je ORM (Object-relational mapping) neboli objektově-relační mapování. Hibernate ORM řeší tzv. objektově-relační nesoulad paradigmatu (*paradigm mismatch*) tím, že provádí automatickou konverzi dat mezi relační databází a objekty vytvořenými v prostředí programovacího jazyka Java. To realizuje pomocí mapování Javovských objektů na datové entity v relační databázi. K mapování Hibernate používá buď mapovací soubory, ve kterých je popsán způsob, jakým se mají data mapovat z databáze do objektu a naopak, nebo je možné použít anotace Javovských objektů [21].

6.2 Soubory jako datové vstupy a výstupy

Pro persistenci dat je často nutné použití souborů, které umožňují jednoduchý přenos a úpravu dat.

6.2.1 XML

XML (eXtensible Markup Language) je rozšířený obecný značkovací jazyk, který byl vytvořen a standardizován W3C konsorciem jako podmnožina staršího SGML (Standard Generalized Markup Language), který není tak rozšířený. XML byl vytvořen primárně pro serializaci, ukládání, distribucia transport dat. Tento jazyk je snadno čitelný i zapisovatelný člověkem a jednoduše strojově zpracovatelný. Jazyk slouží především pro výměnu dat mezi aplikacemi [22, 23].

6.2.2 JSON

JSON (JavaScript Object Notation) je textový formát pro ukládání a výměnu dat, který není závislý na platformě. Jedná se o způsob zápisu (notaci) objektu jazyka Javascript definovaného standardem *ECMA-262 3rd Edition – December 1999* (viz [24]). Je jednoduše čitelný člověkem a snadno zpracovatelný a generovatelný strojově [24, 25].

JSON se skládá ze dvou možných struktur:

- Kolekce dvojic klíč (název), hodnota (řetězec, číslo, objekt, pole, true, false, null)
- Pole – seznam po sobě jdoucích hodnot oddělených čárkou, který má vlastní klíč (název)

JSON je univerzální textový formát, který je podporovaný většinou moderních programovacích jazyků. Z tohoto důvodu je rozšířeně používán jako formát pro výměnu informací [25].

6.2.3 CSV

CSV je běžný souborový formát používaný pro výměnu dat, která jsou oddělovaná znakem čárka. I přesto, že se jedná o velmi běžný a používaný formát, neexistuje pro něj žádná specifikace. Jeho popis je definovaný v rámci *RFC4180* z roku 2005 [26].

Definici CSV formátu lze popsat v několika bodech:

1. Každý záznam je na samostatném řádku ukončeným znakem konce řádku.
2. Poslední řádek v souboru může, ale nemusí končit znakem konce řádku.
3. První řádka může obsahovat hlavičku s názvy jednotlivých polí v celém souboru.

6.3 Knihovny pro práci s datovými soubory

6.3.1 JAXB

Technologie Java a XML se jeví jako dobrá kombinace pro vytváření webových služeb nebo aplikací, které k nim přistupují. Formát XML se stal uznávaným standardem pro výměnu dat mezi různými systémy, který se v praxi velmi osvědčil. Jazyk Java umožňuje vytvářet enterprise webové aplikace včetně webových služeb a multiplatformní aplikace, které s těmito webovými službami mohou komunikovat [27].

Java Architecture for XML Binding (JAXB) je technologie, která definuje API práci s XML v programovacím jazyce Java. Toto API definuje metody pro přístup k datům XML dokumentů, jejich konverzi na Java objekty a naopak, bez nutnosti znalosti XML technologie a jeho zpracovávání [27].

Při zpracovávání XML dokumentu je nejprve nutné svázat schéma dokumentu s množinou Java tříd, které toto schéma reprezentují v aplikaci. Schéma definuje strukturu prvků XML dokumentu, jejich vztahy a pořadí v jakém se musí nacházet. Svazování (*binding*) schématu znamená vytvoření množiny tříd, která XML schéma reprezentuje v Java aplikaci. JAXB poskytuje nástroj pro provedení překladu XML schématu na množinu těchto tříd.

Také je možné použít opačný proces, při kterém se vytvoří schéma XML dokumentu z množiny Java tříd, ve kterých jsou použité anotace pro definici pravidel schématu. Tyto anotace jsou součástí JAXB [27].

JAXB definuje metody marshalling a unmarshalling. Metoda marshalling umožňuje vytvořit XML dokument z Java objektu, který může tvořit komplexní strom komponent dokumentu. Marshalling operaci může provádět objekt *Marshaller*, který obstarává celý proces marshallingu. Před operací marshalling je vhodné provést validaci dat, na kterých se následně provede operace marshalling. Validace je provedena objektem *Validator*, který je součástí JAXB [27].

Opakem je metoda unmarshalling, která z XML dokumentu vytvoří strom Java objektů reprezentujících komponenty dokumentu. Tuto metodu prování objekt *Unmarshaller*. Vytvořený strom Java objektů bude mít stejnou strukturu jako datové elementy v XML dokumentu. Objekty ve vytvořeném stromě jsou instance tříd množiny, která byla vytvořena podle schématu při procesu svazování (*binding*). Objekt *Unmarshaller* je možné nastavit (metodou *setValidating()*) tak, aby prováděl validaci zdrojových dat jako součást unmarshalling operace [27].

Pro vytvoření výše zmíněných objektů *Marshaller*, *Unmarshaller* a *Validator* je nutné použít *JAXBContext*, který slouží jako vstupní bod do JAXB aplikačního rozhraní. Třída *JAXBContext* poskytuje tovární metody pro vytvoření zmíněných objektů. Pro tuto činnost potřebuje uvést jména Java tříd, nebo Java balíčků tříd, které byly vytvořeny procesem *binding* [27].

EclipseLink MOXy je jedna z implementací standardního běhového prostředí JAXB, která ho podstatně rozšiřuje o nové funkce. Umožňuje například pracovat s JSON formátem podobným způsobem jak to definuje JAXB s XML formátem. Je tak možné konvertovat JSON dokumenty do Java objektů a naopak, bez změny množiny Java třídy vytvořených ze schématu v procesu *binding* [28].

6.3.2 SAX

Simple API for XML (SAX) definuje jednoduché rozhraní pro událostně řízené zpracovávání XML dokumentů. Pro čtení XML dokumentů využívá sekvenční přístup, při kterém se nevytváří žádný strom z XML elementů. SAX pouze vytváří události ve chvíli, kdy při analýze narazí na různé symboly XML dokumentu (tokeny) a to v pořadí, v jakém se v dokumentu nacházejí. Klientská aplikace, která SAX využívá, tyto události odchytává a dále zpracovává [29].

Výhodou SAX je vysoká rychlost zpracovávání a relativně nízká pamě-

ťová náročnost algoritmů, které používá. Paměťově nenáročný je proto, že není třeba uchovávat celý dokument a vytvářet strom s XML elementy dokumentu. V případě, že klientská aplikace nebude zpracovávat některé elementy, může SAX tyto celé části přeskočit a tím také zrychlit běh a snížit paměťovou náročnost [29].

Nevýhodou může být až příliš jednoduché API, které umožňuje odchytávat události jednou třídou pro obsluhu těchto událostí pro všechny elementy dokumentu. Je nutné si uchovávat informaci o pozici, na jaké se při sekvenčním zpracovávání v dokumentu SAX analyzátor nachází [29].

7 Návrh řešení

Řešením této práce bude vytvoření webové aplikace, do které bude možné úmyslně zanášet chyby. Samotné vkládání chyb bude provádět další samostatná aplikace. Součástí obou aplikací bude systém logování, který bude zaznamenávat jejich činnost.

7.1 UIS-web aplikace

UIS-web (University Information System webová aplikace) je webová aplikace, která simuluje chování reálného univerzitního informačního systému fungující jako SUT (*System Under Test*). V této aplikaci budou existovat dva typy uživatelů, kteří se budou moci přihlašovat do aplikace a používat její funkce. První typ uživatele bude reprezentovat studenta univerzity, který využívá UIS-web pro zápis a odzápis studovaných předmětů a přihlašování se na termíny zkoušek a odhlašování se z termínů zkoušek studovaných předmětů. Druhým typem uživatele bude učitel, který si bude moci registrovat a odregistrovat vyučované předměty a vytvářet a rušit termíny zkoušek. Také bude moci hodnotit (známkovat) studenty, kteří se účastní termínu zkoušek, které tento učitel vytvořil. Doména UIS by tak měla být na první pohled pochopitelná každému vysokoškolsky vzdělanému uživateli. To byl jeden z důvodů volby této domény.

Aplikaci bude možné sestavit z dílčích entit, které budou implementovat korektní (bezchybnou), nebo naopak chybovou funkčnost. V rámci řešení práce bude existovat kolekce entit, které budou implementovat korektní funkčnost, aby bylo možné sestavit aplikaci bez úmyslně vložených chyb. Bude tedy nutné nejprve vytvořit korektní verzi všech entit a následně sestavenou korektní aplikaci důsledně otestovat. Dále bude vytvořeno několik chybových entit, které budou implementovat konkrétní funkčnost aplikace s různě závažnými uměle vytvořenými chybami pro demonstraci funkčnosti navrženého řešení. Ve chvíli, kdy chybová entita provede chybovou funkčnost, informuje o této události zapsáním podrobností do logu aplikace.

Aplikace bude využívat databázi pro persistenci používaných dat. Na výběr SŘBD nejsou kladeny žádné omezující podmínky. Aplikace bude dále schopná veškerá data z této databáze exportovat ve formě souborů formátu XML nebo JSON. Exporty dat v těchto formátech jsou pro zamýšlený účel plně dostačující a žádné další v rámci této práce nebudou implementovány. Aplikace bude také schopna exportované (a případně modifikované) soubory

zpětně importovat a naplnit databázi daty z těchto souborů. Uživatel aplikace si tak bude moci libovolně upravit data aplikace podle svého uvážení či potřeb. Při importu aplikace nejprve smaže všechna data v databázi a následně ji naplní daty novými. Aplikace bude také obsahovat funkcionalitu pro obnovení databáze do původního stavu, ve kterém se nacházela bezprostředně po spuštění aplikace pro účely testování.

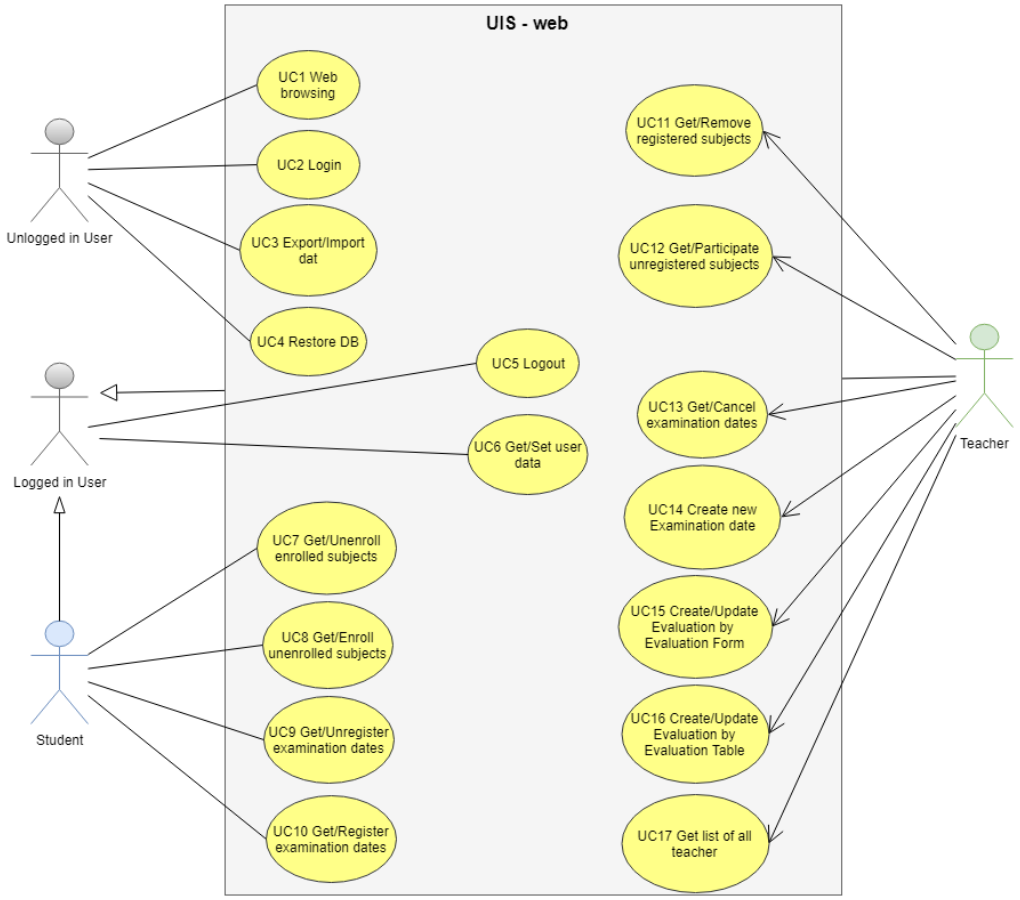
K aplikaci UIS-web bude existovat množina testů, které jednoznačně odhalí všechny úmyslně zanesené chyby nebo ověří, že se jedná o korektní verzi aplikace. Těchto testů musí být dostatečný počet a musí mít dostatečné pokrytí funkčnosti aplikace. Testy bude možné spustit z vnějšku aplikace.

7.1.1 Uživatelé

V aplikaci budou existovat tři typy uživatelů:

- **Nepřihlášený uživatel** – reprezentuje uživatele, který není přihlášený do systému, ale využívá jeho základní (většinou informativní) funkce
- **Student** – reprezentuje uživatele, který chce využívat UIS-web z pohledu hypotetického studenta univerzity
- **Teacher** – reprezentuje uživatele, který chce využívat UIS-web z pohledu hypotetického učitele univerzity

Uživatelé se budou přihlašovat do aplikace pomocí jednotného přihlašovacího formuláře, ke kterému mají přístup všichni uživatelé systému včetně nepřihlášených. Po úspěšném přihlášení bude uživatel autorizován a přesměrován na příslušný pohled aplikace. Uživatelé Student a Teacher mají přístup do odlišných částí systému. Případy užití UIS-web uživatelů jsou zobrazeny na use-case diagramu 7.1. Popisy jednotlivých use-case se nacházejí v následujících kapitolách.



Obrázek 7.1: Use-case diagram UIS-web

7.1.2 Nepřihlášený uživatel – případy užití

1. **UC1 Web browsing** – Use-case umožňuje nepřihlášenému uživateli prohlížení webových stránek, které jsou nepřihlášenému uživateli přístupné (Homepage, Db content stránka, use-cases stránka, export/import stránka, přihlašovací stránka).

2. **UC2 Login** – Use-case umožňuje nepřihlášenému uživateli se přihlásit do systému.

3. **UC3 Export/Import dat** – Use-case umožňuje nepřihlášenému uživateli provést import a export dat systému ve formátu JSON nebo XML.

4. **UC4 Restore DB** – Use-case umožňuje nepřihlášenému uživateli obnovit stav databáze, ve kterém se nacházela po startu aplikace.

7.1.3 Přihlášený uživatel – případy užití

1. **UC5 Logout** – Use-case umožňuje přihlášenému uživateli provést odhlášení ze systému.

2. **UC6 Get/Set user data** – Use-case umožňuje přihlášenému uživateli si prohlédnout jeho údaje (jméno, příjmení, email), případně je změnit.

7.1.4 Student – případy užití

1. **UC7 Get/Unenroll enrolled subjects** – Use-case umožňuje přihlášenému studentovi si prohlédnout jeho zapsané předměty, případně si je odepsat.

2. **UC8 Get/Enroll unenrolled subjects** – Use-case umožňuje přihlášenému studentovi si prohlédnout předměty, které nestuduje, případně si je zapsat.

3. **UC9 Get/Unregister examination dates** – Use-case umožňuje přihlášenému studentovi si prohlédnout termíny zkoušek svých zapsaných předmětů, na kterých je registrován, případně se odepsat z těchto termínů.

4. **UC10 Get/Register examination dates** – Use-case umožňuje přihlášenému studentovi si prohlédnout termíny zkoušek svých zapsaných předmětů, na kterých není registrován, případně se registrovat.

7.1.5 Učitel – případy užití

1. **UC11 Get/Remove registered subjects** – Use-case umožňuje přihlášenému učiteli si prohlédnout předměty, které učí, případně si je odepsat.

2. **UC12 Get/Participate unregistered subjects** – Use-case umožňuje přihlášenému učiteli si prohlédnout předměty, které neučí, případně si je zapsat.

3. **UC13 Get/Cancel examination dates** – Use-case umožňuje přihlášenému učiteli si prohlédnout termíny zkoušek, které vypsal, případně je zrušit.

4. **UC14 Create new Examination date** – Use-case umožňuje přihlášenému učiteli vytvořit nový termín zkoušky z předmětu, který učí.

5. **UC15 Create/Update Evaluation by Evaluation Form** – Use-case umožňuje přihlášenému učiteli oznámkovat studenta, který se účastnil jeho termínu zkoušky pomocí vygenerovaného formuláře a toto hodnocení později změnit.

6. **UC16 Create/Update Evaluation by Evaluation Table** – Use-case umožňuje přihlášenému učiteli oznámkovat studenta, který se účastnil jeho termínu zkoušky pomocí souhrnné hodnotící tabulky a toto hodnocení později změnit.

7. **UC17 Get list of all teacher** – Use-case umožňuje přihlášenému učiteli zobrazit tabulku všech učitelů v systému s předměty, které učí.

7.1.6 Technologie

V rámci návrhu řešení bylo pro tuto aplikaci zvoleno použití Spring frameworku, který pokrývá velkou množinu užitečných nástrojů pro moderní webové (enterprise) aplikace. Důležitou částí Spring frameworku je Spring IoC Container, který v sobě uchovává beany aplikace. Tato vlastnost bude sloužit pro efektivní skládání aplikace z dílčích entit, tedy Java bean.

7.2 Error-Seeder

Dále bude implementována aplikace Error-Seeder (Správce poruchových entit), která bude mít schopnost zanášet do aplikace UIS-web chyby. Tato aplikace bude komunikovat s uživatelem grafickým uživatelským rozhraním, ve kterém si uživatel zvolí podmnožinu z existujících entit implementovaných v UIS-web. Uživatel si bude moci vybrat nejen z chybových entit, ale také z korektně fungujících. Bude možné vybrat pouze bezchybné entity a sestavit tak správně fungující aplikaci UIS-web. Uživatel bude schopen za krátkou dobu vytvořit několik různých verzí UIS-web aplikace s různými zanesenými chybami. Takto vytvořené verze aplikace bude možné použít k ověření správnosti testovacích metod a pro trénovací účely testerů. Aplikace bude také nabízet možnost pojmenovat a uložit uživatelem vybranou podmnožinu entit pro její opětovné použití (šablony).

V aplikaci Error-Seeder bude dále možné spustit testy, které jednoznačně identifikují případně zanesenou chybu. Separátně bude možné spustit test, který bude ověřovat spojení aplikace UIS-web s databází. V případě, že jakýkoliv test selže nebo odhalí chybu, aplikace informuje o této události uživatele vhodným způsobem (viz A.1.3).

Error-Seeder aplikace bude schopná přeložit a sestavit aplikaci UIS-web. Výsledkem sestavení bude samostatný a nezávislý WAR soubor, který bude sloužit pro distribuci UIS-web aplikace koncovým uživatelům, kteří ji budou používat. Uživatel tak bude schopný bez problémů nasadit UIS-web aplikaci na aplikační server (např. Apache Tomcat) pomocí tohoto vytvořeného WAR souboru. Bezproblémové nasazení aplikace by mělo proběhnout také v případě vytvoření aplikace se zanesenými chybami.

Aplikace bude schopná zobrazit uživateli nápovědu, která bude jasně popisovat použití Error-Seederu a vysvětlovat jednotlivé funkce a jejich použití.

7.2.1 Katalog

Ke správnému fungování aplikace Error-Seeder bude nutné vytvořit katalog jako jeho datový vstup. Katalog bude textový XML soubor, který bude uchovávat data využívaná Error-Seederem. Před prvním použitím bude muset uživatel tento katalog připravit. Součástí řešení v rámci této práce bude katalog se základními daty.

Katalog bude uchovávat zejména informace o jednotlivých entitách aplikace UIS-web a jejich korektních nebo chybových verzích (implementacích). Entity a jejich verze budou obsahovat informaci pro jejich jednoznačnou identifikaci v rámci katalogu, jméno a jednoduchý popis, vysvětlující funkci entity nebo konkrétní verze. U verzí entit bude informace, zda se jedná o chybovou nebo korektní verzi. U chybových verzí bude navíc informace o jak závažnou chybu se jedná a scénář, který jednoznačně odhalí vkládanou chybu. Scénář bude představován posloupností kroků, které je třeba provést pro nalezení chyby. Katalog bude dále uchovávat informace o uživatelem vytvořených šablonách (viz kapitola 7.2.3).

Aplikace Error-Seeder bude poskytovat funkci pro export a import katalogu. Import bude načítat ze souboru katalogu data, která se budou v aplikaci používat. Export bude provádět opačný proces. Používaná data Error-Seederu bude ukládat do souboru (nového nebo naposledy importovaného). Exportní funkce je nutná, protože v aplikaci Error-Seeder bude možné editovat data katalogu.

7.2.2 Chybové verze entity

Chybové verze entit budou implementovat konkrétní funkčnost UIS-web aplikace s chybou určité závažnosti. Chybové i korektní verze entit budou implementované na straně UIS-web aplikace, kde bude možné provést jejich syntaktickou validaci a jejich kompilaci. Každá chybová verze entity bude

obsahovat unikátní identifikátor a jméno, které bude odlišné od korektní verze a mělo by odrážet podstatu zanesené chyby. Dále bude obsahovat popis chyby a jak se bude v UIS-web projevovat. Součástí bude také scénář, který jednoznačně odhaluje zanesenou chybu posloupností kroků, které musí uživatel UIS-web provést pro jednoznačnou identifikaci chyby. Tento scénář bude možné zobrazit v aplikaci Error-Seeder pro každou chybovou verzi. Ve chvíli, kdy chyba v chybové verzi nastane, musí být tato informace zanesena do logu aplikace.

V aplikaci Error-Seeder budou všechny verze entit UIS-web aplikace barevně odlišeny podle závažnosti chyby. Chybové verze entit se stejnou závažností chyby budou zvýrazněny stejnou barvou. Pro odlišení korektních verzí entit bude zvolen odstín zelené barvy, aby jí bylo možné jednoznačné odlišit od chybových verzí, které budou zvýrazněné jinou barvou. Pro tento účel bude možné všechny barvy měnit například v konfiguračním souboru nebo souboru pro definování stylů.

Pro každou entitu bude vždy existovat právě jedna korektní verze. Dále může pro každou entitu existovat libovolné množství chybových verzí. Pro provedení procesu injektování vybraných verzí entit do UIS-web aplikace je nutné vybrat pro každou entitu právě jednu její verzi. Není tedy možný výběr kombinace verzí pro jednu entitu.

7.2.3 Šablony

V aplikaci Error-Seeder bude možné vytvářet pojmenované množiny verzí entit, které si uživatel zvolil. Pojmenované a předvybrané množiny budou sloužit pro jejich znovupoužití. Uživatel tak bude mít možnost si vytvořit a pojmenovat několik množin verzí entit a ty následně používat. Bude tak možné vytvořit velice snadno a rychle několik různých variant aplikace UIS-web.

Data šablon budou součástí katalogu. Proto bude nutné uložit katalog po vytvoření nových šablon, jinak budou tyto změny po novém startu Error-Seederu ztraceny. V souboru katalogu budou šablony uloženy jako seznam identifikátorů verzí entit. Proto budou muset být tyto identifikátory unikátní v rámci jednoho katalogu. S tímto seznamem bude uloženo uživatelem vytvořené jméno šablony, které tuto šablonu reprezentuje, a proto bude muset být logicky také unikátní v rámci katalogu.

7.2.4 Maven procesy

UIS-web aplikace bude připravena pomocí buildovacího nástroje Apache Maven, který bude používán pro správu, řízení a automatizaci sestavování. Maven bude využíván aplikací Error-Seeder, která bude externě vyvolávat Maven procedury pro vykonání různých úkolů, které jsou popsány níže. Díky těmto procedurám si bude moci uživatel Error-Seederu např. vytvořit novou verzi UIS-web aplikace, ve které bude jím vložená množina entit, nebo si jednoduše spustí testy, které odhalí zanesené chyby nebo ověří korektně fungující UIS-web aplikaci.

Sestavení

Cílem sestavení je vytvořit samostatný a nezávislý WAR soubor se všemi závislostmi, které aplikace UIS-web vyžaduje. WAR soubor bude možné bez dalších úprav nasadit na aplikační server. Součástí běhu sestavení nebudou spuštěny testy, které by mohly ukončit sestavování aplikace z důvodu přítomnosti chyb v úmyslně vložených chybových entitách.

Spouštění testů

Další Maven úkol bude zajišťovat spuštění všech (jednotkových) testů, které důkladně testují funkcionality UIS-web aplikace. Cílem tohoto úkolu je dokázat, že UIS-web aplikace je teoreticky bez chyb nebo dokázat přítomnost úmyslně zanesených chyb.

Test databázového spojení

Test databázového spojení je Maven úkol, který spouští jediný speciální test UIS-web aplikace pro zkoušku přítomnosti databáze a bezproblémové spojení s UIS-web. Tento test k připojení využívá stejné atributy (adresu a typ databázového serveru, jméno databáze, přihlašovací jméno a heslo) jako výsledná aplikace. Uživatel si tak jednoduše ověří, zda bude po spuštění aplikace UIS-web schopná využívat databázi.

Clean

Tento úkol smaže všechny výstupní soubory Maven sestavení. To je možné použít před sestavováním nové verze UIS-web aplikace.

7.3 Injektování entit

Proces injektování má za cíl vkládání funkčních entit do aplikace UIS-web. Tyto entity mohou existovat ve více verzích (implementacích), které budou implementovat korektní nebo chybovou funkčnost. V UIS-web je definovaná množina bean, které aplikace potřebuje ke své činnosti, a je nutné definovat právě jednu korektní verzi pro každou entitu. Množina verzí těchto entit, ze kterých bude aplikace sestavena, je vybrána uživatelem pomocí aplikace Error-Seeder (viz kapitola 7.2), která provádí i samotný proces injektování. Je tak na uvážení uživatele Error-Seederu, ze kterých připravených verzí entit se bude UIS-web skládat.

Entity budou definovány jako Java beany a budou vkládány do UIS-web aplikace, kde se explicitně uvedou jako položky UIS-web aplikačního kontextu. Aplikační kontext bude zpracován Spring frameworkem, který z tohoto kontextu vytvoří IoC kontejner s beanami používanými v UIS-web. Implementace všech entit bude provedena v rámci UIS-web projektu. To znamená, že při procesu injektování nedochází k fyzickému přesunu entit mezi aplikacemi Error-Seeder a UIS-web. Error-Seeder pouze definuje, které verze bean implementující konkrétní rozhraní se mají v aplikaci UIS-web používat.

7.3.1 Beany

Beany reprezentují položky Spring IoC kontejneru, který spravuje všechny beany aplikace UIS-web. Tyto beany jsou pak pomocí Spring DI vkládány do aplikace UIS-web jako závislosti. Beany jsou v podstatě opakovatelně použitelné komponenty používané napříč celou aplikací UIS-web. Může se jednat například o DAO, servisní, řídící a různé další komponenty.

Beany jsou v UIS-web aplikaci reprezentovány rozhraním, které definuje chování beany. Implementace beany tedy musí implementovat a navenek poskytovat konkrétní funkce. Tyto funkce mohou být používány v dalších beanách.

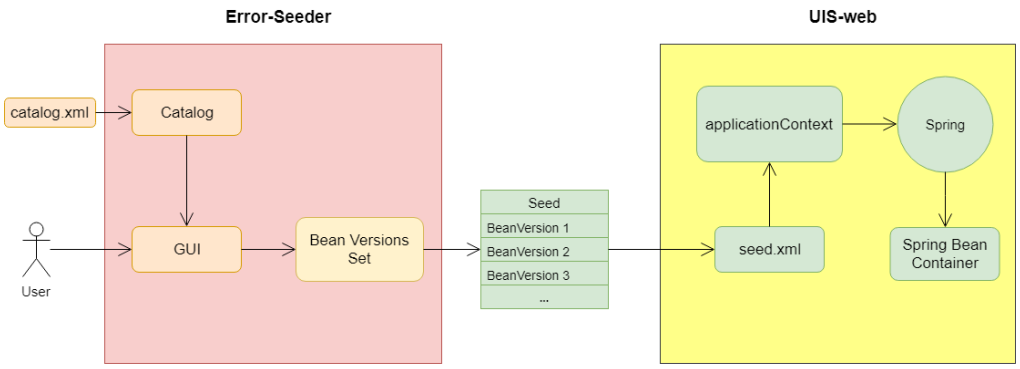
7.3.2 Verze bean

K jedné beaně definované rozhraním může existovat několik implementací (verzí). Bean verze reprezentuje konkrétní implementaci beany, respektive jejího rozhraní, která tak musí poskytovat funkčnost definovanou kontraktem tohoto rozhraní. Verze beany se vztahují vždy k jedné konkrétní beaně, jejíž rozhraní implementují.

Například beana UserDAO, která bude provádět přístup k uživatelským datům v databázi, bude obsahovat metodu pro výběr všech uživatelů z databáze. Tato beana bude existovat v korektní a chybové verzi. V korektní verzi beany bude metoda skutečně vracet všechny uživatele z databáze. V chybové verzi bude metoda vracet např. pouze některé uživatele namísto všech. A právě to bude úmyslně vložená chyba v chybové verzi beany.

7.3.3 Proces injektování

Proces injektování entit do UIS-web aplikace je zobrazen na následujícím diagramu 7.2. V levé části diagramu se nachází aplikace Error-Seeder, která bude s uživatelem komunikovat pomocí grafického uživatelského rozhraní. Error-Seeder nejprve načte načte data ze souboru katalogu. Následně zobrazí beany a jejich verze uživateli, který si pomocí uživatelského rozhraní vybere verze bean, ze kterých chce nechat sestavit UIS-web aplikaci. Z vybrané množiny bean verzí se vytvoří Seed (viz dále), který je následně vložen do aplikačního kontextu UIS-web aplikace.



Obrázek 7.2: Diagram procesu injektování entit

Seed je objekt, který obsahuje seznam verzí bean a jména rozhraní, které verze bean implementují. Aplikace Error-Seeder bude obsahovat funkčnost, která tento objekt jednoduše transformuje do XML formátu. Tento XML formát bude čitelný pro Spring framework, který ho v rámci aplikačního kontextu UIS-web bude zpracovávat.

V pravé části diagramu je UIS-web aplikace, která importuje nově vytvořený XML soubor s vloženými beanami do svého aplikačního kontextu. Z aplikačního kontextu Spring vytvoří Spring Bean Container (IoC kontejner), kde se budou nacházet beany používané UIS-web aplikací a to včetně bean, které vložil Error-Seeder.

8 Architektura

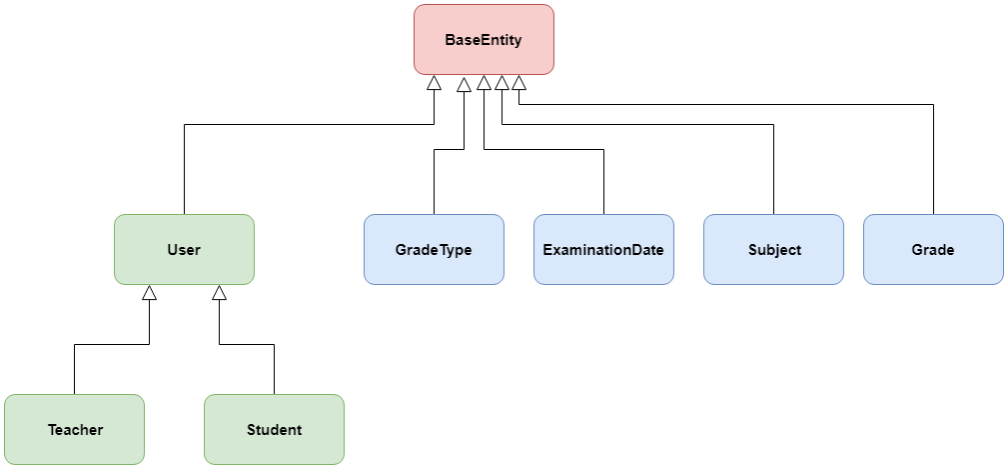
V této kapitole je popsaný návrh architektur aplikací UIS-web a Error-Seeder. Dále je zde navržena struktura a použití katalogu jako datového zdroje pro aplikaci Error-Seeder.

8.1 UIS-web

Jedná se o webovou aplikaci napsanou v jazyce Java, konkrétně Java EE 8 (Enterprise Edition), která umožňuje vytvářet komplexní webové a podnikové aplikace. UIS-web aplikace bude využívat komponenty Spring frameworku, který pokrývá značnou část potřeb při tvorbě nejen webových aplikací. Architektura aplikace bude postavena na modelu Spring Web MVC. Z tohoto důvodu budou pro architekturu aplikace UIS-web odvozena některá architektonická rozhodnutí, která tento rámec doporučuje. Architektura UIS-web bude postavena na upraveném architektonického vzoru MVC, který Spring Web MVC upřednostňuje. Celá řada dalších architektonických rozhodnutí byla provedena v souladu s osvědčenými postupy, které tento framework popisuje.

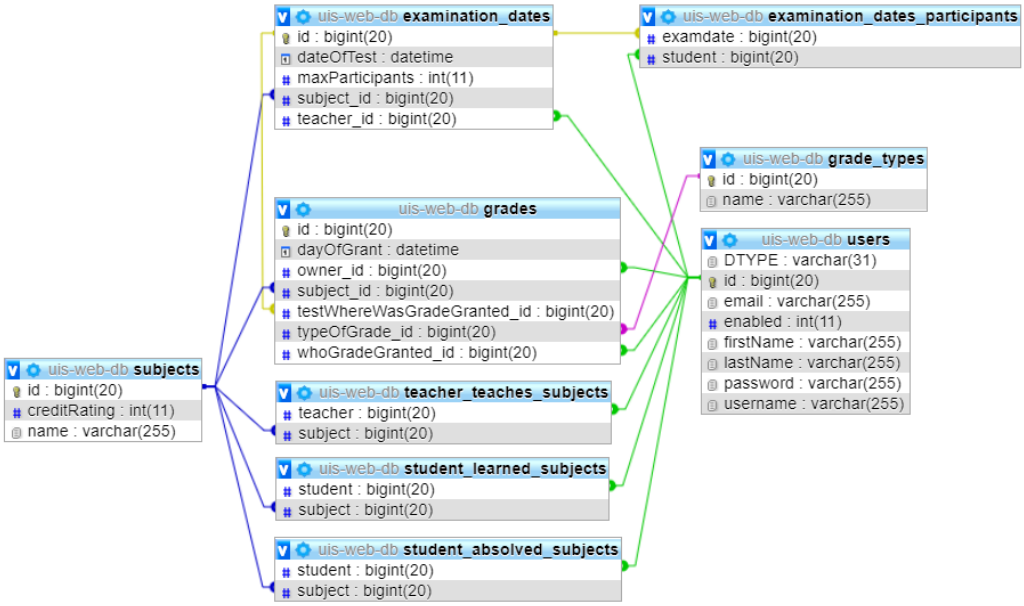
8.1.1 Datový model

Datový model UIS-web aplikace bude obsahovat několik doménových entit pro persistenci dat, které bude aplikace využívat ke své činnosti. Tyto entity budou definovány jako entitní Java třídy v UIS-web, kde budou uvedeny jejich atributy. Pro vytvoření a správu databázového schématu bude použit Hibernate ORM, který bude provádět objektově-relační mapování vytvořených entitních Java tříd na databázové tabulky a jejich sloupečky. Na následujícím diagramu 8.1 je zobrazena hierarchie jednotlivých entit používaných v UIS-web.



Obrázek 8.1: Hierarchie entit v UIS-web

Dále jsou popsané entity, které budou součástí doménového modelu aplikace UIS-web. Entity a jejich vztahy jsou zobrazené na ER diagramu 8.2.



Obrázek 8.2: ER diagram dat v UIS-web

Base Entity

Base Entity reprezentuje obecnou doménovou entitu, která může být uložená v databázi aplikace UIS-web. Všechny ostatní entity tuto entitu rozši-

řují. Base Entity obsahuje jediný atribut, a to identifikátor, který je unikátní v rámci objektů jednotlivých entitních tříd. Tento identifikátor bude v databázi sloužit jako primární klíč. Jedná se tudíž o povinnou položku entit, která bude automaticky generována. Primární klíč bude sloužit k jednoznačné identifikaci instancí jedné entitní třídy v jedné databázové tabulce.

User

User reprezentuje registrované uživatele aplikace UIS-web. Uživatelé mohou být typu Student nebo Teacher (více viz kapitola 7.1.1). Pro rozlišení uživatelů bude existovat rozlišovací hodnota, podle které bude v databázi typ uživatelů jednoznačně určen.

Uživatel typu Student bude ve dvou relacích s entitou Subject. První relace bude reprezentovat studentem studované předměty, kdy jeden student může studovat více předmětů a jeden předmět může být studován více studenty. Druhá relace je podobná, ale týká se absolvovaných předmětů.

Uživatel typu Teacher bude také v relaci s entitou Subject. Tato relace bude reprezentovat předměty, které učitel vyučuje. V této relaci může jeden učitel učit více předmětů a jeden předmět může být vyučován více učiteli.

Subject

Entita typu Subject reprezentuje předmět, který je registrovaný v univerzitním systému UIS-web. Entita Subject bude obsahovat dva atributy. První je řetězec jméno předmětu, který identifikuje předmět. Druhý atribut je celočíselná hodnota, která představuje počet kreditů, které student získá v případě, že předmět absolvuje.

Grade

Entita Grade reprezentuje hodnocení studentů, kteří se účastnili nějakého zkouškového termínu. Grade bude obsahovat datum a čas, kdy bylo hodnocení provedeno. Dále bude Grade ve dvou relacích s entitou User. První určuje studenta, který bude hodnocen. Druhá určuje učitele, který hodnocení provedl. Další tři relace s entitami GradeType, Subject a ExaminationDate určují, jaká známka byla udělena v rámci hodnocení, kterého předmětu se hodnocení týkalo a jakého zkouškovém termínu se hodnocený student účastnil.

ExaminationDate

Entita ExaminationDate reprezentuje zkušební termín. Obsahuje atributy datum a čas, kdy je termín uskutečněn a číselnou hodnotu, která určuje maximální počet účastníku termínu.

Entita bude mít dvě relace s entitou User a relace s entitou Subject. Relace s entitou User bude určovat jednoho učitele, který termín vytvořil a studenty, kteří se termínu účastní. Termínu se může účastnit více studentů, ale musí být splněna podmínka maximálního počtu účastníků termínu. Relace s entitou Subject určuje, v rámci jakého předmětu se termín vytvořil.

GradeType

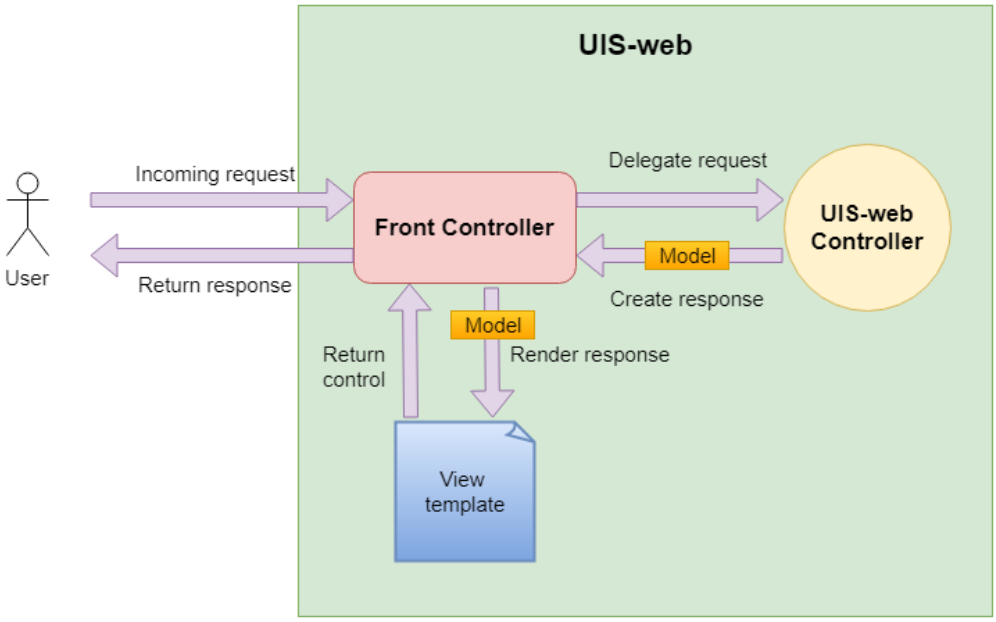
Tato entita reprezentuje známky, kterými je možné hodnotit studenty účastnící se termínu zkoušky. Entita obsahuje atribut jméno typu řetězec s hodnotami A, B, C, D, E, F.

8.1.2 Řídící vrstva – Kontrolery

Spring Web MVC v případě řídící vrstvy implementuje návrhový vzor Front Controller, který definuje jeden hlavní (front) kontroler, který přijímá uživatelské HTTP požadavky a deleguje je dalším kontrolerům. V rámci řídící vrstvy UIS-web bude nutné implementovat několik kontrolerů, které budou obsluhovat uživatelské požadavky.

Kontrolery UIS-web aplikace lze rozdělit do tří skupin. První skupina kontrolerů bude obsluhovat požadavky obecného uživatele, tedy uživatele, který nemusí být v aplikaci přihlášený, ale využívá její základní funkce.

Druhou skupinou jsou kontrolery, které budou obsluhovat požadavky uživatelů typu Student. A poslední skupinou jsou kontrolery, které budou obsluhovat požadavky uživatelů typu Teacher. Uživatelé typu Student a Teacher budou mít přístup do odlišných částí UIS-web a budou mít také odlišné typy požadavků.



Obrázek 8.3: Zpracovávání uživatelských požadavků kontrolery v UIS-web

Zpracovávání uživatelských požadavků je zobrazeno na předchozím diagramu 8.3. Žlutý kruh reprezentuje obecný kontroler, který bude nutné v rámci UIS-web aplikace implementovat. Takový kontroler definuje mapování požadavků, které bude obsluhovat. Pro tyto požadavky implementuje řídící funkčnost, která provede samotnou obsluhu požadavku. Nakonec tento kontroler vrátí hlavnímu kontroleru odpověď včetně vytvořeného modelu s daty použitými při tvorbě pohledů aplikace. Součástí odpovědí je také název pohledu, který se má uživateli zobrazit jako odpověď na jeho požadavek.

Obecné kontrolery

Obecné kontrolery UIS-web aplikace budou obsluhovat požadavky uživatelů, kteří nemusejí být do aplikace přihlášeni. Tyto kontrolery budou obsluhovat požadavky pro import a export dat aplikace, obnovení dat aplikace do původního stavu při jejím spuštění a požadavky na přihlášení uživatele.

Speciálním typem požadavků, které budou obsluhovat kontrolery v této kategorii, jsou požadavky na změnu údajů přihlášeného uživatele. Tyto údaje mohou být křestní jméno, příjmení a email uživatele.

Studentské kontrolery

V této kategorii jsou kontrolery, které budou obsluhovat požadavky přihlášeného uživatele typu Student. Požadavky studentských uživatelů se týkají zejména zapisování a odzapisování studovaných předmětů a registrace a zrušení registrace na zkouškové termíny studovaných předmětů. Studentské požadavky se dále mohou týkat pouze zobrazení pohledů se seznamy studovaných, absolvovaných a nestudovaných předmětů nebo se seznamy termínů zkoušek, kterých se student účastní, nebo neúčastní.

Učitelské kontrolery

V kategorii učitelských kontrolerů jsou kontrolery, které budou obsluhovat požadavky přihlášeného uživatele typu Teacher. Požadavky těchto uživatelů se budou týkat registrování a rušení předmětů, které přihlášený učitel vyučuje, vypisování nových a rušení existujících termínů zkoušek vyučovaných předmětů a vytváření nových a aktualizace existujících hodnocení studentů, kteří se účastnili jím vypsané zkoušky. Další učitelské požadavky se týkají zobrazování dat souvisejících s předešlými požadavky a zobrazení pohledu se seznamem všech učitelů v UIS-web aplikaci a předmětů, které učí.

8.1.3 Prezentační vrstva

Prezentační vrstva UIS-web aplikace se skládá z několika pohledů, které jsou vytvářeny na serverové straně UIS-web a jsou odeslány uživateli, který si je zobrazí ve svém webovém prohlížeči. Pro vytváření pohledů bude použita technologie JSP (Java Server Pages), která generuje stránky ve formátu HTML a jejich obsah je možné upravovat a generovat pomocí Java kódu (scriptletů). Pro zjednodušení vývoje bude použita kolekce knihoven tagů JSTL (JSP Standard Tag Library), která usnadní vývoj a omezí použití scriptletů.

8.1.4 Bezpečnost

O bezpečnost v UIS-web se bude starat komponenta Spring Security, která zajistí proces autentizace a autorizace v aplikaci. Tato volba byla provedena z důvodu vysoké robustnosti komponenty, jednoduchého použití a snadné integrace s ostatními používanými komponentami Spring frameworku.

V rámci procesu autentizace bude Spring Security zprostředkovávat přihlašování uživatelů do aplikace. To bude provádět porovnáváním přihlašovacích údajů, které byly zaslány přihlašovacím formulářem s údaji uživatelů

v databázi. Bude ověřovat shodu přihlašovacího jména a hesla, respektive hashe hesla. Také bude obstarávat přesměrování přihlášeného uživatele do příslušné části systému.

V případě autorizace bude ověřovat přístupová oprávnění uživatelů při vstupu uživatele do zabezpečených oblastí UIS-web aplikace. To znamená, že studentského uživatele vpustí pouze do studentské části aplikace a učitelského uživatele do učitelské části aplikace.

8.1.5 Spring

Jednotlivé vrstvy a komponenty aplikace budou propojeny pomocí Springu, respektive pomocí jeho funkce vkládání závislostí (Spring DI), která propojuje komponenty aplikace s velice volnou vazbou.

Komponenty aplikace budou definovány jako beany, které spravuje Spring IoC kontejner. Tuto definici je možné provést explicitním uvedením beany v aplikačním kontextu (soubor *applicationContext.xml*) nebo uvedením anotace přímo ve třídě konkrétní beany. Beany mohou definovat další beany, které požadují pro jejich korektní funkčnost. Propojení těchto závislostí s instancemi bean provede právě Spring pomocí IoC kontejneru.

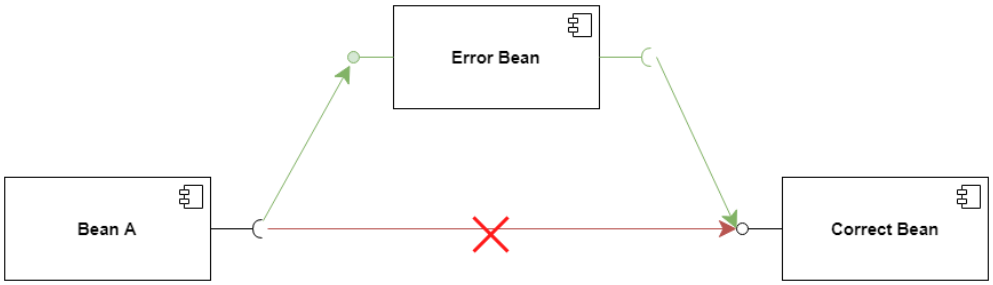
Komponenty Springu budou dále využity pro testování aplikace a další pomocné funkcionality. Při testování budou využity nástroje z komponenty Spring Test, která poskytuje aparát pro důkladné otestování všech částí UIS-web aplikace.

8.1.6 Chybové a korektní verze bean

V aplikaci UIS-web bude implementována vždy jedna korektní verze pro každou beanu, aby bylo možné sestavit teoreticky bezchybnou aplikaci UIS-web. Navíc zde budou implementovány vybrané chybové verze bean, které budou do aplikace úmyslně zanášet chyby. Takové beany budou od korektních odlišeny jménem Java balíku a jménem Java třídy, ve které bude chyba implementována.

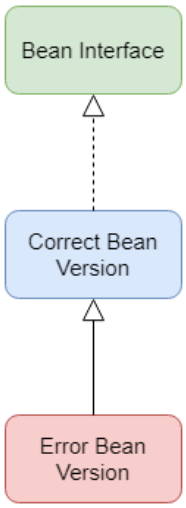
Dále bude nutné implementovat požadavek na odhalování těchto chyb. Jedná se o zápis do logu aplikace ve chvíli, kdy chyba nastane. Zápis bude obsahovat informace pro jednoznačnou identifikaci chyb včetně časového údaje, kdy chyba nastala. Také je nutné implementovat způsob, který bude identifikovat chybu přímo v kódu. V rámci práce bude vytvořena Java anotace, která bude označovat metodu s úmyslně chybovou implementací. Nová anotace bude obsahovat parametr, který ponese informaci s krátkým popisem vzniklé chyby.

Implementaci chybových verzí bean bude možné provádět dvěma způsoby. První způsob je vytvoření zcela nové verze beany, která implementuje stejné rozhraní jako verze beany s korektní funkčností. V tomto případě bude nutné implementovat všechny metody, které rozhraní definuje. Je zde možné uplatnit návrhový vzor proxy, kdy bude chybová verze beany uchovávat referenci na korektní verzi. Tuto referenci může použít pro delegování korektní činnosti v případě potřeby viz diagram 8.4.



Obrázek 8.4: Volba chybové verze beany v případě návrhového vzoru proxy

Druhý způsob je rozšíření korektní verze beany o chybovou funkcionalitu. Tento způsob je znázorněn v diagramu 8.5. Chybová verze beany bude dědit (rozšiřovat) od korektní verze, a díky tomu zdědí všechny neprivátní metody a atributy předka. Chybová verze beany tak může přepsat pouze metody korektní verze beany, do které chce úmyslně zanést chybovou funkcionalitu.



Obrázek 8.5: Vytvoření chybové verze beany rozšířením korektní verze

Spring DI bude vkládat verze bean do ostatních komponent aplikace UIS-web, kde budou definované jako jejich závislosti. Volbu verze beany bude

Spring provádět na základě definice, kterou vytváří uživatel pomocí aplikace Error-Seeder. Na diagramu 8.4 je uveden případ volby chybové verze namísto korektní v případě použití návrhového vzoru proxy, kterou provádí Spring na základě definice verzí bean vytvořené v aplikaci Error-Seeder.

8.2 Error-Seeder

Error-Seeder bude desktopová aplikace schopná úmyslně vkládat do UIS-web aplikace chyby. Pomocí této aplikace bude uživatel moci vytvořit za krátkou chvíli několik verzí aplikace UIS-web s různými chybami.

Chybové funkcionality bude možné implementovat v chybových verzích bean, které bude UIS-web používat pro svojí činnost. Vkládané verze bean (včetně korektních) bude zpracovávat Spring v rámci aplikačního kontextu, ve kterém bude spravovat tyto beany a vkládat v UIS-web jako závislosti. Error-Seeder bude vkládat definici všech uživatelem vybraných verzí bean (včetně korektních) na předem známé místo v UIS-web ve formátu XML, který Spring akceptuje. K tomuto účelu bude použitá knihovna JAXB.

Uživatel bude s aplikací Error-Seeder komunikovat pomocí grafického uživatelského rozhraní. V tomto rozhraní bude možné provést výběr všech existujících verzí bean, které mají být v UIS-web používány, a bude zde možné spustit proces injektování verzí bean. Dále bude v tomto rozhraní možné aplikaci UIS-web sestavit nebo spustit její testy. Také si zde uživatel bude moci vytvářet své pojmenované šablony s předvolenou množinou verzí bean. V uživatelském rozhraní bude také možnost nastavit cestu k souboru, do kterého se budou vkládat vybrané verze bean.

Výsledkem procesu sestavení UIS-web aplikace bude samostatný a nezávislý WAR soubor, který bude sloužit pro distribuci UIS-web aplikace. Spuštěné testy budou muset odhalovat jakoukoliv vloženou chybovou beanu. V UIS-web bude navíc existovat speciální test pro zkoušku dostupnosti databáze separátně spustitelný z aplikace Error-Seeder. Sestavení i spouštění testů bude prováděno pomocí nástroje Maven, který tyto požadavky plně pokryje. Z tohoto důvodu bude v uživatelském rozhraní Error-Seederu možné nastavit cestu ke konfiguračnímu POM souboru projektu UIS-web.

Jako datový vstup bude Error-Seeder využívat katalog, který bude ve formě XML souboru. V katalogu budou uložené veškeré informace, které bude Error-Seeder potřebovat ke své činnosti. Před prvním spuštěním aplikace bude nutné katalog nejprve připravit uživatelem. Uživatelské rozhraní pak bude poskytovat funkce pro ukládání, importování a další úpravy katalogu.

Uživatelské rozhraní bude vytvořené pomocí technologie JavaFX, která nahradila starší knihovny pro vytváření grafického uživatelského rozhraní. Tato knihovna doporučuje použití architektonického vzoru MVC. Z tohoto důvodu volba JavaFX výrazně ovlivní výslednou architekturu Error-Seeder aplikace.

8.2.1 Datový model

Datový model aplikace Error-Seeder se bude skládat ze dvou částí. První část bude tvořena datovými entitami pro práci s daty katalogu. Druhá část bude definovat datové entity reprezentující položky, které budou vytvářet kolekci verzí bean vkládaných do UIS-web – **Seed**.

Entity katalogu:

- **Bean** – tyto entity reprezentují Java beany aplikace UIS-web, které mohou existovat ve více verzích
- **BeanVersion** – reprezentuje konkrétní verzi beany, která je v UIS-web implementovaná
- **Catalog** – entita reprezentující katalog aplikace Error-Seeder
- **ScenarioStep** – krok scénáře, který odhaluje chybu v chybové verzi beany
- **Template** – uživatelská šablona – pojmenovaná kolekce uživatelem předvybraných verzí bean
- **VersionSeverity** – enum, který bude definovat několik hodnot pro úrovně závažnosti chyb v chybových verzích nebo hodnotu pro korektní verze – CORRECT, LOW, MEDIUM, HIGH, CRITICAL

Entity pro injektování:

- **SeedBeen** – entita, která bude reprezentovat jednu položku Seedu, tedy jednu verzi beany, která bude vložena do UIS-web
- **Seed** – tato entita bude reprezentovat samotný Seed – obal, která bude obsahovat všechny zvolené verze bean a bude vložen do UIS-web
- **InjectionVersion** – entita, která bude reprezentovat jméno Seedu (klonu UIS-web), kterou uživatel Error-Seederem vytvořil

- **InjectVersionValue** – entita, která bude reprezentovat konkrétní hodnotu InjectionVersion

8.2.2 Řídící vrstva

Řídící vrstvu budou tvořit dva kontrolery. Jednodušší kontroler bude zpracovávat uživatelské požadavky pocházející z pohledu nápovědy používání aplikace Error-Seeder. Mezi tyto požadavky patří změna textu a obrázku nápovědy, které se uživateli zobrazují v tomto pohledu.

Druhý kontroler bude obsluhovat uživatelské požadavky pocházející z hlavního pohledu aplikace. Požadavky z hlavního pohledu se budou týkat zejména získávání modelu dat, které hlavní pohled zobrazuje. Dále to mohou být požadavky k provedení akce vložení vybrané kolekce verzí bean do UIS-web, spuštění testů, spuštění procesu sestavení nebo vytvoření nové uživatelské šablony.

Požadavky se mohou také týkat práce s katalogem. Error-Seeder bude moci uložit katalog, se kterým pracuje. Také bude moci importovat nový katalog, se kterým bude následně pracovat. Kontroler bude také schopný obsluhovat požadavky na upravení cesty k souboru, kam se budou vkládat verze bean, a souboru *pom.xml* aplikace UIS-web.

8.2.3 Prezentační vrstva

Prezentační vrstvu tvoří dva pohledy, kterými bude uživatel komunikovat s Error-Seederem. Jednoduší pohled bude zobrazovat nápovědu k aplikaci. Budou zde obrázky s ukázkami z aplikace Error-Seeder, které bude možné tlačítky měnit. Ke každému obrázku bude existovat vysvětlující text, který ho popisuje.

Druhý pohled je hlavní pohled aplikace Error-Seeder, kterým bude uživatel celou aplikaci ovládat. Dominantou tohoto pohledu bude komponenta uživatelského rozhraní pro práci s jednotlivými beanami a jejich verzemi. Zde bude uživatel moci provést výběr verzí bean, ze kterých se má UIS-web sestavit. Součástí této komponenty bude také možnost zobrazit si u chybových verzí scénář, kterým bude chyba této chybové verze odhalena. Scénář bude ve formátované textové podobě a bude možné si ho zkopírovat do systémové schránky, ze které ho bude možné kamkoliv vykopírovat.

V pravé části pohledu bude panel, který bude rozdělen na tři části pro ovládání dalších funkcionalit. V první části bude možné spouštět Maven úkoly pro provedení nějaké činnosti. Tyto činnosti budou sestavení UIS-web, spuštění jednotkových testů aplikace UIS-web, vyčištění výstupní složky se

všemi výstupními soubory a spuštění testu, který ověří databázovou dostupnost UIS-web. V některých případech se bude jednat o relativně časově náročné operace. Proto bude vytvořena také komponenta uživatelského rozhraní s konzolí, která bude uživatele informovat o stavu těchto operací. Tuto konzoli bude možné schovat.

V druhé části bude možné vybírat a vytvářet uživatelské šablony. Po výběru existující šablony budou předvyplněné verze bean v centrální komponentě tohoto pohledu.

V poslední části panelu bude možnost pro pojmenování Seedu a spuštění procesu vkládání (injektování) uživatelem vybraných verzí bean do UIS-web. Jméno Seedu bude složeno ze jména vytvořeného uživatelem a části, ze které bude patrné kolik a jak závažných chybových verzích se bude ve výsledné aplikaci UIS-web nacházet. Druhá část bude uživateli v poslední části panelu také zobrazena.

Součástí hlavního pohledu bude menu, které bude nabízet volby pro ukončení aplikace, uložení a načtení katalogu, nastavení cest souborů UIS-web aplikace a volbu pro zobrazení nápovědy k aplikaci.

Pro návrh prezentační vrstvy JavaFX definuje FXML jazyk postavený na jazyce XML, který popisuje strukturu vytvářeného uživatelského rozhraní. Tvorba grafického uživatelského rozhraní tak bude naprosto oddělena od logiky aplikace. Pro upravení základního stylu bude vytvořen a použit jeden CSS soubor, který bude definovat upravené styly aplikace. Budou zde například definované barvy pro odlišení verzí bean s různou úrovní závažnosti chyb.

8.2.4 Katalog

Katalog bude Error-Seederu sloužit jako zdroj dat. Katalog bude uložen ve formě souboru XML a bude uchovávat následující informace.

Bean a BeanVersion

Bean bude obsahovat jméno beany, krátký popis, k čemu beana v UIS-web slouží a identifikátor rozhraní beany, který bude unikátní v rámci katalogu a bude reprezentovat jméno rozhraní v UIS-web, které beanu definuje. Beana bude dále obsahovat seznam referencí všech svých implementovaných verzí BeanVersion. Ten bude obsahovat unikátní identifikátor, jméno a krátký popisek. BeanVersion bude dále obsahovat kompletní jméno třídy, která bude implementovat verzi beany. Dále bude obsahovat informaci definující úroveň závažnosti chyby v případě chybových verzí nebo informaci, že se jedná

o korektní verzi. Součástí chybových verzí by měl být také scénář, který jednoznačně odhaluje chybu zanesenou v chybové verzi. Tento scénář bude obsahovat posloupnost kroků, kterými uživatel UIS-web jednoznačně odhalí konkrétní chybu.

Templates

V druhé části katalogu bude uveden seznam uživatelských šablon. Tyto šablony budou mít jméno, které bude v rámci katalogu unikátní a podle kterého se bude šablona zobrazovat v Error-Seederu. Šablona bude dále obsahovat seznam identifikátorů verzí bean, které uživatel pro tuto šablonu vybere. Proto musí být tyto identifikátory unikátní.

8.2.5 Injektování verzí bean

Injektování všech uživatelem vybraných verzí bean bude Error-Seeder provádět uvedením těchto verzí v souboru XML v aplikaci UIS-web, který bude ve formátu čitelném Springem, aby ho mohl zpracovat (viz diagram 7.2). Z výběru verzí bude nutné vytvořit model (Seed), který bude následně transformován do souboru v UIS-web. K transformaci do XML formátu bude použita knihovna JAXB. Více viz kapitola 7.3.3.

8.2.6 Maven Úkoly

Z aplikace Error-Seeder bude možné spustit několik Maven úkolů pro aplikaci UIS-web. Tyto úkoly budou přesměrovávat svůj výstup do konzole, která je součástí uživatelského rozhraní Error-Seeder aplikace. V konzoli bude uživatel schopný sledovat stav běžících úkolů, které mohou být časově náročné.

Sestavení

Tento úkol bude mít za cíl sestavení aplikace UIS-web do samostatného nezávislého WAR souboru, který bude sloužit pro distribuci aplikace koncovým uživatelům, kteří ji budou používat, nebo pro nasazení aplikace na aplikační server.

Součástí úkolu sestavení nebude spuštění jednotkových testů. V případě přítomnosti chybových verzí bean by tyto testy po nalezení chyby ukončily proces sestavování.

Jednotkové testy

Tento úkol bude spouštět veškeré jednotkové testy, které budou součástí aplikace UIS-web. Těmito testy si uživatel Error-Seederu bude moci ověřit přítomnost zanesené chyby. Výsledky testů budou uživateli přístupné v konzoli, která bude součástí hlavního pohledu Error-Seederu.

Vyčištění

Tento jednoduchý úkol bude mít za cíl vyčištění výstupní složky *target*, kam se vkládají zkompilované a další soubory UIS-web, které Maven vytváří.

Test konektivity databáze

Tento Maven úkol bude spouštět jediný jednotkový test UIS-web aplikace, který bude testovat konektivitu UIS-web s databází. Uživatel Error-Seederu si tak může jednoduše ověřit, zda je možné UIS-web spustit a zda má smysl spouštět ostatní jednotkové testy, které ke svému korektnímu průběhu databázi také potřebují.

8.3 Použité nástroje

V této kapitole jsou popsané nástroje, které budou použité při vývoji aplikací v rámci této práce. Kromě vývojového prostředí jsou zde uvedené nástroje, které usnadní, urychlí nebo určitým způsobem podpoří vývoj.

8.3.1 IDE

Pro vývoj obou aplikací bude použité vývojové prostředí IntelliJ IDEA ULTIMATE 2018.1 se studentskou licencí. Toto vývojové prostředí je určené pro vývoj webových, mobilních a enterprise aplikací. Obsahuje podporu pro nejrůznější používané frameworky a knihovny (Java EE, Spring, GWT, Vaadin, Grails, atd.) a editory různých programovacích jazyků. Také v sobě integruje různé nástroje např. pro statickou kontrolu kódu a jeho analýzu, podporu verzovacích nástrojů (Git, SVN) nebo nástrojů pro sestavení projektu (Maven, Gradle).

8.3.2 JavaFx Scene Builder

Pro vytvoření návrhu pohledů aplikace Error-Seeder ve formátu FXML bude použit nástroj JavaFX Scene Builder verze 2.0, který k tomuto účelu pou-

žívá přehledné uživatelské rozhraní. Díky JavaFX Scene Builderu není nutné žádné psaní kódu a vytváření návrhu pohledů je snadné a rychlé.

8.3.3 phpMyAdmin

Pro účely přístupu a jednoduché správy používané databáze UIS-web aplikace bude použit nástroj phpMyAdmin verze 4.7.0.

8.3.4 JavaDoc

Pro vytváření programátorské dokumentace bude použit nástroj JavaDoc, který vyhledává vytvořené JavaDoc dokumentační komentáře v kódu aplikací a automaticky generuje dokumentaci. Kód obou aplikací bude důkladně komentován. Proto bude možné použít tuto generovanou dokumentaci při detailnějším zkoumání částí aplikací.

8.3.5 GIT

Pro verzování celého projektu této diplomové práce bude použit distribuovaný systém správy verzí GIT, který velice usnadní vývoj a bude sloužit také jako záloha dat. Pro použití GITu byla zvolena webová služba Bitbucket, která nabízí bezplatný hosting několika privátních projektů pro malé týmy. V rámci této práce budou používány GIT klienti SourceTree a TortiesGIT.

9 Implementace řešení

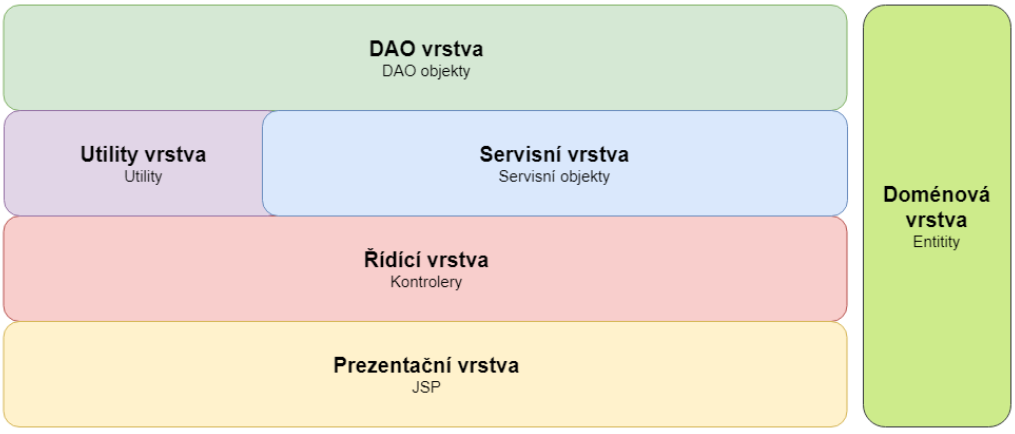
V této kapitole jsou uvedeny podrobnosti týkající se implementace navrženého řešení. Implementovány byly dvě aplikace UIS-web a Error-Seeder. Pro UIS-web i Error-Seeder aplikace byla vytvořena detailní programátorská JavaDoc dokumentace, kterou lze využít v případě nejasností. Tato dokumentace je uložena na přiloženém CD.

9.1 UIS-web

Aplikace UIS-web byla implementována jako webová aplikace napsaná v jazyce Java s použitím komponent frameworku Spring. Tato webová aplikace je schopná běžet na aplikačním serveru (např. Apache Tomcat), jehož součástí je servlet kontejner. Pro persistenci dat používá aplikace databázi. Při vývoji byl použit databázový server MariaDB (nástupce MySQL) verze 10.1.25.

9.1.1 Fyzická struktura aplikace

Na následujícím diagramu 9.1 je znázorněná fyzická struktura vrstev aplikace UIS-web. Komunikace vrstev probíhá směrem zdola nahoru. Nejnižší vrstvou je vrstva prezentační, pomocí které komunikuje UIS-web s uživatelem.



Obrázek 9.1: Fyzické vrstvy UIS-web aplikace

Řídící vrstva obsluhuje a směruje uživatelské požadavky vzniklé v pre-

zentační vrstvě. Tato vrstva využívá servisní vrstvu pro získání dat nebo provedení nějaké akce. Servisní vrstva poskytuje řídící vrstvě rozhraní, které definuje služby servisní vrstvy. Pomocí tohoto rozhraní spolu řídící a servisní vrstva komunikují.

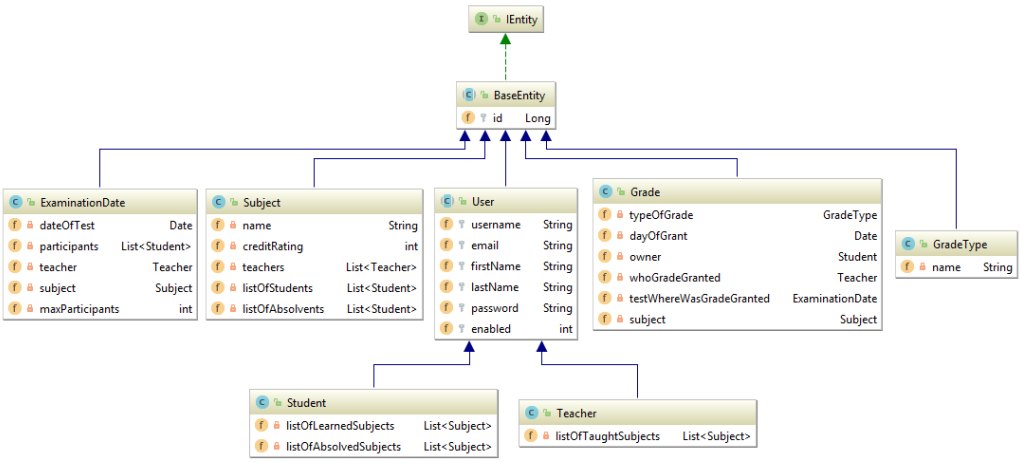
Servisní vrstva může využívat komponenty utility vrstvy, která nabízí užitečné funkce jako např. import a export dat a konverzi kalendářních dat a řetězců. Servisní vrstva komunikuje s utility vrstvou pomocí rozhraní, které utility vrstva navenek poskytuje. Komponenty utility vrstvy využívá pro svoji činnost pouze servisní vrstva.

Pro práci s daty v databázi aplikace UIS-web využívá servisní vrstva komponenty z DAO vrstvy. Komunikace vrstev je opět zajištěna pomocí rozhraní, které navenek DAO vrstva vystavuje. DAO vrstva zodpovídá za komunikaci aplikace s databázovou vrstvou.

Napříč všemi vrstvami jsou využívané prvky doménové vrstvy. Tato vrstva obsahuje entity (doménové objekty), které je možné persistentně uchovat v databázi aplikace. Samotnou persistenci a další databázové operace iniciuje pouze DAO vrstva. Výsledky těchto databázových operací většinou bývají právě entity (doménové objekty) nebo kolekce těchto entit.

9.1.2 Doménová vrstva

Doménová vrstva se skládá z doménových objektů (entit), které reprezentují data používané v aplikaci UIS-web. Doménové objekty se tvoří jako instance entitních tříd popsaných na diagramu 9.2. Tyto objekty mohou být trvale uloženy v databázi aplikace. Vytvoření databázového schématu a datové operace nad databází provádí framework Hibernate verze 5.2.16, který umožňuje ORM pro mapování doménové třídy na tabulky a jejich sloupce, doménové objekty na řádky v těchto tabulkách a atributy doménových objektů na položky v řádcích.



Obrázek 9.2: Diagram tříd doménové vrstvy UIS-web

`IEntity`

Rozhraní, které reprezentuje obecnou entitu, kterou je možné uložit do databáze UIS-web. Definuje metodu pro získání primárního klíče, který identifikuje entity v rámci jedné třídy doménových objektů. Rozhraní nedefinuje přesně jakého typu má primární klíč být, ale definuje, že to musí být serializovatelný datový typ, tedy typ, který implementuje rozhraní `Serializable`.

`BaseEntity`

BaseEntity je abstraktní třída, kterou ostatní implementované doménové třídy v UIS-web rozšiřují. Třída implementuje rozhraní `IEntity` a definuje primární klíč jako atribut `id` datového typu `Long`. Protože všechny ostatní doménové třídy tuto třídu dědí přímo nebo nepřímo, všechny doménové objekty používají primární klíč datového typu `Long`. `BaseEntity` dále definuje, že primární klíč bude automaticky generovaný. Dále implementuje metodu pro získání primárního klíče, která vrátí `id` objektu. Třída nakonec definuje abstraktní metodu `toString()`, která vrací textový řetězec reprezentující instanci třídy. Tuto metodu musí potomci této třídy implementovat.

`User`

`User` je abstraktní třída, která reprezentuje obecného uživatele UIS-web. Tato doménová třída definuje databázovou tabulku ***users***, ve které jsou uloženi všichni uživatelé. Třída `User` dále definuje atributy obecného uživatele jako jsou `firstName`, `lastName`, `email`, `username`, `password`, které jsou

datového typu `String` a celočíselný atribut (`enabled`), který určuje, zda je tento uživatel povolen nebo zakázán v UIS-web aplikaci (Spring Security využívá tento atribut v procesu autentizace). Třída `User` také obsahuje setter a getter metody pro přístup k atributům.

`Student`

Třída `Student` rozšiřuje abstraktní třídu `User`. Jedná se o doménovou třídu, která reprezentuje uživatele aplikace UIS-web typu Student. V této třídě jsou definované dva seznamy předmětů (`Subject`), které obsahují studentem aktuálně studované a absolvované předměty. Třída dále implementuje getter a setter metody pro přístup k těmto seznamům. Pomocí JPA anotací u getter metod je definovaná relační tabulka, která popisuje relaci N:M mezi studenty a předměty. Tato relace umožňuje studentovi, aby mohl studovat a absolvovat více předmětů a naopak umožňuje, aby jeden předmět mohlo studovat a absolvovat více studentů.

`Teacher`

Třída `Teacher` rozšiřuje abstraktní třídu `User`. Objekty této třídy reprezentují uživatele aplikace UIS-web typu Teacher. Třída definuje seznam vyučovaných předmětů učitelem. U getter metody tohoto seznamu je definovaná relační tabulka popisující relaci mezi učiteli a předměty stejným způsobem jako u seznamů studovaných a absolvovaných předmětů u třídy `Student`.

`Grade`

Třída `Grade` reprezentuje hodnocení studentů, které bylo uděleno učitelem. Pro uložení těchto hodnocení v databázi je zde definovaná tabulka se jménem ***grades***, ve které jsou hodnocení uložena. Třída definuje atributy datum a čas udělení hodnocení, stupeň hodnocení (známku), referenci na objekt studenta, který byl hodnocen, referenci na objekt učitele, který hodnocení provedl, referenci na objekt termínu zkoušky, na kterém bylo hodnocení uděleno a předmět, v rámci kterého bylo hodnocení uděleno.

`ExaminationDate`

Třída `ExaminationDate` reprezentuje zkušební termín v rámci konkrétního předmětu v UIS-web. Data zkušebních termínů jsou uložena v tabulce ***examination_dates***. `ExaminationDate` definuje atributy datum a čas termínu, seznam účastníků (studentů) termínu, učitele, který termín vytvořil,

předmět zkušebního termínu a maximální počet účastníků termínu. Pro seznam účastníků musí být opět vytvořena relační tabulka, která popisuje tento vztah N:M.

`Subject`

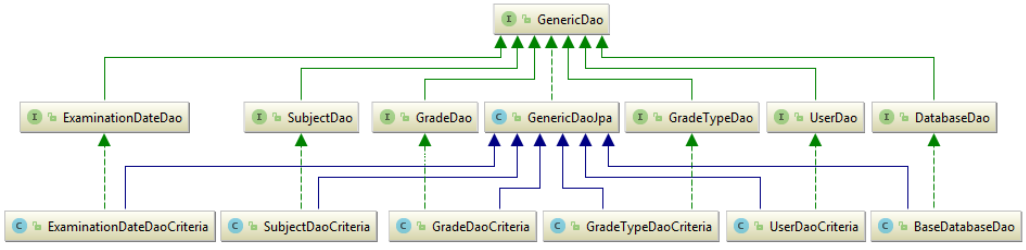
Třída `Subject` reprezentuje vyučované předměty v rámci UIS-web systému. Data předmětů jsou uložena v tabulce ***subjects***. Třída `Subject` definuje atributy jméno předmětu datového typu `String` a celočíselnou hodnotu počtu kreditů, které student získá po absolvování předmětu. Dále jsou zde seznamy studentů, kteří předmět studují nebo ho absolvovali, a seznam učitelů, kteří předmět vyučují. Tyto seznamy jsou výsledkem N:M relací popsaných výše.

`GradeType`

Objekty této třídy reprezentují stupeň známky, která může být udělena v systému UIS-web (A, B, C, D, E, F). Tato data jsou uložena v tabulce ***grade_types***. Třída definuje jediný atribut, a to jméno známky datového typu `String`.

9.1.3 DAO vrstva

DAO vrstva UIS-web aplikace obsahuje objekty, které umožňují přístup k datům. Tyto objekty tvoří poslední vrstvu mezi aplikací a databází. DAO vrstva poskytuje rozhraní pro komunikaci s ostatními komponentami aplikace, které definuje povolené metody pro práci s daty v databázi. Vrstva je zobrazena na následujícím diagramu 9.3. Jsou zde vidět rozhraní, které vrstva poskytuje vně pro komunikaci s ostatními komponentami aplikace.



Obrázek 9.3: Diagram tříd DAO vrstvy UIS-web

Dále jsou popsána jednotlivá rozhraní a jejich implementace. Tato rozhraní jsou definována v aplikačním kontextu jako beany, které mohou být

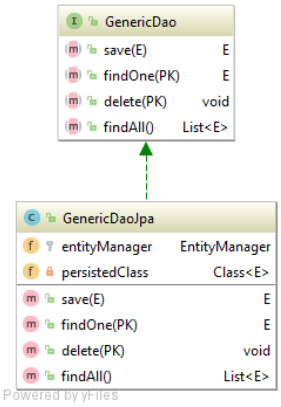
poskytovány dalším komponentám. Je tedy možné vytvořit jejich chybové verze. Pro přehlednost jsou níže uvedeny pouze korektní verze (implementace) těchto bean (rozhraní).

`GenericDao`

`GenericDao` rozhraní definuje obecné metody pro základní manipulaci s datovými objekty. S tímto rozhraním se pojí dva generické datové typy, které jsou použity v parametrech metod rozhraní. První typ reprezentuje doménovou třídu, s jejíž objekty DAO objekt pracuje. Druhý generický typ je primární klíč, podle kterého je možné objekty v databázi vyhledávat.

`GenericDao` definuje metody pro ukládání a mazání objektů v databázi a vyhledávání objektů podle primárního klíče (`id`) nebo vyhledávání všech objektů stejné doménové třídy. Metoda pro ukládání kombinuje dva typy databázových operací, vytvoření (insert) nebo aktualizace (update). Metoda výrazně zjednodušuje práci s objekty v aplikaci. Komponenty v UIS-web nemusí řešit, zda je předaný objekt, se kterým právě pracují, již uložený v databázi a je nutná jeho aktualizace, nebo je nový a je nutné ho vložit do databáze. Rozhraní a jeho korektní implementace jsou znázorněny na diagramu 9.4.

Třída `GenericDaoJpa` implementuje korektní verzi rozhraní `GenericDao` viz diagram 9.4. Implementuje metody rozhraní pro obecnou doménovou třídu a je tedy schopná uložit, smazat a vyhledat jakýkoliv doménový objekt v databázi. K manipulaci s daty využívá Entity Managera (viz kapitola 6.1.1), který zajišťuje samotnou JPA persistentní logiku. V konstruktoru této třídy musí být uvedena entitní třída, pro kterou bude `GenericDaoJpa` objekt zajišťovat persistenci.

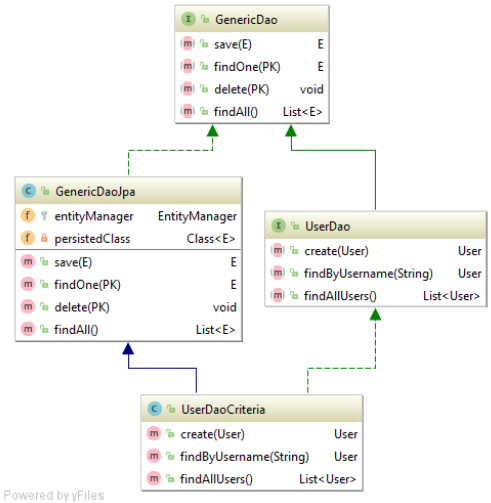


Obrázek 9.4: Diagram třídy a rozhraní `GenericDao`

`UserDao`

`UserDao` rozhraní definuje metody pro práci s objekty v databázi, které reprezentují uživatele aplikace UIS-web. Toto rozhraní rozšiřuje `GenericDao` popisované dříve. Při implementaci tohoto rozhraní bude mimo definovaných metod rozhraní `UserDao` nutné implementovat metody `GenericDao` rozhraní. `UserDao` definuje metody pro vytvoření nového uživatele, hledání uživatele v databázi podle jeho uživatelského jména (`username`) a metodu pro vyhledání všech uživatelů v aplikaci UIS-web.

Rozhraní `UserDao` implementuje třída `UserDaoCriteria`. Tato třída dědí od třídy `GenericDaoJpa`, proto není nutné implementovat metody rozhraní `GenericDao`. Objekty této třídy budou nabízet také funkčnost `GenericDaoJpa`. `UserDaoCriteria` využívá pro tvorbu databázových dotazů `CriteriaBuilder`, který je součástí Criteria API JPA (viz kapitola 6.1.1).

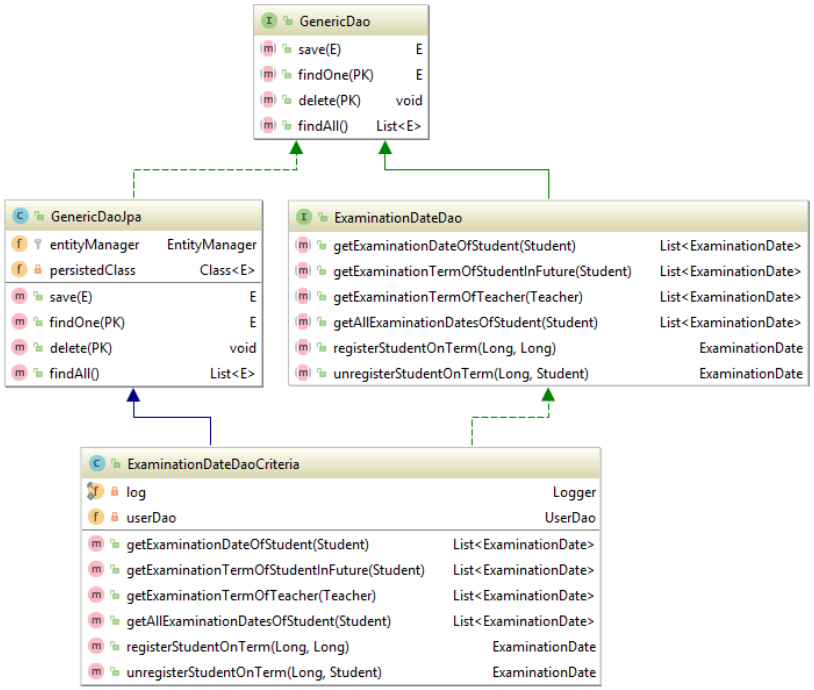


Obrázek 9.5: Diagram třídy a rozhraní `UserDao`

`ExaminationDateDao`

`ExaminationDateDao` rozhraní definuje metody pro práci s objekty v databázi, které reprezentují zkušební termíny předmětů a rozšiřuje rozhraní `GenericDao`. Rozhraní definuje metody pro získání termínů zkoušek, na kterých je student zapsaný, termínů zkoušek, na kterých je student zapsaný a zároveň jsou pouze v budoucnosti a všechny termíny zkoušek studenta, které zapsané má, i ty které zapsané nemá, ale studuje předmět, pro který je termín vytvořen. Dále definuje metodu pro získání termínu zkoušek, které vypsal jeden konkrétní učitel. Poslední dvě definované metody slouží k zapsání studenta na termín a odhlášení studenta z termínu zkoušky.

`ExaminationDateDaoCriteria` implementuje rozhraní `ExaminationDateDao` a všechny jeho metody. Pro vytváření databázových dotazů používá také `CriteriaBuilder` z JPA Criteria API. Třída rozšiřuje třídu `GenericDaoJpa` a dokáže ukládat, mazat a vyhledávat `ExaminationDate` doménové objekty v databázi.

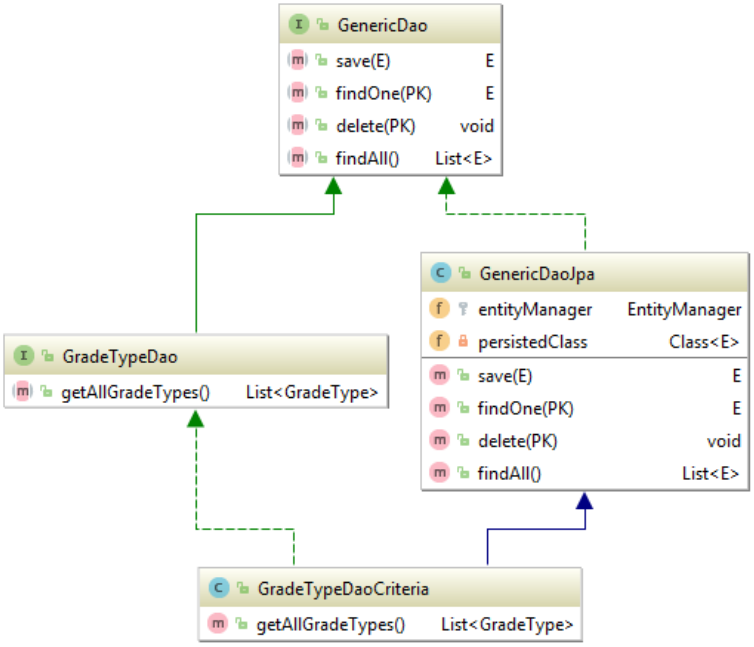


Obrázek 9.6: Diagram třídy a rozhraní `ExaminationDateDao`

`GradeTypeDao`

`GradeTypeDao` je jednoduché rozhraní, které definuje pouze jedinou metodu pro získání všech typů (stupňů) známek, které existují v databázi aplikace UIS-web. Rozhraní rozšiřuje rozhraní `GenericDao`, aby bylo možné ukládat, mazat a hledat `GradeType` doménové objekty v databázi.

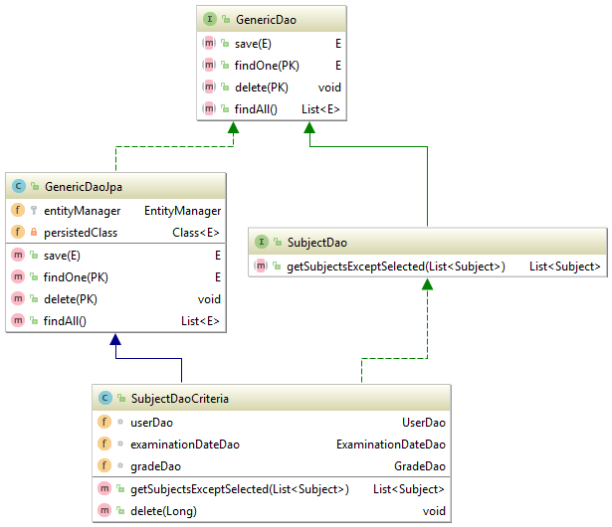
`GradeTypeDaoCriteria` implementuje rozhraní `GradeTypeDao` a implementuje její metodu pro získání všech typů známek. Pro tento účel využívá metodu `findAll()` třídy `GenericDaoJpa`, kterou rozšiřuje.



Obrázek 9.7: Diagram třídy a rozhraní `GradeTypeDao`

`SubjectDao`

`SubjectDao` rozhraní definuje metodu pro práci s předměty aplikace UIS-web. Tato metoda vyhledává v databázi předměty vyjma předaných předmětů v parametru metody. Metoda se používá např. pro vyhledání předmětů, které student aktuálně nestuduje a může si je tedy zapsat.



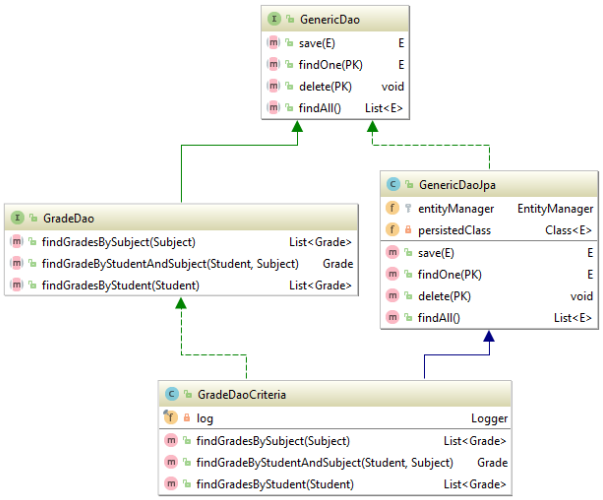
Obrázek 9.8: Diagram třídy a rozhraní `SubjectDao`

`SubjectDaoCriteria` implementuje rozhraní `SubjectDao` a její metodu pro získání předmětů z databáze bez vybraných předmětů uvedených v parametru metody. Třída obsahuje také implementaci metody pro smazání předmětu, která přepisuje metodu smazání obecného doménového objektu z databáze ze třídy `GenericDaoJpa`. K implementaci této metody došlo zejména proto, že smazání předmětu z databáze není tak triviální operace jako smazání obecného doménového objektu. Předmět má několik závislostí různých typů s různými dalšími entitními objekty (např. učitelé, kteří předmět učí, studenti, kteří předmět studují). Proto je nutné při mazání předmětu tyto závislosti vyřešit v `SubjectDao` objektu.

`GradeDao`

`GradeDao` rozhraní definuje metody pro vyhledávání `Grade` doménového objektu v databázi. Dvě metody vyhledávají seznamy známek v prvním případě podle jednoho konkrétního studenta a v druhém případě podle jednoho konkrétního předmětu. Poslední metoda vyhledává jednu známku v databázi podle jednoho studenta udělenou v jednom předmětu.

`GradeDaoCriteria` je korektní implementace rozhraní `GradeDao`, které pro vytváření databázových dotazů využívá `CriteriaBuilder` z JPA Criteria API.

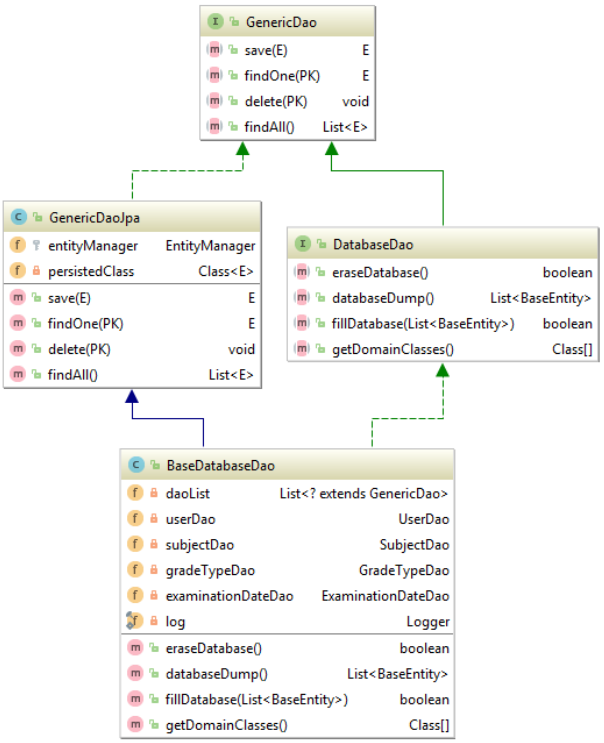


Obrázek 9.9: Diagram třídy a rozhraní `GradeDao`

`DatabaseDao`

Rozhraní `DatabaseDao` definuje metody pro správu celé databáze, nikoliv pouze jednoho doménového objektu. `DatabaseDao` obsahuje metodu pro vy-

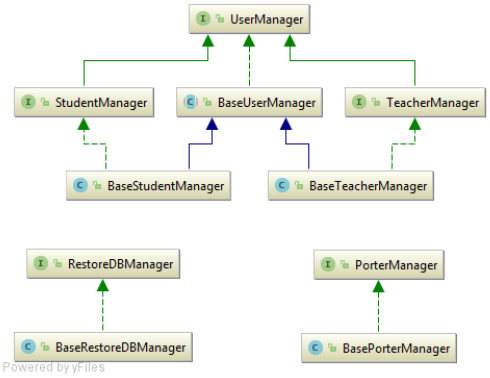
tvoření databázové zálohy a metodu pro zpětné nahrání dat z této zálohy do databáze. Tyto metody slouží pro export a import dat, kdy se z vytvořené zálohy vytvoří exportní soubor, jehož obsah je možné zpětným postupem importovat zpět do databáze. Rozhraní obsahuje další podpůrnou metodu pro smazání všech doménových objektů z databáze a pro získání pole všech doménových tříd, které obsahuje UIS-web aplikace. Pole doménových tříd se využívá např. pro vytvoření instance kontextu pro knihovnu JAXB, která se stará o vytváření a čtení datových souborů. Toto rozhraní implementuje třída `BaseDatabaseDao`.



Obrázek 9.10: Diagram třídy a rozhraní `DatabaseDao`

9.1.4 Servisní vrstva

Servisní vrstva aplikace se skládá ze servisních objektů (bean), které poskytují určité služby (viz diagram 9.11). V aplikaci UIS-web existují tři servisní komponenty. První komponenta poskytuje služby související s uživateli. Druhá komponenta poskytuje služby pro export a import databázových dat. Poslední komponenta zajišťuje služby pro obnovu databáze do původního stavu.



Obrázek 9.11: Diagram tříd servisní vrstvy UIS-web

Servisní objekty pro svojí činnost využívají komponenty DAO vrstvy pro práci s daty a Utility vrstvy pro použití komponent, které nabízejí nějakou užitečnou činnost. Servisní vrstva poskytuje rozhraní, pomocí kterého komunikuje zejména s vrstvou řídící. Řídící vrstva využívá služby pro provedení určité činnosti, kterou vyžaduje uživatel UIS-web.

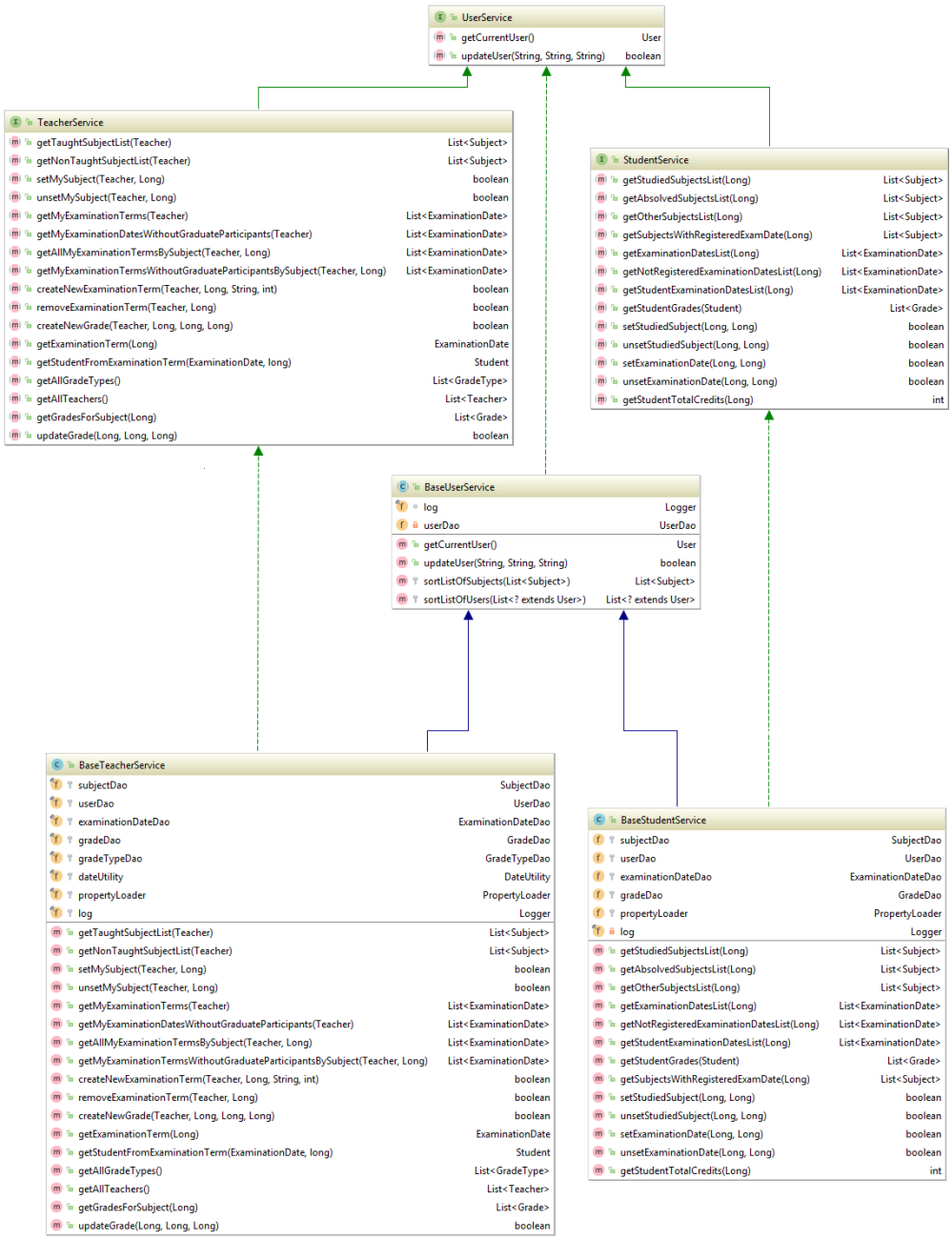
Uživatelské služby

Uživatelské služby se dělí na tři skupiny služeb. První skupina poskytuje služby týkající se obecného přihlášeného uživatele. Tyto služby zajišťuje abstraktní třída `BaseUserService`, která implementuje rozhraní `UserService` viz diagram 9.12. Tato třída poskytuje službu pro aktualizaci údajů právě přihlášeného uživatele a pro získání instance doménového objektu reprezentujícího právě přihlášeného uživatele. Předpokládá se rozšíření této abstraktní třídy třídami, které budou poskytovat služby týkající se konkrétních uživatelů (např. `StudentService` a `TeacherService`).

`StudentService` rozhraní definuje služby poskytované uživatelům typu Student. Jedná se zejména o služby poskytující data, která chce student zobrazit, nebo služby, které provádějí přihlašování nebo odhlašování předmětů a zkušebních termínů. Konkrétní služby lze vypozorovat na diagramu 9.12. Rozhraní `StudentService` implementuje korektní verze této beany, třída `BaseStudentService`, která k poskytování svých služeb využívá zejména DAO komponenty.

Rozhraní `TeacherService` definuje služby poskytované uživatelům typu Teacher. Tyto služby se týkají zejména získávání dat, které chce učitel zobrazit. Další služby zajišťují vytváření a rušení zkušebních termínů a vytváření a aktualizace hodnocení provedených učitelem. Konkrétní seznam služeb je možné vypozorovat na diagramu 9.12. Rozhraní `TeacherService` implemen-

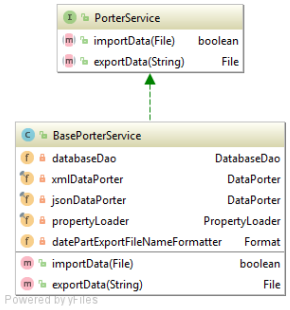
tuje korektní verze této beany, třída `BaseTeacherService`, která k poskytování služeb využívá také DAO komponenty a vybrané utility komponenty.



Obrázek 9.12: Diagram tříd uživatelských služeb

`PorterService`

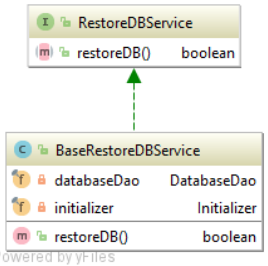
`PorterService` je služba, která poskytuje funkcionality pro export a import dat v databázi. K této činnosti využívá DAO objekty a utility komponenty, které provedou export a import data z a do souborů. Služba je schopná exportovat a importovat data ve formátu XML a JSON. Rozhraní `PorterService` implementuje třída `BeanPorterService` (viz diagram 9.13).



Obrázek 9.13: Diagram tříd `Porter` služeb

`RestoreDBService`

`RestoreDBService` rozhraní definuje metodu, kterou tato služba poskytuje. Jedná se o jednoduchou službu, která je schopná obnovovat stav databáze do původního stavu, ve kterém se nacházela po spuštění aplikace UIS-web. K této činnosti využívá DAO objekt `DatabaseDao`, který může provádět databázové operace. Rozhraní `RestoreDBService` implementuje třída `BaseRestoreDBService` využívající utility komponentu `Initializer`, která je zodpovědná za počáteční inicializaci databáze při startu UIS-web aplikace. Tato komponenta je schopná provést opětovnou inicializaci (reinicializaci) databáze.



Obrázek 9.14: Diagram tříd `RestoreDB` služeb

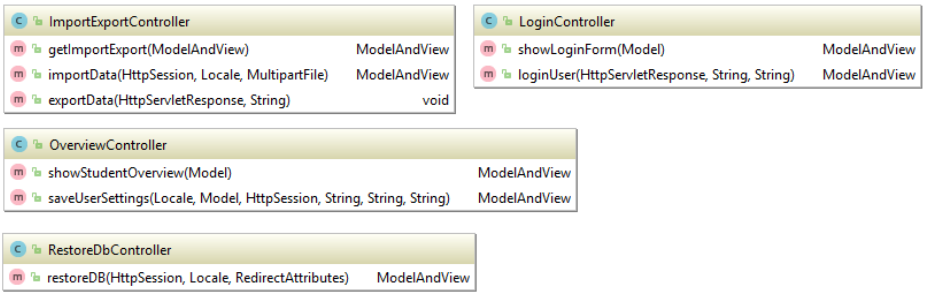
9.1.5 Řídící vrstva

Řídící vrstva aplikace UIS-web je zodpovědná za směrování a řízení uživatelských požadavků. Požadavky nejprve přicházejí na `DispatcherServlet`, který poskytuje Spring Web MVC. Tento servlet slouží jako Front Controller, který přeposílá požadavky ostatním kontrolerům řídící vrstvy. Kontrolery řídící vrstvy pouze zaregistrují mapování uživatelských požadavků, které zpracovávají svými konkrétními metodami. Při zpracovávání uživatelského požadavku připraví model, který zasílají (prostřednictvím `DispatcherServletu`) prezentační vrstvě. Pro získávání dat a provádění operací používají kontrolery servisní vrstvu, se kterou komunikují pomocí jejího rozhraní.

V UIS-web aplikaci je vytvořeno několik kontrolerů, kde každý obsluhuje uživatelské požadavky v rámci jednoho pohledu. To znamená, že každému pohledu aplikace, který vyžaduje určitou logiku nebo data z databáze, odpovídá jeden controller, který obsluhuje požadavky vytvořené v tomto pohledu.

Obecné kontrolery

- **`ImportExportController`** – obsluhuje požadavky uživatele na import a export dat z databáze
- **`LoginController`** – obsluhuje požadavky uživatele na přihlášení
- **`OverviewController`** – obsluhuje požadavky uživatele vytvořené v pohledu uživatelského přehledu
- **`RestoreDBController`** – obsluhuje požadavky obnovení databáze

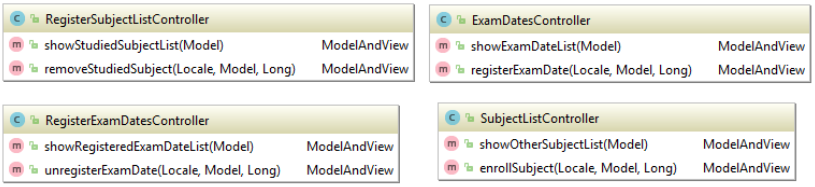


Obrázek 9.15: Diagram tříd obecných kontrollerů

Studentské kontrolery

- **`RegisterSubjectListController`** – obsluhuje požadavky studenta vytvořené v pohledu studovaných předmětů
- **`RegisterExamDatesController`** – obsluhuje požadavky studenta vytvořené v pohledu registrovaných zkušebních termínů
- **`ExamDatesController`** – obsluhuje požadavky studenta vytvořené v pohledu neregistrovaných zkušebních termínů

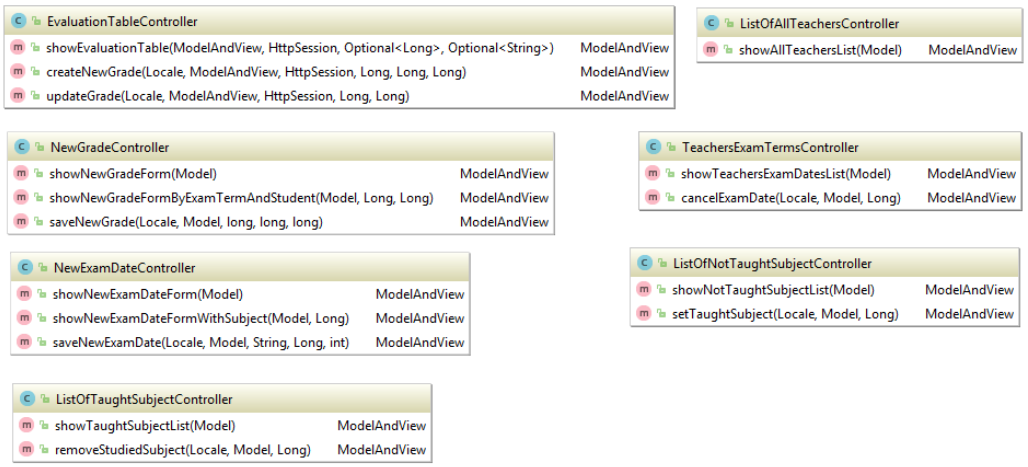
- **`SubjectListController`** – obsluhuje požadavky studenta vytvořené v pohledu nestudovaných předmětů



Obrázek 9.16: Diagram tříd studentských kontrollerů

Učitelské kontrolery

- **`EvaluationTableController`** – obsluhuje požadavky učitele na vytvoření hodnocení v souhrnné hodnotící tabulce
- **`NewGradeController`** – obsluhuje požadavky učitele na vytvoření hodnocení ve formuláři
- **`NewExamDateController`** – obsluhuje požadavky učitele na vytvoření nového zkušebního termínu
- **`ListOfTaughtSubjectController`** – obsluhuje požadavky učitele vytvořené v pohledu všech učitelem vyučovaných předmětů
- **`ListOfAllTeacherController`** – obsluhuje požadavky učitele vytvořené v pohledu se seznamem všech učitelů v UIS-web
- **`TeachersExamTermsController`** – obsluhuje požadavky učitele vytvořené v pohledu zkušebních termínů učitele
- **`ListOfNotTaughtSubjectController`** – obsluhuje požadavky učitele vytvořené v pohledu všech učitelem nevyučovaných předmětů



Obrázek 9.17: Diagram tříd učitelských kontrollerů

9.1.6 Prezentační vrstva

Prezentační vrstva je tvořena technologií JSP (Java Server Pages), které je zaslán model řídící vrstvou. Pomocí dat v tomto modelu tvoří JSP stránky ve formátu HTML, které jsou posílány uživateli pomocí HTTP. Uživatel si tyto stránky může zobrazit ve svém prohlížeči. Pomocí prezentační vrstvy tak může uživatel komunikovat s aplikací.

Pro snadné a rychlé vytvoření responzivních pohledů byl použit framework Bootstrap v4.0. Tento framework obsahuje sadu nástrojů pro vytváření responzivních webových aplikací. Bootstrap obsahuje JavaScriptové knihovny a definici CSS stylů, díky kterým lze v pohledech používat různé komponenty uživatelského rozhraní.

JSP tvoří pohledy, kterých je v aplikaci UIS-web několik. Je možné je rozdělit do několika skupin podle typu uživatelů, kteří k nim mají přístup. První skupina pohledů je přístupná každému uživateli UIS-web (včetně nepřihlášených). Do této skupiny patří úvodní pohled, který uživatel uvidí při přístupu na stránky UIS-web aplikace. Dále je zde pohled umožňující import a export UIS-web dat, pohled pro zobrazení počátečního stavu UIS-web databáze a pohled pro zobrazení use-case UIS-web aplikace. Posledním pohledem v této skupině je stránka s přihlašovacím formulářem, pomocí kterého se mohou uživatelé přihlašovat do UIS-web aplikace.

Druhou skupinou jsou pohledy, které jsou přístupné přihlášenému uživateli typu Student. V této skupině jsou pohledy pro zobrazení seznamu studovaných, absolvovaných a nestudovaných předmětů. V těchto pohledech je možné, aby se student přihlásil na nestudovaný předmět nebo se odhlásil ze studovaného předmětu. Dále jsou zde dva pohledy, které zobrazují seznam zkušebních termínů, kterých se student účastní nebo neúčastní, ale mohl by. V těchto pohledech je možné, aby se student přihlásil na termíny, na kterých přihlášený není nebo aby se odhlásil z termínů, na kterých přihlášený je.

Ve třetí skupině jsou pohledy, které jsou přístupné pouze přihlášenému uživateli typu Teacher. Mezi tyto pohledy patří pohled, který zobrazuje vyučované a nevyučované předměty přihlášeného učitele. Předměty si může učitel zapisovat a odepisovat. Učitel si může odepsat předmět pouze za předpokladu, že na předmětu není přihlášen žádný student. Další dva pohledy se týkají zkušebních termínů. První pohled použije učitel pro zobrazení seznamu se zkušebními termíny, které vytvořil. V tomto pohledu je možnost zrušit vytvořený zkušební termín. Druhý pohled umožňuje vytvořit nový zkušební termín. Další dva pohledy se zaměřují na hodnocení (známkování) studenta, který se účastnil zkušebního termínu. Hodnocení lze provést buď v pohledu, ve kterém je jednoduchý formulář, nebo v pohledu, ve kterém je

souhrnná tabulka pro hodnocení všech studentů filtrovaná podle předmětu. V posledním učitelském pohledu je možné zobrazit seznam všech učitelů v UIS-web aplikaci s předměty, které učí.

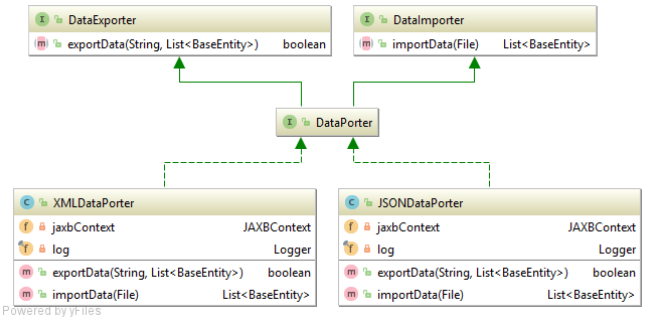
Speciálním případem pohledu, který nepatří do druhé ani třetí skupiny je pohled pro zobrazení a aktualizaci údajů přihlášeného uživatele. K tomuto pohledu má přístup kterýkoliv přihlášený uživatel (student i učitel).

9.1.7 Utility vrstva

Utility vrstva obsahuje komponenty, které poskytují nějakou užitečnou činnost. Utility komponenty poskytují rozhraní, pomocí kterého komunikují s vnějšími komponentami. Utility komponenty využívají zejména komponenty servisní vrstvy, které využívají pomocné funkce ke své činnosti.

`DataPorter`

`DataPorter` utility komponenta slouží k exportu a importu dat aplikace UIS-web ve formátu XML a JSON. Pro tento účel využívá komponenta JAXB knihovnu, konkrétně její implementaci MOXy, která je schopná provést konverzi Java objektů do XML nebo JSON souborů a naopak.

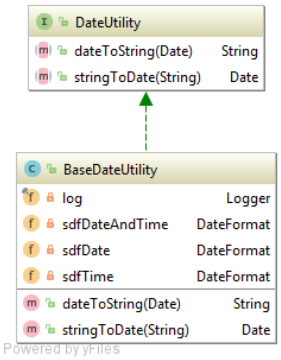


Obrázek 9.18: Diagram tříd `DataPorter` utility

Jak je patrné z diagramu 9.18, pro import a export dat využívá tato komponenta dvou tříd. Každá třída implementuje metody pro import a export v jednom z uvedených formátů. Třídy implementují rozhraní `DataPorter`, které rozšiřuje rozhraní `DataExporter` a `DataImporter`.

`DateUtility`

Utility komponenta `DateUtility` slouží v aplikaci UIS-web pro práci s kalendářními daty. Jak je patrné z diagramu 9.19, `DateUtility` poskytuje metody pro převod data na řetězec a zpětně převod řetězce na datum. Pro tyto převody používá tři `DateFormat` objekty, které určují formát data. Formáty dat jsou určeny konfigurací aplikace, která se nachází v souboru *application.properties*. Jsou zde definované formáty pro datum a pro čas a pro samotný datum a samotný čas používané v UIS-web aplikaci.

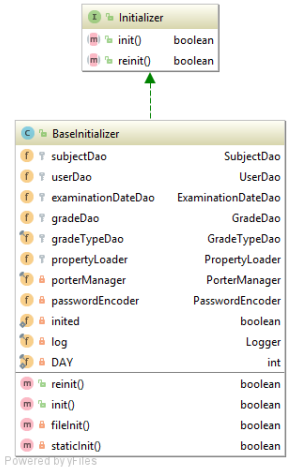


Obrázek 9.19: Diagram tříd `DateUtility`

`Initializer`

`Initializer` komponenta je určena pro inicializaci dat aplikace UIS-web v databázi. Na diagramu 9.20 lze vidět, že `Initializer` rozhraní definuje metody pro inicializaci a znovu-inicializaci dat. První metoda je spuštěna při každém startu aplikace. Inicializace může být provedena buď pomocí inicializačního souboru, který obsahuje inicializační data a nebo statickou inicializací v kódu. Statická inicializace používá inicializační data, která byla zvolena vedoucím této práce tak, aby pokryla většinu případů aplikace. V případě přítomnosti inicializačního souboru se primárně využívá k inicializaci tento soubor, jehož cesta je definovaná v konfiguraci aplikace. V opačném případě je inicializace provedena statickou verzí.

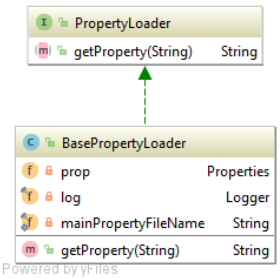
Druhá metoda je volána po uživatelském požadavku na uvedení databáze do původního stavu, což je důležité z hlediska testování. Jedná se o krok smazání všech doménových objektů z databáze a provedení opětovné inicializace stejným způsobem, jakým provádí inicializaci metoda `init()`.



Obrázek 9.20: Diagram tříd `Initializer` utility

`PropertyLoader`

Utility komponenta `PropertyLoader` má jednoduchý úkol. Provádí načítání hodnot konfiguračních atributů z konfiguračního souboru aplikace. Výchozí soubor konfigurace aplikace má název *application.properties* a je součástí UIS-web. V tomto souboru je možné konfigurovat různé atributy aplikace. Komponenta `PropertyLoader` pak tyto hodnoty čte na základě jmen atributů. K tomu využívá metodu, která na základě jména atributu vrací jeho hodnotu (viz diagram 9.21).



Obrázek 9.21: Diagram tříd `PropertyLoader` utility

9.1.8 Bezpečnost

Bezpečnost aplikace zajišťuje Spring Security, který provádí autorizaci i autentizaci uživatelů. Jeho konfigurace se nachází ve třídě `SecurityConfig`, kde jsou implementované metody zajišťující bezpečnostní procesy UIS-web.

Pro proces autentizace využívá databázového dotazu, který vrací údaje uživatele uloženého v databázi podle uživatelského jména autentizovaného uživatele. Tyto údaje porovnává s údaji uživatele, který se chce přihlásit pomocí přihlašovacího formuláře. Porovnává přihlašovací jméno a otisk hesel. Pro vytvoření otisku hesel využívá UIS-web `BCryptPasswordEncoder`, který je součástí Spring Security a poskytuje robustní metodu pro vytvoření otisku BCrypt.

V rámci procesu autorizace zajišťuje přístup přihlášených uživatelů pouze do částí aplikace UIS-web, do které mají oprávnění. Studentské uživatele vpouští do studentské části a učitelské uživatele do učitelské části aplikace. Součástí je také funkce, která přesměrovává uživatele po přihlášení přímo do příslušné části UIS-web.

9.2 Korektní a chybové entity

V rámci této práce jsou implementované korektní a vybrané chybové entity, ze kterých je možné sestavit různé varianty UIS-web. Jedná se o různé verze stejné Java beany, které používá Spring IoC kontejner pro vkládání závislostí. Pro zanášení chyb do UIS-web jsou vhodné komponenty z DAO, Utility a Servisní vrstvy, které jsou pro zanášení chyb v rámci této práce ověřené. V těchto vrstvách existují vybrané implementované chybové verze. Pro zanášení chyb do komponent řídící vrstvy bude nutné hotové řešení rozšířit viz kapitola 9.2.4.

Chybové verze se od korektních liší v názvu třídy, ve které je verze beany implementovaná, a názvem balíčku, ve kterém se třída nachází. Chybová verze musí vždy začínat velkým písmenem `E` následovaným dvouciferným pořadovým číslem chybové verze beany (tři chybové verze stejné beany budou tedy začínat `E01`, `E02` a `E03`). Chybové verze mají navíc označené metody, ve kterých se nachází chybová funkcionalita, anotací, která je součástí této práce. V anotaci `@ErrorMethod` je vždy uvedený krátký popis, který chybovou funkcionalitu vystihuje, jako např. v následující ukázce kódu. V metodě s chybovou funkcionalitou je také nutné informovat o této situaci zápisem do logu aplikace. Tento zápis musí obsahovat dostatečné informace, které vystihují vzniklé chybové chování.

```
@ErrorMethod(errorMessage = "Error method description")
public void methodWithErrorFunctionality() {
  ...
}
```

V dalších odstavcích jsou popsané implementované chybové verze bean.

9.2.1 DAO vrstva

Objekty v DAO jsou (speciální) beany, do kterých je možné zanést chybovou funkčnost.

`E01GradeTypeDao`

`E01GradeTypeDao` je první chybová verze beany, která je definovaná rozhraním `GradeTypeDao`. Tato beana provádí přístup k doménovým objektům typu `GradeType` v databázi. `E01GradeTypeDao` implementuje rozhraní `GradeTypeDao` a jeho jedinou metodu `getAllGradeTypes()` do které vkládá chybovou funkčnost. Metoda bude v této chybové verzi vracet vždy chybný seznam typů známek [1, 2, 3, 4].

`E01UserDao`

`E01UserDao` je chybová verze beany, která je definovaná rozhraním `UserDao`. Tato chybová verze rozšiřuje korektní verzi `UserDaoCriteria` o chybovou funkcionalitu v metodě `findAllUsers()`. `E01UserDao` bude v této metodě vrace pouze seznam všech studentů namísto všech uživatelů UIS-web (nevrací učitele).

9.2.2 Servisní vrstva

`E01StudentService`

`E01StudentService` je první chybová verze beany, kterou definuje rozhraní `StudentService`. Tato chybová verze rozšiřuje korektní verzi `BaseStudentService` o chybovou funkcionalitu v metodě `getStudiedSubjectsList()`, která v chybové verzi vrací vždy `null` namísto seznamu předmětů.

`E02StudentService`

`E02StudentService` je druhá chybová verze beany, kterou definuje rozhraní `StudentService`. `E02StudentService` rozšiřuje korektní verzi `BaseStudentService` o chybovou funkcionalitu v metodě `getStudentTotalCredits()`. Tato metoda vrací vždy číslo 20 namísto skutečného součtu kreditů studenta.

`E01TeacherService`

`E01TeacherService` je první chybová verze beany, kterou definuje rozhraní `TeacherService`. Tato chybová verze rozšiřuje korektní verzi `BaseTeacherService` o chybovou funkcionalitu v metodě `getTaughtSubjectsList()`,

která v chybové verzi vrací vždy `null` namísto seznamu vyučovaných předmětů.

9.2.3 Utility vrstva

V Utility vrstvě jsou implementovány dvě chybové verze beany, kterou definuje rozhraní `DateUtility`.

`E01DateUtility`

`E01DateUtility` implementuje rozhraní `DateUtility`. Do implementované metody `stringToDate()`, kterou definuje toto rozhraní, vnáší chybu. Metoda vždy vrací objekt `Date` vytvořeného z předaného řetězce s přidaným jedním dnem.

`E02DateUtility`

`E02DateUtility` implementuje rozhraní `DateUtility`. Do implementované metody `stringToDate()` tohoto rozhraní vnáší chybu. Metoda vždy vrací objekt `Date` vytvořený z konstantního řetězce. To znamená, že metoda vrací vždy stejný `Date` objekt pro předané řetězce stejného formátu (yyyy-MM-dd HH:mm ⇒ 2100-12-12 20:00, yyyy-MM-dd ⇒ 2100-12-12, HH:mm ⇒ 20:00).

9.2.4 Řídící vrstva

V rámci této práce nebyla implementována žádná chybová verze beany kontroleru. Kontroler je speciální druh beany, který vyžaduje definici mapování uživatelských požadavků. Definici mapování požadavků lze provést v XML souboru, kde je explicitně definováno, jaké verze bean se mají v UIS-web aplikaci použít. Z tohoto důvodu by bylo nutné rozšířit také Error-Seeder aplikaci.

9.2.5 Vytvoření chybové verze beany

Vytvoření nové verze beany s chybovou funkcionalitou lze provést několika způsoby, které jsou popsané níže. Každý způsob má určité výhody i nevýhody, je proto na uvážení programátora, který způsob zvolí.

V obou případech je nutné v implementaci označit metodu s chybovou funkčností anotací `@ErrorMethod` s parametrem `errorMessage`, který výstižně popíše vytvořenou chybovou funkcionalitu. Dále je nutné vytvořit

v implementaci zápis do logu aplikace, který bude obsahovat informace s popisem chyby, která právě nastala včetně časových údajů.

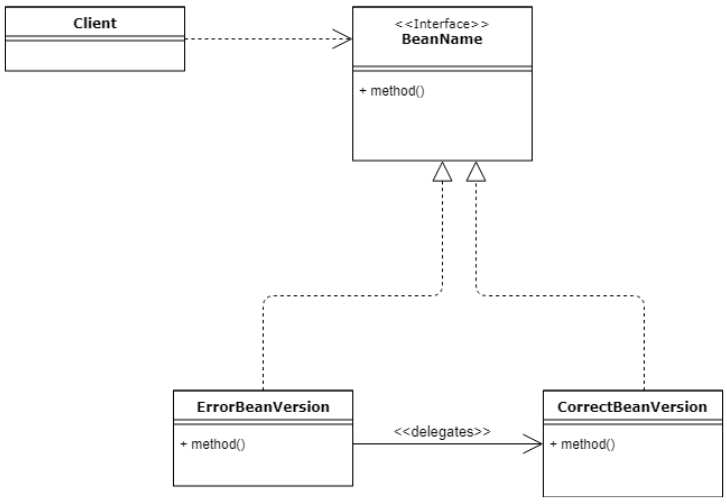
Také je vhodné po vytvoření nové chybové verze do souboru aplikačního kontextu vložit před import Seed souboru korektní verzi beany, která bude následně přepsaná chybovou verzí v Seed souboru. V případě, že se Seed soubor nesprávně importuje nebo bude nová chybová verze v Seedu chybět, bude aplikace používat korektní verzi.

```
<!-- Default definition of correct beans. -->
<!-- This beans might be overridden in seed.xml -->
<bean id="correctBeanVersion" class="full.path.correctBean"/>
...
<!-- Import Seed file -->
<import resource="seed.xml"/>
```

Implementace rozhraní

Prvním způsobem vytvoření chybové verze beany je implementace rozhraní, které beanu definuje. To znamená implementaci všech metod tohoto rozhraní. Díky tomu je možné modifikovat veškerou funkčnost beany.

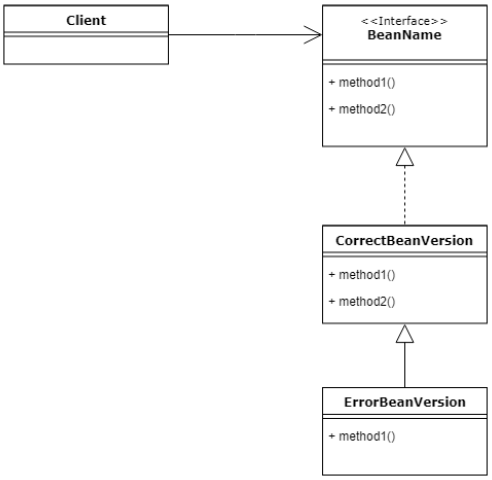
V tomto způsobu implementace chybové verze beany lze použít návrhový vzor proxy, který bude využívat referenci na původní korektní verzy beany a používat její korektní funkcionalitu v případě potřeby(viz diagram 9.22).



Obrázek 9.22: Návrhový vzor proxy při vytváření nové chybové verze beany

Rozšíření korektní verze

Druhým způsobem je rozšíření korektní verze beany o chybovou funkcionalitu. V tomto případě není nutné v chybové verzi implementovat rozhraní, které beanu definuje. Chybová verze beany pouze uvede v implementaci jako svého předka verzi korektní (dědí od ní). Následně mohou být přepsány pouze metody s úmyslem zanést chybovou funkcionalitu (viz diagram 9.23).



Obrázek 9.23: Rozšíření korektní beany o chybovou funkcionalitu

Vložení do katalogu

Po vytvoření chybové verze beany je nutné tuto verzi uvést v katalog souboru. Ten využívá aplikace Error-Seeder pro přehled o všech beanách a jejich verzích. V katalogu musí být pro každou beanu uveden seznam všech jejích chybových i korektních verzí. Pro jednotlivé verze zde musí být uvedeno jméno, popis a celý název třídy, která verzi implementuje. U chybových verzí musí být uveden také scénář odhalení zanesené chyby. Scénář je posloupnost kroků, které musí být provedeny uživatelem pro jednoznačnou identifikaci chyby chybové verze. Níže je uveden příklad chybové verze beany, která bude vložena do katalogu mezi verze stejné beany.

```
<version id="2"> <!-- Identifikator verze beany -->
  <name>Error Bean Name</name>
  <class>path.to.error.class.E01BeanName</class>
  <versionSeverity>HIGH</versionSeverity>
  <description>Short ErrorBean description</description>
  <scenario>
    <step number="1">Step 1.</step>
    <step number="2">Step 2.</step>
```

```
  </scenario>
</version>
```

Anotace `@ErrorMethod`

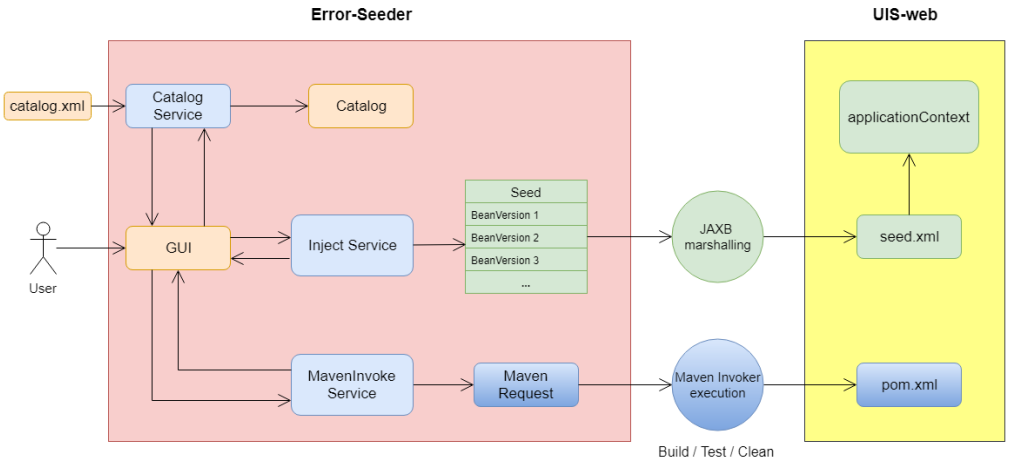
Anotace `@ErrorMethod` slouží pro jednoznačnou identifikaci metody se zanesenou chybou v kódu. Anotace má jeden povinný parametr, který vyjadřuje v čem spočívá chybová funkcionalita. Programátor zde vkládá krátký řetězec s popisem chyby, kterou implementovaná chybová metoda obsahuje.

```
@Override
@ErrorMethod(errorMessage = "Method returns just students
                             instead of all users.")
public List<User> findAllUsers() { ... }
```

9.3 Error-Seeder

Aplikace Error-Seeder je implementována v jazyce Java s použitím knihovny JavaFX pro vytvoření grafického uživatelského rozhraní. V aplikaci může uživatel provést výběr verzí bean, které chce vložit do UIS-web. Výběr verzí bean, který bude vkládán do UIS-web se nazývá **Seed**.

Na následujícím diagramu jsou naznačené základní činnosti aplikace Error-Seeder. Šipky vycházející z modulu GUI reprezentují uživatelské požadavky, které obsluhují kontrolery aplikace. Světle modré moduly reprezentují služby, které může uživatel využívat v rámci Error-Seeder aplikace. Šipky z těchto modulů do GUI modulu reprezentují zpětnou vazbu, která byla vytvořena jako odpověď na konkrétní uživatelský požadavek.

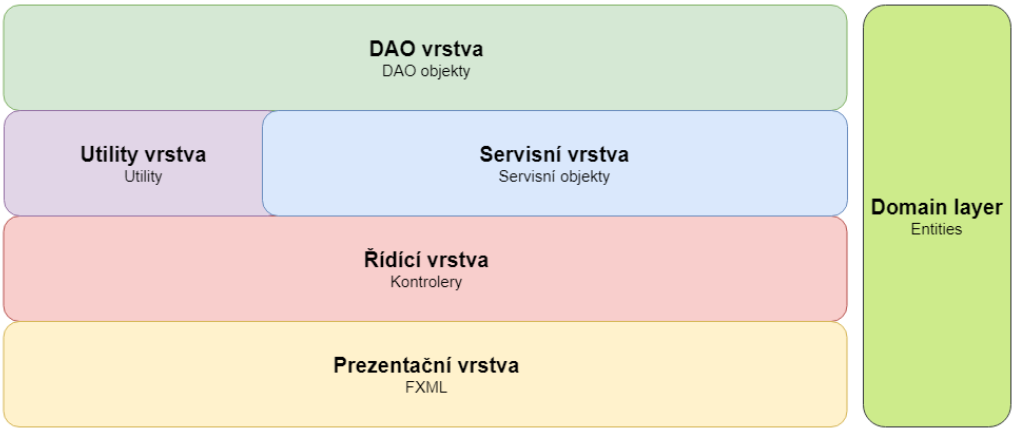


Obrázek 9.24: Diagram fungování Error-Seederu

Catalog Service zajišťuje služby týkající se práce s katalogem. *Inject Service* poskytuje služby pro provedení procesu injektování vybraných verzí bean do UIS-web aplikace. Pro tento proces využívá knihovnu JAXB. *MavenInvoke Service* nabízí služby pro spouštění Maven úkolů pro sestavení, testování a vyčištění výstupní složky UIS-web aplikace. Tyto úkoly jsou provedeny pomocí Apache Maven Invoker API. Detailnější popis těchto komponent je v kapitole 9.3.4.

9.3.1 Fyzická struktura aplikace

Fyzická struktura vrstev aplikace Error-Seeder je podobná aplikaci UIS-web. Prezentační vrstva je vytvořena pomocí jazyka FXML, který definuje strukturu pohledů aplikace. Model scény pohledů vytvoří z FXML popisu `FXMLLoader`, který je dostupný jako součást knihovny JavaFX.



Obrázek 9.25: Fyzické vrstvy Error-Seeder aplikace

Propojení prezentační a řídící vrstvy je zajištěno `@FXML` anotacemi, které v kontrolerech označují metody pro obsluhu uživatelských požadavků. V definici FXML jsou u komponent uživatelského rozhraní, které mohou generovat uživatelské požadavky, definovaná jména příslušných metod kontroleru, které tyto požadavky obsluhují.

Ostatní vrstvy mezi sebou komunikují pomocí rozhraní, která si přilehlé vrstvy navzájem poskytují. Speciálním případem je Utility vrstva, která poskytuje užitečnou činnost přilehlým vrstvám (včetně servisní).

Doménová vrstva neposkytuje žádné rozhraní. Obsahuje pouze entity pro manipulaci s daty, která jsou používána napříč celou aplikací.

9.3.2 Doménový model

Doménový model je tvořen doménovými objekty, které se dělí do dvou skupin. V první skupině jsou objekty, které reprezentují data katalogu. Tyto doménové objekty využívá aplikace Error-Seeder jako data, která jsou zobrazována uživateli a která používá pro svou vlastní činnost. Druhou skupinou jsou doménové objekty, pomocí kterých se vytváří Seed pro přenos dat při procesu injektování.

Objekty obou skupin je nutné serializovat do XML souborů. Z tohoto důvodu jsou u entitních objektů z obou skupin použité JAXB anotace pro definici modelu konverze Java objektů do XML formátu a naopak. První skupina je ukládána do katalog souboru a druhá skupina do Seed souboru.

`Bean`

`Bean` je katalogový doménový objekt, který reprezentuje beanu používanou v UIS-web aplikaci. Obsahuje identifikátor rozhraní beany, který musí být v rámci katalogu unikátní. Tento identifikátor odpovídá identifikátoru beany ve Spring IoC kontejneru aplikace UIS-web, a je zde většinou volen podle jména rozhraní, které beanu definuje. `Bean` dále obsahuje atributy jméno a krátký popis, které jsou zobrazovány v Error-Seederu. Dále `Bean` obsahuje seznam svých verzí, ve kterém se vždy předpokládá přítomnost korektní verze.

`BeanVersion`

`BeanVersion` reprezentuje konkrétní verzi beany, respektive konkrétní implementaci rozhraní, které beanu definuje. `BeanVersion` obsahuje unikátní identifikátor, jméno, krátký popis a celý název Java třídy v UIS-web, která verzi beany implementuje. Dalším atributem `BeanVersion` je hodnota výčtového typu `VersionSeverity`, který určuje závažnost chyby, která se nachází v chybové verzi, nebo určuje korektní verzi beany. U chybových verzí bean je navíc scénář, který definuje posloupnost kroků, které musí uživatel UIS-web vykonat, aby byla vložená chyba jednoznačně odhalena.

`Catalog`

Tento objekt reprezentuje celý katalog Error-Seederu, který byl do aplikace importován z katalog souboru. `Catalog` obsahuje atributy definující cestu k souboru *pom.xml* a k Seed souboru *seed.xml* aplikace UIS-web. Dále obsahuje seznam všech `Bean` a `Template` katalogu.

`ScenarioStep`

`ScenarioStep` reprezentuje jeden krok scénáře pro odhalení zanesené chyby. Tento krok obsahuje řetězec, který krok popisuje, a pořadové číslo kroku.

`Template`

`Template` reprezentuje uživatelem vytvořenou šablonu, která obsahuje uživatelem předvolené verze bean. `Template` obsahuje unikátní jméno v rámci katalogu a seznam verzí bean předvolených uživatelem.

`VersionSeverity`

Tento výčtový typ definuje několik hodnot pro závažnost (severitu) chyb úmyslně zanesených v chybových verzích bean. Enum definuje také hodnotu pro korektní verzi beany k odlišení od chybových. Hodnoty enumu jsou `CORRECT`, `CRITICAL`, `HIGH`, `MEDIUM`, `LOW`. Součástí hodnot tohoto enumu je také atribut reprezentující jméno třídy (kategorie), který bude použit ve stylování (obarvení) verzí tohoto typu v uživatelském rozhraní pomocí CSS.

Dále jsou popsané doménové objekty aplikace Error-Seeder, které tvoří model pro injektování verzí bean.

`SeedBean`

`SeedBean` reprezentuje jednu konkrétní verzi beany, která má být injektovaná do UIS-web (v souboru `seed.xml`), kde bude tuto beanu reprezentovat. Obsahuje atribut identifikátor beany a kompletní jméno třídy s implementací v UIS-web.

`Seed`

Obsahuje nastavení XML schematu Seed souboru v UIS-web určeném pro injektování verzí bean. Dále obsahuje seznam všech `SeedBean`, které se budou injektovat.

`InjectVersion`

`InjectVersion` reprezentuje jméno Seedu, který bude do UIS-web injektován. Jméno bude vytvořeno jako jednoduchá beana typu `java.lang.String`, kterou bude v UIS-web možné používat. `InjectVersion` atribut id bude reprezentovat jméno beany v UIS-web (výchozí hodnota „`injectVersion`“).

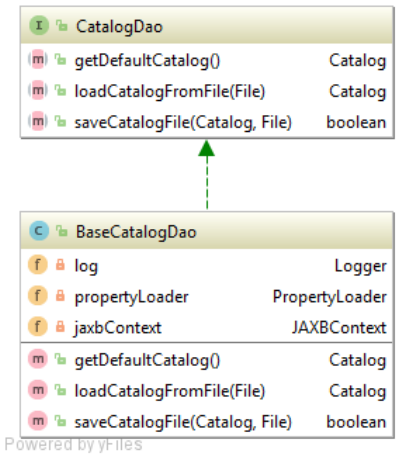
Dále `InjectVersion` obsahuje hodnotu `InjectVersionValue`, který obsahuje konkrétní jméno Seedu.

`InjectVersionValue`

`InjectVersionValue` reprezentuje konkrétní hodnotu jména Seedu, který chce uživatel injektovat do UIS-web. Obsahuje pouze atribut jméno verze.

9.3.3 DAO vrstva

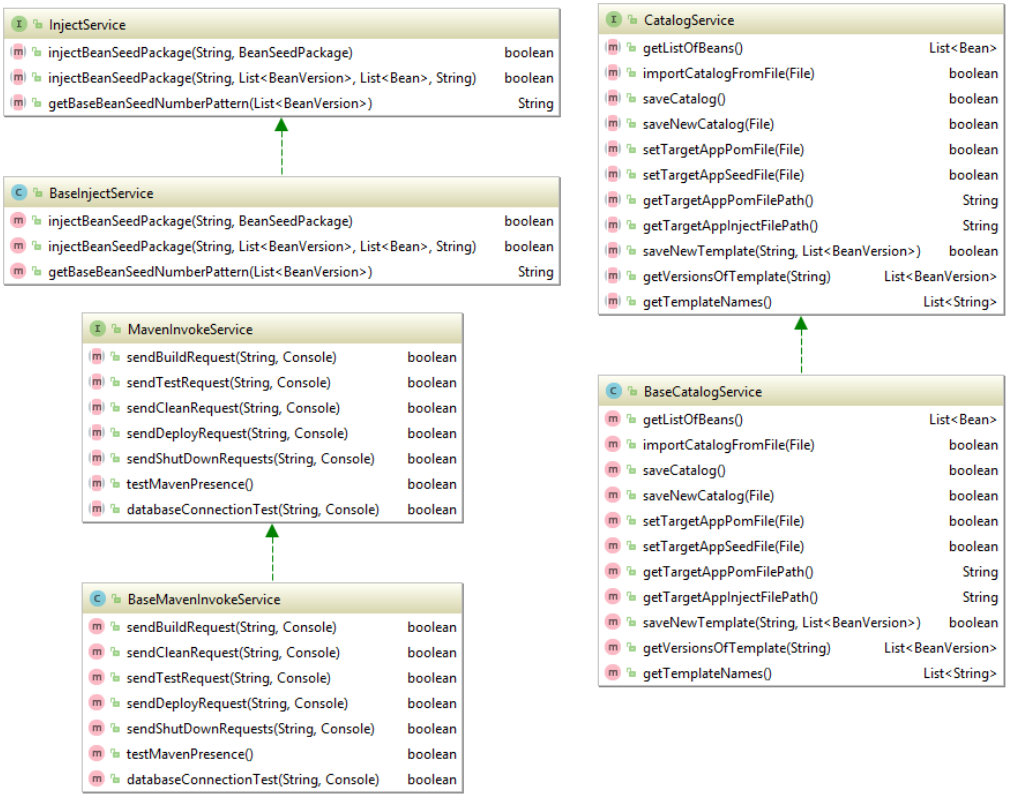
DAO vrstva obsahuje pouze jeden DAO objekt pro přístup k datům katalogu (viz diagram 9.26). Třída tohoto objektu implementuje rozhraní `CatalogDao`, které poskytuje servisní vrstvě použití metod pro přístup k datům katalogu. Rozhraní definuje metody pro získání výchozího katalogu, načtení katalogu ze souboru a uložení rozpracovaného katalogu do souboru. Metoda pro získání výchozího katalogu zkusí načíst soubor katalogu, jehož cesta je uvedena v konfiguračním souboru (*application.properties*) Error-Seederu. V případě, že tento soubor najde a úspěšně z něj načte data, vrací tato metoda objekt `Catalog` s daty z tohoto souboru. Metoda pro import katalogu ze souboru provádí podobnou činnost pro soubor, který je předán v parametru metody. Poslední metoda pro uložení rozpracovaného katalogu do souboru provádí opačnou činnost předchozích metod.



Obrázek 9.26: Diagram tříd DAO vrstvy

9.3.4 Servisní vrstva

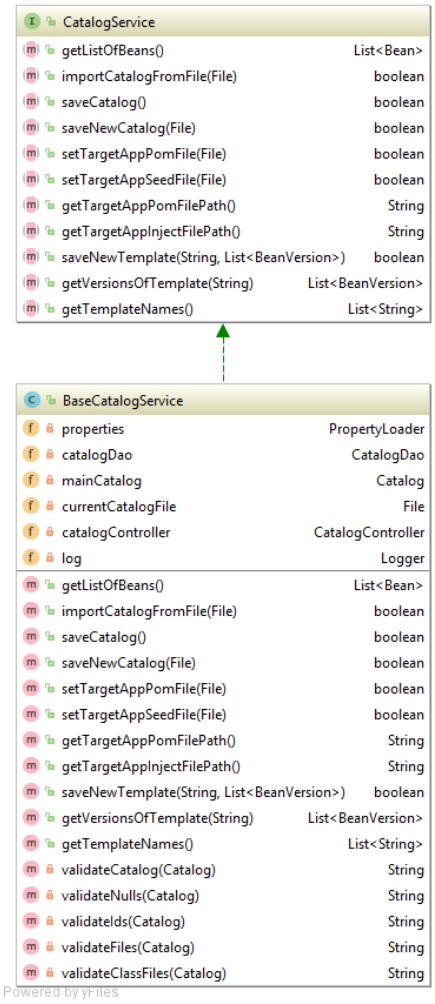
Servisní vrstva obsahuje komponenty, které poskytují služby (viz diagram 9.27). Služby jsou používané zejména kontrolery z řídící vrstvy, které pomocí těchto služeb obsluhují uživatelské požadavky. Servisní vrstva tyto služby poskytuje pomocí rozhraní, které navenek vystavuje. Jednotlivé komponenty, které tyto rozhraní implementují jsou popsány níže.



Obrázek 9.27: Diagram tříd Servisní vrstvy

`CatalogService`

`CatalogService` je objekt servisní vrstvy, který poskytuje služby pro práci s katalogem. Obsahuje metody pro získávání dat z katalogu nebo jeho úpravu viz diagram 9.28. Získaná data z tohoto objektu využívá `MainController` pro vytvoření modelu používaného v hlavním pohledu.



Obrázek 9.28: Diagram tříd služby `CatalogService`

`BaseCatalogService` se při své inicializaci pokusí načíst výchozí katalog ze souboru, který je uveden v konfiguračním souboru Error-Seederu jako výchozí souboru katalogu. Dále umožňuje importovat katalog z předaného souboru. Při každém pokusu o import katalogu je provedena validace dat katalogu, která musí splňovat určité podmínky. Validace testují, zda jsou data korektně načtena, zda jsou veškeré identifikátory unikátní a zda v cílovém projektu UIS-web existují všechny soubory, se kterými bude Error-Seeder pracovat (soubory Java tříd, `pom.xml` soubor a `seed.xml` soubor). `BaseCatalogService` dále umí uložit aktuálně používaný katalog do souboru, ze kterého vznikl, nebo do nového souboru.

Metodami `BaseCatalogService` objektu lze dále vytvářet předvyplněné uživatelské šablony v katalogu. Pro uchování šablon po ukončení aplikace je nutné uložit používaný katalog. V opačném případě se všechny vytvořené

šablony a další změny provedené v katalogu ztratí.

`BaseCatalogService` objekt je také schopný měnit cesty k souborům v cílovém UIS-web projektu. Cesty k souborům (`pom.xml` soubor a `seed.xml` soubor) je možné nastavit pomocí dvou metod, které tato služba implementuje.

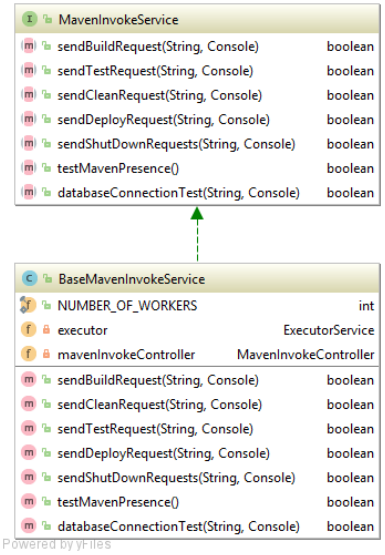
`MavenInvokeService`

`MavenInvokeService` implementuje metody pro vykonávání Maven úkolů v projektu aplikace UIS-web (viz diagram 9.29). K tomuto účelu využívá `MavenInvokeService` nástroj Apache Maven Invoker API, který byl vytvořen v rámci projektu Apache Maven Project jako robustní aplikační rozhraní pro jednoduchou práci s Maven projekty v aplikacích. Hlavním úkolem tohoto API je spouštění Maven úkolů jednoduchým způsobem z dalších aplikací.

Jelikož běh Maven úkolů může být časově náročný, je nutné aby běžel ve vlastním vlákně. K tomu `MavenInvokeService` využívá službu `ExecutorService`, které předává Maven úkoly k vykonání. `ExecutorService` spravuje pool vláken, kterým deleguje přijaté úkoly.

Pro korektní běh služby `MavenInvokeService` je nutné mít na počítači, kde je Error-Seeder spuštěn, správně nainstalovaný a správně nastavený Maven verze 3 a nastavenou proměnnou `M3_HOME` v proměnném prostředí. `MavenInvokeService` získá cestu k Maven instalaci právě pomocí proměnné `M3_HOME`.

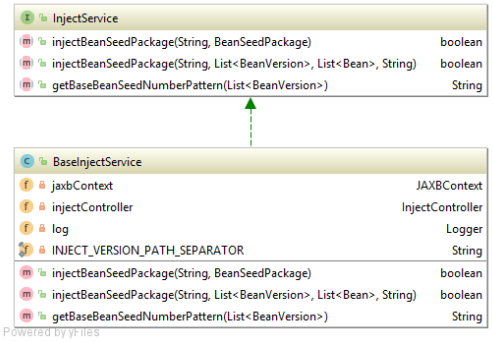
V procesu spouštění Maven úkolů je nejprve vytvořen pomocí tovární metody třídy `MavenInvokeRequestFactory` příslušný `InvocationRequest` objekt, který reprezentuje požadavek k vyvolání Maven úkolu. Dále je vytvořena callback funkce tovární metodou třídy `MavenTaskCallbackFactory`, která bude provedena po dokončení Maven úkolu. Tato callback funkce informuje `MainController` o stavu dokončení Maven úkolu. Z `InvocationRequest` objektu je vytvořen Maven úkol, který je předán `ExecutorService`. Ta ho předá volnému vláknu z poolu vláken ke zpracování. Výstup spuštěných Maven úkolů je přesměrován do předané konzole.



Obrázek 9.29: Diagram tříd služby MavenInvokeService

`InjectService`

`InjectService` poskytuje služby pro injektování verzí bean do cílové aplikace UIS-web. Implementuje a v rámci jeho rozhraní poskytuje metody pro injektování uživatelem vybraného seznamu verzí bean (viz diagram 9.30), ze kterého vytvoří model pro injektování (Seed). Tento model následně transformuje pomocí knihovny JAXB do podoby XML souboru v projektu UIS-web. K vytvoření modelu (Seedu) využívá tovární metodu třídy `SeedFactory`, která model vytvoří z předaného seznamu vybraných verzí bean a jména Seedu.



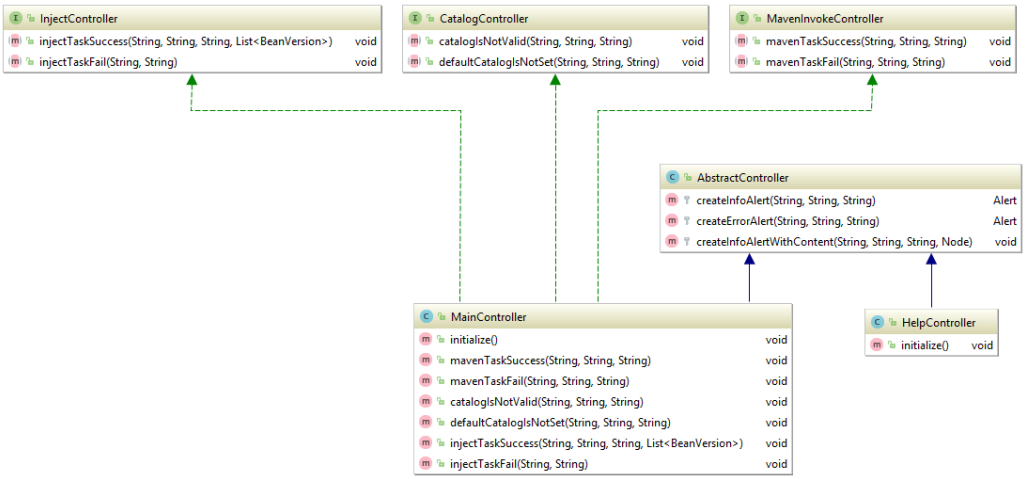
Obrázek 9.30: Diagram tříd služby `InjectService`

`InjectService` dále implementuje metodu pro získání statické části jména Seedu, která vychází z uživatelem vybraného seznamu verzí bean. Tento ře-

těžec popisuje počty chybových verzí bean ve výsledném Seedu. Seed se statickou částí jména „**C1.H2.M3.L4**“ znamená, že ve výsledném Seedu je jedna verze beany s chybou kritické severity, dvě verze s chybou vysoké severity, tři verze s chybou střední severity a čtyři verze beany s chybou nízké severity.

9.3.5 Řídící vrstva

Řídící vrstvu v Error-Seederu představují dva kontrolery a rozhraní, pomocí kterých komunikují se servisní vrstvou viz diagram 9.31. Tyto kontrolery zpracovávají uživatelské požadavky vytvořené v prezentační vrstvě uživatelem. Pro obsluhu požadavků používají kontrolery komponenty servisní vrstvy pro provedení určitých akcí nebo získání dat zobrazovaných uživateli.



Obrázek 9.31: Diagram tříd řídící vrstvy

`HelpController` obsluhuje uživatelské požadavky pocházející z pohledu nápovědy. Uživatelské požadavky se zde týkají změny obrázku nápovědy a souvisejícího textu, který obrázek popisuje. Požadavky uživatel zasílá dvěma tlačítky, které se v pohledu nápovědy nacházejí.

`MainController` obsluhuje požadavky, které uživatel posílá z hlavního pohledu aplikace. Jedná se hlavně o požadavky pro spuštění Maven úkolů, spuštění procesu injektování verzí bean do UIS-web, vytváření uživatelských šablon, požadavky na práci s katalogem atd.

K obsluze těchto požadavků většinou využívá servisní vrstvu pro získání dat nebo provedení určitých operací. Pro zpětnou vazbu poskytuje kompo-

nentám servisní vrstvy několik rozhraní, které definují metody pro informování uživatele o výsledcích požadovaných operacích.

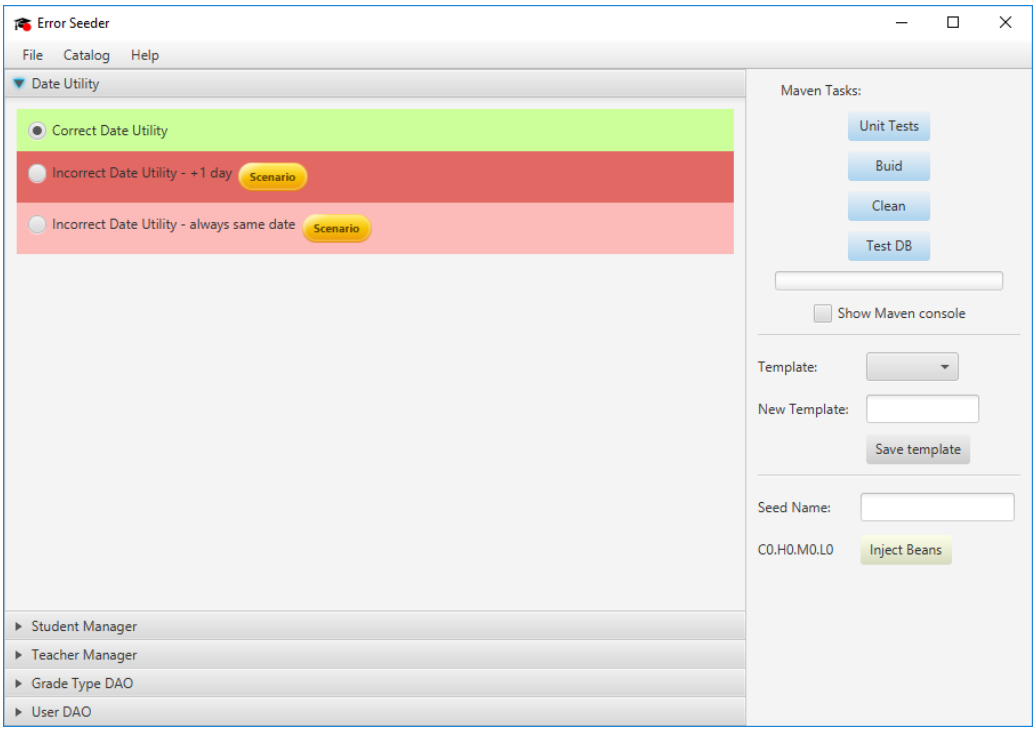
V řídící vrstvě existuje také třída abstraktního kontroleru `AbstractController`, který implementuje často používané metody pro vytváření informačních oken různých typů. Ostatní kontrolery řídící vrstvy tento kontroler rozšiřují a mohou tak používat jeho metody.

9.3.6 Prezentační vrstva

Prezentační vrstva aplikace je tvořená dvěma pohledy, pomocí kterých komunikuje uživatel s aplikací. Tyto pohledy jsou vytvořeny FXML popisem struktury pohledů. Prezentační vrstva využívá `FXMLLoader`, který z FXML definice vytvoří pohledy s komponentami uživatelského rozhraní.

V Error-Seeder aplikaci existují dva pohledy. Jednodušší pohled používá uživatel jako nápovědu k této aplikaci. Pohled zobrazuje pomocný obrázek a dvě tlačítka, které po jejich stisknutí iniciují akci změny obrázku nápovědy. K tomuto obrázku je vždy zobrazen text v dolní částí pohledu, který obrázek popisuje (viz kapitola A.1.3).

Druhý pohled uživatel používá pro ovládání aplikace. Jsou zde zobrazeny všechny komponenty uživatelského rozhraní, kterými je možné provádět práci s katalogem, výběr verzí bean, které chce uživatel vkládat do UIS-web, spouštění Maven úkolů, vytváření uživatelských šablon a spouštění samotného procesu injekce (viz obrázek pohledu 9.32).



Obrázek 9.32: Hlavní pohled aplikace Error-Seeder

V centrální části pohledu je zobrazena komponenta uživatelského rozhraní typu `Accordion` (rozbalovací výčty), která umožňuje zobrazit a skrýt verze jednotlivých bean. Zobrazené verze bean si může uživatel prohlédnout, případně vybrat pro vytvoření injektovaného Seedu nebo pro vytvoření vlastní šablony. U jednotlivých verzí je možné zobrazit tooltip s popiskem verze. U chybových verzí je možné také zobrazit scénář pro odhalení její chyby. Jak lze vidět z obrázku 9.32, jednotlivé verze bean jsou barevně odlišené podle závažnosti chyby, kterou implementuje, nebo zda se jedná o korektní verzi.

V pravé části je zobrazen panel, který má tři části. V první části je možné spouštět jednotlivé Maven úkoly popsané v kapitole 8.2.6. Je zde zaškrtávací pole umožňující zobrazovat a skrývat konzoli, do které spuštěné Maven úkoly přesměrovávají svůj výstup. V druhé části jsou ovládací prvky pro výběr existující nebo vytvoření nové šablony. Při výběru existující šablony se centrální část předvyplní verzemi bean z této šablony. V poslední části pohledu je možné spouštět proces injektování. Je zde textové pole, ve kterém může uživatel definovat jméno Seedu, který hodlá injektovat do UIS-web. Tlačítko v této části spustí proces injektování. Před tímto tlačítkem je statická část jména s informací kolik a s jakou závažností se ve výsledném Seedu

bude nacházet chybových verzí bean.

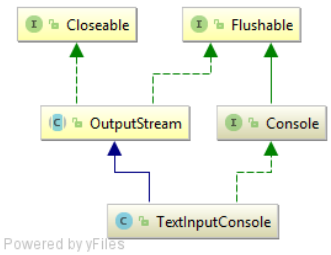
O výsledku všech operací je uživatel informován pomocí vyskakovacích informačních oken s informací o jakou akci se jedná a s jakým výsledkem byla dokončena. V případě chybového dokončení nějaké operace by měl být součástí také popis, proč k chybě došlo.

Součástí hlavního pohledu je také menu, ve kterém může uživatel ukončit aplikaci, provádět načtení, uložení a další akce týkající se katalogu, a zobrazení pohledu s nápovědou. Vybrané akce položek menu lze vyvolat také klávesovou zkratkou.

9.3.7 Utility vrstva

Utility vrstva obsahuje dvě komponenty, které jsou v Error-Seeder aplikaci využívané. První je jednoduchá komponenta pro načítání hodnot z konfiguračního souboru aplikace `application.properties`, ze kterého aplikace načítá různé konfigurační hodnoty potřebné k její činnosti.

Druhá komponenta je použitá pro vytváření konzole pro sledování činnosti probíhajících Maven úkolů viz diagram 9.33. Tato konzole bude zobrazená v grafickém uživatelském rozhraní, kde bude přístupná uživateli aplikace. Také jí budou používat jednotlivé spuštěné Maven úkoly, které do ní budou přesměrovávat svůj výstup.



Obrázek 9.33: Diagram komponenty Console

Velikost a rychlost výstupního proudu těchto úkolů může být velmi velká. Při zapisování každého jednotlivého znaku do konzole by došlo k vysokému zatížení vlákna uživatelského rozhraní. Z tohoto důvodu má konzole buffer, do kterého je výstup z Maven úkolů vkládán, a při naplnění bufferu je pak následně text vložen do konzole. Při ukončení úkolu se do konzole vloží data, která v bufferu zbyla a buffer se vyprázdní.

10 Odhalení chyb z chybových modulů

Pro chybové entity, vytvořené v rámci této práce, bylo nutné navrhnout několik způsobů pro odhalení úmyslně vložených chybových funkcionalit. V následujících kapitolách jsou uvedené scénáře a sada funkčních testů, které jsou schopné jednoznačně odhalit chyby všech vytvořených chybových entit.

10.1 Scénáře odhalení

Ke každé chybové verzi bean, která byla implementována v rámci této práce, existuje scénář, který úmyslně vloženou chybu jednoznačně odhaluje. Tyto scénáře jsou součástí přílohy B a také je možné je zobrazit v aplikaci ErrorSeeder. Krok scénáře, který je v příloze vyznačen červenou barvou, odhaluje vloženou chybu.

Scénáře jsou závislé na staticky inicializovaném stavu databáze, ve kterém se aplikace UIS-web nachází bezprostředně po jejím spuštění. Po manipulaci uživateli s datovým modelem UIS-web aplikace, nemusejí být uvedené scénáře platné.

Všechny scénáře předpokládají stejný počáteční stav, kdy se nepřihlášený uživatel nachází na úvodním pohledu (homepage) aplikace UIS-web. Všechny scénáře odhalení jsou popsané v příloze B.

10.2 Funkční testy

V UIS-web projektu je vytvořena obsáhlá sada jednotkových testů, které testují jednotlivé dílčí funkce komponent aplikace, ale i komplexnější funkce aplikace. Sada testů pokrývá téměř celý kód aplikace a testy jsou schopné odhalit všechny implementované chyby v chybových entitách popsaných v kapitole 9.2. Tyto testy jsou dostatečně komentovány dokumentačními komentáři a jsou vhodně pojmenované, aby bylo možné identifikovat chybu v případě, že tyto testy selžou. Součástí řešení je JavaDoc dokumentace těchto testů.

10.3 Další způsoby

Dalším způsobem odhalením zanesené chyby v UIS-web je např. nahlédnutí do logu běžící aplikace, ve kterém je možné chybu nalézt a identifikovat. Chyba se do logu aplikace zapíše ve chvíli, kdy nastane. Chybový záznam logu obsahuje datum a čas, kdy chyba nastala, jméno chybové třídy a krátký popis začínající řetězcem „`Deliberate error`“, který vzniklou chybu popisuje. Logger Log4j směruje logy do konzole aplikace, ale je možné ho jednoduše rozšířit o logování do souboru v příslušném konfiguračním souboru *log4j.properties*.

Dále je v kódu možné chybovou funkcionalitu odhalit pomocí anotace `@ErrorMethod`, která označuje metodu s implementovanou chybou. Anotace obsahuje parametr `errorMessage` s krátkým popisem implementované chyby. V této metodě by se měl nacházet také příkaz zápisu do logu aplikace, který je popsaný výše.

11 Dosažené výsledky

V této práci byly vytvořeny dvě aplikace. První je webová aplikace UIS-web, která může běžet na webovém serveru a uživatelé ji mohou používat pomocí webového prohlížeče jako klienta. Aplikace funguje podle očekávání, splňuje všechny zadané funkční i mimofunkční požadavky a všechny zadané případy užití. Aplikace ošetřuje veškeré vstupy od uživatele a počítá tak i s nevalidními daty. Pro úschovu a případnou úpravu dat byl vytvořen import a export, který aplikace umožňuje. Import nevalidních dat je rovněž ošetřen. Pro zajištění validity UIS-web aplikace byla použita relativně velká množina různých druhů testů. Důsledné otestování aplikace byl jeden z požadavků na tuto práci. Dalšími splněnými požadavky byla vhodná architektura aplikace, která umožňuje snadné rozšíření aplikace o chybové entity, a implementace způsobu, jak jimi nahradit korektní entity.

Druhou vytvořenou aplikací je deskopová aplikace Error-Seeder pro vkládání chybových entit do první UIS-web aplikace. Aplikace opět funguje dle očekávání a ovládá se intuitivně. S nevalidními vstupy uživatele aplikace počítá a ošetřuje je. Hlavním datovým vstupem je XML soubor, který aplikace validuje pro kontrolu, zda jsou data v konzistentním stavu. Navíc aplikace Error-Seeder umožňuje uživateli spouštět sadu připravených testů v aplikaci UIS-web pro ověření, zda se ve výsledné aplikaci nachází zanesené chyby, nebo se jedná o verzi (klon) bez chyb. Také je možné z Error-Seederu spustit proces sestavení UIS-web pro jednoduchou a rychlou distribuci aplikace.

11.1 Diskuse výsledků

V rámci této práce byly vytvořeny dvě aplikace, kterými bylo zadání práce splněno. Dle mého názoru mají obě vytvořené aplikace velký potenciál pro budoucí používání. Jsou připravené na budoucí rozšiřování a v této práci jsou demonstrovány jejich možnosti. Práce tak může sloužit např. pro ověřování testovacích metod a trénování testerů v univerzitním prostředí. Obě aplikace také mohou sloužit jako prototypy pro vývoj dalších podobných aplikací.

V této práci je navrženo několik způsobů snadné implementace chybových entit aplikace UIS-web, kterými bude moci uživatel Error-Seederu snadno nahrazovat korektní entity. Uživatel si tak může velice snadno a rychle vytvořit libovolné množství různých verzí aplikace UIS-web s různými chybami.

Podle zadání práce měla UIS aplikace existovat také v desktopové verzi. Z rozhodnutí vedoucího práce se tato verze nakonec neimplementovala. Webová verze pro účely této práce byla shledána jako plně dostačující a také nabízí více možností pro black-box testování. Navíc webové aplikace jsou dnes aktuálnější a i z pohledu testování zajímavější. Z tohoto důvodu desktopová verze nebyla vytvořena a vyhrazený čas na její vývoj byl věnován vylepšování a rozšiřování webové verze. V případě budoucí implementace desktopové verze lze díky použité architektuře využít většinu kódu webové verze pouze s drobnými změnami a úpravou prezentační vrstvy.

11.2 Ověření kvality

V rámci této práce byla vytvořena sada automatických testů a provedeno nesčetně manuálních testů pro zaručení kvality vytvořeného software. Z povahy vytvářeného řešení byl na testování kladen velký důraz.

11.2.1 Jednotkové testování

Pro jednotkové testování byl zvolen framework JUnit verze 4.12, který nabízí ověřené a robustní nástroje pro jednotkové testování Java aplikací. Pomocí JUnit frameworku byly vytvořeny nejen jednotkové testy, ale také integrační a regresní testy.

UIS-web

V aplikaci UIS-web je pro testování použita také Spring komponenta pro testování verze 4.3.2, která poskytuje nástroje pro testování aplikací postavených na frameworku Spring (nebo jeho částech).

Pro UIS-web aplikaci byla vytvořena sada 216 jednotkových testů, která testuje jednotlivé dílčí části aplikace s různými vstupy pro otestování různých větví programu. Jednotkové testy většinou pokrývají přes 80 % kódu pro jednotlivé entity programu. Často to bývá i přes 90 %.

Error-Seeder

Pro testování aplikace Error-Seeder existuje sada automatizovaných testů, které byly vytvořeny také pomocí frameworku JUnit. Pro testování s použitím komponent uživatelského rozhraní byla použita knihovna TestFX verze 4.0.1-alpha, která umožňuje jednoduché a korektní testování JavaFX aplikací a jejich komponent.

11.2.2 Funkcionální testování statického obsahu

Pro UIS-web aplikaci byla (vedoucím práce) vytvořena sada několika stovek funkčních testů, které používají kombinaci frameworků JUnit a Selenium, pro testování statického obsahu jednotlivých stránek. Mezi statický obsah patří aktivní i pasivní prvky stránek.

Testy jsou logicky strukturované do různých kategorií podle druhu testů. Je tak možné spustit pouze některé kategorie testů namísto spouštění všech testů. Tato vlastnost může ušetřit čas a usnadnit práci s testy.

11.2.3 Funkcionální testování podle scénářů

Pro UIS-web aplikaci byla dále (vedoucím práce) vytvořena sada funkcionálních testů, které testují různé scénáře pomocí kombinace frameworků JUnit a Selenium.

11.2.4 Manuální testování

UIS-web aplikace byla dále testována četnými manuálními testy od různých lidí, kteří aplikaci ručně testovali. Tímto testování bylo ověřeno splnění všech případů užití, které byly v aplikaci požadovány. Také byla ověřena stabilita UIS-web aplikace a její ošetření různých vstupů od uživatele. Dále byly použity techniky průzkumného testování a odhady chyb (error guessing).

11.2.5 Zátěžové testy

Pro UIS-web aplikaci bylo provedeno několik zátěžových testů, při kterých bylo spuštěno několik funkčních testů statického obsahu běžících na různých zařízeních ve stejné sítí jako běžící UIS-web aplikace. Všechny funkční testy statického obsahu úspěšně doběhly a ověřily tak dostupnost a stabilitu UIS-web aplikace.

11.2.6 Statická kontrola kódu

Pro zajištění kvalitního kódu byl pro obě aplikace použit komplexní nástroj Code Inspection, který je integrovaný v použitém vývojovém prostředí IntelliJ IDEA. Code Inspection nabízí rozsáhlé možnosti pro robustní a flexibilní statickou analýzu kódu. Detekuje syntaktické chyby použitého programovacího jazyka a runtime chyby. Navrhuje opravy chyb a další vylepšení kódu v nejrůznějších případech (např. potenciálně nebezpečný kód nebo neošetřené vstupy metod). Dokáže také detekovat neefektivní kód a nabídnout efek-

tivnější alternativu. Také kontroluje nedosažitelný, nepoužívaný nebo duplicitní kód. Nabízí kontroly různých technologií (např. JSP), jazyků (např. XML, CSS, JavaScript), frameworků (Spring, JPA) a dokonce pravopisu. Nalezené problémy, chyby a výstrahy roztřídí do kategorií a seřadí podle priority. Nalezené závažné nedostatky byly díky tomuto nástroji z obou aplikací odstraněny.

11.3 Omezení a hardwarové nároky

Obě aplikace jsou napsané v jazyce Java (Java SDK 1.8). To umožňuje jejich běh na různých platformách, na kterých je nainstalované běhové prostředí JRE.

UIS-web aplikace vyžaduje pro svůj běh prostředí webového serveru se servlet kontejnerem např. Apache Tomcat 7, na kterém byla aplikace testována, nebo Jetty.

Pro bezproblémový běh aplikace Error-Seeder je doporučeno použít Java SE Runtime Environment 8, ve kterém byla testována. Pro korektní činnost Error-Seederu je nutné mít správně nainstalovaný a nastavený Maven 3.0 a vyšší. Také je nutné mít korektně nastavené systémové proměnné JAVA_HOME a M3_HOME.

Pro bezproblémový běh aplikace UIS-web je doporučeno alespoň 1 GB volné operační paměti. Aplikace bude pravděpodobně fungovat i s menším množstvím dostupné operační paměti, nicméně není zaručen její maximální výkon.

11.3.1 UIS-web – Podpora prohlížečů

V následujících tabulkách jsou znázorněné podporované prohlížeče, pro které je použitým CSS frameworkem Bootstrap předpokládané bezproblémové fungování aplikace UIS-web. Zvlášť jsou uvedené prohlížeče mobilních a desktopových zařízení.

	Android	**iOS**	**Windows 10 Mobile**
Chrome	OK	OK	N/A
Firefox	OK	OK	N/A
Safari	N/A	OK	N/A
Android Browser	Android v5.0 + OK	N/A	N/A
Microsoft Edge	OK	OK	OK

Tabulka 11.1: Podporované prohlížeče mobilních zařízení

	Mac	Windows
Chrome	OK	v10+ OK
Firefox	v6+ OK	v6+ OK
Internet Explorer	N/A	IE 10+ OK
Microsoft Edge	N/A	OK
Opera	v10+ OK	v10+ OK
Safari	OK	Nepodporováno

Tabulka 11.2: Podporované prohlížeče desktopových zařízení

11.4 Budoucí práce

Budoucí práce by se měly zaměřit zejména na vytváření nových chybových verzí bean, které by měly v rámci UIS-web aplikace sloužit k ověřování kvality existujících a nově vyvíjených testovacích metod jako SUT a pro trénink testerů. Implementovanou množinu chybových verzí entit lze například rozšířit o chybové entity v řídící vrstvě.

V souvislosti s rozšiřováním kolekce chybových modulů by se další vývoj měl také zaměřit na rozšiřování UIS-web aplikace o další funkce. Vzhledem k použité architektuře (a technologiím) by rozšiřování aplikace o další funkce mělo být velice jednoduché a bezproblémové.

Velkým přínosem by bylo rozšíření současného řešení o nástroj, který by prováděl logování v okamžiku začátku vykonávání kódu metody, která je označena připravenou anotací *ErrorMethod*. Zápisy tohoto speciálního logu by popisovaly nastalou chybovou činnost, která je implementovaná v metodě s touto anotací. Zápisy logu by se ukládaly do tabulky databáze pro jejich snadnější zpracování.

Tato a další vylepšení by jistě zvýšily hodnotu a použitelnost této práce, která má vysoký potenciál pro praktické použití, nejen v univerzitním prostředí.

12 Závěr

V rámci této práce se podařilo vytvořit SUT Java webovou aplikaci UIS-web, která simuluje funkčnost reálného univerzitního informačního systému. Aplikace využívá vybrané komponenty frameworku Spring, které velice usnadnily a urychlily vývoj. V UIS-web existují uživatelé typu student a učitel, kteří se mohou do aplikace přihlásit. Oba typy uživatelů mají přístup do odlišných částí aplikace, ve kterých mohou provádět různé akce. Jako hlavní zdroj dat používá aplikace databázi a jako datové vstupy a výstupy lze použít XML a JSON soubory.

Nejdůležitější využívanou funkcí Spring frameworku je Dependency injection, která umožňuje aplikaci skládat z dílčích částí tzv. bean s velice volnou vazbou. Bean může v UIS-web aplikaci existovat v několika verzích implementace. Tyto verze bean jsou děleny na korektní (bezchybné) a úmyslně chybové s tím, že musí být implementovaná právě jedna korektní verze pro každou beanu. Je tedy možné UIS-web sestavit pouze z korektních verzí a vytvořit aplikaci bez chyb. Chybové verze bean implementují podobnou funkcionalitu jako korektní verze stejné beany, nicméně s úmyslně vloženou chybou. Součástí práce je několik vybraných chybových verzí bean, které demonstrují funkčnost implementovaného řešení.

Dále byla navržena a implementována aplikace Error-Seeder pro úmyslné injektování chyb do aplikace UIS-web. Error-Seeder definuje, které verze bean má Spring Dependency injection v UIS-web používat pro vkládání závislostí. Volbu používaných verzí bean provádí uživatel pomocí uživatelského rozhraní Error-Seederu. V aplikaci je možné spustit proces sestavení UIS-web pro snazší distribuci a spouštět testy, které odhalují vložené chyby.

K implementovaným chybovým verzím v aplikaci UIS-web byly vytvořeny scénáře, díky kterým může uživatel jednoznačně chyby v aplikaci identifikovat. Tyto scénáře jsou přístupné k nahlédnutí v aplikaci Error-Seeder. Také byla vytvořena obsáhlá sada funkčních testů, které všechny chyby v implementovaných chybových verzích spolehlivě odhalují. V rámci práce byla také implementována anotace, která s krátkým popisem v kódu označuje metodu s chybnou implementací. Chyby je možné odhalit také nahlédnutím do logu aplikace, ve kterém je vytvořen zápis ve chvíli, kdy chyba nastane.

UIS-web aplikace plně pokrývá požadavky, které na ní byly kladeny. Díky navržené architektuře a zvoleným technologiím je aplikace připravena na další rozšiřování a bude sloužit jako dobrý základ pro budoucí vývoj. Existuje několik různých jednoduchých způsobů, jak aplikaci rozšířit o chybové verze

bean. Hotové řešení může sloužit pro řízené experimenty, ověřování kvality testovacích metod a edukační účely v univerzitním prostředí.

Přehled zkratek

AOP	Aspect Oriented Programming
API	Application Programming Interface
CBSE	Component-Based Software Engineering
CD	Compact Disc
CSRF	Cross-Site Request Forgery
CSS	Cascading Style Sheets
CSV	Comma-separated values
CWE	Common Weakness Enumeration
DAO	Data Access Object
DI	Dependency Injection
ECMA	European Computer Manufacturers Association
EJB	Enterprise JavaBeans
ER diagram	Entity-Relationship diagram
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IoC	Inversion of Control
J2EE	Java Enterprise Edition
JAR	Java Archive
JAXB	Java Architecture for XML Binding
JAX-WS	Java API for XML Web Services
JDBC	Java Database Connectivity
JMS	Java Message Service
JPA	Java Persistence API
JPQL	Java Persistence Query Language
JRE	Java Runtime Environment
JSON	JavaScript Object Notation
JSP	Java Server Pages
JSR	Java Specification Request
JSTL	JSP Standard Tag Library
JTA	Java Transaction API
MOXy	kódový název pro EclipseLink Object-XML
MVC	Model-view-controller
ORM	Object-relational mapping
OSGi	Open Services Gateway initiative
PHP	Hypertext Preprocessor

POM	Project Object Model
RESTful API	Representation State Transfer Application Programming Interface
RFC	Request For Comments
RMI	Remove Method Invocation
SAX	Simple API for XML
SGML	Standard Generalized Markup Language
SIR	Software-artifact Infrastructure Repository
SoC	Separation of Concerns
SQL	Structured Query Language
SŘBD	Systém řízení báze dat
SUT	System Under Test
UC	Use Case
UIS	University Information System
URL	Uniform Resource Locator
WAR	Web Archive
XML	eXtensible Markup Language
XSD	XML Schema Definition
XSS	Cross-site scripting

Literatura

[1] Budd, Timothy Alan *Mutation Analysis of Program Test Data.* New Haven, CT, USA: Yale University, 1980.

[2] ADITYA P. MATHUR. *Foundations of software testing: fundamental algorithms and techniques : an undergraduate and graduate text, a reference for the practicing software engineer.* 2. impr. Delhi: Pearson Education, 2008. ISBN 9788131716601.

[3] *Mutation Testing: Complete Guide* [online]. [cit. 2018-04-27]. <`https://www.guru99.com/mutation-testing.html`>

[4] *Mutator* [online]. [cit. 2018-04-27]. <`http://ortask.com/mutator/`>

[5] *μJava* [online]. [cit. 2018-04-27]. <`https://cs.gmu.edu/~offutt/mujava/`>

[6] *BACTERIO MUTATION TEST SYSTEM* [online]. [cit. 2018-04-27]. <`https://alarcos.esi.uclm.es/per/preales/bacterio/bacteriomanual.pdf`>

[7] *Software-artifact Infrastructure Repository* [online]. [cit. 2018-04-28]. <`http://sir.unl.edu/portal/`>

[8] *CWE – Common Weakness Enumeration* [online]. [cit. 2018-04-20]. <`http://cwe.mitre.org/`>

[9] PATTON, Ron. *Software testing. 2nd ed.* Indianapolis, IN: Sams Pub., c2006. ISBN 9780672327988.

[10] HEROUT, Pavel. *Testování pro programátory.* České Budějovice: Kopp, 2016. ISBN 978-80-7232-481-1.

[11] HEINEMAN, George T. a William T. COUNCILL. *Component-based software engineering: putting the pieces together.* Boston: Addison-Wesley, c2001. ISBN 0-201-70485-4.

[12] *Spring Framework Documentation* [online]. [cit. 2018-03-31]. <`https://docs.spring.io/autorepo/docs/spring-framework/5.0.5.BUILD-SNAPSHOT/spring-framework-reference/`>

[13] *OSGi Framework Documentation* [online]. [cit. 2018-03-31]. <http://enroute.osgi.org/book/210-doc.html>

[14] *Enterprise JavaBeans Technology Specification* [online]. [cit. 2018-03-31]. <http://www.oracle.com/technetwork/java/javaee/ejb/>

[15] *Apache Maven Project* [online]. [cit. 2018-04-28]. <https://maven.apache.org/>

[16] *Spring Framework reference documentation* [online]. [cit. 2018-04-21]. <https://docs.spring.io/autorepo/docs/spring-framework/5.1.0.BUILD-SNAPSHOT/spring-framework-reference/>

[17] JOHNSON, Rod. *Professional Java development with the Spring Framework.* Indianapolis, IN: Wiley Pub., 2005. ISBN 978-0-7645-7483-2.

[18] *Spring Security Reference documentation* [online]. [cit. 2018-04-22]. <https://docs.spring.io/spring-security/site/docs/5.0.5.BUILD-SNAPSHOT/reference/html/>

[19] KEITH, Mike. a Merrick SCHINCARIOL. *Pro EJB 3: Java persistence API.* Berkeley: Apress, 2006. ISBN 978-1-59059-645-6.

[20] *The Java EE 6 Tutorial* [online]. [cit. 2018-04-01]. <https://docs.oracle.com/javaee/6/tutorial/doc/>

[21] *Hibernate ORM* [online]. [cit. 2018-04-01]. <http://hibernate.org/orm/>

[22] *XML Tutorial* [online]. [cit. 2018-03-31]. <https://www.w3schools.com/xml/>

[23] *Extensible Markup Language (XML) 1.1 (Second Edition)* [online]. [cit. 2018-03-31]. <https://www.w3.org/TR/xml11/>

[24] *ECMAScript® 2017 Language Specification* [online]. [cit. 2018-04-01]. <http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf>

[25] *Introducing JSON* [online]. [cit. 2018-04-01]. <https://www.json.org/>

[26] Shafranovich, Y. *Common Format and MIME Type for Comma-Separated Values (CSV) Files.* RFC 4180, DOI 10.17487/RFC4180. October 2005. [online]. [cit. 2018-04-23]. <https://tools.ietf.org/html/rfc4180>

[27] *Java Architecture for XML Binding (JAXB)* [online]. [cit. 2018-04-23]. <http://www.oracle.com/technetwork/articles/javase/jaxb-141136.html>

[28] *EclipseLink MOXy documentation* [online]. [cit. 2018-04-25]. <http://www.eclipse.org/eclipselink/documentation/2.7/moxy/toc.htm>

[29] MEANS, W. Scott. a Michael A. BODIE. *The book of SAX: the simple API for XML.* San Francisco: No Starch Press, c2002. ISBN 1886411778.

A Uživatelská příručka

V této příloze je popsaná uživatelská příručka pro aplikace UIS-web a Error-Seeder implementované v rámci této práce. Příručka poskytuje informace nutné pro korektní práci s aplikacemi.

A.1 Error-Seeder

A.1.1 Sestavení aplikace

Pro sestavení aplikace lze použít buildovací nástroj Maven, který z projektu aplikace vytvoří samostatný spustitelný JAR soubor se všemi potřebnými závislostmi. Pro vytvoření tohoto JAR souboru je nutné provést následující Maven příkaz v kořenové složce projektu, kde se nachází POM soubor. Po úspěšném sestavení aplikace bude vytvořený JAR soubor ve výstupní složce *target*.

```
mvn clean install assembly:single
```

A.1.2 Prerekvizity

Pro korektní fungování aplikace Error-Seeder je nutné mít na PC, kde aplikace poběží, nainstalovaný Maven minimálně verze 3.0. Maven musí být také správně nastavený v prostředí systému, kde musí existovat proměnná **JAVA_HOME** (s cestou k používané Javě) a **M3_HOME** (s cestou k použité instalaci Mavenu).

Dále musí být nastavený katalog s daty, které bude aplikace Error-Seeder používat. V následující ukázce je znázorněn formát zápisu jednotlivých bean a jejich verzí v souboru katalogu (reálný soubor je součástí přiloženého CD). Atribut *interfaceName* beany musí být v rámci katalogu unikátní a určuje název rozhraní definující beanu v UIS-web. Id verze beany musí být také v rámci katalogu unikátní.

```
<beans>
<bean>
    <description>Kratky popis beany</description>
    <!-- Jmeno rozhrani v UIS-web -->
    <interfaceName>nameOfInterface</interfaceName>
    <name>Jmeno beany</name>
    <versions>
        <version id="1"> <!-- Verze beany -->
            <name>Jmeno beany</name>
            <description>Kratky popis verze beany</
               description>
            <class>cesta.ke.tride.TridaVerzeBeany</class>
            <versionSeverity>CORRECT/LOW/MEDIUM/HIGH/CRITICAL
               </versionSeverity> <!-- Severita verze beany
               -->
            <scenario> <!-- Scenar odhaleni musi byt pouze u
               chybovych verzi bean -->
                <step number="1">Krok 1</step>
                <step number="2">Krok 2</step>
                ...
            </scenario>
        </version>
        <version id="2"> <!-- id musi byt unikatni -->
            ...
        </version>
    </versions>
</bean>
<bean> <!-- Definice dalsi beany -->
...
</bean>
</beans>
```

V následující ukázce ze souboru katalogu jsou dva povinné elementy katalogu. Jedná se o cesty k souborům v projektu aplikace UIS-web. První element určuje cestu k souboru POM ke spouštění Maven úkolů pro projekt aplikace UIS-web z aplikace Error-Seeder. Druhý element určuje cestu k *seed.xml* souboru s definicí, které verze bean se budou v UIS-web používat.

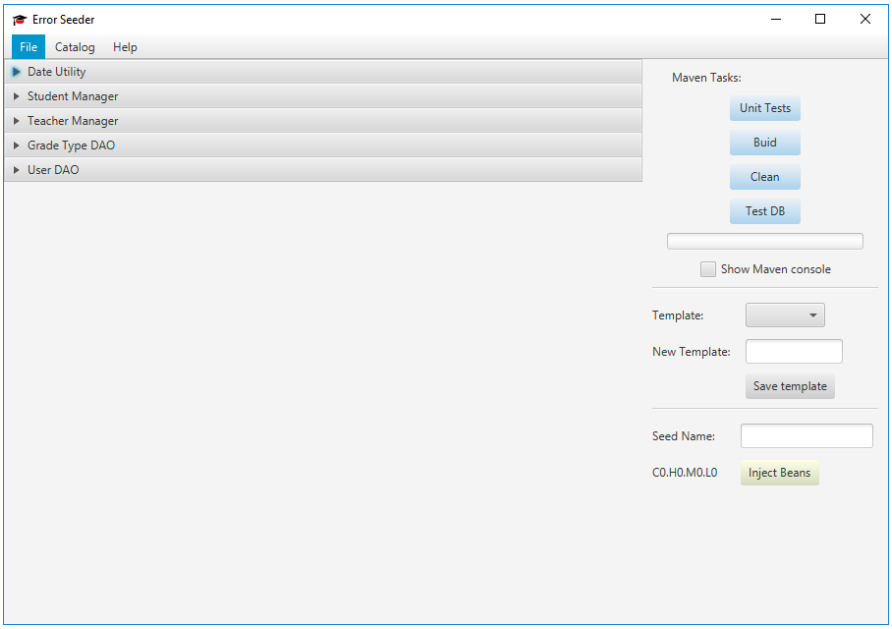
```
<pathToPOM>cesta/k/souboru/pom.xml</pathToPOM>
<pathToBeanSetFile>cesta/seed.xml</pathToBeanSetFile>
```

V poslední ukázce je zobrazen způsob, jakým je možné vytvořit uživatelskou šablonu. V elementu *template* je nutné vyplnit atribut *name*, který definuje jméno uživatelské šablony. Dále je pro každou šablonu uveden seznam elementů s id verzí bean, které šablona obsahuje. Z tohoto důvodu musí být id atributy verzí bean unikátní. Šablony je také možné vytvořit v uživatelském rozhraní aplikace Error-Seeder.

```
<templates>
    <template name="All_OK">
        <versionId>1</versionId>
        <versionId>4</versionId>
        <versionId>7</versionId>
        <versionId>11</versionId>
        <versionId>13</versionId>
    </template>
</templates>
```

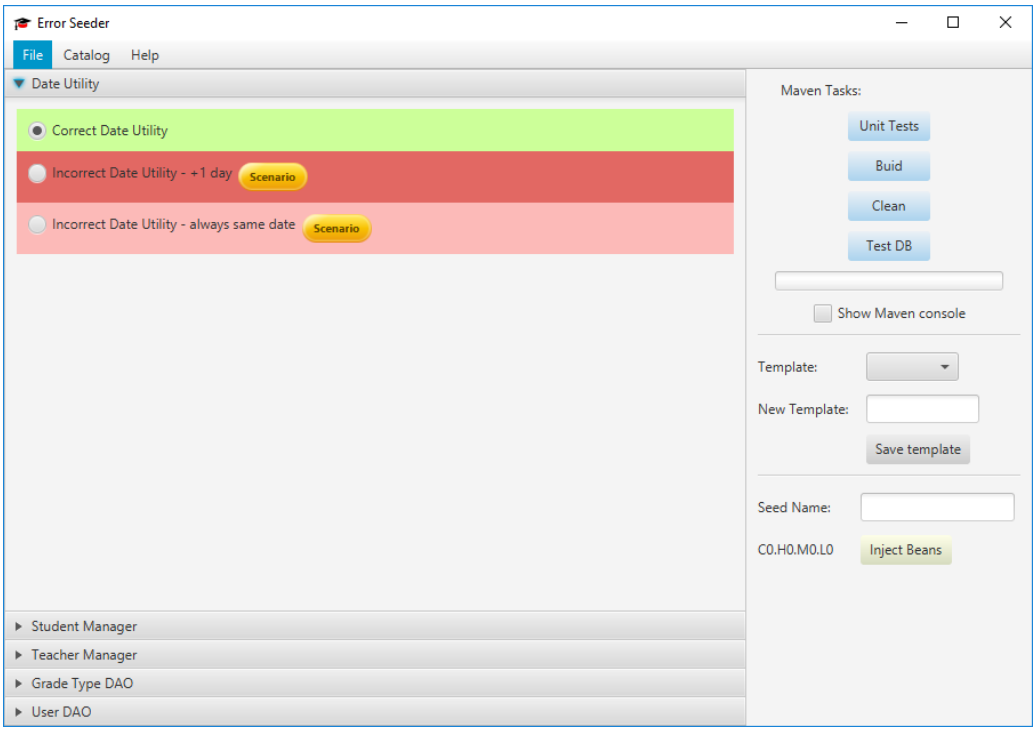
A.1.3 Ovládání aplikace

Aplikace Error-Seeder využívá ke své činnosti data z katalog souboru, který načítá buď ze složky, ve které byla aplikace spuštěna, nebo importem souboru katalogu v běžící aplikaci Error-Seederu. Data z katalog souboru musí být ve správném formátu. V opačném případě neprojdou validačním procesem při importu katalog souboru. Na následujícím obrázku je vidět hlavní pohled aplikace Error-Seeder, který se uživateli zobrazí při spuštění aplikace.



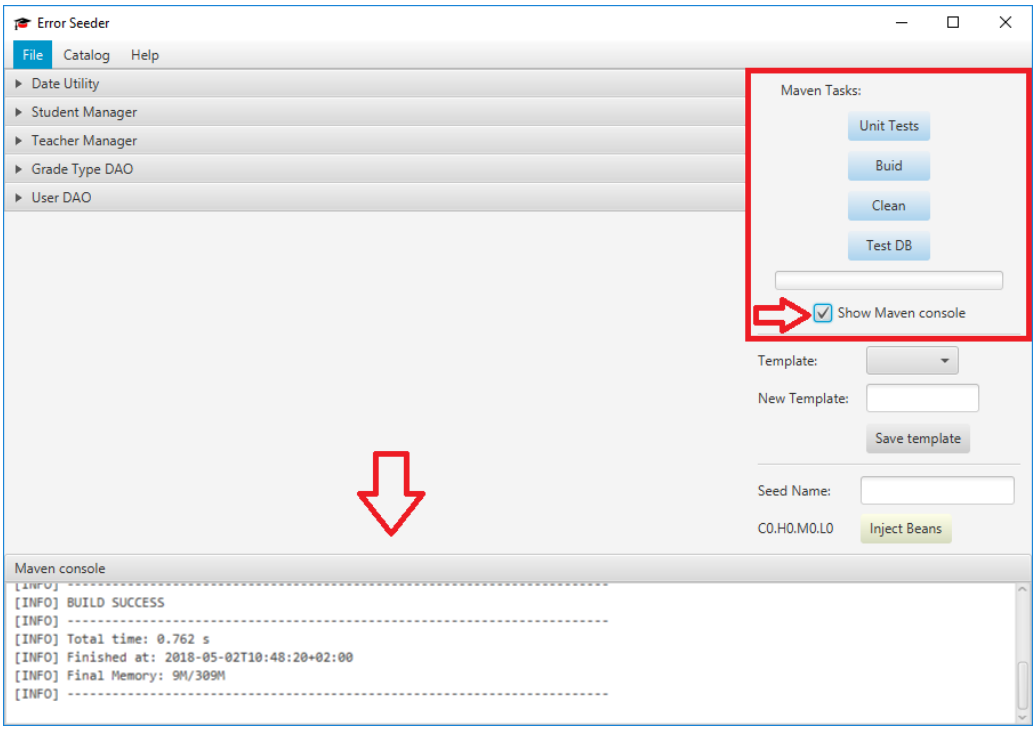
Obrázek A.1: Hlavní pohled

V levé části pohledu se nachází jednotlivé beany aplikace UIS-web, pro které může uživatel nastavit jejich používanou verzi. Pro každou beanu by měla existovat korektní verze bez chyb, která je zvýrazněna zelenou barvou. Pro chybové verze bean vždy existuje scénář, který lze zobrazit kliknutím na příslušné tlačítko *Scenario*. V novém informačním okně se zobrazí posloupnost kroků scénáře, která může být zkopírována do systémové schránky kliknutím na tlačítko.



Obrázek A.2: Beany a jejich verze

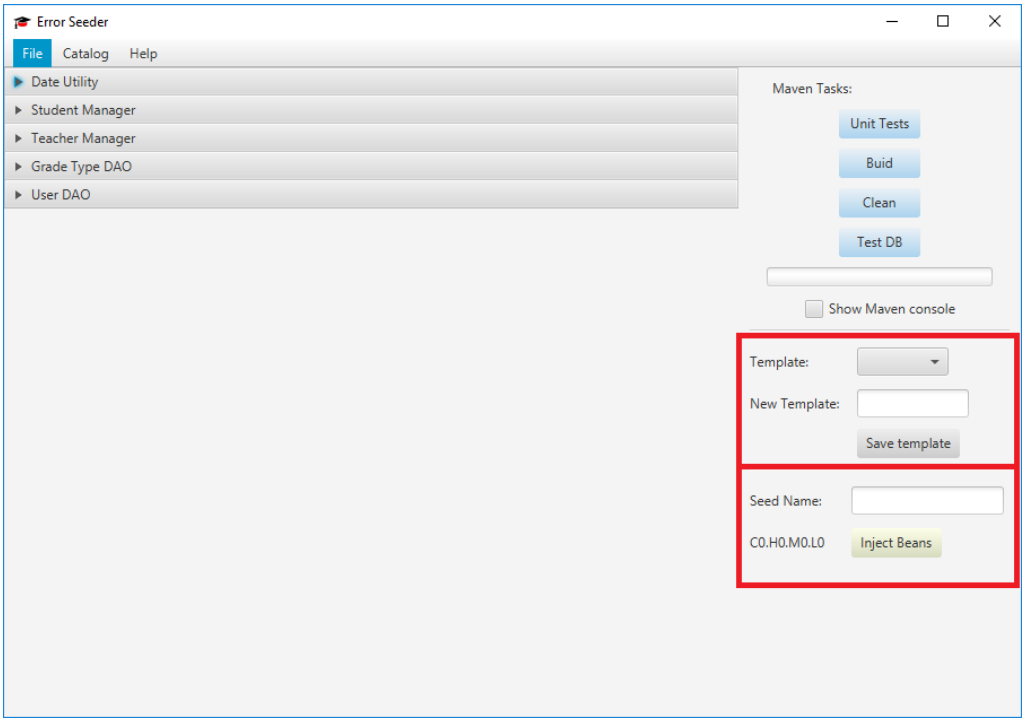
V pravé části hlavního pohledu je zobrazený panel, který má několik částí. V první části (červeně zvýrazněné) se nachází několik tlačítek pro spouštění Maven úkolů nad projektem UIS-web. Uživatel tak může spustit úkol provedení jednotkových testů UIS-web, sestavení, vyčištění výstupní složky, a spuštění testu databázového spojení UIS-web. Pří spuštění Maven úkolu začne progress bar signalizovat probíhající Maven úkol. Po úspěšném i neúspěšném ukončení Maven úkolu je uživatel o této události informován informačním oknem.



Obrázek A.3: Maven úkoly s otevřenou konzolí

Posledním prvkem této části je zaškrtávací okénko, které umožňuje uživateli zobrazovat a skrývat konzoly, do které je převeden výstup spuštěných Maven úkolů.

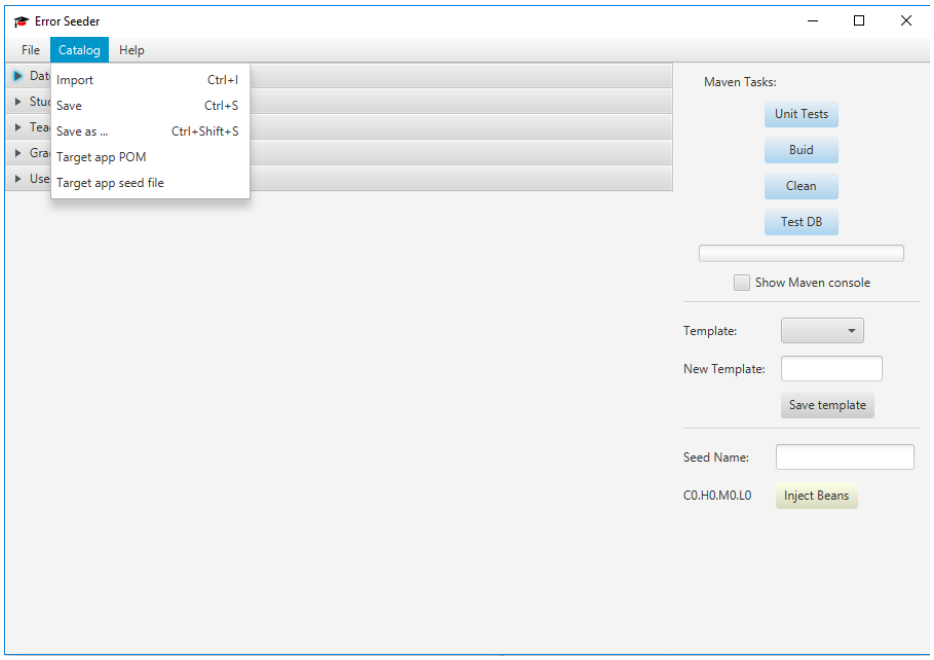
V další části se nachází prvky pro práci s uživatelskými šablonami. Je zde pole pro výběr existující šablony. Po provedení výběru některé z existujících šablon se předvyplní verze bean v hlavním panelu pro všechny beany podle uživatelské šablony. Dále je zde textové pole pro jméno nové šablony a tlačítko, které spouští proces vytvoření nové uživatelské šablony. Uživatel si nejprve vybere verze bean, které chce uložit ve své šabloně, a následně šablonu uloží. Po úspěšném vytvoření šablony se uživateli zobrazí informační okno s touto informací. Pro trvalé uložení nově vytvořených šablon je nutné uložit katalog do souboru (zkratka Ctrl+S).



Obrázek A.4: Uživatelské šablony a Injektování verzí bean do UIS-web

V poslední části se nachází prvky uživatelského rozhraní pro vytvoření nového Seedu s vybranými verzemi bean a jeho injektování do UIS-web projektu. V této části se nachází textové pole pro uživatelskou část jména Seedu, které si může uživatel zvolit libovolně. Dále je zde zobrazena statická část jména Seedu, ze kterého lze poznat kolik chybových verzí bean a s jakou severitou se bude v Seedu nacházet (např. C1.H2.M3.L4 – 1 *Critical*, 2 *High*, 3 *Medium*, 4 *Low*). Nakonec lze vytvořený a pojmenovaný Seed injektovat do UIS-web stisknutím tlačítka *Inject Beans*. Po úspěšném injektování se zobrazí informační okno s tabulkou vložených verzí bean. Celé jméno injektovaného Seedu bude zobrazeno na úvodní straně běžící aplikace UIS-web.

Na dalším obrázku je vidět rozbalené menu s možnými akcemi nad katalogem. Kliknutím na položky z tohoto menu lze načíst nový katalog ze souboru (Ctrl+I), uložit aktuálně používaný katalog do původního katalog souboru (Ctrl+S) nebo do nového katalog souboru (Ctrl+Shift+S) např. po vytvoření nové uživatelské šablony. Dále jsou zde položky pro nastavení souborů pro vkládání Seedu a POM v cílovém UIS-web projektu.



Obrázek A.5: Správa katalogu

V dalším menu *Help* je položka pro zobrazení okna s nápovědou aplikace, která popisuje korektní používání aplikace Error-Seeder. Uprostřed tohoto pohledu je obrázek nápovědy. V dolní části okna je textový popisek, který popisuje konkrétní případ užití Error-Seederu na aktuálně zobrazeném obrázku nápovědy. Po stranách obrázku jsou tlačítka pro přesun na další a předchozí obrázek nápovědy.

Typický postup použití Error-Seederu

1. Uživatel si předpřipraví katalog soubor se jménem *catalog.xml* a vloží ho do složky, ve které je spustitelný JAR soubor Error-Seederu.

2. Uživatel spustí Error-Seeder. Ten by měl automaticky načíst data (beany, verze bean, šablony, cesty k souborům cílového UIS-web projektu) z připraveného katalog souboru. Načtené beany a jejich verze by měly být vidět v levé části hlavního pohledu (viz obrázek A.2).

3. Uživatel si v levé části hlavního pohledu navolí jaké verze bean bude chtít v sestavené aplikaci UIS-web použít.

4. Po výběru verzí bean provede uživatel proces injektování verzí bean do UIS-web aplikace. To provede tak, že vyplní textové pole *Seed Name* libovolným názvem jména nového Seedu a stiskne tlačítko *Inject Beans*.

5. Přítomnost chybových verzí bean může uživatel ověřit spuštěním Maven úkolu pro běh testů aplikace UIS-web. To uživatel provede stisknutím tlačítka *Unit Tests* v horní polovině panelu v pravé části hlavního pohledu.

6. Následně může uživatel vytvořit WAR soubor sestavením UIS-aplikace s vybranými verzemi bean. Vytvoření uživatel provede stisknutím tlačítka *Build* pro spuštěním Maven úkolu sestavení aplikace UIS-web. Po úspěšném sestavení aplikace UIS-web se bude výsledný WAR soubor nacházet ve složce *target* v projektu aplikace UIS-web.

A.2 UIS-web

A.2.1 Sestavení aplikace

Pro sestavení aplikace UIS-web pomocí nástroje Maven do podoby WAR souboru, který je možné nasadit na webový server se servlet kontejnerem (např. Tomcat), je možné použít následující Maven příkaz v kořenové složce UIS-web projektu s POM souborem:

```
mvn clean install -DskipTests
```

Po úspěšném sestavení aplikace bude výsledný WAR soubor pro distribuci aplikace umístěn ve složce *target*. Cíl `-DskipTests` umožňuje přeskočení testů, které by mohly ukončit proces sestavení v případě použití beany s úmyslně vloženou chybou.

A.2.2 Nasazení aplikace

Spuštění aplikace pomocí Maven pluginu

UIS-web aplikaci lze spustit pomocí použitého Maven tomcat7 pluginu, který umožňuje manipulovat s WAR projekty uvnitř vestavěného servlet kontejneru Tomcat verze 7.x. Pro nové spuštění aplikace pomocí WAR souboru s čistým sestavením lze použít následující příkaz:

```
mvn clean install -DskipTests tomcat7:run-war
```

Cíl *-DskipTests* definuje přeskočení testů, které by mohly ukončit proces sestavení s úmyslně vloženou chybovou verzí beany.

Nasazení UIS-web na Tomcat server

Sestavenou aplikaci v podobě WAR souboru lze nasadit na webový server se servlet kontejnerem jako např. Tomcat. War soubor je na Tomcat server možné nasadit dvěma způsoby. První je vložit vytvořený WAR soubor do složky s ostatními webovými aplikacemi a server spustit (nebo restartovat). Tomcat složku prohlédne a všechny nové nebo aktualizované webové archivy WAR rozbalí a zpřístupní webovou aplikaci. Druhá možnost je tzv. *Hot Deployment*, která umožňuje nasadit samostatnou web aplikaci za běhu serveru. Na serveru Tomcat lze *Hot Deployment* použít pomocí nástroje ke správně serveru *Tomcat Manager*.

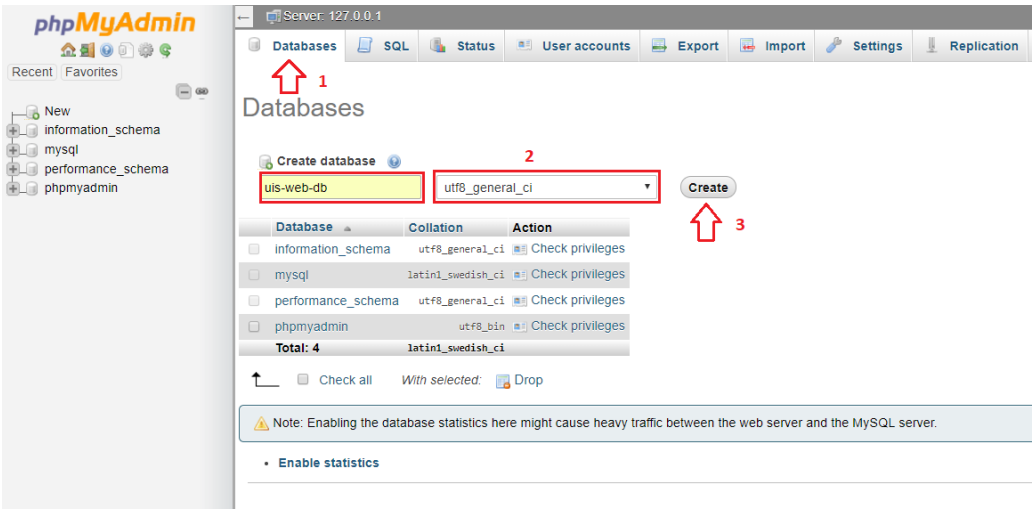
A.2.3 Nastavení databáze

Před prvním spuštěním aplikace UIS-web je nutné vytvořit novou databázi, kterou bude aplikace používat pro persistenci dat. Pro používání této databáze aplikací UIS-web je také nutné nastavit atributy user, password, url databáze (jsou zde i další atributy) pro JDBC konektor databáze v souboru (cesta ve WAR souboru) *WEB-INF/classes/META-INF/persistence.xml*.

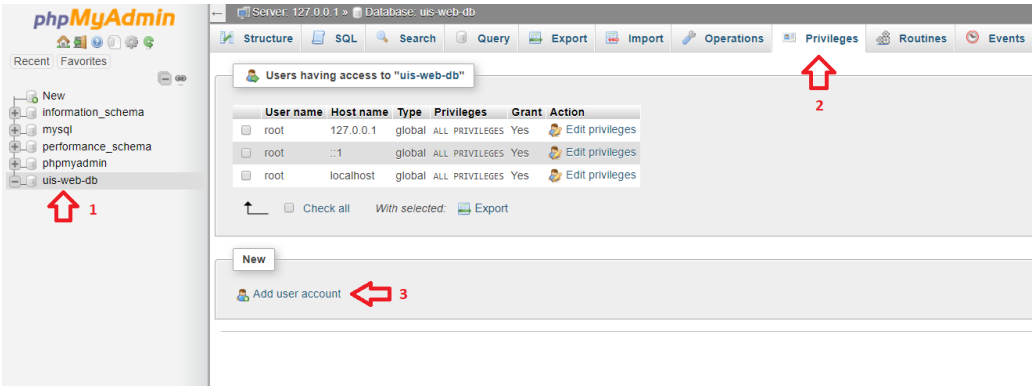
Základní nastavení databáze:

Jméno databáze:	uis-web-db
Porovnání:	UTF8_general_ci
Username:	uis-web
Heslo:	uis

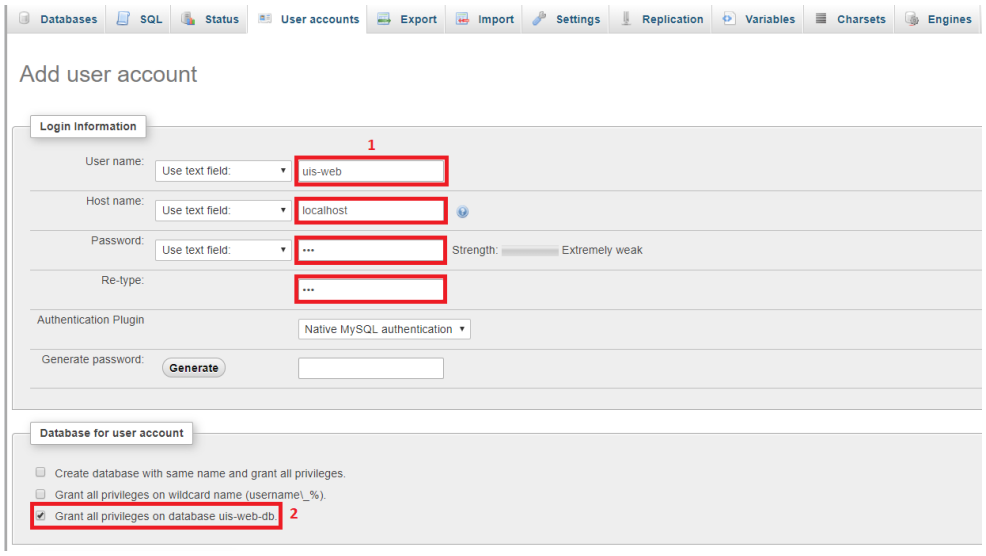
Nastavení databáze v nástroji phpMyAdmin



Obrázek A.6: Vytvoření databáze

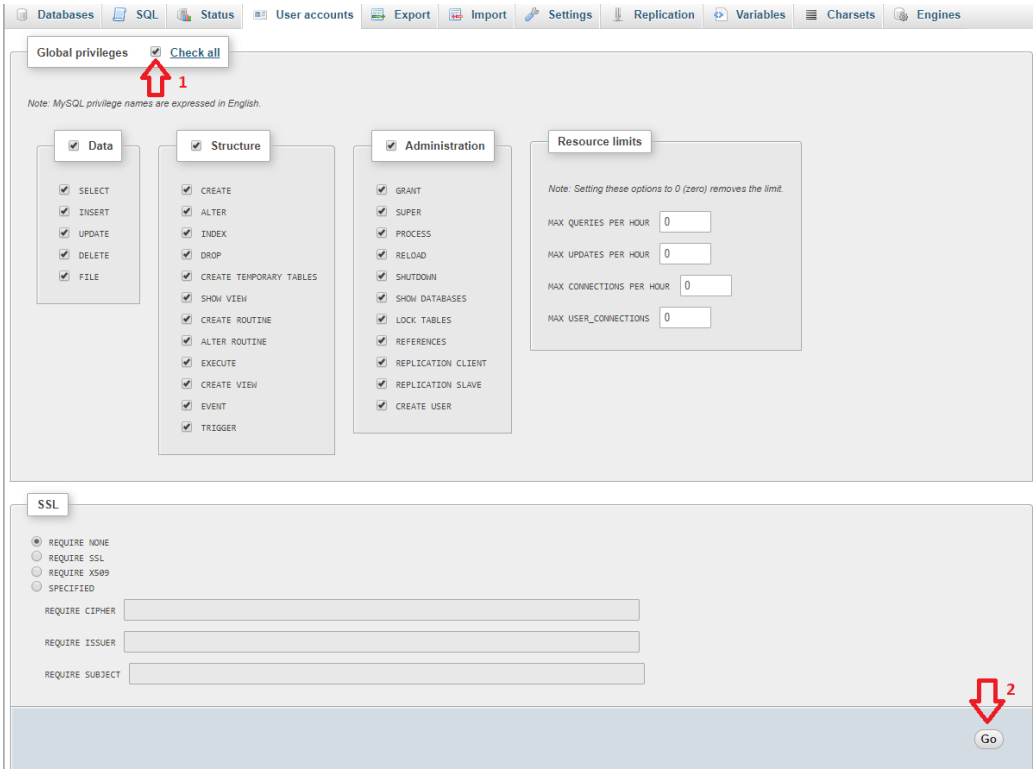


Obrázek A.7: Vytvoření *uis-web* uživatele databáze



Obrázek A.8: Nastavení přihlašovacích údajů

Poznámka: *Host name* = *localhost* v případě, že aplikace UIS-web a databáze běží na jednom počítači



Obrázek A.9: Nastavení práv a uložení přístupových údajů

B Scénáře odhalení

V této příloze jsou vypsané všechny scénáře odhalení pro každou implementovanou chybovou verzi bean v rámci této práce. Červeně jsou vyznačené kroky scénářů, které identifikují chybovou funkcionalitu implementovanou v chybové verzi beany.

B.0.1 E01GradeTypeDao – scénář odhalení

1. Nepřihlášený uživatel přejde na stránku *Login* (položka horního menu).
2. Přihlásí se za učitele s uživatelským jménem *pedant* (heslo *pass*).
3. Přejde na pohled *Set Evaluation* (položka uživatelského menu).
4. Výběr známky nabízí statickou množinu známek 1, 2, 3, 4 namísto známek z databáze A, B, C, D, E, F.
5. Při pokusu vytvoření nového hodnocení vrátí aplikace chybu a nové hodnocení/známku nevytvoří.

B.0.2 E01UserDao – scénář odhalení

1. Nepřihlášený uživatel přejde na stránku *Login* (položka horního menu).
2. Přihlásí se za učitele s uživatelským jménem *pedant* (heslo *pass*).
3. Přejde na pohled *List of All Teachers* (položka uživatelského menu).
4. Tabulka v tomto pohledu neobsahuje žádné položky

B.0.3 E01StudentService – scénář odhalení

1. Nepřihlášený uživatel přejde na stránku *Login* (položka horního menu).
2. Přihlásí se za studenta s uživatelským jménem *pink* (heslo *pass*).
3. Přejde na pohled *My Subjects* (položka uživatelského menu).
4. Tabulka *Enrolled Subjects* je vždy prázdná (nehledě na studované předměty)

B.0.4 E02StudentService – scénář odhalení

1. Nepřihlášený uživatel přejde na stránku *Login* (položka horního menu).
2. Přihlásí se za studenta s uživatelským jménem *pink* (heslo *pass*).
3. Přejde na pohled *My Subjects* (položka uživatelského menu).
4. Počet získaných kreditů je vždy 20 (nehledě na seznam absolvovaných předmětů)

B.0.5 E01TeacherService – scénář odhalení

1. Nepřihlášený uživatel přejde na stránku *Login* (položka horního menu).
2. Přihlásí se za učitele s uživatelským jménem *pedant* (heslo *pass*).
3. Přejde na pohled *My Subjects* (položka uživatelského menu).
4. Tabulka *Taught Subjects* je vždy prázdná (nehledě na vyučované předměty)

B.0.6 E01DateUtility – scénář odhalení

1. Nepřihlášený uživatel přejde na stránku *Login* (položka horního menu).
2. Přihlásí se za učitele s uživatelským jménem *pedant* (heslo *pass*).
3. Přejde na pohled *New Exam Dates* (položka uživatelského menu).
4. Vytvoří nový zkušební termín pro předmět *Computer System Engineering* s maximálním počtem účastníků 10 studentů a s dnešním datem a časem 12:00.
5. Přejde na pohled *My Exam Dates* (položka uživatelského menu).
6. Nově vytvořený termín je posunutý o 1 den (zítřejší datum s časem 12:00)

B.0.7 E02DateUtility – scénář odhalení

1. Nepřihlášený uživatel přejde na stránku *Login* (položka horního menu).
2. Přihlásí se za učitele s uživatelským jménem *pedant* (heslo *pass*).
3. Přejde na pohled *New Exam Dates* (položka uživatelského menu).

4. Vytvoří nový zkušební termín pro předmět *Computer System Engineering* s maximálním počtem účastníků 10 studentů a s dnešním datem a časem 12:00.

5. Přejde na pohled *My Exam Dates* (položka uživatelského menu).

6. Nově vytvořený termín má vždy nastavený datum a čas 2100-12-12 20:00.

C Obsah CD

- **bin** – složka s přeloženými a spustitelnými aplikacemi
  - **catalog.xml** – soubor katalogu (použitelný pro strukturu CD)
  - **Error-Seeder.jar** – Spustitelná verze aplikace Error-Seeder
  - **UIS-web.war** – Nasaditelná verze aplikace UIS-web
- **Error-Seeder** – Složka projektu aplikace Error-Seeder
  - **src** – složka se zdrojovými soubory aplikace Error-Seeder
  - **pom.xml** – Error-Seeder Maven Project Object Model
- **JavaDoc** – JavaDoc dokumentace aplikací
  - **Error-Seeder** – složka s dokumentací aplikace Error-Seeder
  - **UIS-web** – složka s dokumentací aplikace UIS-web
- **Poster** – složka se soubory posteru této práce
  - **Matyas_Jiri_2018.pdf** – Microsoft Publisher soubor posteru
  - **Matyas_Jiri_2018.pdf** – pdf soubor posteru
- **UIS-web** – Složka projektu aplikace UIS-web
  - **src** – složka se zdrojovými soubory aplikace UIS-web
  - **pom.xml** – UIS-web Maven Project Object Model
- **Matyas_Jiri_Diplomova_prace.pdf** – pdf soubor této práce