FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Refaktoring backend části portálu dbs.fit.cvut.cz - semestrální práce
Student:	Jakub Lukačín
Vedoucí:	Ing. Jiří Hunka
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2021/2022

Pokyny pro vypracování

Cílem této práce je refaktoring backendové části semestrálních prací portálu dbs.fit.cvut.cz, dále jen portál. Důvodem je nutná modernizace a pokrok k pohodlnějšímu a efektivnějšímu následnému vývoji.

Postupujte v těchto krocích:
1. Analyzujte současný stav portálu se zaměřením na semestrální práce. Nezapomeňte na provázanost s ostatními částmi.
2. Na základě analýzy důkladně navrhněte strukturu nové databáze dané části portálu s ohledem na již existující backend který vyvíjel Bc. Andrii Plyskach.
3. Nad databází postavte vhodný backend. Vzhledem k rozsahu se počítá s prototypovou implementací.
4. Na Vámi implementovaný kód realizujte vhodné testy - minimálně ukázkové jako šablonu pro další vývoj.
5. Navrhněte budoucí směr rozvoje Vámi připraveného prototypu.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 16. ledna 2021 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalárska práca

Refaktoring backend části portálu dbs.fit.cvut.cz - semestrální práce

Jakub Lukačín

Katedra softwarového inženýrství
Vedúci práce: Ing. Hunka Jiří

13. mája 2021

Poďakovanie

Chcel by som poďakovať vedúcemu tejto bakalárskej práce, pánovi Ing. Jiřímu Hunkovi za odborné rady, ochotu a čas venovaný tejto práci. Takisto by som chcel poďakovať Ing. Janovi Matouškovi za kvalitnú spätnú väzbu a rady pri konzultácii práce. Moja vďaka patrí aj Martinovi Hanzlovi za pomoc s nasadením Doctrine frameworku a tímu študentov, ktorí so mnou spolupracovali na ďalšom vývoji modulu. Veľká vďaka patrí mojej rodine a kamarátom za podporu počas štúdia a písania tejto práce.

Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval(a) samostatne a že som uviedol(uviedla) všetky informačné zdroje v súlade s Metodickým pokynom o etickej príprave vysokoškolských záverečných prác.

Beriem na vedomie, že sa na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Sb., autorského zákona, v znení neskorších predpisov, a skutočnosť, že České vysoké učení technické v Praze má právo na uzavrenie licenčnej zmluvy o použití tejto práce ako školského diela podľa § 60 odst. 1 autorského zákona.

V Prahe 13. mája 2021 . . . . . .

Odkaz na túto prácu

Lukačín, Jakub. *Refaktoring backend části portálu dbs.fit.cvut.cz - semestrální práce*. Bakalárska práca. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2021.

Abstrakt

Hlavným cieľom práce je vytvorenie analýzy aktuálneho riešenia modulu semestrálna práca, návrh nového riešenia a následná implementácia časti návrhu. Zvolenými implementačnými technológiami sú framework Nette a framework Doctrine. K analýze aktuálneho riešenia je použitý Use Case model. Analýza zachytáva 49 prípadov užitia aktuálneho riešenia. Návrh databázy je rozdelený do 11 sekcií a približuje takmer 40 rozličných databázových tabuliek. Dátová a logická vrstva tvoria návrh systému. Hlavnou časťou implementácie je „mapovanie“ entít pre Doctrine. V implemetácii je taktiež popísaný proces vývoja repository a service tried. Pre časť implementácie sú vytvorené unit a integračné testy. Budúci rozvoj modulu je čiastočne vyriešený spoluprácou so študentským tímom počas práce na tejto bakalárskej práci. Výsledky tejto práce pokladajú základy novému modulu semestrálnej práce DBS portálu, ktorý raz v praxi nahradí aktuálny používaný modul.

Klíčová slova refaktoring backendu, DBS portál, modul semestrálni práce, databázové systémy, objektovo-relačné mapovanie, framework Doctrine, PHP

Abstract

The main goal of this thesis is to analyse the current solution of the semester work module, design new solution and lastly implement part of the design. The chosen implementation technologies are the Nette framework and the Doctrine framework. The Use Case model is used to analyse the current solution, with 49 use cases of the current solution analysed. The database design is divided into 11 sections and uses almost 40 different database tables. The data and logical layers form the system design. The main part of the implementation is the „mapping“ of entities for Doctrine. The implementation also describes the process of developing repository and service classes. Unit and integration tests are created for part of the implementation. The future development of the module is partially solved by cooperation with the student team during the work on this bachelor thesis. The results of this work lay the foundations for a new semester work module of the DBS portal, which will once in practice replace the currently used module.

Keywords backend refactoring, DBS portal, Semester Work module, database systems, object–relational mapping, framework Doctrine, PHP

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

Každý študent, ktorý plánuje získať bakalársky titul na Fakulte informačních technologií Českého vysokého učení technického v Prahe, sa počas štúdia povinného predmetu DBS stretne s DBS portálom sídliacim na adrese `https://dbs.fit.cvut.cz/`. Práve na tomto portáli prebieha študentova práca na semestrálnej práci z DBS, splnenie ktorej je jednou z podmienok úspešného ukončenia predmetu, ako aj písanie testu v semestri a skúškového testu.
Portál bol študentmi prvý krát používaný v LS 2016 a to len na cvičeniach, ktoré viedol Jiří Hunka. V skúškovom období LS 2016 bol taktiež otestovaný na jednom zo skúškových termínov. Začínajúc od LS 2017 je portál využívaný všetkými cvičiacimi. Portál bol vyvíjaný v rámci viacerých bakalárskych prác, ako aj predmetov SP1 a SP2.
Moja bakalárska práca sa zaoberá refaktoringom časti semestrálna práca DBS portálu, čo je časť ktorá, ako už názov napovedá, obsahuje systém pre editáciu semestrálnej práce študentom, jej kontroly a opravy cvičiacim. V práci nadviažem na bakalársku prácu Andriiho Plyskacha[1], ktorý prevádzal refaktoring modulu testy.
Na vývoji DBS portálu sa vystriedalo veľa študentov. Kód je neprehľadný a ťažko rozširovateľný, je teda požiadavka na vytvorenie nového backendu, ktorý bude jednotný a jednoduchšie udržiavateľný a rozširovateľný.
Motiváciou výberu témy je to, že výstup mojej práce pomôže vývojárom DBS portálu jednoduchšie rozširovať portál. Na tomto portáli som pracoval počas štúdia. Motiváciou je aj to, že vylepším výučbu predmetu DBS, keďže chcem pridať novú funkcionalitu.
Prvým cieľom práce je zvoliť vhodnú implementačnú technológiu s ohľadom na existujúci prototyp modulu testy. Následne vytvoriť analýzu aktuálneho riešenia a rozšírení. Ďalším cieľom je podľa analýzy navrhnúť nové riešenie, z pohľadu databázy a aj z pohľadu systému. Časť navrhnutého riešenie naimplementovať a otestovať. Posledným cieľom je navrhnúť budúci rozvoj modulu.

Kapitola 1

Teoretická časť

1.1 Priblíženie portálu

V prvej sekcii priblížim čitateľovi portál. Bližšie sa zameriam sa na časť semestrálna práca.

1.1.1 Zameranie portálu

Skratka DBS ukrýva **d**ata**b**ázové **s**ystémy. Tento predmet ma naučil ako fungujú databázy, základy jazyka SQL a Relačnej algebry. Pokiaľ by Vás zaujímal detailnejší popis tohto predmetu, z Bílej knihy FIT ČVUT som si vypožičal jeho anotáciu: „*Student se seznámí s architekturou databázového stroje a typickými uživatelskými rolemi. Dále stručně pozná různé databázové modely. Naučí se navrhovat menší databáze (včetně integritních omezení) pomocí konceptuálního modelu a poté je implementovat v relačním databázovém stroji. Prakticky se seznámí s jazykem SQL a také s jeho teoretickým základem - relačním databázovým modelem. Seznámí se s principy normalizace relačního databázového schématu. Pochopí základní koncepce transakčního zpracování, řízení paralelního přístupu uživatelů k jednomu datovému zdroji a obnovy databázového stroje po havárii. Stručně se seznámí se speciálními způsoby uložení dat v relačních databázích s ohledem na rychlost přístupu k velkému množství dat. Tento základní kurz nepokrývá témata: administrace databázových systémů, ladění a optimalizace databázových aplikací, distribuované databázové systémy a datové sklady.*“ [2]

1.1.2 Rozdelenie portálu

DBS portál sa v čase písania tejto bakalárskej práce skladá z viditeľných modulov:

1. Semestrálna práca
2. Testy
3. Administrácia
4. Pripojenie
5. Data modeller (kreslítko)

Medzi jeho ďalšie časti, ktoré pre bežného užívateľa nie sú viditeľné, patrí:

prekladač z relačne algebry do SQL vypracovaný v rámci bakalárskej a diplomovej práce Ing. Martina Kubiša [3],

služba automatickej opravy zjednodušeného relačního zápisu ako bakalárska práca Filipa Machalu [4].

1.1.3 Bodovanie v BI-DBS

V semestroch, ktoré neboli ovplyvnené Covid19 pandémiou, bolo rozdelenie bodov z predmetu ako v tabuľke 1.1.

Tabuľka 1.1: Bodovanie BI-DBS [5]

časť	max. bodov	min. požadovaných bodov
semestrálna práca	22	11
test počas semestru	18	8
skúškový test	60	0
ústna skúška	12	0

1.1.4 Priblíženie semestrálnej práce

Semestrálna práca tvorí zhruba štvrtinu známky študenta. V aktuálnom prebiehajúcom semestri je to dokonca 50%, kvôli prechodu na dištančnú výuku [6]. Študentovou úlohou je vymyslieť si projekt, pre ktorý následne navrhne a vyvinie dátové úložisko v SQL databáze [7].
Body za semestrálnu prácu študent získava v troch krokoch, tzv. Iteráciách. Každá z iterácii ma špecifické požiadavky na obsah semestrálnej práce, ktorých nesplnenie pripraví študenta o určitý počet bodov.

1.1.4.1 Téma semestrálnej práce

Témy projektov sú ľubovoľné, učiteľ ich však musí schváliť. Téma by mala obsahovať viacero objektov. Príkladom témy môže byť Medzinárodná odsoľovacia korporácia – futuristická firma zaoberajúca sa odsoľovaním znečistenej vody v 4.tisícročí.

1.1.4.2 Entity, atribúty

Objekty nazývame entitami. Entity majú atribúty – vlastnosti objektov. Medzi entitami existujú vzťahy. Príkladmi entít v prípade projektu Medzinárodná odsoľovacia korporácia je služba, objednávka, zákazník, zamestnanec, more, oceán. Atribútom bude meno, priezvisko, vek a email zamestnanca.

1.1.4.3 Vzťahy

Vzťahy sú viacerých typov, rozdelené podľa násobnosti.

1.1.4.4 Časti semestrálnej práce v predmete DBS

- Názov
- Popis
- Konceptuálna schéma
- Diskusia smyčiek
- Relačná schéma
- Dotazy
- Create script
- Insert script
- Záver
- Zdroje

Priblížim termíny, ktoré by čitateľom mohli byť neznáme.

1.1.4.5 Konceptuálna schéma

Graficky definuje dátové úložisko, konkrétne jeho entity, atribúty entít a vzťahy medzi entitami.

1.1.4.6 Diskusia smyčiek

V konceptuálnej schéme môže vzniknúť smyčka, teda určité entity majú spoločné vzťahy, ktoré vytvoria uzol. V tomto prípade je vhodné zamyslieť sa nad tým či niektorý zo vzťahov nie je prebytočný.

1.1.4.7 Relačná schéma

Podobná konceptuálnej schéme s rozdielom, že atribúty entít majú svoje dátové typy, určité väzby sú definované vlastnou tabuľkou.

1.1.4.8 Dotazy

Príkazy, ktorými chceme vyfiltrovať špecifické dáta z databázy. Dotazy spadajú pod rozdielne kategórie podľa toho, čo vykonávajú. V DBS sú vyučované v troch jazykoch: prirodzený jazyk, SQL a relačná algebra. Dotazy v SQL a RA sú posielané na databázu, ktorá vracia dotazované dáta.

1.1.4.9 Create script

Skript v jazyku SQL, ktorý databázovému stroju diktuje, ako má vyzerať dátové úložište pre študentov projekt, teda názvy entít, atribútov, typy atribútov, vzťahy medzi entitami [8].

1.1.4.10 Insert script

Skript v jazyku SQL, ktorý vypĺňa tabuľky[i] v študentovom projekte konzistentnými dátami, na základe ktorých sa testuje správnosť pripravených dotazov [9].

1.2 Priblíženie prototypu testového modulu

Pre testový modul portálu už existuje prototyp nového riešenia. Bol vypracovaný Bc. Andriim Plyskachom ako jeho bakalárska práca.
Dovolím si citovať z jeho abstraktu „*Tato bakalářská práce se zabývá návrhem a implementací prototypu nového backendu*[ii] *testového modulu DBS portálu.*“[1]
Ako ste zrejme postrehli, téma a cieľ jeho bakalárskej práce sú podobné mojej, s rozdielnym zameraním – zatiaľ čo on riešil testový modul, ja sa sústredím na modul semestrálnej práce.
Víziou vedenia DBS portálu je v budúcnosti plný prechod na nové moduly testov a semestrálnej práce, ktoré vzniknú rozširovaním spomínaného prototypu testového modulu, ako aj prototypu, ktorý v rámci bakalárskej práce naimplementujem ja.
Bc. Plyskach vo svojej práci použil frameworky Nette a Doctrine. Tieto technológie a výrazy predstavím v nasledujúcej kapitole.

1.3 Priblíženie a voľba technológií

Na začiatku tejto sekcii priblížim čitateľovi pár pojmov z oblasti programovania, ktoré budem v práci spomínať. Následne priblížim technológie, vhodné na splnenie zadanej úlohy. Na koniec vyberiem najvhodnejšie technológie a ich voľbu odôvodním.

[i] entity sú v databáze reprezentované tabuľkami

[ii] Funkcionalita webovej aplikácie, ktorú bežný užívateľ nevidí. Je to kód, ktorý spája web s databázou, zabezpečuje pripájanie užívateľov a chod aplikácie [10].

1.3.1 Priblíženie základných pojmov

1.3.1.1 PHP

PHP je „populárny open source[iii] programovací jazyk, ktorý sa používa najmä na programovanie klient–server[iv] aplikácií na strane servera a pre vývoj dynamických webových stránok. Medzi známe aplikácie založené na PHP patrí napríklad phpBB a MediaWiki, software, na ktorom beží Wikipédia.“ [13]
O jeho populárnosti hovoria aj štatistiky. Podľa webu w3techs.com 79,1% stránok, ktorých programovací jazyk serverovej časti poznáme, používa jazyk PHP [14].

1.3.1.2 Knižnica, framework

Knižnica je „v informatike označenie pre súbor funkcií a procedúr (v objektovom programovaní aj objektov, dátových typov a zdrojov), ktorý môže byť zdieľaný viacerými počítačovými programami. Knižnica uľahčuje programátorovi tvorbu zdrojového kódu tým, že umožňuje použiť už vytvorený kód aj v iných programoch. Knižnica navonok poskytuje svoje služby pomocou API (aplikačné rozhranie), čo je zbierka zvonka prístupných funkcii knižnice s popisom ich činnosti a spôsobom ich volania (odovzdávanými parametrami a návratovými hodnotami).“ [15]
Framework je „jednoducho povedané súbor mnohých knižníc, ktoré sú prispôsobené tak, aby k sebe pasovali a tvorili jeden funkčný celok. Zvyčajne všetky pochádzajú od jedného autora alebo skupinu autorov.“ [16]

1.3.1.3 MVC

MVC je jeden zo spôsobov návrhu aplikácie. Princípom je rozdelenie aplikácie na tri časti – na Model, View a Controller. Funkcie jednotlivých častí:

Model je reprezentáciou vlastností objektu s ktorým pracujeme a predpisuje tzv. business logic, teda spôsob, ako sa s dátami pracuje.

View je vizuálnou reprezentáciou dát.

Controller vytvára inštancie jednotlivých modelov, napĺňa ich dátami a zobrazuje príslušný View a to na základe požiadavky používateľa. Je „lepidlom“ medzi požiadavkou používateľa, jednotlivými modelmi a View. [17]

[iii] „Open source softvér je softvér so zdrojovým kódom, ktorý môže hocikto vidieť, upravovať a vylepšovať.“ [11]

[iv] „Klient/server opisuje vzťah medzi dvoma počítačovými programami, z ktorých jeden, klient, odošle požiadavku na službu z druhého programu, servera, ktorá požiadavku splní.“ [12]

1.3.1.4 Template engine

Softwér navrhnutý na skombinovanie šablóny s dátovou vrstvou, produkuje výsledné dokumenty. [18]
Na príklade sa pokúsim zjednodušene vysvetliť, ako to funguje pri webovej aplikácii. Predstavte si, že si otvárate domovskú stránku Youtube.

Šablónou v tomto prípade je šablóna domovskej stránky Youtube, ktorá je nahraná na serveroch Youtube a pripravil ju programátor.

Dátovou vrstvou sú dáta, ktoré má Youtube uložené o Vašom účte a potrebuje ich na vyplnenie šablóny domovskej stránky Youtube.

Výsledným dokumentom je zdrojový kód stránky, ktorá sa Vám zobrazí, teda vidíte videá kanálov, ktoré odoberáte, obľúbené videá vo Vašej krajine a iné personalizované dáta.

1.3.1.5 ORM

„*Objektově relační mapování je programovací technika v softwarovém inženýrství, která zajišťuje automatickou konverzi dat mezi relační databází a objektově orientovaným programovacím jazykem.*“ [19]
V princípe to znamená, že programátor pre prístup k databáze nepotrebuje písať zložité SQL príkazy, ale k jednotlivým entitám v databáze pristupuje ako k objektom, ktoré majú svoje metódy.
Príkladom môže byť osoba, ktorá sa chce na mojej stránke zaregistrovať. Ak od nej požadujem email a prihlasovacie meno, ktoré si uložím do databázy, v programe môžem namiesto zložitejšieho SQL príkazu typu „INSERT INTO user (name, email) VALUES (lukacja6, l6@cvut.cz);“ použiť jednoduchší PHP kód „$newUser = new User(”lukacja6”, ”l6@cvut.cz”)“.

1.3.2 Voľba programovacieho jazyka

Backend DBS portálu je naprogramovaná jazykom PHP. Hlavnou technológiou mojej implementácie bude teda programovací jazyk PHP.

1.3.3 Rozbor vhodných frameworkov

Z dôvodu uľahčenia práce, ako aj možnosti jednoduchého rozširovania aplikácie, ktorú vytvorím, použijem pri vývoji PHP frameworky. Podľa webu builtwith.com ku dňu 10. 3. 2021 50,49 % z milióna najnavštevovanejších stránok využíva niektorý z PHP frameworkov [20]. Anglická Wikipedia má v čase písania tejto semestrálnej práce 39 stránok v kategórii PHP Framework [21]. Z tohto veľkého počtu vyberiem pár fameworkov, ktoré by sa na plánovanú prácu hodili, priblížim ich výhody ako aj nevýhody a následne zvolím tie, ktoré pri práci využijem.

1.3.3.1 Laravel

Laravel je prvým frameworkom, ktorý predstavím. Jeho počiatočné vydanie bolo v júni 2011 [22].
Aktuálne patrí podľa viacerých zdrojov medzi najpoužívanejšie frameworky. Ku dňu 11. 3. 2021 je latest release[v] v8.32.1[23]. Vývoj tohto frameworku je teda relatívne rýchly.
Laravel funguje na MVC patterne.
Výhody:

- nie je zložitý na naučenie,
- ako template engine využíva Blade[vi], teda nie je potrebné využiť ďalší programovací jazyk,
- ponúka sortiment bezpečnostných vlastností, ako autentifikácia, autorizácia, verifikácia emailu, resetovanie hesla,
- obsahuje Eloquent ORM a Fluent Query Builder ktoré chránia proti SQL injection útoku[vii]. [25]

1.3.3.2 Nette

Nette Framework je open source framework na tvorbu webových aplikácií v PHP 5 a PHP 7[viii]. Pôvodným autorom je český programátor David Grudl [26].
Je vo vývoji viac než 10 rokov. Ku dňu 11. 3. 2021 je latest release v3.1.0 [27]. Podľa stránky builtwith.com ide o 6. najpoužívanejši framework na Slovensku a 2. najpoužívanejší framework v Českej republike [28].
Tento framework je použitý na vývoj aktuálneho riešenia portálu dbs.fit.cvut.cz. Stretol som sa s ním na predmetoch BI-SP1 a BI-SP2, takisto som sa v ňom snažil vytvoriť vlastný projekt.
Výhody:

- skúsenosť s technológiou,
- kompatibilita s aktuálnym riešením portálu,
- zameraný na bezpečnosť,
- kvalitne spracovaná dokumentácia [29].

[v] Najnovšia dostupná verzia softwéru.

[vi] Programátor môže použiť PHP v Blade, čo nie je možné v iných template enginoch.

[vii] SQL injection je technika napadnutia databázovej vrstvy aplikácie, spočívajúca vo vsunutí (odtiaľ „injection“) kódu prostredníctvom neošetreného alebo nesprávne ošetreného vstupu a vykonaní vlastného SQL príkazu. Zneužitie môže viesť k získaniu citlivých údajov, ako napr. prihlasovacie údaje, osobné údaje (čísla bankových účtov, rodné čísla...); pozmeneniu, doplneniu alebo odstráneniu údajov, prípadne aj k ovládnutiu celého serveru. [24]

[viii] verzie jazyka PHP

1.3.3.3 Doctrine

„*Doctrine je ORM framework pro jazyk PHP. Nabízí vysokou míru abstrakce databázové vrstvy použitím objektového přístupu. Umožňuje tak pracovat s daty jako s objekty. Jedna z klíčových vlastností Doctrine je psaní databázových dotazů použitím Doctrine Query Language (DQL), který vychází z ORM frameworku Hibernate určeným pro Javu*[ix].“ [30]
Výhody:

- výborná abstrakcia,
- jednoduché používanie,
- kvalitne spracovaná dokumentácia,
- objektovo–orientované. [31]

1.3.4 Voľba frameworku

Keďže sa programátori pracujúci na DBS portáli často menia a nie všetci sú na začiatku skúsení a pripravení naučiť sa veľa nových technológií, je vhodné zvoliť framework, ktorý je kompatibilný s ostatnými časťami portálu, ktoré používajú framework Nette.
V budúcnosti má rozšírený prototyp Bc. Plyskacha, založený na Nette a Doctrine, byť v prevádzke vedľa môjho rozšíreného modulu.
Zvolím frameworky Nette, Doctrine a Nettrine[x].
Ak by som nebol obmedzený spomínanou kompatibilitou, zvolil by som Laravel, nakoľko ide o veľmi rozšírený framework a rád by som ho vyskúšal.

1.4 Priblíženie modelu prípadov užitia

Tento model využijem k analýze aktuálneho riešenia. Najprv ho čitateľovi priblížim.
Use Case model[xi] jednoznačne vymedzuje vlastnosti softvéru. Takisto popisuje, ako užívatelia v rozdielnych rolách interagujú so systémom. Približuje cieľe uživateľa, interakcie užívateľa a systému ako aj požadovanú odpoveď systému na konanie užívateľa.
Najdôležitejšou časťou modelu je textový popis, definujúci prípad užitia a jeho scenáre. Medzi ďalšie prvky patrí:

aktér: môže ním byť osoba(rola, v ktorej voči systému osoba vystupuje), externý systém ale aj čas. Čas sa využíva v prípade, ak modelovaný

[ix] programovací jazyk
[x] balík knižníc uľahčujúci integráciu Doctrine do Nette frameworku
[xi] model prípadov užitia

systém obsahuje funkčnosti, ktoré sú spúšťané automaticky na základe časovej udalosti.

pre-condition: podmienka, ktorá musí byť splnená pred spustením prípadu užitia.

post-condition: podmienka, ktorá musí byť splnená po ukončení prípadu užitia.

scenár: existuje hlavný a vedľajší, tieto scenáre popisujú interakcie aktéra so systémom v podobe jednotlivých krokov. Hlavný scenár popisuje najbežnejší spôsob, akým aktér môže v systéme daný prípad užitia previesť. Pomocou vedľajšieho scenára môže aktér dosiahnuť rovnaký výsledok, ale iným spôsobom. [32]

1.5 Zhrnutie

Na začiatku kapitoly som čitateľa oboznámil s DBS portálom a predmetom DBS. Následne som priblížil existujúci prototyp modulu testy. Pokračoval som popísaním základných odborných pojmov používaných v práci a priblížením vhodných technológií. Zvolil som technológie, ktoré v práci využijem. V závere kapitoly som čitateľovi priblížil model prípadov užitia, ktorý využijem k analýze.

Kapitola 2

Analýza

Keďže rozsah modulu nie je zanedbateľný, potrebujem čo najlepšie porozumieť aktuálnemu riešeniu, ako aj problémom aktuálneho riešenia. Na veľkú časť analýzy použijem postupy vysvetľované na predmete SI1. Táto časť takisto obsahuje požiadavky, ktoré v aktuálnom riešení nie sú a bude mojou úlohou začleniť ich do môjho prototypového riešenia. K analýze aktuálneho riešenia použijem Use Case model. Tento model som zvolil z dôvodu skúsenosti s týmto modelom, stretol som sa s ním počas štúdia. Takisto detailne špecifikuje funkčné požiadavky [33], čo je dobrý predpoklad pre kvalitný návrh.

2.1 Aktéri na portáli

V rámci aktuálneho riešenie semestrálnej práce existujú 3 role, ktoré majú rozdielne právomoci a budeme ich spomínať v modeloch prípadov užitia. Sú to garant, učiteľ a študent.

študent – osoba, ktorá má v aktuálnom semestri zapísaný predmet DBS, jej hlavným cieľom by malo byť vypracovanie a včasné odovzdanie semestrálnej práce v jednotlivých iteráciách.

učiteľ – osoba poverená vyučovaním predmetu DBS. Jej hlavným cieľom je kontrola a ohodnotenie iterácií semestrálnej práce odovzdaných študentmi ich paraleliek. Paralelka je skupina študentov, ktorí majú rovnaký dátum cvičení a rovnakého učiteľa.

garant – je v tejto hierarchii na najvyššom mieste, určuje požiadavky na jednotlivé iterácie.

2.2 Prípady užitia

V tejto sekcii zadefinujem, čo a ako užívateľ v aktuálnej verzii portálu vykonáva. Zvolil som rozdelenie na menšie súdržné časti. Pre veľký rozsah tejto časti som sa rozhodol podrobné prípady užitia uložiť do prílohy. Konkrétne príloha UC.pdf približuje jednotlivé prípady užitia podrobnejšie. Tieto podrobné UC majú naviac prioritu, náročnosť, pre-conditions a scenáre.

2.2.1 Spravovanie bodovania a požiadaviek na iterácie

Podsekcia približuje, ako sa v portáli nastavuje bodovanie a požiadavky na jednotlivé iterácie. Graficky túto podsekciu približuje diagram 2.1.

UC1 – Zobrazenie klasifikácie a požiadaviek na iterácie

Prípad užitia umožňuje aktérovi prezrieť si, ako je nastavené bodovanie semestrálnej práce, teda maximum a minimum bodov za semestrálnu prácu a minimum bodov zo semestrálnej práce potrebných na zápočet. Aktér taktiež uvidí v koľkých iteráciach sa semestrálna práca odovzdáva, aké je bodovanie jednotlivých iterácií a aké su požiadavky na jednotlivé iterácie. Pri jednotlivých požiadavkách aktér uvidí, koľko bodov mu môže byť stiahnutých za nesplnenie konkrétnej požiadavky.
Aktér: Garant, Učiteľ, Študent

UC2 – Zmena klasifikácie

Prípad užitia umožňuje garantovi upraviť klasifikáciu, konkrétne maximum, minimum bodov zo semestrálnej práce, minimum bodov zo semestrálnej práce potrebných na zápočet, pre jednotlivé iterácie ich základne body, maximum bonusových bodov a maximum stiahnutých bodov za neskoré odovzdanie.
Aktér: Garant

UC3 – Pridanie požiadavky na iteráciu

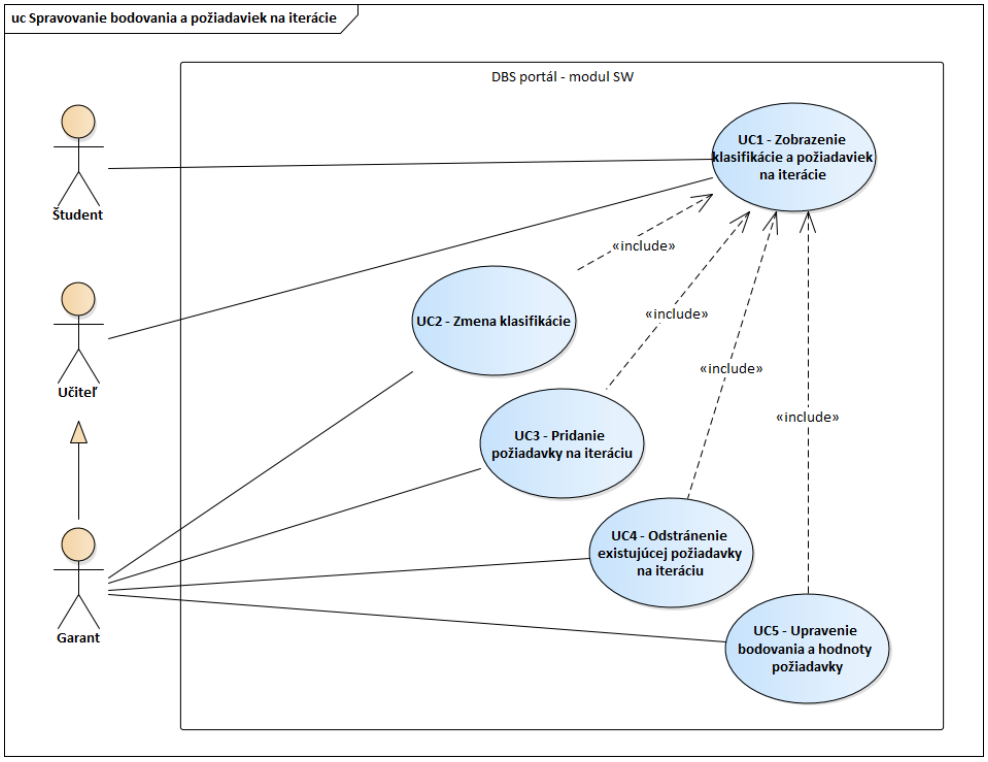
Prípad užitia umožňuje garantovi pridať požiadavku pre konkrétnu iteráciu.
Aktér: Garant

UC4 – Odstránenie existujúcej požiadavky na iteráciu

Prípad užitia umožňuje aktérovi odstrániť konkrétnu požiadavku na jednu z iterácií.
Aktér: Garant

UC5 – Upravenie bodovania a hodnoty požiadavky

Prípad užitia umožňuje garantovi upraviť body, strhnuté za nedodržanie zvolenej požiadavky. Aktér môže takisto nastaviť hodnotu požiadavky, ak je požiadavka valuable[xii].
Aktér: Garant



Obr. 2.1: Use Case diagram Spravovanie bodovania a požiadaviek na iterácie

2.2.2 Spravovanie paraleliek

Časť zaoberajúca sa informáciami o paralelkách a dátumom odovzdania jednotlivých iterácií pre paralelky. Graficky zachytená na obrázku 2.2.

UC6 – Prehľad termínov odovzdania paraleliek

Prípad užitia umožňuje aktérovi prezrieť si nastavené termíny odovzdania pre jednotlivé paralelky. V prípade učiteľa ide o paralelky, ktoré vyučuje. V prípade garanta ide o všetky paralelky patriace do kurzu, ktorého je garantom. Aktér môže spravované paralelky zoradiť a vyfiltrovať podľa filtrov cvičiaci,

[xii] príklad valuable požiadavky je „dĺžka popisu musí byt väčšia alebo rovná 300 slov“, kde 300 je hodnota požiadavky

miestnosť, deň, čas, týždeň, a dátumy odovzdania jednotlivých iterácií.
Aktér: Garant, Učiteľ

UC7 – Upravenie termínu odovzdania

Prípad užitia umožňuje aktérovi upraviť dátum a čas odovzdania každej iterácie pre konkrétnu paralelku, ktorú spravuje. Učiteľ môže upravovať termíny odovzdania svojich paraleliek, garant všetkých paraleliek.
Aktér: Garant, Učiteľ

UC8 – Import termínov odovzdania

Prípad užitia umožňuje aktérovi importovať termíny odovzdania pre zvolené paralelky z .csv súboru.
Aktér: Garant, Učiteľ

UC9 – Export termínov odovzdania

Prípad užitia umožňuje aktérovi stiahnuť jednotlivé nastavené termíny odovzdania ako .csv súbor. Môže si zvoliť, či chce exportovať všetky termíny, alebo len vyfiltrované.
Aktér: Garant, Učiteľ

UC10 – Zaslanie emailu paralelke

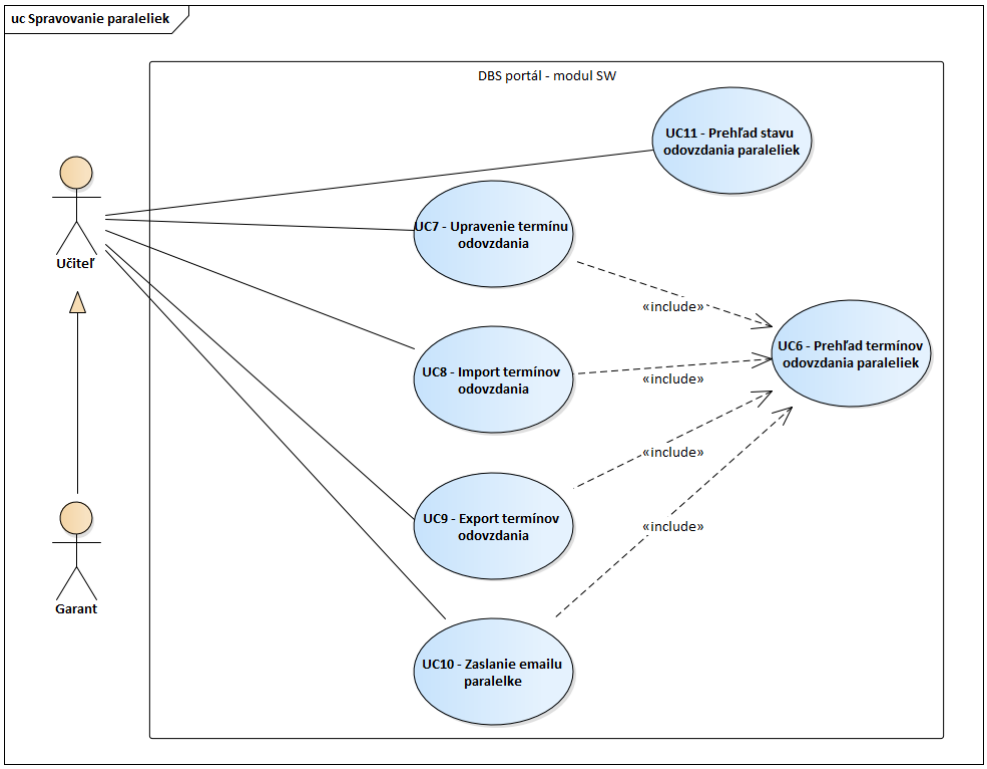
Prípad užitia umožňuje aktérovi zaslať email študentom patriacim do zvolenej paralelky. Garant môže zaslať email ľubovoľnej paralelke, učiteľ len spravovaným paralelkám.
Aktér: Garant, Učiteľ

UC11 – Prehľad stavu odovzdania paraleliek

Prípad užitia umožňuje aktérovi prezerať a filtrovať informácie o jednotlivých študentoch a ich odovzdaniach semestrálnej práce. Medzi filtrami je užívateľské meno, meno a priezvisko, email študenta, paralelka, vyučujúci, stav odovzdania jednotlivých iterácii, počet odovzdaní jednotlivých iterácií a dátum posledného prihlásenia užívateľa.
Aktér: Garant, Učiteľ

2.2.3 Spravovanie semestrálnej práce

Podsekcia popisuje prípady užitia z pohľadu študenta v rámci interakcií s jeho semestrálnou prácou. Zahŕňa editáciu jednotlivých častí semestrálnej práce, spúšťanie skriptov nad pripojením semestrálnej práce a prehľad semestrálnej práce. Graficky zachytené na diagrame 2.3.



Obr. 2.2: Use Case diagram Spravovanie paraleliek

UC12 – Zobrazenie semestrálnej práce

Prípad užitia umožňuje aktérovi zobraziť semestrálnu prácu pozostávajúcu z položiek „Nadpis“, „Popis“, „Konceptuální schema“, „Diskuze smyček“, „Relační schema“, „Create script“, „Insert script“, „Dotazy“, „Kategorie pokryté dotazy“, „Závěr“, „Zdroje“.
Pre lepšiu orientáciu v semestrálnej práci má aktér možnosť jednotlivé položky zabaliť, rozbaliť. Systém takisto umožňuje rozbaliť všetky časti semestrálnej práce zakliknutím „Rozbalit vše“ a zabalit všetky časti zakliknutím „Sbalit vše“.
Aktér: Študent

UC13 – Stiahnutie aktuálnej verzie semestrálnej práce

Prípad užitia umožňuje študentovi stiahnuť aktuálnu verziu semestrálnej práce v .zip súbore. Súbor obsahuje SQLDevSource, obrázok relačnej schémy, insert script, create script a ostatné textové časti semestrálnej práce v súbore main.xml.
Aktér: Študent

UC14 – Importovanie časti semestrálnej práce

Prípad užitia umožňuje študentovi importovať zvolenú časť semestrálnej práce. Podrobnejšie je tento Use Case popísaný v priloženom súbore UC14.pdf. Medzi importovateľné časti semestrálnej práce patrí „SQL Developer ZIP“, „Relační schema“, „Create script“, „Insert script“ a „Soubor XML“ - prepisujúci ostatné textové časti semestrálnej práce, menovite „Nadpis“, „Popis“, „Diskuze smyček“, „Dotazy“, „Závěr“ a „Zdroje“.
Aktér: Študent

UC15 – Editácia pripojenia semestrálnej práce

Prípad užitia umožňuje študentovi nastaviť, ktoré pripojenie chce nastaviť ako pripojenie pre semestrálnu prácu. Na tomto pripojení budú spúšťané skripty a dotazy zo semestrálnej práce.
Aktér: Študent

UC16 – Editácia jednej z textových častí semestrálnej práce

Prípad užitia umožňuje študentovi editovať zvolenú textovú časť semestrálnej práce. Podrobnejšie je tento Use Case popísaný v priloženom súbore UC16.pdf. Medzi textové časti semestrálnej práce patria „Nadpis“, „Popis“, „Diskuze smyček“, „Závěr“, „Zdroje“.
Tento prípad je priblížený v prílohe UC16.pdf. **Aktér:** Študent

UC17 – Editácia konceptuálnej schémy

Prípad užitia umožňuje študentovi editovať konceptuálnu schému patriacu k semestrálnej práci pomocou nástroja data modeller alebo pomocou importovaného SQL Developer Zipu.
Aktér: Študent

UC18 – Spravovanie dotazov patriacich k semestrálnej práci

Prípad užitia umožňuje študentovi spravovať dotazy patriace k jeho semestrálnej práci. Medzi spravovanie dotazov patrí:

- pridanie nového dotazu,
- editácia existujúceho dotazu,
- odstránenie existujúceho dotazu,
- priradenie kategórie dotazu,
- automatické nastavenie kategórií dotazu,
- prevedenie dotazu nad pripojením semestrálnej práce.

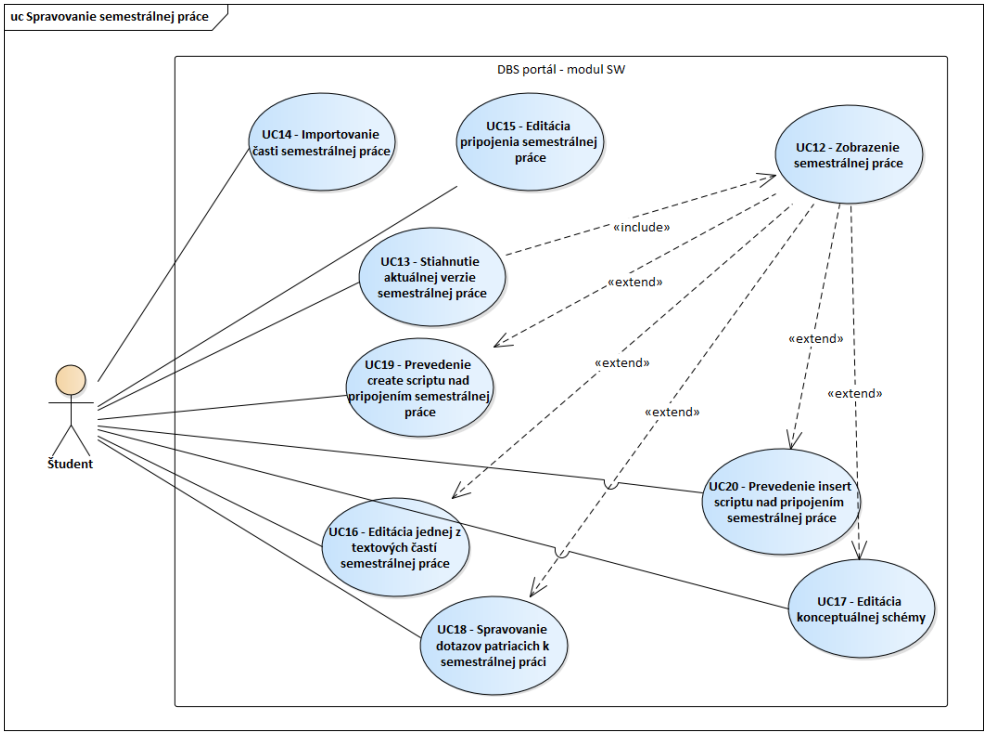
Jednotlivé aktivity patriace k tomuto Use Casu sú vlastnými Use Case modelmi popísané v prílohe UC18.pdf.
Aktér: Študent

UC19 – Prevedenie create scriptu nad pripojením semestrálnej práce

Prípad užitia umožňuje študentovi previesť create script nad pripojením semestrálnej práce. V prípade úspechu vytvorí na pripojení semestrálnej práce databázové úložisko.
Aktér: Študent

UC20 – Prevedenie insert scriptu nad pripojením semestrálnej práce

Prípad užitia umožňuje študentovi previesť insert script nad pripojením semestrálnej práce. V prípade úspechu naplní databázové úložisko na pripojení semestrálnej práce dátami z insert scriptu.
Aktér: Študent



Obr. 2.3: Use Case diagram Spravovanie semestrálnej práce

2.2.4 Kontrola a odovzdávanie semestrálnej práce

Táto časť združuje aktivity spojené s odovzdávaním semestrálnej práce v rámci jednotlivých iterácií, ako aj kontroly splnenia požiadaviek na iterácie. Časť je graficky približená diagramom 2.4.

UC21 – Kontrola dodržania požiadaviek iterácie

Prípad užitia umožňuje študentovi požiadať systém o kontrolu splnenia jednotlivých požiadaviek na zvolenú iteráciu. Systém kontroluje, či aktuálna verzia semestrálnej práce spĺňa požiadavky na zvolenú iteráciu. Verzia semestrálnej práce o ktorej kontrolu aktér požiadal sa archivuje v systéme.
Aktér: Študent

UC22 – Prehlaď skontrolovaných verzií semestrálnej práce

Prípad užitia umožňuje študentovi prezrieť si zoznam všetkých verzií semestrálnej práce, ktoré študent dal v minulosti skontrolovať systému. Systém mu v scrollbare pri každej verzii ukáže podmienky iterácie, ktoré nesplnil. Aktér má možnosť verzie vyfiltrovať podľa ID, iterácie, dátumu spracovania, dátumu odovzdania, získaných bodov a stavu.
Aktér: Študent

UC23 – Detailný náhľad skontrolovanej verzie semestrálnej práce

Prípad užitia umožňuje študentovi detailne zobraziť verziu semestrálnej práce, ktorú dal systému skontrolovať v rámci jednej z iterácií. Detailný náhľad zahŕňa výpis požiadaviek na iteráciu, jednotlivé časti semestrálnej práce ako aj farebne vyznačené splnenie a nesplnenie požiadaviek na jednotlivé časti semestrálnej práce. Ak ide o verziu, ktorá bola ohodnotená učiteľom, vidí študent aj bodový zisk a komentáre učiteľa.
Aktér: Študent

UC24 – Porovnanie skontrolovanej verzie so staršou verziou

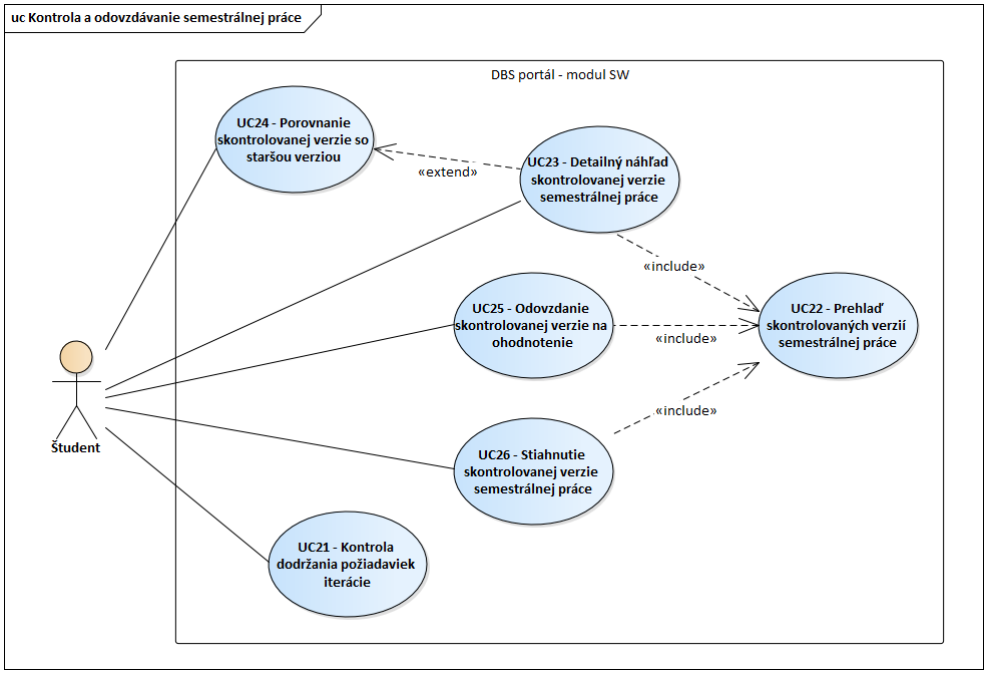
Prípad užitia umožňuje študentovi porovnať skontrolovanú verziu semestrálnej práce s jednou z predošlých skontrolovaných verzií.
Aktér: Študent

UC25 – Odovzdanie skontrolovanej verzie na ohodnotenie

Prípad užitia umožňuje študentovi odovzdať skontrolovanú verziu semestrálnej práce v rámci jednej z iterácii na ohodnotenie učiteľovi.
Aktér: Študent

UC26 – Stiahnutie skontrolovanej verzie semestrálnej práce

Prípad užitia umožňuje študentovi stiahnuť verziu semestrálnej práce, ktorú dal skontrolovať systému, vo formáte .zip.
Aktér: Študent



Obr. 2.4: Use Case diagram Kontrola a odovzdávanie semestrálnej práce

2.2.5 Spravovanie odovzdaných semestrálnych prác

Graficky priblížené na obrázku 2.5. Poznáme 2 typy odovzdaní. Ak ide o prvé odovzdanie študenta v rámci iterácie ide o štandardné odovzdanie. V opačnom prípade rozprávame o takzvanom „znovuodovzdaní“.

UC27 – Prehľaď odovzdaných verzií semestrálnej práce

Prípad užitia umožňuje učiteľovi zobraziť odovzdané verzie semestrálnej práce všetkých spravovaných študentov na jednej stránke a filtrovať ich podľa zvolených filtrov. Medzi dostupné filtre patria iterácia, dátum odovzdania, dátum ohodnotenia, užívateľské meno študenta, meno a priezvisko študenta, paralelka, získane body, stav odovzdanej iterácie.
Aktér: Učiteľ

UC28 – Náhľad verzie odovzdanej semestrálnej práce

Prípad užitia umožňuje aktérovi detailne zobraziť verziu semestrálnej práce študenta, ktorú odovzdal k ohodnoteniu.
Aktér: Učiteľ

UC29 – Oprava odovzdanej verzie semestrálnej práce

Prípad užitia umožňuje učiteľovi opraviť odovzdanú iteráciu študenta. Pri každej požiadavke má učiteľ možnosť stiahnuť študentovi body, môže každý z bodov okomentovať. Učiteľ takisto môže pridať bonusové body a stiahnuť body za neskoré odovzdanie. Učiteľ ma možnosť okomentovať odovzdanie celkovo.
Aktér: Učiteľ

UC30 – Vyhodnotenie odovzdanej verzie semestrálnej práce

Prípad užitia umožňuje aktérovi vyhodnotiť odovzdanú verziu semestrálnej práce, v rámci jednej z iterácií oboduje študentovu prácu.
Aktér: Učiteľ

UC31 – Stiahnutie odovzdanej verzie semestrálnej práce

Prípad užitia umožňuje učiteľovi stiahnuť odovzdanú iteráciu študenta v .zip subore, ktorý obsahuje jednotlivé časti semestrálnej práce.
Aktér: Učiteľ

UC32 – Porovnanie odovzdanej verzie s predošlou verziou

Prípad užitia umožňuje učiteľovi porovnať odovzdanie s jedným z predošlých odovzdaní alebo znovuodovzdaní.
Aktér: Učiteľ

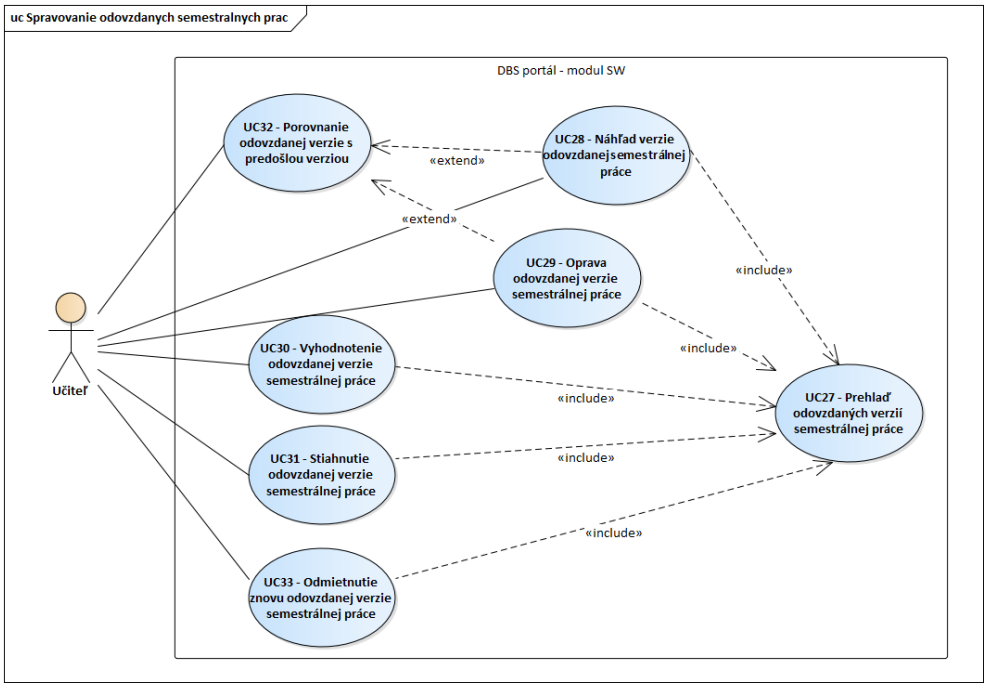
UC33 – Odmietnutie znovu odovzdanej verzie semestrálnej práce

Prípad užitia umožňuje učiteľovi neakceptovať znovuodovzdanie. Študentovi v tomto prípade ostávajú body z predošlého hodnotenia danej iterácie.
Aktér: Učiteľ

2.3 Definícia rozšírenia

V tejto časti predstavím požiadavky na modul, ktoré v aktuálnom riešení nie sú naimplementované a ja sa pri návrhu budem snažiť o ich implementáciu. Požiadavky na portál rozdelím do dvoch kategórií:

funkčné: už z názvu dokážeme určiť že definujú funkcionalitu systému – veci, ktoré užívateľ môže v systéme vykonávať. Bližšie sú definované



Obr. 2.5: Use Case diagram Spravovanie odovzdaných semestrálnych prác

práve modelom prípadov užitia, s ktorým sme sa stretli v predošlej časti analýzy.

nefunkčné: jedná sa o požiadavky, ktoré:

- určujú obmedzenia kladené na systém,
- majú zásadný dopad na návrh architektury
- určujú dodržiavanie štandardov. [34]

2.3.1 Funkčné požiadavky

F1 – DEMO semestrálna práca

Systém umožňuje učiteľovi vytvoriť DEMO semestrálku, teda semestrálnu prácu vedenú pod učiteľovým menom, ktorú môže vypĺňať rovnako ako študenti. Učiteľ môže svoju DEMO semestrálku odkryť zvolenej paralelke, ktorú vyučuje. Študenti paralelky uvidia nazdieľanú DEMO semestrálnu prácu a môžu si prezrieť jej jednotlivé časti.
Priorita: vysoká
Náročnosť: vysoká

2.3.2 Nefunkčné požiadavky

N1 – Voľba technológií

Ostatné časti portálu sú naprogramovaný v PHP s využitím frameworku Nette. Nový prototyp modulu testy naviac využíva framework Doctrine. Keďže je žiaduce, aby spravovanie a rozširovanie portálu bolo čo najjednoduchšie, je potrebné pri návrhu zvoliť rovnaké technológie, konkrétne Doctrine a Nette.
Priorita: vysoká
Typ: implementácia
Náročnosť: vysoká

N2 – Nízka previazanosť modulu

Nízka previazanosť modulu semestrálna práca povedie k jednoduchšej úprave iných modulov bez potrebných veľkých zmien v module semestrálna práca. Takisto je nízka provázanost jedným z princípov kvalitného kódu a objektovo-orientovaného programovania.
Priorita: vysoká
Typ: supportability[xiii]
Náročnosť: vysoká

N3 – Zachovanie starých semestrálných prác

Počas minulých behov predmetu DBS študenti vytvárali svoje semestrálne práce, ktoré sú na portáli uložené. Bývalí študenti si po prihlásení školským účtom svoje práce môžu prezerať. Je žiaduce, aby to si ho svoje práce mohli prezerať aj po nasadení tohto modulu.
Priorita: vysoká
Typ: použiteľnosť
Náročnosť: stredná

2.4 Závažné nedostatky aktuálneho riešenia

Nezistil som žiadne zásadné nedostatky aktuálneho riešenia modulu semestrálna práca v rámci kódu. Medzi pár nie závažných, ale stále nedostatkov, by som zaradil:

Nekonzistencia kódu – ako som už spomínal, vývojársky tím pracujúci na portáli sa často obmieňa. V rámci projektu je určený code style[xiv] ktorý je navyše kontrolovaný automatickými testami pri nahrávaní upraveného a nového kódu na GitLab[xv]. Napriek týmto pravidlám a ich dodržiavaniu sa stretávame s nekonzistentným kódom ktorý sa ťažšie udržiava.

[xiii] podporovateľnosť / rozšíriteľnosť
[xiv] pravidlá, ako písať a formátovať zdrojový kód
[xv] webový portál zameraný na údržbu a ukladanie kódu s viacerými ďalšími funkciami

Rozsiahlosť kódu – niektoré z tried majú veľký rozsah. Konkrétne trieda CheckPresenter má aktuálne 945 riadkov. Údržba a prípadne vylepšovanie takto rozsiahlej triedy je časovo náročné. Z môjho pohľadu sa do tohto stavu triedy dostali jednoduchým spôsobom – v počiatkoch vývoja bol dobrý návrh rozloženia tried, avšak postupným pridávaním požiadaviek na portál sa namiesto pridania nových tried a delegovania funkcionality na ne rozširovali existujúce triedy.

Vysoká previazanosť logickej a databázovej vrstvy – aktuálne riešenie vo viacerých metodách dotazuje dáta z databázy, s ktorými následne pracuje. Toto dotazovanie často prebieha písaním SQL príkazov, v ktorých sa neskúsený programátor môže jednoducho pomýliť a dlho hľadať chybu. Framework Doctrine poskytuje výborné riešenie tohto problému, repository triedy, ktoré predstavím v kapitole Návrh.

Kontrola splnenia požiadavok – v aktuálnom riešení výsledok kontroly splnenia požiadavok na iteráciu odovzdaním nie je ukladaný do databázy. Znamená to, že pri každom načítaní stránky s náhľadom odovzdaných verzií semestrálnej práce sa splnenie požiadaviek kontroluje opätovne.

Všetky zo spomenutých nedostatkov sa pokúsim v ďalšej kapitole Návrh vziať do úvahy a eliminovať v novom riešení.

2.5 Previazanosť s ostatnými časťami portálu

Aktuálne riešenie modulu je v určitých častiach previazané s ostatnými časťami portálu. Konkrétne:

Administrácia - priradzovanie študentov a učiteľov k paralelkám, ako aj nastavovanie dátumu odovzdávania iterácii je z môjho pohľadu administratívna činnosť. Napriek tomu som v analýze aktuálneho riešenia zistil, že ide o funkcionalitu modulu semestrálna práca a preto na tieto aktivity nemôžem zabudnúť v návrhu.

Pripojenie - má v portále vlastný malý modul. Keďže však jedno z pripojení patrí k semestrálnej práci, musím sa naň v návrhu zamerať.

Notifikácie - trieda NotificationService a ďalšie triedy app/Notification namespacu[xvi] obsluhujú notifikácie v module semestrálnej práce. Tento systém bude vhodný aj pre nové riešenie, keďže je používaný v rámci celého portálu.

[xvi] slúži k združovaniu tried, ktoré spolu súvisia

Bezpečnosť a prístupnosť v module rieši trieda AclFactory, ktorá používa funkcionalitu frameworku Nette. Tento framework som zvolil ako jednu z implementačných technológií a je v projekte zaužívaný, preto nevidím dôvod hľadať niečo iné a považujem ho za vhodné riešenie tejto problematiky pre modul.

2.6 Zhrnutie

V analýze som sa zoznámil s jednotlivými aktérmi modulu. Následne som zistil, čo ktorý aktér môže v module vykonávať a tak som pochopil funkcionalitu aktuálneho riešenia. Zadefinoval som si nové funkčné a nefunkčné požiadavky na systém. Spísal som aj nedostatky aktuálneho riešenia. Nakoniec som sa vyjadril k previazanosti modulu s ostatnými časťami. Všetky tieto poznatky sa pokúsim zúžitkovať k vytvoreniu lepšieho návrhu.

Kapitola 3

Návrh

V návrhu sa sústredím hlavne na databázu a systém, okrajovo sa vyjadrím aj k iným veciam. Využijem poznatky získané v analýze.

3.1 Návrh vymenovaných typov

V rámci databázy ako aj kódu je viacero prípadov, v ktorých je potrebné rozlíšiť, k akej časti semestrálnej práce sa daný komentár alebo požiadavka viaže, prípadne pre akú časť semestrálnej práce je daná funkcia volaná. Na zjednodušenie práce v týchto prípadoch, ako aj udržanie prehľadnosti kódu využijem návrhový vzor vymenovaný typ, známy pod anglickou skratkou **enum**. Ide o konečnú obmedzenú množinu pomenovaných hodnôt. Každý člen je tvorený názvom a hodnotou. [35]

3.1.1 Vymenovaný typ SWPartEnum

Tento vymenovaný typ reprezentuje časť semestrálnej práce. Členmi tejto množiný sú:

- TITLE = 0
- DESCRIPTION = 1
- CONCEPTUAL_MODEL = 2
- LOOP_DISCUSSION = 3
- RELATIONAL_MODEL = 4
- CREATE_SCRIPT = 5
- INSERT_SCRIPT = 6
- CATEGORY_TABLE = 7

- QUERY = 8
- CONCLUSION = 9
- REFERENCES = 10
- SQL_DEV_SOURCE = 11

Bude využitý pri ukladaní dát do databázy, ako aj

3.1.2 Vymenovaný typ SWTurnStatus

Druhý vymenovaný typ som prevzal z aktuálneho riešenia. Popisuje stavy, v ktorých sa môže počas svojho životného cyklu nachádzať jedna verzia semestrálnej práce. Obrázok 3.1 približuje členov tejto množiny. Obrázok som prevzal z aktuálneho kódu DBS portálu. V rámci mojej implementácie preň vytvorím novú triedu.

```
public const IN_QUEUE = 0;
public const PROCESSED = 1;
public const SUBMITTED = 2;
public const RESUBMITTED = 3;
public const LOCKED = 4;
public const RELOCKED = 5; // "In evaluation"
public const EVALUATED = 6;
public const REJECTED = 7;
```

Obr. 3.1: Vymenovaný typ SWTurnStatus

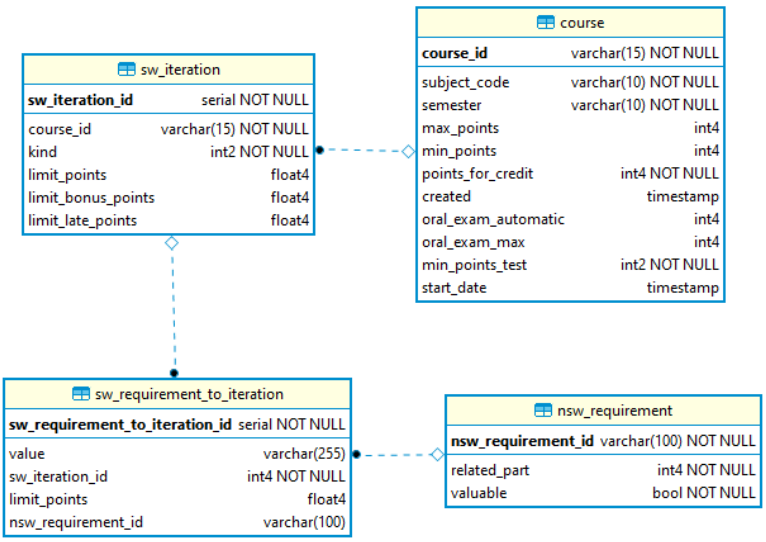
3.2 Návrh databázy

Nie je veľa nových funkčných požiadaviek, návrh novej databázy môže čerpať zo starej databázy s pridaním tabuliek riešiacich nové požiadavky a upravením pár tabuliek na z môjho pohľadu vhodejšie riešenie.
Novú databázu postupne predstavím.

3.2.1 Ukladanie bodovania a požiadaviek na iterácie

Tabuľky patriace do tejto kategórie sú na obrázku 3.2.
Tabuľka **course** zastupuje jeden kurz, je v nej uložené bodovanie semestrálnej práce. Course je vo vzťahu s iteráciami. Pod jeden kurz môže patriť viac iterácií.
Iterácia je zastúpená tabuľkou **sw_iteration**. Iterácia je definovaná typom[xvii]

[xvii] v rámci BI-DBS prvá, druhá a tretia



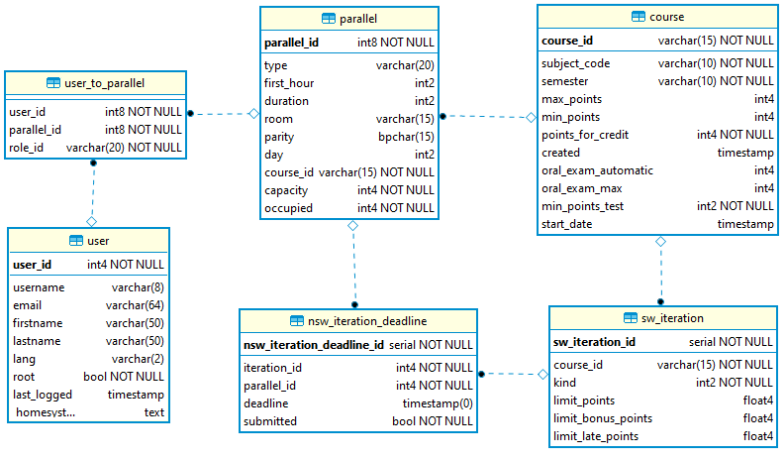
Obr. 3.2: Databáza bodovanie

a takisto má zadefinovaný počet získateľných bodov, počet získateľných bonusových bodov a limit bodov strhnuteľných za neskoré odovzdanie.
Jednotlivé typy požiadaviek sú uložené v tabuľke **nsw_requirement**. Pod ID je uložený typ požiadavky, tabuľka takisto ukladá, na akú časť semestrálnej práce sa požiadavka vzťahuje pomocou enumu SWPartEnum a informácie o tom, či je požiadavka valuable.
Priradzovanie jednotlivých požiadaviek k iteráciam prebieha pomocou tabuľky **sw_requirement_to_iteration**. Každá z požiadaviek má body za nesplnenie a hodnotu, ak je príslušná požiadavka valuable.

3.2.2 Ukladanie informácii o paralelkách

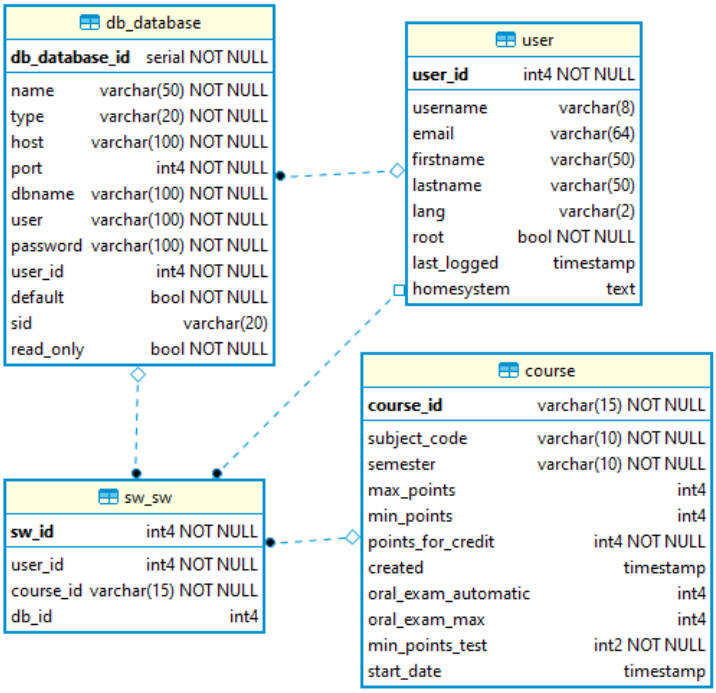
Obrázok 3.3 zachytáva tabuľky patriace do tejto sekcie.
Tabuľka **parallel** je prepojená s nám známou tabuľkou **course**. Parallel obsahuje informácie o jednotlivých paralelkách, ktoré získava z KOSu[xviii].
Tak ako iterácií, aj paraleliek môže pod jeden course patriť viacero. Dátum deadlinu konkrétnej iterácie pre konkrétnu paralelku je uložený v tabuľke **nsw_iteration_deadline**. Atribút submitted oznamuje, či učiteľ vyplnil deadline.
Užívatelia zastúpeni tabuľkou **user** sú priradzovaní k paralelkám pomocou tabuľky **user_to_parallel**.

[xviii] informačný systém používaný k administrácii na ČVUT



Obr. 3.3: Databáza paralelky

3.2.3 Semestrálna práca

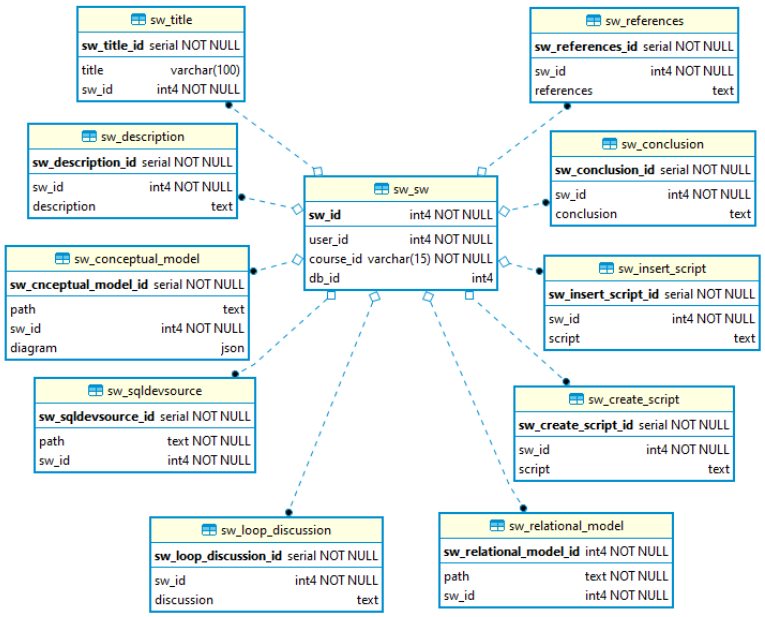


Obr. 3.4: Databáza semestrálna práca

Semestrálna práca je zastúpená tabuľkou **sw_sw**, priradená užívateľovi a patriaca pod kurz. Pripojenie semestrálnej práce je vyriešené vzťahom s

tabuľkou **db_database**.
Tabuľka **db_database** zastupuje jedno z pripojení uživateľa, pričom jeden užívateľ môže mať viacero pripojení, ale len jedno z nich môže nastaviť ako pripojenie svojej semestrálnej práce. Tieto tabuľky sú zachytené na obrázku 3.4. User ma vlastný **db_database**, ktorých môže mat viac. **db_database** je tabuľka ukladajúca informácie o pripojeniach pre daného usera.

3.2.4 Časti semestrálnej práce



Obr. 3.5: Databáza časti semestrálnej práce

Časti semestrálnej práce sú všetky zastúpené vlastnými tabuľkami, ktoré sú vo vzťahu so semestrálnou prácou, ku ktorej patria. Konkrétne ide o tabuľky:

sw_title ukladá názov semestrálnej práce,

sw_description ukladá popis semestrálnej práce,

sw_conceptual_model ukladá konceptuálny model semestrálnej práce,

sw_sqldevsource ukladá naimportovaný SQL Developer source,

sw_loop_discussion ukladá diskusiu smyčiek k semestrálnej práci,

sw_relational_model ukladá relačný model semestrálnej práce,

sw_create_script ukladá create script semestrálnej práce,

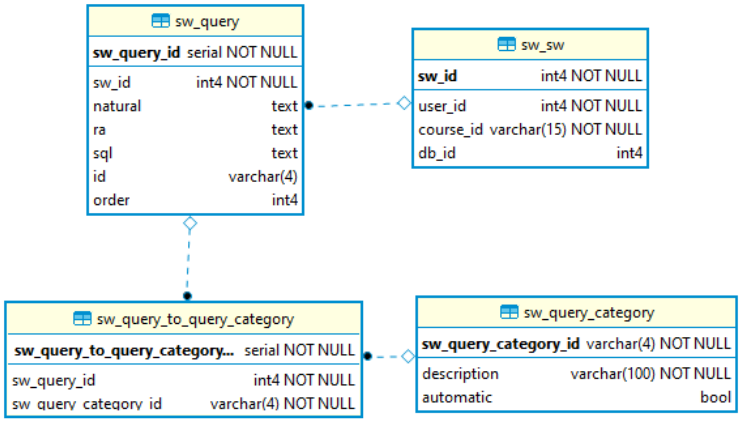
sw_insert_script ukladá insert script semestrálnej práce,

sw_conclusion ukladá záver semestrálnej práce,

sw_references ukladá zdroje semestrálnej práce.

Tieto časti sú zachytené na obrázku 3.5.

3.2.5 Dotazy semestrálnej práce



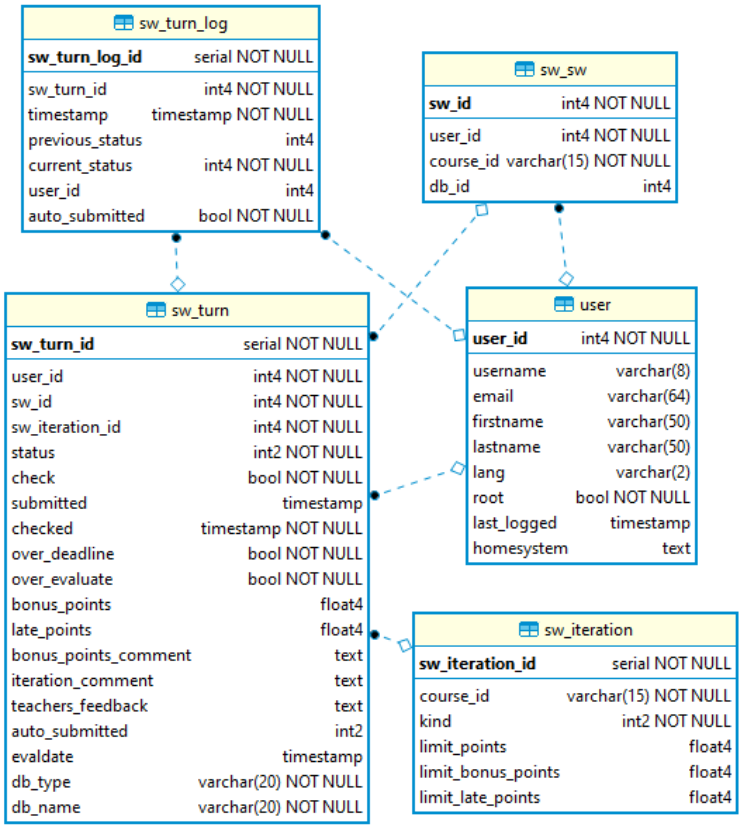
Obr. 3.6: Databáza dotazy

K jednej semestrálnej práci môže patriť viac než jeden dotaz. Dotaz je zastúpený tabuľkou **sw_query** ktorá ukladá znenie dotazu v normálnom jazyku, znenie dotazu v relačnej algebre, znenie dotazu v SQL, ID dotazu v rámci konkrétnej semestrálnej práce a poradie dotazu v prehľade semestrálnej práce[xix].
Jednotlivé kategórie dotazov sú uložené v tabuľke **sw_query_category**, ktorá okrem ID a popisu kategórie má uložené aj to, či ide o automaticky priradzovanú kategóriu alebo ju študent musí ručne k dotazu priradiť.
Tabuľka **sw_query_to_query_category** je zodpovedná za ukladanie priradených kategórií k dotazom.
Táto časť databázy je zachytená na obrázku 3.6.

3.2.6 Kontrola a odovzdávanie semestrálnej práce

Ku kontrole a následnému odovzdávaniu k ohodnoteniu semestrálnej práce je navrhnutá primárne tabuľka **sw_turn**. Pre každú verziu semestrálnej práce,

[xix] poradie sa môže líšiť od ID

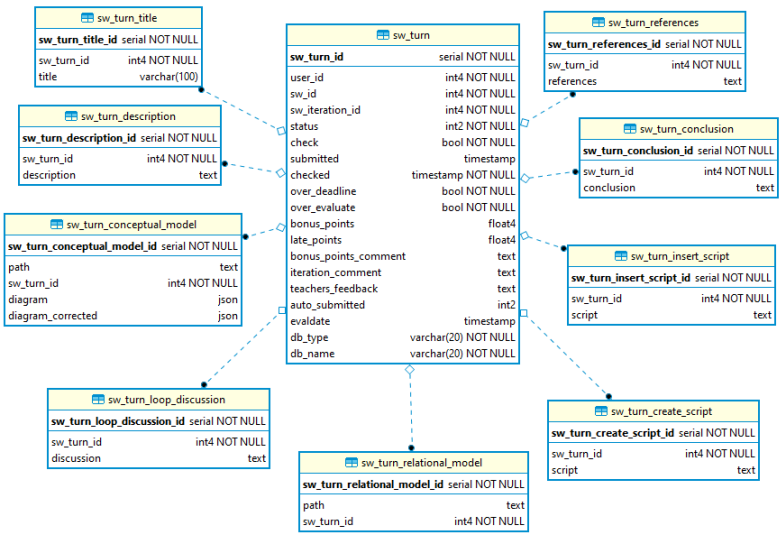


Obr. 3.7: Databáza kontrola a odovzdávanie

ktorú chce študent skontrolovať systémom je v nej vytvorený nový záznam. Turn je vo vzťahu s príslušnou iteráciou, užívateľom, semestrálnou prácou. Medzi nadôležitejšie argumenty tejto tabuľky patrí aktuálny status odovzdania, dátum kontroly, dátum odovzdania, informácia či ide o znovuodovzdanie ako aj informácia či ide o odovzdanie po deadline. Táto tabuľka takisto ukladá bonusové body a body strhnuté za neskoré odovzdanie, ako aj komentár učiteľa k týmto bodom.
Tabuľka **sw__turn__log** slúži na ukladanie životného cyklu semestrálnej práce. Vždy, keď sa zmení stav jedného z odovzdaní, vznikne nový zápis vo vzťahu s konkrétnym odovzdaním. Okrem dátumu zmeny statusu je ukladaný aj predošlý a nový status odovzdávania.

3.2.7 Časti odovzdávania semestrálnej práce

Je žiaduce, aby študent mal prístup ku svojim starším odovzdaniam. Je teda potrebné jednotlivé časti odovzdanej semestrálnej práce niekam ukladať. Slúžia k tomu tabuľky zachytené na obrázku 3.8. Všetky sú vo vzťahu s tabuľkou



Obr. 3.8: Databáza časti odovzdávania

konkrétneho odovzdania. Ide o tabuľky:

sw_turn_title pre názov odovzdanej verzie semestrálnej práce,

sw_turn_description pre popis,

sw_turn_conceptual_model pre konceptuálny model,

sw_turn_sqldevsource pre naimportovaný SQL Developer source,

sw_turn_loop_discussion pre diskusiu smyčiek,

sw_turn_relational_model pre relačný model,

sw_turn_create_script pre create script,

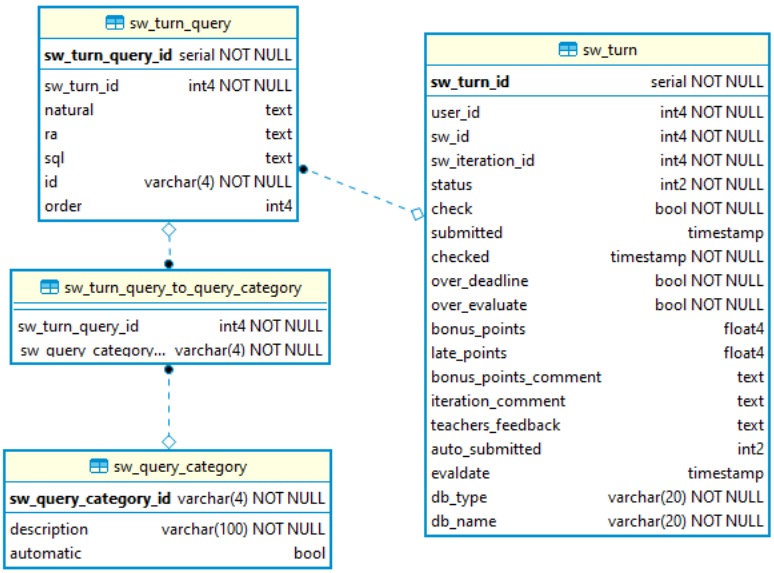
sw_turn_insert_script pre insert script,

sw_turn_conclusion pre záver,

sw_turn_references pre zdroje.

3.2.8 Dotazy odovzdanej semestrálnej práce

Tabuľka **sw_turn_query** slúži k ukladaniu dotazov patriacich k odovzdaniu. Každý z dotazov je vo vzťahu s odovzdaním, ku ktorému patrí. Takisto je vo vzťahu s už spomínanou tabuľkou kategórie **sw_query_category**. Spomínané tabuľky sú na obrázku 3.9.



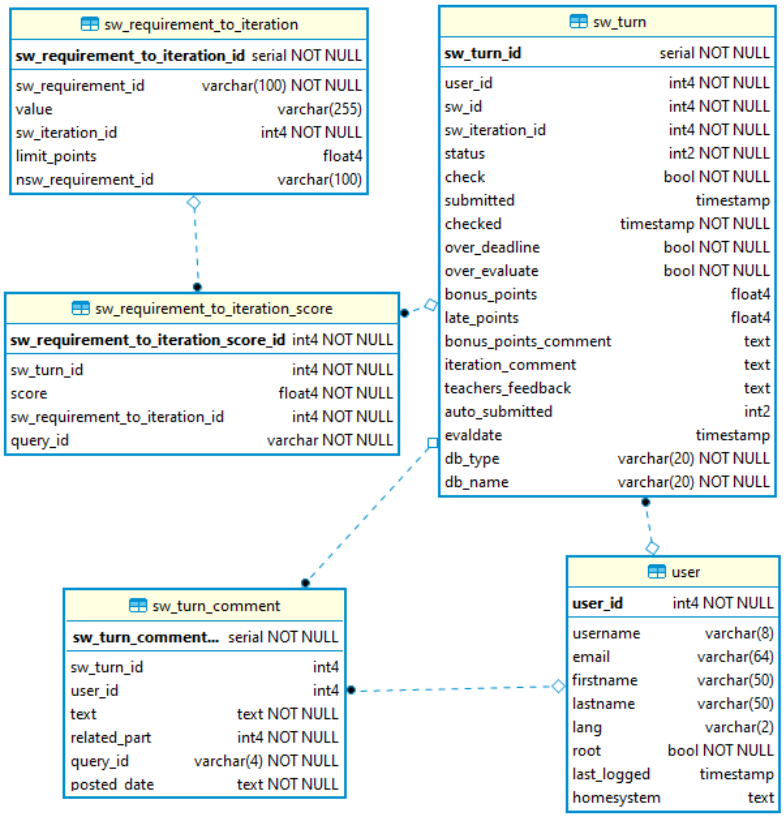
Obr. 3.9: Databáza dotazov odovzdávania

3.2.9 Komentáre a bodovanie odovzdania

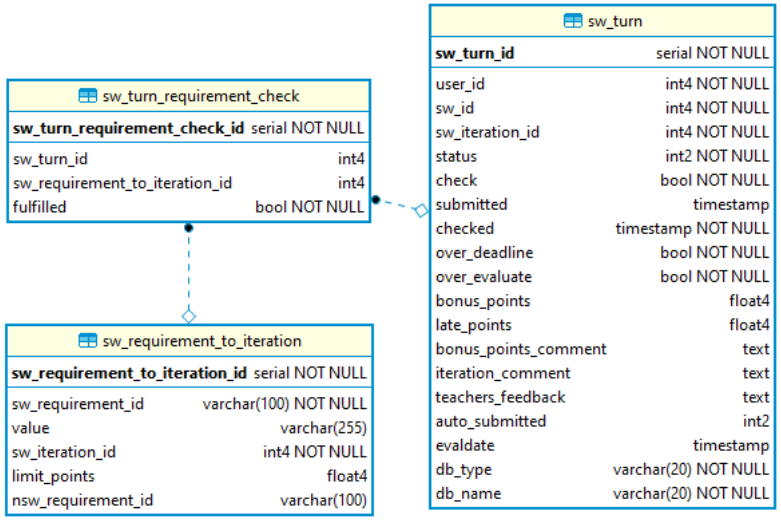
Komentár študenta k odovzdaniu je v tabuľke **sw_turn**, rovnako ako učiteľov feedback na celé odovzdania a komentár k bonusovým bodom. Bonusové body a body stiahnuté za neskoré odovzdanie sú taktiež atribútmi tabuľky **sw_turn**.
Tabuľka **sw_turn_comment** slúži na ukladanie komentárov učiteľa k jednotlivým častiam semestrálnej práce. Je vo vzťahu s užívateľom, ktorý komentár zadal, ako aj odovzdaním, ku ktorému patrí. Atribútmi sú okrem samotného komentára a dátumu zapísania komentára aj časť semestrálnej práce, na ktorú sa komentár vzťahuje. V rámci databázy ide o integer, v rámci kódu sa pracuje s SWPartEnum spomenutým v sekcii o enumoch. Atribút query_id je využívaný v prípade dotazu, aby sme vedeli o ktorý konkrétny dotaz ide.
Body stiahnuté za nesplnenie požiadaviek na iteráciu sú ukladané do tabuľky **sw_requirement_to_iteration_score**, ktorá je vo vzťahu so zadefinovanou požiadavkou ako aj odovzdaním, ku ktorému hodnotenie patrí.
Jednotlivé tabuľky sú na obrázku 3.10.

3.2.10 Kontrola splnenia požiadaviek

Ako som spomenul v kapitole analýza, jednou z vecí, ktoré by som chcel oproti aktuálnemu riešeniu zlepšiť, je kontrola splnenia požiadaviek na iteráciu odovzdaním. Kontrola sa bude prevádzať len raz a jej výsledky budú uložené do databázy. Bude k tomu slúžiť tabuľka **sw_turn_requirement_check**. Je vo vzťahu s konkrétnym odovzdaním a požiadavkou, jej atribút fulfilled



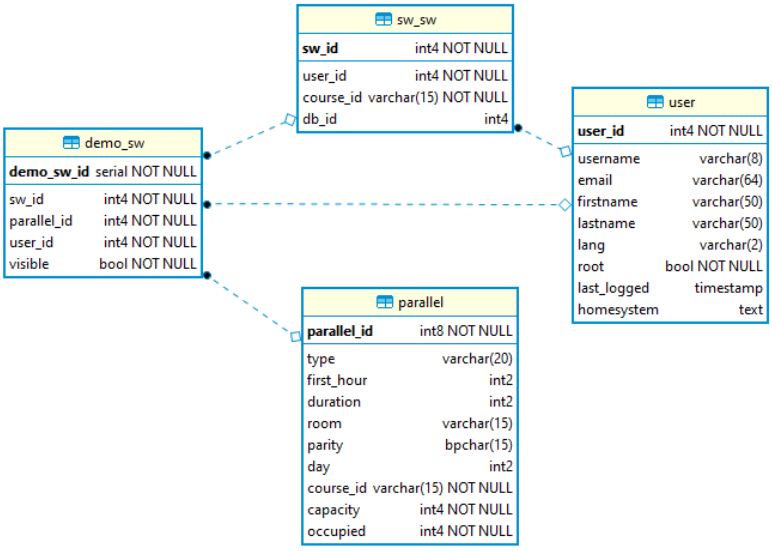
Obr. 3.10: Databáza komentárov a bodovania odovzdávania



Obr. 3.11: Databáza splnenie požiadaviek

značí, či bola daná požiadavka splnená. Graficky je tento prípad zachytený na obrázku 3.11.

3.2.11 DEMO semestrálna práca



Obr. 3.12: Databáza DEMO semestrálna práca

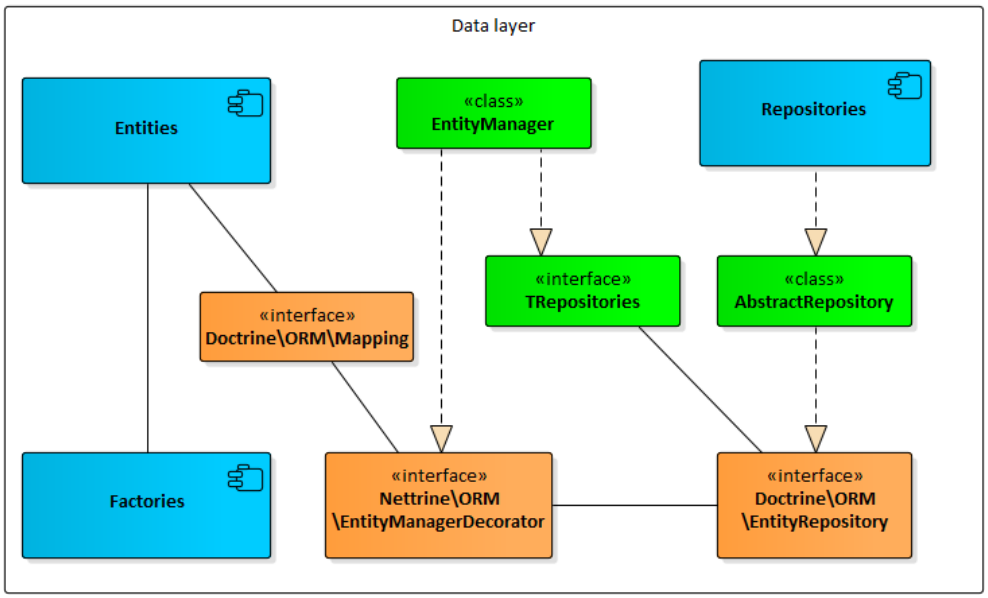
Semestrálna práca, ktorú budú mať učitelia možnosť vypracovať sa bude ukladať do tabuľky **sw_sw**. K následnému nazdieľaniu študentom dopomôže tabuľka **demo_sw**, ktorá má bool visible, teda či práca je pre paralelku viditeľná a je vo vzťahu so semestrálna prácou, ktorú chce učiteľ nazdieľať, paralelkou, ktorej ju chce nazdieľať a učiteľom samotným. Zachytené je to na obrázku 3.12.

3.3 Návrh systému

Systém ktorý navrhujem, rozdelím na dve časti. Dátová vrstva pracuje s databázou, zatiaľ čo logická vrstva je sústredená na logické procesy. Výsledkom kvalitného návrhu by malo byť také rozdelenie backendu, aby bola dátová vrstva s logickou čo najmenej previazané. Rozdelenie vrstiev nedosiahneme stopercentné, nakoľko hlavnou výhodou frmaeworku Doctrine je práca s entitami databázy ako objektmi PHP.

3.3.1 Dátová vrstva

Dátovú vrstvu graficky približuje obrázok 3.13. Procesy s databázou vieme rozdeliť do 3 kategórii.



Obr. 3.13: Dátová vrstva

1. čítanie existujúcich dát
2. vkladanie nových dát
3. úprava existujúcich dát

V rámci modulu bude všetko čítanie dát na repository triedach. Tieto triedy budú mať funkcie, vracajúce instancie databázových entít.
Vkladanie nových dát bude prebiehať vytváraním nových instancí tried a ich následným persistnutím do databázy. Factories a trieda EntityManager sa postarajú o tieto úkony.
Úprava existujúcich dát bude prebiehať úpravou atribútov instancií entít a následným persistnutím instancie do databázy. Použijeme EntityManager.
Nasledujúce časti predstavia spomínané triedy.

3.3.1.1 Entity

Triedu entity potrebuje každá databázová tabuľka, ktorá má primárny kľúč. Entita je v PHP kóde zadefinovaná ako trieda[xx], ktorej atribútmi sú jednotlivé stĺpce príslušnej tabuľky.
Atribúty budú privátne, teda vonkajšie triedy k ním budú môcť pristupovať a meniť ich len pomocou funkcií na to určených, takzvaných getters[xxi] a setters[xxii].

[xx] class
[xxi] na čítanie atribútu
[xxii] na zmenu atribútu

Triedy entít a ich atribúty „namapujem“ pre framework Doctrine pomocou cudzieho interfacu Doctrine\ORM\Mapping. Príklad mapovania ukážem v časti implementácia.

3.3.1.2 Repositories

Doctrine štandartne pre každú entitu vytvorí implementáciu jeho repository. Pod repository si čitateľ môže predstaviť triedu, ktorej funkcie slúžia na prístup k dátam z tabuľky príslušnej entity.
Štrandartná verzia repository implementovaná frameworkom Doctrine obsahuje len základné funkcie. Programátor štandartnú verziu môže rozšíriť vytvorením vlastnej repository triedy, ktorú následne pomocou anotácií priradí ku konkrétnej entite.
Väčšina entít bude mať custom repository class a získavanie dát z databázy bude prebiehať výhradné volaním funkcí zadefinovaných v repository triedach entít.
Hlavnou triedou v tejto sekcii je AbstractRepository, ktorá dedí z Doctrine\ORM\EntityRepository a je rodičovskou triedou všetkých entity repository tried. Pridanie tejto triedy umožní budúcim vývojárom jednoducho pridať funkcie, ktoré budú prístupne pre všetky repository triedy.

3.3.1.3 EntityManager

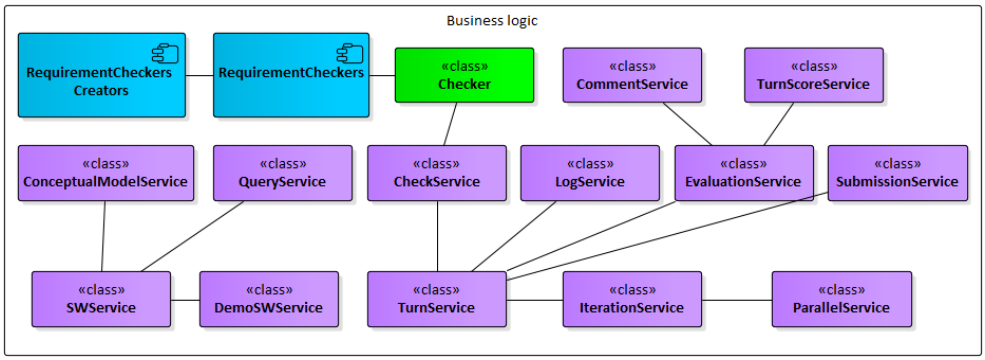
EntityManager je hlavná trieda, ktorá bude riešiť komunikáciu logickej vrstvy s databázou. Dedí z Nettrine\ORM\EntityManagerDecorator, čo je jeden z interfacov frameworku Nettrine. Vďaka tomuto spojeniu dokáže EntityManager pracovať s jednotlivými entitami a databázou.
EntityManager navyše využíva funkcie interface triedy TRepositories, ktorá definuje funkcie pre získavanie jednotlivých custom repositories. Spájajú sa v nej všetky činnosti s databázou a je teda univerzálnym mostom medzi dátovou a logickou vrstvou.

3.3.1.4 Factories

Sú triedy, ktorých funkcie vytvárajú nové instancie tried s predloženými argumentami. Naimplementujem ich pre triedy, ktorých nové záznamy často pridávame do databázy. Ide o dotazy k semestrálnej práci, odovzdania semestrálnej práce, časti odovzdania, komentáre k odovzdaniu, hodnotenie odovzdania, tabuľka pre kontrolu splnenia požiadavky na iteráciu odovzdaním, ...

3.3.2 Logická vrstva

Logická vrstva bude zložená z viacerých tried, najmä však service tried ktoré definujú logiku príslušnej časti modulu. Každá service trieda je prepojená s



Obr. 3.14: Logická vrstva

EntityManager triedou z dátovej vrstvy. Piatimi základnými sú:

- SWService,
- DemoSWService,
- TurnService,
- IterationService,
- ParallelService.

Všetky service triedy pracujú s EntityManager triedou, ktorá im umožňuje komunikovať s databázou. Niektoré sú navzše v spojení s inými service triedami, ktoré môžeme nazvať pomocnými. Pomocné triedy definujú vlastnú časť logiky a zabezpečujú prehľadnosť kódu. Vrstva a prepojenia sú priblížené obrázkom 3.14.

3.3.2.1 Iterácie

Už podľa názvu vieme určiť, že **IterationService** obsluhuje logiku spojenú s iteráciami. Je vo vzťahu s ParallelService a TurnService, ktoré ešte len predstavím.

3.3.2.2 Paralelky

Logika spojená s paralelkami je zadefinovaná v triede **ParallelService**. Je v asociácii s IterationService.

3.3.2.3 Kontrola a odovzdávanie

Hlavnou triedou tejto sekcie je **TurnService**. Má viacero pomocných tried, konkrétne:

SubmissionService zameraná na odovzdanie skontrolovanej verzie učiteľovi k ohodnoteniu.

EvaluationService – definuje metódy využívané učiteľom pri kontrole a opravovaní semestrálnych prác. Má dve nápomocné triedy,

- **CommentService** zaoberajúca sa komentármi,
- **TurnScoreService** zaobrajúca sa bodovaním.

LogService rieši životný cyklus odovzdania semestrálnej práce.

CheckService je zamerný na kontrolu splnenia požiadaviek iterácie. Spolupracuje s triedou **Checker** ktorá je vo vzťahu s RequirementChekers. Ide o triedy ktoré dedia z interfacu **RequirementChecker**. Sú zadefinované pre jednotlivé typy požiadaviek a každá z nich kontroluje splnenie svojho konkrétneho požiadavku.
Tieto triedy budú vytvárane RequirementCheckers Creatormi.

3.3.2.4 Semestrálna práca a jej editácia

Trieda SWServcice bude definovať vačšinu metód týkajúcich sa semestrálnej práce a jej častí. Má dve pomocné triedy, ktoré pomáhajú udržiavať prehľadnosť kódu:

QueryService je trieda obsluhujúca dotazy. Vlastnú service triedu má preto, že je najrozšírenejšou časťou semestrálnej práce s rozsiahlou funkcionalitou.

ConceptualModelService je ďalšou z častí semestrálnej práce, ktorá má špeciálnu service triedu. Dôvodom je to, že konceptuálny model nie je entitou s jedným atribútom a existujú dva spôsoby ako nastaviť konceptuálny model.

3.3.2.5 DEMO Semestrálna práca

Obsluhou požiadaviek DEMO semestrálnej práce sa zaoberá trieda **DemoSWService**. Je vo vzťahu so SWService.

3.4 Zachovanie starých semestrálných prác

Tabuľky, ktoré sú zodpovedné za ukladanie častí semestrálnej práce, som oproti aktuálnemu riešeniu nemenil. Semestrálne práce vytvorené pred nasadením tohto prototypu by teda mali byť dostupné k náhľadu aj po nasadení tohto prototypu. Veci týkajúce sa odovzdávania som v návrhu pozmenil. K tejto časti sa vyjadrím v kapitole Budúci rozvoj.

3.5 Zhrnutie

Návrhom som si pripravil podklady pre implementáciu. Podľa návrhu databázy sa v časti implementácia pokúsim prerobiť štruktúru aktuálneho riešenia databázy. Pre každú tabuľku vytvorím entity triedu a repository triedu. Časť návrhu logickej vrstvy mi poslúži pri implementácii tried pre logickú vrstvu.

Kapitola 4

Implemetácia

Priblížim celý postup implementácie, teda ktorými aktivitami som začínal a ako aktivity na seba nadväzovali. Takisto sa pokúsim zamerať na problémy, s ktorými som sa stretol a spôsoby, akými som ich vyriešil. Verím, že to bude nápomocné programátorom snažiacim sa naimplementovať aplikáciu s použitím Doctrine a Nette, ako aj programátorom DBS portálu, ktorí by radi spoznali základy modulu semestrálnej práce.

4.1 Pridanie a registrácia nového modulu

Úplne prvým krokom v rámci implementácie bolo pridanie nového modulu pre môj prototyp do portálu. Modul som nazval newSW. Tento postup pozostával z viacerých krokov, konkrétne:

editácia config súborov: modul newSW som zaregistroval v config súboroch extension.neon a header.neon. Takisto som pridal nové config súbory pre modul a to newsw.neon a sidebar.newsw.neon, ktoré budú využívané na ukladanie konfigurácie modulu. Týmto krokom som do hlavičky DBS portálu pridal záložku newSW modulu.

editácia lang súborov: názov nového modulu som uložil do lang súborov[xxiii] v oboch jazykoch. Vďaka tomuto kroku vie portál, ako nazvať záložku newSW modulu v hlavičke portálu.

registrácia modulu v RouterFactory: RouterFactory je v rámci portálu trieda priradzujúca modulom portálu ich URL. Využíva Nette framework. V tejto triede som zaregistroval modul newSW a priradil som mu URL **newSW/**.

[xxiii] skupina súborov ukladajúca preklad textu viditeľného užívateľom portálu do všetkých dostupných jazykov

pridanie HomepagePresentera pre nový modul: ďalším krokom bolo vytvorenie presenter triedy a ďalších súborov potrebných HomepagePresenterom. Tieto súbory definujú stránku, ktorú užívateľ portálu vidí po zakliknutí záložky newSW v hlavičke portálu. Zároveň mi tento krok pomohol otestovať funkčnosť pridania modulu do portálu.

4.2 Integrácia frameworkov

4.2.1 Pridanie frameworkov

Nasledujúcim krokom v implementácii bolo pridanie a konfigurácia frameworkov, ktoré využijem. Ide o Nette a Doctrine, ktoré som predstavil a o ich použití som informoval v prvej kapitole. Namiesto pridávania Doctrine použijem Nettrine[xxiv].
Pridávanie frameworkov vyriešim pomocou softwaru **Composer**[xxv]. Počas implementácie používam verziu Nette 2.4 a verziu Nettrine 0.3. Tieto informácie sú uložené v **composer.json** súbore.

4.2.2 Konfigurácia Doctrine

Do súboru **config.neon** som pridal konfiguráciu pre framework Doctrine, graficky zachytené na obrázku 4.1. Hlavnými časťami konfigurácie sú:

- rozšírenia, ktoré využijem,
- pripojenie k databáze,
- cesta k triede EntityManager,
- cesta k zložke obsahujúcej triedy entít.

4.2.3 Pridanie doctrine príkazov do konzoly

Časťou Doctrine je Doctrine Console, rozhranie príkazového riadku uľahčujúce programátorovi prevádzanie administratívnych úloh na projekte používajúcom Doctrine. Popis rozhrania ako aj priblíženie jeho jednotlivých príkazov je dostupné na tomto zdroji [37].
Ako si pridať toto rozhranie do konzoly newDBS projektu mi poradil Martin Hanzl a mnou spísaný postup je v návode doctrine.pdf na priloženom pamäťovom médiu.
Takisto bolo potrebné pridať PHP extension **apcu** do vagrant boxu. Ide o rozšrenie jazyka PHP ktoré je vyžadované frameworkom Doctrine. Ako ho

[xxiv] ide o frameworkov integrujúci Doctrine do Nette, získam teda framework Doctrine + nápomocné triedy
[xxv] nástroj určený na správu knižníc a iných závislostí v PHP [36]

```
extensions:
    replicator: Kdyby\Replicator\DI\ReplicatorExtension
    userLogger: App\Model\DI\UserBlueScreenExtension

    # DBAL configuration
    dbal: Nettrine\DBAL\DI\DbalExtension
    dbal.console: Nettrine\DBAL\DI\DbalConsoleExtension

    # ORM configuration
    orm: Nettrine\ORM\DI\OrmExtension
    orm.cache: Nettrine\ORM\DI\OrmCacheExtension
    orm.console: Nettrine\ORM\DI\OrmConsoleExtension
    orm.annotations: Nettrine\ORM\DI\OrmAnnotationsExtension

dbal:
    # database connection
    connection:
        host: localhost
        user: adminDBS
        password: ******
        dbname: newdbs
        driver: pdo_pgsql

orm:
    configuration:
        autoGenerateProxyClasses: true
    entityManagerDecoratorClass: App\Modules\NewSW\Database\EntityManager

orm.annotations:
    # paths to entity classes
    paths:
        - %appDir%/Modules/NewSW/Database

orm.cache:
    defaultDriver: apcu
```

Obr. 4.1: Konfigurácia Doctrine

pridať som sa dozvedel na newDBS Slacku[xxvi] v kanáli development, postup je priblížený v spomínanom návode na nasadenie Doctrine.

[xxvi] softvér na komunikáciu používaný vývojármi newDBS projektu

4.3 Enums

```php
class SWPartEnum
{
    use StaticClass;

    public const TITLE = 0;
    public const DESCRIPTION = 1;
    public const CONCEPTUAL_MODEL = 2;
    public const LOOP_DISCUSSION = 3;
    public const RELATIONAL_MODEL = 4;
    public const CREATE_SCRIPT = 5;
    public const INSERT_SCRIPT = 6;
    public const CATEGORY_TABLE = 7;
    public const QUERY = 8;
    public const CONCLUSION = 9;
    public const REFERENCES = 10;
    public const SQL_DEV_SOURCE = 11;
}
```

Obr. 4.2: Vymenovaný typ SWPartEnum

Vytvoril som triedy pre enumy. Obrázok množiny SWTurnStatus bol v kapitole Návrh. Pridaný typ SWPartEnum je na obrázku 4.2.

4.4 Mapovanie entít

Vytvorenie tried pre entity a ich následné namapovanie pre Doctrine bolo mojou ďalšou aktivitou. Príklad namapovanej triedy User je na obrázku 4.3. Prvým krokom je vytvorenie triedy a priradenie atribútov v jazyku PHP. Druhým krokom je priradenie anotácií k triede a jej atribútom.

4.4.1 Priblíženie anotácií

V anotáciach skratka ORM zastupuje Doctrine\ORM\Mapping. Anotácie pozostávajú z názvu a prípadných argumentov. Najčastejšie som použil anotácie:

@ORM\Entity(repositoryClass=) pri definícii každej triedy. Symbolizuje, že daná trieda je Doctrine Entitou. Nepovinný argument repositoryClass určuje repository triedu danej entity.

@ORM\Table(tabuľka, schema=) je taktiež pri definíci triedy. Atribút tabuľka ukladá názov tabuľky, ktorú entita predstavuje. Schema určuje v ktorej schéme sa tabuľka nachádza.

@ORM\ID() pri atribúte triedy značí, že ide o atribút zastupujúci ID tabuľky.

```php
use Doctrine\ORM\Mapping as ORM;
use Nette\Utils\DateTime;

/**
 * @ORM\Entity(repositoryClass="App\Modules\NewSW\Database\Repository\General\UserRepository")
 * @ORM\Table("`user`", schema="public")
 */
class User
{
    /*============================================  Fields  ================================
    /**
     * @var int
     * @ORM\Id()
     * @ORM\Column(type="integer", name="user_id")
     */
    private $id;

    /**
     * @var string|null
     * @ORM\Column(type="string", length=8, nullable=true)
     */
    private $username;

    /**
     * @var string|null
     * @ORM\Column(type="string", length=64, nullable=true)
     */
    private $email;

    /**
     * @var string|null
     * @ORM\Column(type="string", length=50, nullable=true, name="firstname")
     */
    private $firstName;
```

Obr. 4.3: Trieda User

@ORM\Column() pri atribúte značí, že ide o atribút tabuľky. Má viacero argumentov, najpoužívanejšie:

- type určujúci dátový typ argumentu
- length určujúci dĺžku argumentu
- name určujúci názov argumentu v tabuľke, potrebné vyplniť ak sa názov líšil od zvoleného názvu atribútu v PHP.

@ORM\JoinColumn(name=, referencedColumnName=) pri atribúte ktorý zastupoval vzťah s inou tabuľkou databázy. Argument name rovnako ako v anotácii Column zastupuje názov argumentu v tabuľke. Argument referencedColumnName zastupuje názov atribútu ID tabuľky s ktorou je táto vo vzťahu.

@ORM\ManyToOne(targetEntity=) pri atribúte zastupujúcom vzťah symbolizuje o aký typ vzťahu ide. Argument targetEntity značí triedu entity s ktorou je tabuľka vo vzťahu. Ak obe entity nie sú uložené v rovnakej zložke, je potrebné vyplniť plný názov triedy targetEntity. Použil som aj iné kategórie vzťahov, avšak tento bol najpoužívanejší.

@ORM\MappedSuperclass() pri definiíci supertriedy, bližsie popísané v sekcii o supertriedach. [38]

4.4.2 Kontrola správnosti mapovania

Počas implementácie entít bol veľmi dobrou pomôckou, ktorú by som odporučil všetkým, ktorí plánujú pracovať s Doctrine, príkaz Doctrine Console **orm:info**. Tento príkaz vypísal všetky entity namapované na Doctrine a v prípade syntaktickej alebo logickej chyby v mapovaní ma upozornil, kde sa chyba nachádza a o aký typ chyby ide. Tento príkaz takisto vypíše počet úspešne namapovaných tried, v mojom prípade to po ukončení implementácie bolo 44 tried.

4.4.3 Supertriedy v mapovaní

```
/**
 * Class SWPart
 * @package App\Modules\NewSW\Database\Entity\Superclasses
 * @ORM\MappedSuperclass()
 */
class SWPart
{
    /**
     * @var SemesterWork
     * @ORM\OneToOne(targetEntity="App\Modules\NewSW\Database\Entity\General\SemesterWork")
     * @ORM\JoinColumn(name="sw_id", referencedColumnName="sw_id")
     */
    protected $sw;
```

Obr. 4.4: Trieda SWPart

Superclassy sú funkcionalita Doctrine umožňujúca zadefinovať rodičovskú triedu, ktorá sama nie je entitou. Potomkovia rodičovskej triedy dedia atribúty rodičov [39].
V rámci implementácie s som vytvoril dve superclassy:

SWPart: rodičovská trieda všetkých tried zastupujúcich časť semestrálnej práce. Všetky časti semestrálnej práce majú jeden spoločný atribút a to semestrálnu prácu ku ktorej patria. Tento atribút ako aj jeho getter a setter je zadefinovaný v triede SWpart. Obrázok 4.4 zobrazuje spomínanú triedu a jej atribút.

TurnPart: rodičovská trieda všetkých častí odovzdania. Časti jedného odovzdania majú spoločný atribút práve odovzdanie, v kóde zastúpene triedou SWTurn.

4.4.4 Getters a Setters

Podľa štandardov Doctrine sú všetky atribúty entít private a všetky atribúty supertried protected. Na prístup k týmto atribútom je potrebné vytvoriť getters a setters. K tejto práci som využil funkcionalitu PhpStorm[xxvii] Generate ktorá tieto metódy generuje.

4.5 Repositories

Mojím nasledujúcim krokom bolo naimplementovanie repository tried. Začal som implementáciou repository tried. Triedu AbstractRepository som pridal až neskôr, využiť ma napadlo až počas implementácie. V návrhu som spomínal, že repositories budú slúžiť na získavanie dát z databázy. Pomocou Doctrine funkcií môžem posielať 2 typy dotazov na databázu:

špecifický: očakávam, že v databáze je len jeden záznam, ktorý spĺňa môj dotaz. Doctrine pre tento prípad má funkciu **getSingleResult**. Používanie tejto funkcie som v kóde zabalil do try-catch bloku, ktorý v prípade neunikátného alebo žiadneho výskytu dotazovaného záznamu v databáze spustí časť kódu prichystanú pre danú situáciu. Pri požívaní týchto dotazov som to väčšinou vyriešil tak, že v prípade žiadneho záznamu som vracal null a v prípade neunikátneho záznamu som vyhodil výnimku, ktorá programátora informuje o chybe.

množinový: očakávam, že v databáze je viacero záznamov spĺňajúcich môj dotaz. Doctrine pre tento prípad má funkciu **getResult**. Vždy vráti pole, či je v databáze odpovedajúcich záznamov 0, 1 alebo 15. [38]

Obrázok 4.5 približuje jednu z repository tried, konkrétne IterationDeadlineRepository. Jej prvá funkcia približuje množinový dotaz a druhá dotaz špecifický.

4.6 Migrácia databázy

4.6.1 Priblíženie migrácie

Databázu je potrebné dostať do stavu popísaného v kapitole Návrh. Úpravy databázy potrebujem zdieľať aj s vývojármi, ktorí budú na tomto prototype

[xxvii] softvér uľahčujúci programátorovi písanie kódu a ďalšie aktivity spojené s vývojom, je zameraný na jazyk PHP a ČVUT študentom zabezpečuje licenciu

```
class IterationDeadlineRepository extends AbstractRepository
{
    /**
     * @param Parallel $parallel
     * @return IterationDeadline[]
     */
    public function getAllDeadlinesForParallel(Parallel $parallel): array
    {
        $query = $this->_em->createQuery('SELECT i FROM App\Modules\NewSW\Database\Entity\Iterations\IterationDeadline i
            WHERE i.parallel = :parallel');
        $query->setParameter( key: 'parallel', $parallel->getId());
        return $query->getResult();
    }

    /**
     * @param Parallel $parallel
     * @param Iteration $iteration
     * @return IterationDeadline|null
     */
    public function getDeadline(Parallel $parallel, Iteration $iteration): ?IterationDeadline
    {
        $query = $this->_em->createQuery('SELECT i FROM App\Modules\NewSW\Database\Entity\Iterations\IterationDeadline i
            WHERE i.parallel = :parallel AND i.iteration =:iteration');
        $params = new ArrayCollection(['parallel' => $parallel->getId(), 'iteration' => $iteration->getId()]);
        $query->setParameters($params);
        try {
            return $query->getSingleResult();
        } catch (NoResultException $e) {
            return null;
        } catch (NonUniqueResultException $e) {
            throw new InvalidStateException( message: "non unique result in database" . $e);
        }
    }
}
```

Obr. 4.5: Trieda IterationDeadlineRepository

pracovať v budúcnosti. Tento proces sa nazýva databázová migrácia a newDBS projekt naň používa nástroj **nextras/migrations** [40]. SQL skripty upravujúce databázu sú ukladané do súborov v špeciálnych zložkách používaných spomínaným nástrojom a nahraním na GitLab sú zdieľané s ostatnými vývojármi.

4.6.2 Vytvorenie SQL skriptov

Pri vytváraní SQL skriptov ktoré, upravia databázu do požadovaného stavu, som si pomohol príkazom **orm:schema-tool:update** z Doctrine Console. Tento príkaz prevedie SQL príkazy, ktoré upravia štruktúru databázy do stavu popísaného namapovanými entitami. Príkaz som žiaľ nemohol plne využiť, niektoré namapované entity, ktoré už v databáze boli, chcel Doctrine vytvárať opäť, čím sa proces úpravy databázy zastavil. Mohol som vymazať štruktúru celej databázy a vytvoriť novú pomocou tohto príkazu, avšak vytvorila by sa len databáza z namapovanými entitami a celý portál by bol nefunkčný.
Spomínaný príkaz je spustiteľný s prepínačom **dump-sql**, ktorý namiesto prevádzania príkazov SQL skripty vypíše do konzoly. Tieto skripty som si uložil do súboru a vyfiltroval tie, ktoré som potreboval. Celú migráciu som rozdelil do dvoch súborov:

2021-03-10-134403-newsw-structures.sql ktorý upravoval štruktúru tabuliek a pridával nové,

2021-03-14-114925-newsw-relations.sql vytvárajúci vzťahy medzi tabuľkami.

Vytvorením týchto skriptov som dostal štruktúru databázu do stavu, kedy je databáza kompatibilná s namapovanými entitami a zároveň prechádzajú insert príkazy, ktorými si vývojári zapĺňajú databázu na svojom lokálnom prostredí. K tejto téme sa ešte vyjadrím v kapitole Budúci rozvoj.

4.7 Implementácia service tried a factories

Naimplementoval som jednotlivé service triedy spomínané v kapitole Návrh. Vytvoril som 9 service tried a zaregistroval ich v konfiguračnom súbore. V niektorých service triedach som naiplementoval aj ich funkcie, konkrétne viac než 30 funkcií. Pri niektorých triedach som len pridal kostry funkcií, ktoré bude potrebné doimplementovať.
Vytvoril som 6 factory tried s funkciou create. Táto funkcia vráti programátorovi entitu konkrétneho objektu s parametrami špecifikovanými pri volaní danej funkcie.

4.8 Testovanie

Informácie k testovaniu som primárne čerpal z [41]. Testovanie je rozdelené do 2 kategórií, statické a dynamické. Oba typy testov sú v rámci vývoja newDBS portálu zautomatizované, primárne vďaka práci Pavla Kováře, detailnejšie popísané v jeho bakalárskej práci [42].

4.8.1 Statické testovanie

Statické testovanie pomáha nájsť chyby v kóde bez toho, aby sme kód museli spustiť. V rámci tohto testovania sa vyhľadajú najčastejšie chyby typu:

- Nevalidný syntax
- Nedosiahnuteľný alebo nepoužívaný kód
- Nedodržanie stanovených konvencií pre tvorbu kódu
- Používanie neexistujúcich či neprístupných premenných, metód a funkcií
- Používanie premenných s nedefinovanou hodnotou
- Predávanie premenných nesprávného typu [41]

Statickému testovaniu som sa venoval najmä v čase pred aktivitou Nasadenie modulu tímom[xxviii].
Mojou najčastejšou chybou bolo nedodržanie stanovených konvencií pri tvorbe kódu. Patrili sem:

- chýbajúce medzery okolo **if** statementu,
- chýbajúcí nový riadok na konci súboru,
- prebytočné prázdne riadky v hlavičke triedy a na jej konci.

Všetko to sú chyby zhoršujúce prehľadnosť kódu.
Ďalšou kategóriou, v ktorej som mal chyby bolo predávanie premenných nesprávneho typu. Týkalo sa to najmä funkcií definovaných v service a factory triedach.
Využíval som spomínané zautomatizované testy pre newDBS projekt. Najprv som ich spúšťal pri nahrávaní nových verzií kódu na GitLab. Neskôr som ich spúšťal na mojom zariadení. Všetky chyby s výnimkou jednej, ktoré boli zistené počas statického testovania, sa mi podarilo opraviť. Dôkazom tohto tvrdenia sú výpisy z pipelines na GitLabe.

[xxviii] táto aktivita je bližšie popísaná v nasledujúcej kapitole

4.8.1.1 UserRepository vs ObjectRepository

V rámci statického testovania bol ťažšie opraviteľným errorom v triede EntityManager nasledujúci error

```
Method EntityManager::getUserRepository() should return
Repository\General\UserRepository but returns
Doctrine\Common\Persistence\ObjectRepository.
```

V spomínanej getUserRepository funkcii je použitá metóda **getRepository** triedy Nettrine\EntityManagerDecorator ktorá podľa dokumentácie vracia **ObjectRepository**, avšak v anotácii je nastavené, že vracia UserRepository. Funkcia v skutočnosti vráti instanciu objektu, ktorá bude mať dostupné všetky metódy triedy UserRepository. Je teda vhodnejšie v anotácii nastaviť ako návratový typ UserRepository, po zavolaní tejto funkcie IDE[xxix] bude vnímať vytvorený objekt ako UserRepository a programátorovi bude našepkávať funkcie, ktoré preň sú zadefinované.
Error som sa pokúsil opraviť pomocou zdrojového kódu vzorového projektu Nettrine s názvom FoREST Project[43]. Tento pokus nebol úspešný, napriek tomu že som anotácie v častiach s chybou nastavil podľa vzorového projektu, error pretrvával. K riešeniu tejto chyby nakoniec stačilo použitie kľúčového slova **isInstanceOf**. Na toto riešenie prišiel člen tímu s ktorým som spolupracoval[xxx].

4.8.2 Dynamické testovanie

Na rozdiel od statického testovania sa snažíme overiť správanie aplikácie alebo aspoň jej časti na spustnom zdrojovom kóde. Nie je vhodné testovať každú časť aplikácie rovnakým spôsobom, dokonca to niekedy ani nie je možné. Preto delíme dynamické testy do určitých skupín podľa zvolenej techniky. [41]
Pri implementácií testov som okrem práce Pavla Kováře použil aj zdrojový kód modulu vytvoreného Andriim[44].

4.8.2.1 Jednotkové testy

Nazývané aj unit testy. V týchto testoch sa snažím overiť správanie vymedzenej časti zdrojového kódu v izolovanom prostredí. Nezávislosť na okolitom prostredí často nie je jednoduché dosiahnuť, a preto využijem špecializované nástroje. [41]
Pretože sa nechcem spoliehať na správnosť implementácie okolitých tried, nahradím ich dvojníkom, ktorého správanie si dopredu zadefinujem. Tento proces sa nazýva odstienenie závislosti.
Ja som pri prísaní testov k odstieneniu závislosti použil nástroj Phony[45],

[xxix] program v ktorom programátor píše kód
[xxx] detailnejšeie v nasledujúcej kapitole

ktorý je používaný a odporúčaný v newDBS projekte[46].
Vytvoril som 3 unit test triedy, pre Query, Title a IterationService.

4.8.2.2 Integračné testy

Cieľom týchto testov je otestovať vzájomnú interakciu komponent či spoluprácu jednotlivých tried. V týchto testoch už nepracujem v izolovanom prostredí, teda beriem ohľad taktiež na okolité prostredie. [41]
V mojom prípade je okolitým prostredím testovacia databáza, na ktorej testujem repository triedy. Naimplementoval som triedu AbstractEManager, ktorá aktuálne vytvára entitu triedy EntityManager na testovacej databáze. Táto trieda je rodičom integračných testov, ktoré som naimplementoval.

4.9 Zhrnutie

Na začiatku implementácie som sa venoval pridaniu nového modulu a integrácii frameworku Doctrine. Vytvoril som triedy pre enumy. Pre jednotlivé databázové tabuľky som vytvoril ich entity triedy. Následné som ich s použitím Doctrine funkcionality namapoval na príslušné databázové tabuľky. Pre entity som vytvoril repository triedy. Vytvoril som migrácie upravujúce databázu do stavu kompatibilného s novým návrhom. Naimplementoval som časť navrhnutých service tried a factory tried. Celý vytvorený kód som otestoval statickými testami. Pre 2 entity a jednu service triedu som vytvoril unit testy. Naimplementoval som triedu vytvarajúcu entitu EntityManager triedy na testovacej databáze. Táto trieda je použiteľná k vytváraniu integračných testov repository tried na testovacej databáze. Vytvoril som 2 integračné testy.
Na branch `https://gitlab.fit.cvut.cz/dbs/newDBS/tree/BW_newSW` som nahrával mnou implementovaný kód.

Kapitola 5

Budúci rozvoj

V tejto kapitole spomeniem moju spoluprácu s SP1 tímom a vyjadrím sa k ďalšiemu rozvoju modulu.

5.1 Spolupráca s SP1 tímom

Návrh a implementácia, ktoré som previedol, sú len malou časťou práce, ktorú je potrebné vykonať, kým nový modul semestrálnej práce bude používaný k výučbe. Ing. Hunka si bol toho vedomý a už na začiatku našej vzájomnej komunikácie mi navrhol osloviť študentov v rámci predmetu SP1, ktorý vyučuje. Myšlienkou spolupráce bolo začleniť týchto študentov do vývoja nového modulu. V spomínanom semestri boli študenti SP1 patriaci pod projekt newDBS rozdelení do 3 tímov s priemerným zastúpením 6 členov na tím. Jeho návrh som zvážil a rozhodol som sa vyskúšať to.

5.1.1 Začiatok spolupráce

Dňa 3.3.2021 som sa zúčastnil na hovore vývojárov newDBS projektu, kde som odprezentoval základný koncept mojej bakalárskej práce a navrhol im vzájomnú spoluprácu. Pár minút po hovore ma kontaktoval vedúci tímu 2 Matěj Frnka so žiadosťou o bližšie priblíženie spomínanej spolupráce. Dohodli sme sa na hovore 5.3. a vytýčili si veci, ktoré ich najviac zaujímajú.
Pripravil som si prezentáciu skladajúcu sa z pár hlavných bodov:

priblíženie BP: obšírnejšie som popísal moju bakalársku prácu, spomenul som technológie ktoré plánujem použiť, plánovaný postup a aktuálny stav.

čím by bol projekt pre Vás prínosom? medzi hlavné výhody pre študentov som zaradil prácu na začínajúcom projekte, kde môžu vytvárať nové

veci namiesto opravovania nefunkčného kódu. Ďalšou výhodou je možnosť pokračovať v práci na tomto module v rámci predmetu SP2 a prípadne ich bakalárskych prác.

čím by bola spolupráca s Vami prínosom pre mňa? práca, ktorej som venoval veľa času nebude odložená ale bude ďalej vyvíjaná. Vedúci práce snáď odcení môj záujem a aktivitu o ďalší rozvoj modulu.

ako si predstavujem spoluprácu? v tejto časti som spomenul aktivity, na ktorých môžeme spolupracovať a víziu komunikácie a zodpovednosti.

Dňa 5.3. sme sa stretli na Discord serveri, ktorý spomínaný tím používa na schôdze a ja som podľa spomínanej osnovy odprezentoval moju bakalársku prácu a víziu spolupráce. Zodpovedal som otázky, ktoré na mňa mali. Uistil som ich, že spolupráca nie je záväzná, teda ak jednotlivec alebo celý tím zistí, že nemá záujem pracovať na module, môžeme ju ukončiť. Po hovore sa medzi sebou poradili a oznámili mi, že o spoluprácu majú záujem. Ďalej budem týchto študentov volať tím.

5.1.2 Spoznanie modulu

Prišlo mi vhodné aby sa tím zoznámil s modulom, na ktorom bude pracovať. Dohodli sme sa, že prvou aktivitou, na ktorej budeme spolupracovať, budú Use Case modely modulu semestrálna práca. Use Case je vyučovaný na predmete SI1, ktorého hodnotenie je spojené s predmetom SP1. Išlo teda o obojstranne výhodnú aktivitu. Tímu som ukázal Use Case modely a diagramy, ktoré som vypracoval. Prezreli si modul semestrálna práca a dali mi spätnú väzbu a navrhli, čo by bolo vhodné zmeniť. Pozmenil som znenie pár viet ako aj typ väzby v 3 diagramoch. Tím sa lepšie zoznámil s funkcionalitou modulu.

5.1.3 Výber REST API technológie

Olda je senior developer a manažér newDBS projektu, preto som ho po rozhovore s Ing. Hunkom oslovil otázkou o mojej spolupráci s tímom na ďalšom rozvoji modulu. V komunikácii mi oznámil, že backend by mal poskytovať REST API. Túto požiadavku som dňa 19.3. prezentoval tímu na spoločnej schôdzi, kde som im takisto priblížil, ako som aktuálne postúpil v rámci implementácie. Dohodli sme sa, že vypracujú dokument popisujúci použitie REST API v rámci projektu newDBS a navrhnú technológie vhodné na implementáciu tohto požiadavku. Výsledok ich práce mi poslal Karel Zanáška. Preštudoval som si dokument a zhodli sme sa použiť technológiu **ublaboo/api-router**. Neskôr tím zistil, že funkcionalita tejto technológie je nedostačujúca a rozhodli sa použiť druhú možnosť spomínanú v dokumente, technológiu **apitte/core**.

5.1.4 Nasadenie modulu tímom

Dňa 26.3. mi vedúci tímu Matěj oznámil, že chcú začať pracovať na module. Dohodli sme sa, že pripravím tutoriál ako si modul na svojom počíťači spustiť, aby mohli programovať. Vytvoril som návod doctrine.pdf, ktorý sa skladá z častí:

Pridanie PHP apcu extensionu do vagrantu: približuje postup pridania, obsahuje príkazy ktoré je potrebné zadať do konzoly. Takisto obsahuje podsekciu známe errory, kde je priblížený zatiaľ jediný známy error, s ktorým sa stretol Pavel Jordán a jeho postup vyriešenia daného erroru.

Pridanie orm príkazov do vagrant ssh konzoly: popisuje postup ktorý mi poradil Martin Hanzl.

Ako získať nové databázové tabuľky pre nový prototyp: ukazuje ako previesť databázové migrácie.

Okrem návodu som v HomepagePresenter nachystal kód, ktorý využíval Doctrine a v prípade funkčnosti vypísal na domovskej stránke modulu newSW dáta. Dňa 27.3. som tímu tento návod nazdieľal. V ten istý deň som mal s Matějom hovor, kde som mu predstavil návrh systému a novej databázy. Po hovore si vyskúšal podľa návodu spustiť Doctrine a previesť databázovú migráciu. Podarilo sa mu to a nestretol sa so žiadnymi problémami. V priebehu nasledujúceho týždňa si celý tím úspešne nasadil modul. Vedúci tímu na GitLabe vytvoril nový tímový branch, kde tím verzuje svoju prácu.

5.1.5 Práca na module

Po nasadení modulu chcel tím začať s implementáciou. Mali nejaké otázky ohľadom modulu a Doctrine, tak ma oslovili. Dohodli sme sa, že 2.4. im odprezentujem môj návrh systému a ako používam Doctrine. V spomínaný deň sme sa opäť spojili cez Discord. Podľa kapitoly semestrálnej práce návrh som im odprezentoval ako vidím databázovú a logickú vrstvu. Ukázal som im, v čom vidím výhodu novej databázovej vrstvy a ako si predstavujem prácu s databázovou vrstvou. Demonštroval som aj prácu s Doctrine entitami a repositories. Po prezentácii som zodpovedal pár otázok a poslal som im dokument popisujúci návrh novej databázy a návrh jednotlivých častí systému.

5.1.6 Príručka pre nového vývojára

Vytvoril som dokument, ktorý popisuje základy backendu modulu. Obsah dokumentu som predstavil tímu. Tento dokument bude slúžiť novým vývojárom projektu newDBS pracaujúcim na module k jednoduchšiemu zoznámeniu s používanými technológiami a rozdelením kódu. Nachádza sa v prílohách pod názvom prirucka.pdf.

5.1.7 Výsledky práce na module

Pred odovzdávaním práce som oslovil vedúceho tímu Matěja, aby mi spísal, čo sa im ako tímu podarilo počas práce na novom module docieliť. Citujem jeho odpoveď:
„*Mezi nejvýznamnější věci kterých jsme docílili patří:*
1) Rozdělili jsme projekt na API v Apitte/Core a frontend v Vue. Hlavní výhodou je, že lze frontend psát v moderním frameworku vue.js které funguje jako single-page application.
2) Začali jsme využívat novou technologii doctrine která ulehčuje práci programátorům při psaní jednoduchých dotazů na databázi.
3) Uzamkli jsme všechny verze závislostí na backendu a smazali složku vendor z gitlab repozitáře. Před uzamčením nebylo možné aktualizovat konkrétní závislost protože se tím aktualizovali i všechny ostatní závislosti na verze které už nefungovali s dbs portálem. Díky uzamknutí verzí lze aktualizovat konkrétní závislosti.
4) Přepsali jsme knihovnu na zobrazování dat v tabulce pro frontend která blokovala upgrade z vue2 na vue3.“[47]
Z jeho odpovede je jasné, že rozvoj modulu je na správnej ceste.
`https://gitlab.fit.cvut.cz/dbs/newDBS/tree/team-sp-2021-2` je ich aktuálnou tímovou branchou, ktorá je založená z mojej branche.

5.2 Čo je potrebné vykonať pred nasadením

5.2.1 Service triedy a Checker

Ďalším krokom je implementácia nedokončených service tried, ako aj tried zaoberajúcich sa kontrolou požiadaviek na iteráciu. Časť implementácie triedy kontrolujúcej splnenie požiadaviek je možné prebrať z aktuálneho riešenia. Pozor na zmenu pri tabuľke ukladajúcej jednotlivé typy požiadaviek.

5.2.2 Testy

Pred nasadením bude potrebné vytvoriť ďalšie dynamickě testy pre modul. Verím, že testy ktoré som vypracoval, trieda AbstractEManager a aj dynamické testy aktuálneho riešenia budú dobrým odrazovým mostíkom pre túto aktivitu.

5.2.3 Frontend

K backendu bude potrebný vhodný frontend. Na tejto časti už pracuje spomínaný tím, prevzali prototyp nového frontendu modulu semestrálna práca, ktorý bol vyvíjaný počas minulých cyklov predmetu SP.

5.2.4 Úprava importu

Pred nasadením modulu bude potrebné upraviť kód, ktorý je zodpovedný za import dát z z KOSu tak, aby odpovedal novej štruktúre databázy. Ide hlavne o tabuľky spojené s iteráciami a požiadavkami.

5.2.5 Zachovanie semestrálných prác

V návrhu som spomenul, že štruktúra databázy ukladajúca semestrálnu prácu je rovnaká ako v aktuálnom riešení. Vďaka tomu bude náhľad starých[xxxi] semestrálných prác schopný používať rovnaký frontend, ako nové semestrálne práce. Tento fakt bude potrebné pred nasadením modulu otestovať.
S odovzdávaním je to inak. Prvý rozdiel je v komentároch, aktuálna verzia má špeciálnu tabuľku pre komentár ku každej časti odovzdania, môj návrh používa jednu tabuľku pre všetky typy komentárov. Ďalší rozdiel je v kontrole splnenia požiadaviek, aktuálna verzia spúšťa kontrolu pri každom načítaní stránky, môj návrh prevádza kontrolu raz a ukladá výsledok do databázy. Rozdiel je takisto v databázových tabuľkaćh pre deadliny iterácií. Napadlo ma pár riešení tohto problému:

1. Ponechať pre staré odovzdania starý frontend.
2. Nepovoliť absolventom zobrazovať staré odovzdania.
3. Pre všetky staré odovzdania prepísať dáta zo starých tabuliek do odpovedajúcich nových tabuliek a spustiť kontrolu starých odovzdaní tak, aby pre všetky boli dostupné informácie o splnení alebo nesplnení požiadaviek.

5.3 Aký smer rozvoja navrhujem

5.3.1 DEMO semestrálna práca

DEMO semestrálna práca je časť portálu, ktorá obsahuje veľa priestoru pre novú funkcionalitu. V mojej práci som počítal len s prezeraním DEMO semestrálnej práce študentmi. Ďalším krokom by mohlo byť spúštanie predvytvorených dotazov na databáze DEMO semestrálnej práce študentmi, prípadne umožniť študentovi naklonovať si DEMO prácu a vytvárať si na nej vlastné dotazy, ktoré môže následne testovať.

5.3.2 Prechod celého portálu na Doctrine

Vhodným krokom vpred by bol prechod celého portálu na Doctrine. Najlepším termínom prevedenia tejto aktivity by bol čas pred nasadením nového modulu

[xxxi] staré znamená vytvorené pred nasadením nového modulu

semestrálnej práce a nového modulu testov. Existujú entity, ktoré sú v oboch moduloch, preto bude pri tomto kroku potrebné zaistiť, aby neboli namapované duplicitne. Navyše bude potrebné pridať mapovanie entít, ktoré neboli zahrnuté v žiadnom module. Prevedenie tejto aktivity by uľahčilo údržbu a vývoj portálu, avšak bude časovo náročné.

5.4 Ja a ďalší vývoj modulu

S tímom som sa dohodol, že aj po ukončení implementácie a písania bakalárskej práce im poradím pri otázkach ďalšieho vývoja modulu. Mám podanú prihlášku na magisterské štúdium na Fakulte informačných technológí ČVUT. Ak budem prijatý a v čase volenia témy diplomovej práce zostanú moje programátorské záujmy podobné, určite oslovím môjho aktuálneho vedúceho práce ohľadom ďalšieho vývoja modulu v rámci mojej diplomovej práce.

Záver

Zvolil som implementačnú technológiu, ktorá je kompatibilná s aktuálnym riešením ako aj novým prototypom modulu testy. Vytvoril som dôkladnú analýzu funkcionality aktuálneho riešenia. Našiel som nedostatky aktuálneho riešenia a popísal nové funkčné a nefunkčné požiadavky.
Z aktuálneho riešenia som prebral jeden vymenovaný typ a navrhol som druhý pre zjednodušenie práce. Na základe analýzy som vytvoril návrh novej databázy. Časť tabuliek som ponechal z aktuálneho riešenia, časť som nahradil z môjho pohľadu vhodnejšími tabuľkami a pridal tabuľku pre novú funkcionalitu. Následne som podľa analýzy vytvoril návrh systému rozdelený na dátovú a logickú vrstvu. Pri návrhu som bral ohľad na zachovanie starých semestrálnych prác a previazanosť s ostatnými časťami portálu. Počas návrhu a čiastočne aj implementácie som sa snažil brať ohľad na existujúci prototyp modulu testy.
Počas implementácie som namapoval jednotlivé databázové tabuľky na triedy. Pre väčšinu entít som vytvoril ich repository triedy, v časti ktorých som naimplementoval funkcie na komunikáciu s databázou. Vytvoril som skripty pridávajúce navrhnuté tabuľky do aktuálnej databázy. Naimplementoval som časť navrhnutých service tried ako aj pár factory tried. Počas vývoja som používal automatické statické testy projektu newDBS a testoval nimi vytvorený kód. Vytvoril som unit testy entity tried a service tried. Naimplementoval som triedu, ktorá umožňuje otestovať repository triedy na testovacej databáze projektu a následne s jej využitím vytvoril integračné testy.
Vyjadril som sa k budúcemu rozvoju modulu aj portálu, ako z pohľadu vývojárov portálu tak aj z môjho pohľadu. Taktiež som spomenul aktivity, ktoré bude nevyhnutné pred nasadením modulu vykonať.
Cieľami práce bolo zvoliť vhodnú implementačnú technológiu, vytvoriť analýzu aktuálneho riešenia a rozšírení, navrhnúť nové riešenia, časť navrhnutého riešenie naimplementovať a otestovať, navrhnúť budúci rozvoj modulu. Všetky tieto ciele som splnil. Naviac som komunikoval a spolupracoval s jedným zo študentských tímov vyvíjajúcich DBS portál. Priblížil som im moju prácu,

aktuálny modul a víziu nového modulu. Pomohol som im s nasadením mnou vyvíjaného modulu. Odprezentoval som im všetky, z môjho pohľadu dôležité, časti návrhu a implementácie. Poradil som im v otázkach vývoja, ktoré nadväzovali na moju implementáciu. Taktiež som vypracoval príručku pre nového vývojára modulu približujúcu najmä backend modulu.

Literatúra

[1] PLYSKACH, Andrii. *Refaktoring testové části backendu portálu dbs.fit.cvut.cz* [online]. Praha, 2020 [cit. 2021-03-11]. Dostupné z: https://dspace.cvut.cz/handle/10467/88282?show=full. Bakalárska práca. České vysoké učení technické v Praze. Vedoucí práce Ing. Jiří Hunka.

[2] *Bíla kniha FIT ČVUT: Popis předmětu BI-DBS, Anotace* [online]. Fakulta informačných technológií ČVUT. [cit. 2021–03–02]. Dostupné z: http://bk.fit.cvut.cz/cz/predmety/00/00/00/00/00/00/01/12/24/p1122406.html

[3] KUBIŠ, Martin. *Rozšíření překladače relační algebry* [online]. Praha, 2019 [cit. 2021-03-22]. Dostupné z: https://dspace.cvut.cz/bitstream/handle/10467/82310/F8-DP-2019-Kubis-Martin-thesis.pdf?sequence=-1&isAllowed=y. Diplomová práca. České vysoké učení technické v Praze. Vedoucí práce Ing. Jiří Hunka.

[4] MACHALA, Filip. *Automatická oprava zjednodušeného relačního zápisu* [online]. Praha, 2018 [cit. 2021-03-22]. Dostupné z: https://dspace.cvut.cz/bitstream/handle/10467/76661/F8-BP-2018-Machala-Filip-thesis.pdf?sequence=-1&isAllowed=y. Bakalárska práca. České vysoké učení technické v Praze. Vedoucí práce Ing. Jiří Hunka.

[5] *Courses BI-DBS B182: Hodnocení* [online]. Fakulta informačných technológií ČVUT. [cit. 2021-03-22]. Dostupné z: https://courses.fit.cvut.cz/BI-DBS/@B182/classification/index.html

[6] *Courses BI-DBS: Hodnocení* [online]. Fakulta informačných technológií ČVUT. [cit. 2021-03-10]. Dostupné z: https://courses.fit.cvut.cz/BI-DBS/classification/index.html

[7] *Courses BI-DBS: Semestrální práce* [online]. Fakulta informačných technológií ČVUT. [cit. 2021-03-10]. Dostupné z: https://courses.fit.cvut.cz/BI-DBS/project/index.html

[8] *Courses BI-DBS: Create script* [online]. Fakulta informačných technológií ČVUT. [cit. 2021-03-10]. Dostupné z: https://courses.fit.cvut.cz/BI-DBS/project/index.html#_create-script

[9] *Courses BI-DBS: Insert script* [online]. Fakulta informačných technológií ČVUT. [cit. 2021-03-10]. Dostupné z: https://courses.fit.cvut.cz/BI-DBS/project/index.html#_insert-script

[10] *LearnToCodeWithMe: Backend Development* [online]. [cit. 2021-03-10]. Dostupné z: https://learntocodewith.me/posts/backend-development/

[11] What is open source? *Opensource.com* [online]. [cit. 2021-5-12]. Dostupné z: https://opensource.com/resources/what-open-source

[12] Klient-server. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2021-03-10]. Dostupné z: https://sk.wikipedia.org/wiki/Klient-server

[13] PHP (skriptovací jazyk). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2021-03-10]. Dostupné z: https://sk.wikipedia.org/wiki/PHP_(skriptovac%C3%AD_jazyk)

[14] *W3Techs: Usage statistics of server-side programming languages for websites* [online]. [cit. 2021-03-11]. Dostupné z: https://w3techs.com/technologies/overview/programming_language

[15] Knižnica (programovanie). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2021-03-11]. Dostupné z: https://sk.wikipedia.org/wiki/Kni%C5%BEnica_(programovanie)

[16] *ITnetwork.sk: Kurzy programovania, 1. diel - Úvod do knižníc a frameworkov v PHP* [online]. [cit. 2021-03-10]. Dostupné z: https://www.itnetwork.sk/php/kniznice/php-tutorial-uvod-do-kniznic-a-frameworku

[17] *Blog Bart: Po lopate: Čo je to MVC* [online]. [cit. 2021-03-11]. Dostupné z: https://blog.bart.sk/co-je-mvc-model-view-controller-po-lopate/

[18] Template processor. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2021-03-11]. Dostupné z: https://en.wikipedia.org/wiki/Template_processor

[19] Objektově relační mapování. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2021-03-11]. Dostupné z: https://cs.wikipedia.org/wiki/Objektov%C4%9B_rela%C4%8Dn%C3%AD_mapov%C3%A1n%C3%AD

[20] *Builtwith: Framework Usage Distribution in the Top 1 Million Sites* [online]. [cit. 2021-03-10]. Dostupné z: https://trends.builtwith.com/framework

[21] *Wikipedia: Pages in category "PHP frameworks"* [online]. [cit. 2021-03-10]. Dostupné z: https://en.wikipedia.org/wiki/Category:PHP_frameworks

[22] SURGUY, Maks. History of Laravel PHP framework, Eloquence emerging. *MAKSIM SURGUY PERSONAL WEBSITE AND BLOG* [online]. 27. 07. 2013 [cit. 2021-5-12]. Dostupné z: https://maxoffsky.com/code-blog/history-of-laravel-php-framework-eloquence-emerging/

[23] *GitHub: Laravel releases* [online]. [cit. 2021-03-11]. Dostupné z: https://github.com/laravel/framework/releases/

[24] SQL injection. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2021-03-11]. Dostupné z: https://sk.wikipedia.org/wiki/SQL_injection

[25] *Kinsta: The Most Popular PHP Frameworks to Use in 2021* [online]. [cit. 2021-03-11]. Dostupné z: https://kinsta.com/blog/php-frameworks/

[26] Nette Framework. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2021-03-11]. Dostupné z: https://cs.wikipedia.org/wiki/Nette_Framework

[27] *GitHub: Nette releases* [online]. [cit. 2021-03-11]. Dostupné z: https://github.com/nette/nette/releases

[28] *Builtwith: Nette Framework Usage Statistics* [online]. [cit. 2021-03-11]. Dostupné z: https://trends.builtwith.com/framework/Nette-Framework

[29] *Nette.org: Why Use Nette?* [online]. [cit. 2021-03-11]. Dostupné z: https://doc.nette.org/en/3.1/why-use-nette

[30] Doctrine (PHP). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2021-03-11]. Dostupné z: https://cs.wikipedia.org/wiki/Doctrine_(PHP)

[31] *Stackshare: Doctrine 2* [online]. [cit. 2021-03-11]. Dostupné z: https://stackshare.io/doctrine2

[32] ING. MLEJNEK, Jiří. *UML diagram případů užití* [online]. Fakulta informačných technológií ČVUT. 2019 [cit. 2021-04-06]. Dostupné z: https://moodle-vyuka.cvut.cz/pluginfile.php/388199/mod_resource/content/1/Use%20Case%20Model.pdf

[33] ING. MLEJNEK, Jiří. *Analýza a sběr požadavků* [online]. Fakulta informačných technológií ČVUT. 2020, 18 [cit. 2021-04-06]. Dostupné z: https://moodle-vyuka.cvut.cz/pluginfile.php/388197/mod_resource/content/4/03.prednaska.pdf

[34] ING. MLEJNEK, Jiří. *Analýza a sběr požadavků* [online]. Fakulta informačných technológií ČVUT. 2020, 3–4 [cit. 2021-04-06]. Dostupné z: https://moodle-vyuka.cvut.cz/pluginfile.php/388197/mod_resource/content/4/03.prednaska.pdf

[35] Výčtový typ. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2021-03-22]. Dostupné z: https://cs.wikipedia.org/wiki/V%C3%BD%C4%8Dtov%C3%BD_typ

[36] *Composer: Introduction* [online]. [cit. 2021-04-03]. Dostupné z: https://getcomposer.org/doc/00-intro.md

[37] *Doctrine: Tools* [online]. [cit. 2021-04-03]. Dostupné z: https://www.doctrine-project.org/projects/doctrine-orm/en/2.8/reference/tools.html

[38] *Doctrine 2 ORM´s documentation* [online]. [cit. 2021-04-09]. Dostupné z: https://www.doctrine-project.org/projects/doctrine-orm/en/2.8/index.html

[39] *Doctrine: Inheritance Mapping* [online]. [cit. 2021-04-05]. Dostupné z: https://www.doctrine-project.org/projects/doctrine-orm/en/2.8/reference/inheritance-mapping.html

[40] *NewDBS Redmine Wiki: Databázové migrace* [online]. [cit. 2021-04-05]. Dostupné z: https://dbs.fit.cvut.cz/redmine/projects/new-dbs/wiki/Datab%C3%A1zov%C3%A9_migrace

[41] KOVÁŘ, Pavel a Andrii PLYSKACH. Testování: Základní informace. *NewDBS redmine: Wiki* [online]. [cit. 2021-04-13]. Dostupné z: https://dbs.fit.cvut.cz/redmine/projects/new-dbs/wiki/Z%C3%A1kladn%C3%AD_informace

[42] PAVEL, Pavel. *Automatizované testování webového portálu dbs.fit.cvut.cz* [online]. Praha, 2017 [cit. 2021-04-13]. Dostupné z: https://dspace.cvut.cz/bitstream/handle/10467/69954/F8-BP-2017-Kovar-Pavel-thesis.pdf?sequence=1&isAllowed=y. Bakalárska

práca. České vysoké učení technické v Praze. Vedoucí práce Ing. Jiří Hunka.

[43] *Forest Project* [online]. [cit. 2021-04-01]. Dostupné z: https://github.com/contributte/apitte-skeleton

[44] NewDBS: branch bw_new_tests. *GitLab fit cvut* [online]. [cit. 2021-04-13]. Dostupné z: https://gitlab.fit.cvut.cz/dbs/newDBS/tree/bw_new_tests

[45] *Phony documentation* [online]. [cit. 2021-04-14]. Dostupné z: https://eloquent-software.com/phony/latest/

[46] KOVÁŘ, Pavel. Odstínění závislostí. *NewDBS redmine: Wiki* [online]. [cit. 2021-04-14]. Dostupné z: https://dbs.fit.cvut.cz/redmine/projects/new-dbs/wiki/Odst%C3%ADn%C4%9Bn%C3%AD_z%C3%A1vislost%C3%AD

[47] Téma: *Dosiahnuté veci na novom module* Písomná komunikácia s Matejom FRNKOM, vedúci SP1 tímu. 11. 05. 2021.

Dodatok A

Zoznam použitých skratiek

csv comma-separated values

ČVUT České vysoké učení technické v Praze

DBS predmet „Databázové systémy“ vyučovaný na ČVUT

LS Letný semester

RA Relačná Algebra

SI1 predmet „Softwarové inženýrství I“ vyučovaný na ČVUT

SP1 predmet „Softwarový týmový projekt 1“ vyučovaný na ČVUT

SP2 predmet „Softwarový týmový projekt 2“ vyučovaný na ČVUT

SQL Structured Query Language

UC Use Case

URL Uniform Resource Locator

Dodatok **B**

Obsah priloženého média

- `readme.txt` ..... stručný popis obsahu CD
- `documents`
  - `doctrine.pdf` ..... návod na nasadenie doctrine a nového modulu
  - `prirucka.pdf` ..... príručka pre nového programátora modulu
- `images` ..... obrázky a diagramy použité v práci
- `src`
  - `thesis` ..... zdrojový kód práce vo formáte LaTeX
- `text`
  - `thesis.pdf` ..... text práce vo formáte PDF
- `usecase`
  - `UC.pdf` ..... podrobné UC z analyzačnej časti
  - `UC14.pdf` ..... detailný UC14
  - `UC16.pdf` ..... detailný UC16
  - `UC18.pdf` ..... detailný UC18