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

Bakalářská práce

Softwarová podpora organizace předmětů TSP

Plzeň 2022 Hinterholzinger Jan

Místo této strany bude zadání práce.

Prohlášení

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

V Plzni dne 19. dubna 2022

Hinterholzinger Jan

Abstract

The text of the abstract (in English). It contains the English translation of the thesis title and a short description of the thesis.

Abstrakt

Text abstraktu (česky). Obsahuje krátkou anotaci (cca 10 řádek) v češtině. Budete ji potřebovat i při vyplňování údajů o bakalářské práci ve STAGu. Český i anglický abstrakt by měly být na stejné stránce a měly by si obsahem co možná nejvíce odpovídat (samozřejmě není možný doslovný překlad!).

Obsah

1 Úvod **7**
- 1.1 Motivace . . . 7
- 1.2 Cíl práce . . . 7

2 Specifikace požadavků **8**
- 2.1 Organizace předmětů TSP . . . 8
- 2.2 Uživatelé . . . 9
  - 2.2.1 Nepřihlášený uživatel . . . 9
  - 2.2.2 Vedoucí týmu . . . 9
  - 2.2.3 Mentor . . . 9
  - 2.2.4 Garant . . . 9

3 Dostupné technologie **10**
- 3.1 PHP . . . 10
- 3.2 PHP frameworky . . . 10
  - 3.2.1 Laravel . . . 10
  - 3.2.2 Symfony . . . 11
  - 3.2.3 Nette . . . 11
- 3.3 Front-end . . . 11
  - 3.3.1 HTML a CSS . . . 11
  - 3.3.2 Bootstrap . . . 12
  - 3.3.3 AdminLTE . . . 12
- 3.4 Databáze MariaDB . . . 12
- 3.5 Nástroje pro řízení projektu . . . 13
  - 3.5.1 Verzovací systém . . . 13
  - 3.5.2 Plánování úkolů . . . 14
  - 3.5.3 Issues . . . 14
  - 3.5.4 Kanban . . . 15
  - 3.5.5 MantisBT . . . 16

4 Návrh aplikace **17**
- 4.1 Dekompozice specifikace . . . 17
  - 4.1.1 Zadání . . . 17
  - 4.1.2 Tým . . . 17
  - 4.1.3 Projekt . . . 18
- 4.2 Případy užití . . . 18

4.2.1 Nepřihlášený uživatel . . . 18
4.2.2 Přihlášený uživatel . . . 18
4.2.3 Vedoucí týmu . . . 19
4.2.4 Mentor . . . 19
4.2.5 Garant . . . 21
4.3 Architektura aplikace . . . 23
4.3.1 Třívrstvá architektura . . . 23
4.3.2 Komponenty aplikace v Nette . . . 24
4.3.3 Dependency Injection v Nette . . . 25
4.3.4 Formuláře v Nette . . . 25
4.3.5 Database Explorer . . . 26
4.3.6 Testovatelnost aplikace . . . 26
4.3.7 Responzivita . . . 27
4.4 Databázová struktura . . . 27
4.4.1 Základní tabulky . . . 29
4.4.2 Číselníkové tabulky . . . 30
4.4.3 Spojovací tabulky . . . 30
4.4.4 Další tabulky . . . 31
4.5 Uživatelské rozhraní . . . 31
4.6 Návrh loga . . . 31

5 Realizace **32**
5.1 Adresářová struktura . . . 32
5.2 Konfigurace aplikace . . . 32
5.2.1 Konfigurace databázového spojení . . . 32
5.2.2 Vlastní nastavení aplikace . . . 33
5.3 Vylepšování aplikace . . . 33
5.3.1 Oprava nalezených defektů . . . 33
5.3.2 Úpravy dle aktualizování specifikace . . . 33
5.3.3 Připomínky budoucích uživatelů . . . 33
5.4 Požadavky na zavedení aplikace . . . 33

6 Testování **34**

7 Závěr **35**

Literatura **37**

1 Úvod

1.1 Motivace

Na Katedře informatiky a výpočetní techniky vzniká nový předmět „Týmový softwarový projekt“ (KIV/TSP1 a KIV/TSP2) určený pro studenty navazujícího studia. Podstatou předmětu je vypracování zadaného tématu ve skupinkách studentů, kdy studenti přijdou do styku s týmovou prací, řízením projektu a dalšími interními procesy.

Zvláštností předmětu je doba pro řešení projektu, dva semestry rozdělené do dvou předmětů. Budou zapojeny čtyři skupiny lidí. Garant, zadavatelé, mentoři a studenti, což bude klást poměrně značné nároky na organizaci předmětu. Pro takový rozsah bylo rozhodnuto o vytvoření webové aplikace, která všem zúčastněným stranám zjednoduší prohlížení obsahu a evidenci postupu prací projektu. Cílem aplikace je umožnit lepší informovanost a zapojení studentů, řešení evidence zadavatelů a možnosti sdílení mezi mentorem a garantem. Aplikace tak má nahradit způsob organizace pomocí Excel tabulky, jako je tak tomu u jiných obdobných předmětů.

1.2 Cíl práce

Cílem práce je vyvinout zmíněnou webovou aplikaci dle specifikace předmětu a požadavků jeho garanta. Předpokládá se, že aplikace bude v budoucnu nadále rozšiřována. Je proto žádoucí, aby aplikace byla dostatečně robustní a důkladně otestována.

2 Specifikace požadavků

Specifikace požadavků se řídí průběhem a organizací předmětů TSP. V době vývoje aplikace se díky aktivní přípravě těchto předmětů několikrát požadavky změnily a tedy muselo být počítáno s průběžnou aktualizací specifikace. Díky tomu je potřeba aplikaci psát s dostatečným odstupem a přiměřeně obecně, aby odolávala drobným změnám specifikace.

2.1 Organizace předmětů TSP

Výuka předmětů TSP je rozdělena do dvou semestrů, a tedy do dvou předmětů KIV/TSP1 vyučovaného v letním semestru 1. ročníku a KIV/TSP2 v zimním semestru 2. ročníku. Řešení týmového projektu tedy bude překračovat hranice ročníku a v mnoha částech se organizace předmětů bude překrývat.

Jednotlivý cyklus se v nejčastějším případě bude probíhat následovně.

- Garant přijímá od domluvených zadavatelů jejich témata, která mají studentské týmy řešit. Toto zadání společně s možným PDF souborem vloží do podpůrného softwaru pro zvolené období předmětu.
- Před začátkem TSP budoucí vedoucí týmu zažádá garanta o založení účtu v podpůrném softwaru s rolí „Vedoucí“. Vedoucí týmu společně s dalšími studenty sestaví tým a vloží informace o týmu do této podpory.
- Poté co jsou témata zadání projektů zveřejněny, vedoucí týmu (na základě rozhodnutí v týmu) projeví zájem o zvolené téma.
- Mentor ohodnotí vhodnost zvoleného tématu pro daný tým na základě zájmu, který vedoucí týmu učinil.
- Garant následně podle rozhodnutí mentora a zájmu týmu definitivně přiřadí nebo nepřiřadí téma týmu. Přiřazením tématu vzniká projekt.
- Tým řeší projekt a postup řešení vkládá do softwarové podpory
- Mentor na základě postupu řešení týmu kontroluje a potvrzuje postup týmu
- Na závěru předmětu TSP se koná obhajoba projektu, jejíž výsledek se zapíše do softwarové podpory.

2.2 Uživatelé

2.2.1 Nepřihlášený uživatel

Nepřihlášený uživatel je každý návštěvník aplikace, který se neprokáže svými přihlašovacími údaji. Takový uživatel může ve skutečnosti být student předmětu TSP, proto je mu umožněno zobrazovat vypsaná témata a týmy, které hledají nové členy.

2.2.2 Vedoucí týmu

Student předmětů TSP, který zažádal garanta o vytvoření konta. Tento uživatel je pověřen tvorbou týmu. Je mu umožněno vyjadřovat zájmy o volná zadání, které následně ohodnocuje mentor a na základě zájmu je garantem téma přiděleno. Také má možnost evidovat v projektu splnění kritérií, postupu a využitých zkušeností. Vedoucí vyplňuje údaje o týmu do softwarové podpory a eviduje postup projektu.

2.2.3 Mentor

Vyučující předmětů TSP, který má v softwarové podpoře založený účet- Jeho úkolem je evidovat postup a kontrolovat výstupy z týmu. Uživateli dále bude umožněno převzít si téma, které bude mentorovat a určovat vhodnost tématu dle vytvořených zájmů týmů o konkrétní téma, které mentoruje.

2.2.4 Garant

Garant je osoba, která koordinuje činnost všech ostatních skupin uživatelů. Mezi jeho hlavní náplní patří také přidávání nových témat, správa uživatelů a dalších nastavení softwarové podpory.

3 Dostupné technologie

3.1 PHP

[1]

3.2 PHP frameworky

Existuje mnoho úspěšných PHP frameworků, které mají různé typy zaměření. Mezi nejúspěšnější a nejpoužívanější patří Laravel, Symfony a Nette. Všechny tyto frameworky si zakládají na vytváření znovupoužitelných komponent a služeb. Kromě toho v sobě obsahují rozsáhlé nástroje pro usnadnění například práce s databází, přesměrování, ošetření bezpečnosti, atd. Díky tomu ulehčují vývojářům vlastní vývoj aplikace a nemusí tak vyvíjet úsilí pro tvorbu vlastních řešení. Výrazné zjednodušení představuje funkce dependency injection, která zajišťuje propojení mezi jednotlivými komponentami a službami a hlídá dostupnost všech závislostí.

Součástí těchto frameworků jsou také šablonovací enginy, které zjednodušují tvorbu front-endu. Tyto šablonovací systémy umožňují dědičnost jednotlivých pohledů, jejich členění na sekce a obecně jejich cílem je jednodušeji prezentovat data z back-endu. Jednotlivé šablonovací enginy se liší svými funkcemi a rozšířeními, ale typicky je vybíráme dle osobních preferencí nebo dle použitého back-end frameworku.

3.2.1 Laravel

Framework Laravel je možné považovat jako za ten nejrozšířenější. Mezi jeho hlavní východy patří jeho jednoduchost používání a rychlost. Pro svůj přístup k jednoduchému použití je Laravel doporučován jako vhodný pro začátečníky ale i pro profesionály. Framework se hodí pro vytváření méně komplexních projektů.

Laravel využívá šablonovací engine Blade, který je standardně dodáván společně se samotným frameworkem. Tento engine umožňuje oproti jiným rozšiřovat PHP kód, a tak provádět různé jednoduché operace pro přizpůsobení dat k samotnému front-endu.

3.2.2 Symfony

Tento framework se vyznačuje zakládáním si na striktním dodržování nejen PHP standardů a snaží se maximálně využívat různé návrhové vzory. Díky tomu jsou komponenty frameworku robustnější, což může znamenat větší časovou náročnost, ale také výraznou stabilitu frameworku, a proto je vhodný pro použití na komplexnějších projektech. Další předností mohou být rozsáhlé možnosti pro vývojáře, který si může prostředí přizpůsobit svým potřebám. To však vyžaduje hlubší znalosti jazyka PHP a struktury frameworku. Pro nováčky se tedy Symfony více náročný na naučení.

Jako výchozí šablonovací engine je využíván Twig, který se také řadí mezi nejpoužívanější šablonovací systémy. Oproti systému Blade obsahuje navíc další bezpečnostní vrstvu a další funkce. Twig je často využíván i samostatně, tedy bez použití back-end frameworku. To podtrhuje jeho flexibilitu.

3.2.3 Nette

S frameworkem společně přichází i šablonovací engine Latte.

3.3 Front-end

3.3.1 HTML a CSS

Základem webových aplikací je způsob zobrazování. Webové technologie nabízejí tvorbu elementů pomocí formátu HTML[1] založeného na XML[2]. Vlastností tohoto formátu je tvorba webových elementů pomocí tagů, kterým se specifikují jejich vlastnosti. Dnes již naprostá většina prohlížečů zobrazuje webové stránky pomocí tohoto formátu a stává prakticky synonymem pro internetové stránky [2].

Další obdobně oblíbenou technologií je stylovací kaskádový jazyk CSS[3]. Díky tomuto formátu je možné nastavovat atributy jednotlivých elementů a upravovat tak zejména jejich vzhled. Tyto styly lze aplikovat přímo do HTML tagu daného elementu nebo do blokového elementu společný pro celou stránku. Nejčastěji se však využívá importování přiloženého `.css` souboru. CSS umožňuje vzhled stránky upravovat i podle velikosti displeje zařízení a tak přizpůsobit obsah i pro menší obrazovky [4].

Hlavním cílem jazyka CSS je oddělovat formátovací a stylizační pravidla od obsahu stránky. Díky tomu je možné stránky snadněji udržovat a

[1] Hyper Text Markup Language - Značkovací jazyk pro tvorbu webových technologí
[2] Extensible Markup Language - Víceúčelový značkovací jazyk
[3] Cascading Style Sheet - technologie pro úpravu vzhledu nejen webových stránek

přinutit je pracovat správně v různých webových prohlížečích, na různých platfomách, zařízení nebo dokonce při tisku [2].

3.3.2 Bootstrap

Bootstrap představuje nadstavbu nad CSS styly. Umožňuje upravovat prvky pomocí tříd a dát jim tak moderní vzhled bez potřeby upravovat styly ručně. Při správném použití tohoto frameworku je velice snadno zajištěn responzivní design webu.

3.3.3 AdminLTE

Ačkoli Bootstrap přichází již s hotovou sadou stylů pro určité prvky a pro responzivitu, tak stavba celé webové aplikace by se neobešla bez četného přidávání vlastních stylů. Existují však nástavby nad technologií Boostrap a nabízejí kompletní sadu stylů k vytvoření informačních systémů. Jedním z takových projektů je front-endový framework AdminLTE.

Tento framework nabízí celou řadu placených šablon, ale poskytuje i jednu velmi oblíbenou šablonu zdarma. Součástí tohoto frameworku jsou příklady použití stylů jednotlivých elementů a také předpřipravené šablony pro celé stránky.

Framework využívá Bootstrap, je tedy stejně jednoduché využívat responzivitu a čerpat jeho dalších výhod. Stylování stránek tedy vypadá tak, že na hotovou HTML strukturu aplikujeme třídy, na které se následně navazují styly těchto frameworků.

3.4 Databáze MariaDB

Databáze je důležitá součást informačního systému, která zpracovává nějaká data a je potřeba jejich uložení a rychlé přístupnosti k nim.

MariaDB databáze je relační databáze oblíbená vývojáři zejména webových projektů malé a střední velikosti. Systém MariaDB původně vznikl jako alternativní větev MySQL od původních vývojářů po zakoupení firmy Sun společností Oracle. MariaDB staví na své značné kompatibilitě s databází MySQL, která je považována z různých průzkumů za jednu z nejoblíbenějších databází pro webové aplikace společně s databází Oracle a MS SQL. Další přednosti systému MariaDB je její cíl zůstat open-source projektem.

Díky velké kompatibilitě s databází MySQL je možné využívat nástroje určené nativně pro MySQL i pro MariaDB. Proto pro vývoj využíváme ná-

stroj MySQL Workbench, který pro většinu případů poskytuje stejné funkce jako pro databázi MySQL.

3.5 Nástroje pro řízení projektu

Aplikace, kterou vyvíjíme, bude patřit mezi rozsahově náročnější, proto je její řešení rozděleno do dvou prací. Jedna práce (tato) se věnuje samotnému vývoji aplikace. Druhá je zaměřená na důkladné otestování aplikace, čímž má být zajištěna její kvalita spolehlivost.

Ze skutečnosti, že na aplikaci takového rozsahu pracuje více lidí, je potřeba využít takové procesy, které usnadní jednotlivé části vývoje a domluvu mezi vývojem, testováním a vedoucím práce.

3.5.1 Verzovací systém

V oblasti vývoje informačních projektů je častým nástrojem pro efektivní a produktivní vývoj verzovací systém ze skupiny Git.

Tato technologie totiž umožňují práci více vývojářů na jednom projektu současně. Jednotlivé změny v repozitáři vývojáři seskupují do tzv. commitů. Ty totiž v systémech Git představují nejmenší jednotku změny v úložišti. Commit v sobě obsahuje informace o upravených řádkách textových a změny dalších souborů.

Další důležitou funkcí je větvení. V základu je v repozitáři hlavní větev „main“ nebo „master“, který obsahuje centrální vývoj projektu. Od této větve se se oddělují další větve, dle určení. Typicky se do jednotlivých větví postupně commitují úpravy rozsáhlejší funkcionality aplikace. Díky tomu ostatní vývojáři nejsou těmito změnami zasaženi a tak si nepřekážejí.

Po dokončení úprav nebo přidávání funkcionality jsou commity větve sloučeny do rodičovské větve, kde tvoří větší celek. Tato funkce se jmenuje Merge request a její starostí je přemístit commity jedné větve do jiné větve tak, aby byly změny zaneseny. Může se stát, že se změny v obou větvích dostanou do konfliktu, poté záleží na uživateli jak konflikt vyřeší.

Dostupných nadstaveb verzovacích systémů existuje celá řada. Mezi nejvýznamnější na trhu patří GitHub a GitLab.

GitHub je pravděpodobně nejúspěšnější systém pro tvorbu projektů od jednotlivců nebo malých skupin. Obsahuje však výrazná omezení, která jsou sice odstraněny v prémiovém plánu, ale v našem případě výhody ostatních nadstaveb převažují nad výhodami GitHubu.

Z tohoto důvodu se zdá být přínosnější použít systém GitLab, který pokrývá veškeré naše požadavky.

Důležitou zmínkou je, že projekt GitLab je veden jako open-source. Díky tomu Katedra informatiky a výpočetní techniky provozuje svoji vlastní instalaci tohoto verzovacího nástroje na katedrálním serveru.

To má pro studenty nesmírnou výhodu, protože během dosavadního studia již GitLab používali a mají s ním zkušenosti. Navíc všichni zúčastnění mají na této platformě vytvořené své účty, nemusíme tedy organizačně řešit vstup na novou platformu.

Z těchto důvodů jsme se v naší volbě verzovacího systému rozhodli zvolit právě variantu s katedrálním serverem.

3.5.2 Plánování úkolů

S rozsahem práce přichází i potřeba rozvržení práce na časové úseky a stanovení cílů. Zároveň je žádoucí, aby k plánu měli přístup všechny zúčastněné osoby. Tyto požadavky sice splňuje množství známých nástrojů nebo sdílených úložišť. My jsme se ale rozhodli využít již námi používaný nástroj GitLab.

Tento nástroj obsahuje možnosti vytváření wiki dokumentace ve formátu MarkDown[4], který navíc zajišťuje základní formátování.

V této dokumentátorské sekci máme vytvořenou jednu stránku, ve které je po týdnech rozdělena struktura naplánovaných úkolů a vzniklých dotazů.

3.5.3 Issues

Různé požadavky evidujeme jako jednotlivé issues. Issues je záznam o požadavku na aplikaci nebo úkolu, který je potřeba vykonat. Issue tak eviduje přiřazení k člověku, který daný požadavek řeší, kompletní historii řešení požadavku, termíny pro jeho splnění, atd.

Systém také poskytuje štítkování těchto issues a tím pomáhá v jejich řazení, filtrování, seskupování i výběru. Díky štítkům můžeme totiž přiřazovat prioritu, závažnost, druh požadavku na aplikaci a jiné označení.

Issues nám pomáhají ve fragmentaci jednotlivých požadavků na aplikaci do přibližně stejně náročných dílů, které se poté ve vývoji dobře plní a je tak zajištěna přehlednost. Zároveň má tento princip i pozitivní psychologický dopad, kdy vývojáři mají dobrý pocit z dokončeného úkolu a mají tak chuť pokračovat dalším issue.

V systému GitLab tyto issues používáme také ve funkci Board, který využívá principy kanbanu.

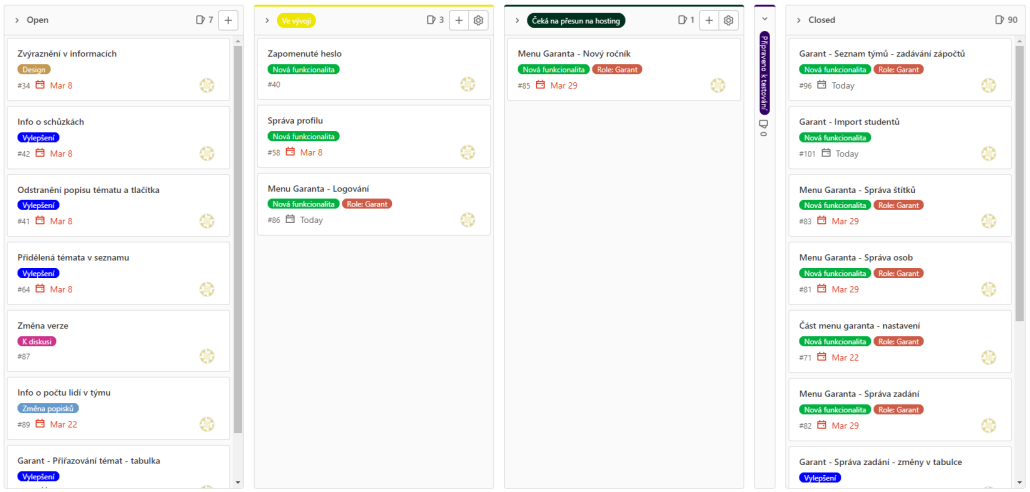
[4]Značkovací jazyk pro vytváření formátovaného textu

3.5.4 Kanban

Systém kanban je strategie řízení projektu, kdy si mezi sebou části výroby předávají výrobek. Cílem je využívat pouze ty nejnutnější zdroje.

V praxi se tento systém zobrazuje jako tabule, kde jsou rozvržené jednotlivé sloupce dle možných stavů vývoje. Následně zde máme rámečky představující výrobek, které obsahují popis jak má být produkt upraven. Podstatou věci je, že tyto rámečky následně přemisťujeme mezi jednotlivými sloupci v závislosti na reálném stavu produktu.

Obrázek 3.1 obsahuje snímek z reálného použití při vývoji. Jednotlivé issues jsou rozděleny do sloupců dle jejich stavu rozpracovanosti.



Obrázek 3.1: Ukázka použití tabule s issues (zdroj: vlastní)

V informačních technologiích se kanban používá ve velké míře pro organizaci různých úkolů a požadavků. Různé části vývoje si mezi sebou požadavek vyměňují a tím mění jeho stav.

Například když vývojář dokončí práci na nové funkcionalitě, tak přesune příslušný rámeček ze sloupce „Ve vývoji“ do sloupce „Připraveno k testování“. Tím se tester dozví, že je daný úkol připraven k testování a může si úkol přiřadit a pracovat na něm.

V GitLabu je systém kanbanu implementován v souvislosti s issues a jejich štítky. Každý issue je zobrazen jeden rámeček na tabuli kanban, kam se automaticky po vytvoření issue promítne a po přidělení štítků zařadí do připravených sloupců.

Propojenost s issue nám ulehčí práci s přepisováním jednotlivých požadavků do jiných nástrojů, které mají obdobné funkce.

3.5.5 MantisBT

Mantis Bug Tracker je webová aplikace pro nahlašování a evidenci chyb (defektů) vytvořených v průběhu vývoje aplikace. Přidávané defekty lze velmi podrobně popsat, určit prioritu a štítkovat.

MantisBT využíváme pro nahlašování objevených selhání nalezených především, ale ne výhradně pomocí komplexního automatizovaného testování aplikace. Na základě reportování defektů jsou následně žádány jejich opravy chyb a po opravě jsou znovu testovány. Pro naše potřeby používáme vlastní instalaci MantisBT, která je přístupná pro anonymně přihlášené, aby mohli v budoucnu nahlašovat defekty i uživatelé aplikace.

4 Návrh aplikace

4.1 Dekompozice specifikace

4.1.1 Zadání

V moment zveřejnění zadání pro mentory je mentorům umožněno zvolit si téma k mentorování. Po zvolení se automaticky přidělí danému mentorovi.

Po úplném zveřejnění tématu je zadání zobrazováno ve veřejném seznamu témat. Vedoucí týmů mohou o takové zadání projevit zájem hodnotou na pětibodové stupnici vážnosti.

Zadání se skládá z částí:

- název zadání
- stručný popis zadání
- PDF soubor s podrobným zadáním
- doporučovaná velikost týmu
- časová náročnost řešení
- kontaktní informace zadavatele
- požadovaný profil týmu.

Požadovaný profil týmu obsahuje štítky ze stejného souboru štítků jako v profilu týmu.

4.1.2 Tým

Tým je seskupení několika studentů. Má svého vedoucího, který vede tým a hájí jeho zájmy. Cílem týmu je řešení společného projektu na základě přiřazené tématu.

Tým je vytvořem na požádání vedoucího garantem předmětu. Po vytvoření nového týmu se založí nový uživatelský účet pro vedoucího. Následně jsou vedoucímu na jeho studentský email zaslány přihlašovací údaje.

Nastavení týmu provádí z zejména vedoucího týmu včetně přidávání studentů do týmu ze seznamu volných studentů zapsaných na předmět.

Vedoucí týmu je zodpovědný za obsah, který do softwarové podpory vkládá.

4.1.3 Projekt

4.2 Případy užití

Příklady užití popisují v krocích způsob používání různých částí aplikace uživatelem nebo systémem. Tyto případy představují se zadavatelem dohodnutou funkčnost systému. Vytvořené případy užití usnadňují tvorbu požadavků na systém pro následné sepsání testovacího plánu.

Z podrobné specifikace požadavků byly sestaveny případy užití, které popisují práci s výslednou aplikací. Rozdělení případů užití jsou ilustrovány v use-case diagramu (viz obrázek 4.1). Následuje seznam sestavených případů užití s popisem funkcí, které pokrývají.

4.2.1 Nepřihlášený uživatel

Přihlášení (UC.01)

Velká část aplikace jsou přístupné pouze uživatelům dle jejich role. To vytváří potřebu autentizace v naší aplikaci. Uživatel má možnost se přihlásit pomocí přihlašovací stránky pomocí registrované kombinace přihlašovacího jména a hesla. Pokud je uživatel autentizován, je přesměrován na domovskou obrazovku pro jeho konkrétní roli.

Zobrazení témat (UC.02-03)

Nepřihlášený uživatel (např. běžný student) si může zobrazit seznam volných témat v přehledu, kde se dozví základní informace o tématech.

Tyto témata lze rozkliknout do podrobnější podoby, kde je zobrazen název, stručný popis, časová náročnost, doporučená velikost týmu, zadavatel a mentor zadání. Také lze stáhnout PDF soubor s podrobným zadáním.

Neúplné týmy (UC.04)

Nepřihlášený uživatel má k dispozici seznam neúplných týmů, které hledají nové členy. Případný student má k dispozici seznam obsazených rolí a kontakt na vedoucího týmu.

4.2.2 Přihlášený uživatel

Odhlášení (UC.05)

Přihlášený uživatel má možnost ukončit svou relaci a odhlásit se. Po této akci je uživatel přesměrován na hlavní stránku nepřihlášeného uživatele.

4.2.3 Vedoucí týmu

Schůzky týmu (UC.06-09)

Vedoucí týmu může v aplikaci naplánovat schůzku. Schůzka může být obsahuje čas a místo konání a v jakém uskupení. Schůzka může být například interní, kdy tým se schází s jeho členy, aby například probrali další postup nebo může být za účasti mentora, zadavatele apod. Pokud je zvoleno, že se schůzky účastní mentor projektu týmu, zobrazí se schůzka kromě stránky týmu i na stránce mentorovaných projektů mentora.

Řešení projektu (UC.10-13)

V řešení projektu vedoucí týmu zaznamenává postup svého týmu pomocí čtyřstavové hodnoty v několika kategoriích. Na základě těchto záznamů přiřazený mentor kontroluje a potvrzuje tento postup týmu volbou ze stejného čtyřstavového výběru.

Vedoucí týmu má na stránce řešení projektu možnost vyplnit a přidat užitečné odkazy.

Editace týmu (UC.14-16)

Vedoucí má možnost přidávat nebo odebírat členy týmu a signalizovat jednotlivé stavy týmu. Členům týmu může přiřazovat a odebírat týmové role. Také může vyplňovat týmový profil z předem definovaného seznamu schopností. Řádně stanovený profil dle opravdových schopností členů týmu může orientačně pomoci s výběrem tématu.

Stránka tématu a zájem o téma (UC.17-19)

Vedoucí jehož tým nemá přiřazené žádné téma může na stránce detailu volného zadání projevit zájem. Vážnost zájmu o téma je vybírána ze stupnice pěti hodnot od méně významné po nejvážnější. Vedoucí nemůže projevit zájem o téma s vážností, s kterou již má aktivní zájem o jiné téma.

Projevené zájmy o témata (i ty neaktivní) jsou zobrazeny na stránce týmu, kde aktivní zájmy může vedoucí odvolat.

4.2.4 Mentor

Změna kontaktních údajů (UC.20)

Mentor má ve správě profilu zpřístupněnou úpravu kontaktních údajů. Tyto údaje se následně zobrazují například u detailu zadání, kde je zobrazena

„vizitka“ přiřazeného mentora.

Mentorované projekty (UC.21-25)

Jednou z hlavních tabulek mentora jsou mentorované projekty. V této tabulce uvidí o každém mentorovaném projektu stav řešení projektu v kompaktním zobrazení, datum přiřazení tématu týmu a samozřejmě prokliky na projekt a tým.

Předpokládá se, že tato tabulka bude hlavním rozcestníkem mentora v průběhu mentorování projektů a tedy stránka na které se nachází je nastavena jako mentorova domovská stránka.

Schůzky (UC.09-10,26)

Mentor má přístup k vytváření schůzek týmů stejně jako vedoucí týmu. Navíc však disponuje tabulkou obsahující schůzky všech týmů, kterých má být mentor zúčastněný.

Volba tématu (UC.27-28)

Součástí navigace mentora je i položka „Položka bez mentora“ vedoucí na stránku s tabulkou témat, která nemají mentora. Tyto témata jsou buď zveřejněná pouze pro mentory nebo zcela veřejná. Témata z této nabídky si mohou mentoři v detailu zadání přiřadit k mentorování. Po přidělení je téma z tabulky odstraněno a naopak se objeví v tabulce „Témata bez týmu“, pokud téma dosud není přiřazeno žádnému týmu, nebo na stránce „Témata bez týmu“.

Témata bez týmu (UC.29-30)

Stránka témat bez týmu je rozdělena na dvě tabulky. Jedna tabulka obsahuje prostý výčet témat, která má mentor přidělené k mentorování, ale dosud žádný tým o ně neprojevil zájem. Druhá komplexnější tabulka obsahuje veškeré projevené zájmy týmu o zadání. Účel tabulky pochází ze specifikace, že mentor posuzuje vhodnost tématu pro daný tým. Výsledek posouzení mentor v příslušnému zájmu přidělí označení „Nevhodné“ nebo „Vhodné“. Výchozí hodnota, kdy mentor dosud není rozhodnut je „Neohodnoceno“.

Historie mentorování (UC.31)

Aplikace má v sobě udržovat historii předchozích ročníků TSP. Proto mentor má k dispozici výpis témat, která dosud mentoroval. Mentor se může na dané

projekty prokliknout a prohlížet je.

4.2.5 Garant

Témata bez mentorů

Garant vidí tabulku se seznamem témat, která nemají dosud přiřazeného mentora. Součástí tabulky je i název zadavatele.

Seznam týmů

Obdobná tabulka jako u pohledu mentora „Mentorovaná tabulka“, obsahuje tedy podobné . Tabulka však navíc obsahuje i týmy, které nemají přiřazené žádné zadání. V tom případě tak proklik odkazuje na stránku týmu.

Tabulka umožňuje přistoupit k editaci daného týmu, kde lze upravit údaje o týmu, zadávat datumy zápočtů a vytvářet záznamy o obhajobách.

Seznam mentorů

Stránka ukazující propojení témat a jejich mentorů mentorů. Proklikem na jméno mentora je mentor zvolen a garant může skrz navigační menu přistupovat k pohledům mentora.

Správa osob

Garant má k dispozici tabulky se seznamy všech uživatelů, kteří se mohou přihlásit. Tito uživatelé jsou rozděleni dle jejich role na mentory a vedoucí. Garant má oprávnění těmto uživatelům odebrat přístup čímž jim zamezí v jejich dalším přihlášení.

Správa témat

Garant má přístupnou tabulku všech zadání, kde o nich může vidět základní informace.

Součástí správy témat je i vytváření a editace nového zadání. Garant kromě všech dat zadání může přiřadit zadání mentora, určit ročník ve kterém je zadání vyvěšeno a upravovat jeho viditelnost.

Správa štítků

Používané štítky může garant také upravovat, vytvářet a znemožnit další použití. V aplikace jsou dva druhy štítků. Týmové role, které vedoucí přiřazuje členům jeho týmu a štítky pro profil týmu a zadání. Pokud garant

pro jakýkoliv štítek nastavení, že není dostupný pro další užití, není možné jej nově použít. Naopak při znepřístupnění štítku není odebráno již stávající přiřazení, aby nebyla nijak upravována historie již proběhlých projektů.

Přidělování témat

Speciální tabulka, kde každý sloupec reprezentuje tým bez přiřazeného tématu a každý řádek jedno volné zadání. V tabulce jsou vyplněné ty buňky, kde existuje zájem týmu o dané zadání. Garant zde vidí vážnost zájmu a datum vytvoření zájmu. Zobrazuje se zde i doporučení mentora, které mentor vyplní na stránce „Témata bez týmu“. Na základě těchto informací garant definitivně přiřadí zadání jednomu z přihlášených týmů.

Správa studentů

Na stránce správy studentů má garant k dispozici tabulku všech přidaných studentů. Z těchto studentů vedoucí vkládají členy do svých týmů. Garant může nad těmito studenty vyvolávat akce mazání a stanovení vedoucího nového týmu. Při vytvoření nového týmu je potřeba vyplnit počáteční název týmu. Studenti se do této tabulky přidávají buď jednotlivě pomocí formuláře nebo z přiloženého CSV[1] souboru.

Vkládání studenta jednotlivě vyžaduje znalost přihlašovacího jména do systému StaG, křestní jméno, příjmení, email, osobní číslo, ročník ve kterém je TSP studováno a případně obor studia se studovanou fakultou.

Import studentů pomocí CSV umožňuje přidání velkého množství studentů najednou. Jedná se o vhodnou variantu pro hromadné přidávání studentů například na začátku semestru. Je počítáno s CSV souborem formátu, který poskytuje systém StaG při exportu studentů zapsaných na daný předmět. I v tomto případě je potřeba určit do kterého ročníku TSP importovaní studenti náleží.

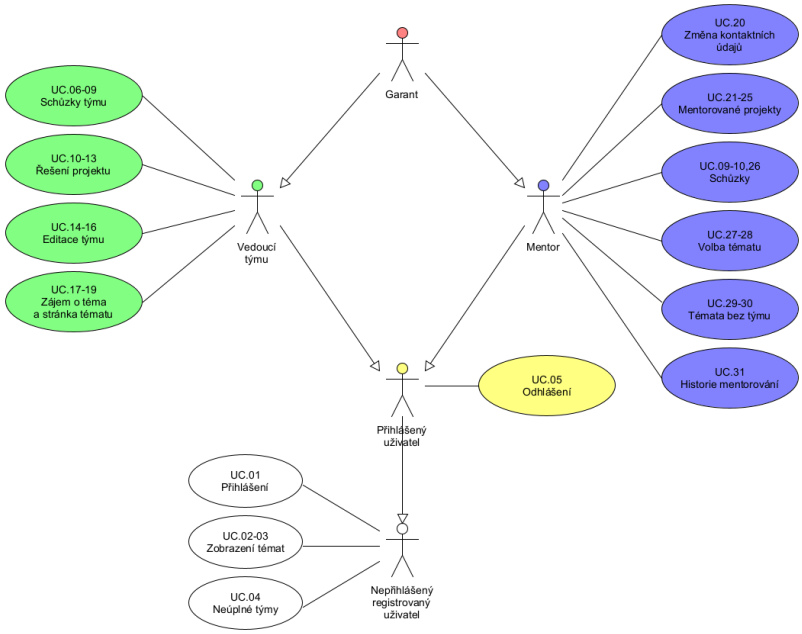
Správa ročníků

Garant ve správě ročníků může vidět informace o jednotlivých proběhlých ročnících. Zároveň může tvořit nové ročníky a nastavovat aktuální ročník TSP. Po zvolení nového aktuální cyklu TSP se původní cyklus považuje za historický a je tak i zobrazován ve zbytku aplikace. Tímto způsobem je řešeno zachování předchozích ročníků TSP, kdy v běžném stavu aplikace se zobrazují zejména údaje vztažené k aktuálnímu ročníku TSP.

[1] Comma-separated values - jednoduchý souborový formát pro výměnu tabulkových dat

Logování

Pro sledování správní činnosti aplikace má garant přístup k zobrazení logovacích záznamů. Tato funkce je zprostředkována pomocí parsování sekvencí záznamů do tabulky. Garant si může zvolit logovací soubor k nabídky k jednoduchému zobrazení. Zvolený záznam může také stáhnout v textové podobě do počítače, kde s ním může provádět další operace.



Obrázek 4.1: Schéma případů užití (zdroj: vlastní)

4.3 Architektura aplikace

4.3.1 Třívrstvá architektura

[8]

Architektura MVP[2] je třívrstvový návrh struktury aplikace pro rozdělení souborů dle funkčnosti s ohledem na rozšiřitelnost. MVP architektura se skládá z datové části (model), která se stará o práci s daty a umožňují vytvářet abstrakci nad jednotlivými strukturami. Například jednotlivé akce s databází v kontextu dat se kterými aplikace pracuje. Další část je aplikační vrstva (presenter), která řídí chod aplikace, předává data z šablon do modelů a naopak. Poslední vrstvou je vrstva prezentační. Ta se skládá z

[2]Model-View-Presenter - třívrstvá architektura

jednotlivých šablon a prezentace dat. Představuje tak rozhraní, přes které uživatel komunikuje s aplikací.

Mezi další velice rozšířenou architekturou patří MVC architektura. Hlavním rozdílem mezi MVP a MVC architekturami je, zpracovávání vstupů od uživatele. V MVP architektuře jsou vstupy obsluhovány prezentovací vrstvou.

4.3.2 Komponenty aplikace v Nette

Komponenty jsou části aplikace, které vkládáme do stránek. Nejčastěji se jedná o formuláře, tabulky a další objekty, které je možné používat opakovaně.

Nette má v sobě vestavěný komponentový systém, který umožňuje komponenty vkládat na stránky a někdy dokonce do jiných komponent, kdy se tak tvoří tzv. komponentový strom. Velkou výhodou je, že se komponenta tvoří až tehdy, kdy je opravdu použita. To těží výhodu zejména zpracování AJAX požadavků, kde je typicky výsledkem operace pouze část stránky, kdy není komponenta využita. Díky tomu se vůbec nevytváří tím šetří výkon serveru.

Komponenty se vytváření v továrních funkcích v presenteru. Tyto funkce mají prototyp `createComponent<Name>(): Control` [7].

Ukázka tovární metody v presenteru

```
class DefaultPresenter extends Nette\Application\UI\Presenter
{
   protected function createComponentPoll(): PollControl
   {
      $poll = new PollControl;
      $poll->items = $this->item;
      return $poll;
   }
}
```

Ukázka kódu 1: Ukázka tovární metody v presenteru

Komponentu lze definovat přímo v tovární metodě, ale pro účely znovupoužitelnosti je možné pro komponentu vytvořit vlastní třídu dědící třídu `Nette\Application\UI\Control`.

4.3.3 Dependency Injection v Nette

Vkládání závislostí je technika programování vedoucí k čistému a přehlednému kódu. Princip tohoto způsobu programování spočívá v předávání zdrojů, které konkrétní funkce nebo třída vyžaduje, aby splnila svoji úlohu. Cílem je zamezit získávání zdrojů přímo v daném objektu, kdy programátor bez žádné další informace neví jaký zdroj se získá a jak se získá. V těchto případech říkáme, že se vytváří v aplikaci skryté vazby, díky kterým vzniká nepřehledný a špatně udržitelný kód [5].

Pro implementaci DI se využívá tzv. Dependecy Container. Tento nástroj slouží k uchování instancí jednotlivých služeb a jejich vytváření, pokud se jedná o závislosti aktuálně potřebných částí aplikace.

Nette pro implementaci Depencency Containeru využívá vlastní knihovnu Nette DI Container. Knihovna se stará o automatické generování a aktualizaci tříd kontejneru na základě konfiguračního souboru formátu NEON. Díky této technologii programátor nemá potřebu implementace vlastního řešení a může se soustředit na vývoj logiky aplikace. Při vytváření jednotlivých služeb však nesmí zapomenout tuto službu zaregistrovat zapsáním do konfiguračního souboru.

Další technologií, kterou Nette využívá, je nástroj Autowire. Ten se stará o automatické předávání závislostí z dependency kontejneru do konstruktorů a parametrů dalších funkcí [6].

4.3.4 Formuláře v Nette

Formuláře jsou důležitou součástí webového systému. Jedná se o jednu z mála možností jak získat uživatelský vstup. Zároveň tak představují bezpečností riziko, které je nutno ošetřit. Typů útoků skrz uživatelský vstup přes formuláře je velké množství a jejich ošetření klade nemalé časové a znalostní na programátora. Také i proto je tvorba a validace formulářů je velmi rutinní činnost. Framework Nette proto přichází se svým řešením Nette Forms. Jedná se o možnost tvorby formulářů, kdy programátor definuje základní prvky formuláře a framework sám z těchto informací sestaví formulář. Výhodou tohoto přístupu spočívá v bezpečnosti. Nette Forms totiž validuje vstupy uživatele podle definovaných validačních pravidel a to jak na straně backendu, tak v případě připojených příslušných JavaScript skriptů i na straně frontendu. Aplikaci tedy komponenta vrátí validovaná data, která může bez starostí použít.

Tvorba formuláře probíhá jako obyčejná komponenta. Tovární funkce vrací instanci třídy `Nette\Application\UI\Form`. Funkce instanci vytvoří a postupně do ní přidává formulářové prvky, kterým lze definovat další různá

pravidla nebo vlastnosti. Je také potřeba implementovat kód, který se spustí po nějaké události ve formuláři. Zejména je myšlena událost úspěšně odeslaného formuláře, kdy je pravděpodobně nutné vykonat nějakou akci aplikace (např. uložit data).

Vytvořenou komponentu formuláře je možno automaticky vykreslit dle předem vytvořené šablony pomocí šablonovacího systému Latte a tak jednoduše vložit formulář do příslušné stránky. Je však ponechána možnost vytvoření si jednotlivých formulářových prvků v šabloně a dle identifikátoru je propojit s prvky v komponentě [6].

4.3.5 Database Explorer

4.3.6 Testovatelnost aplikace

Testování robot frameworkem

Aby bylo možné aplikaci automaticky otestovat je potřeba umožnit testovacímu softwaru ovládat aplikaci. Proto musí být veškeré ovládací prvky jednoznačně identifikovatelné. Toho docílíme zavedením jednoznačných identifikátorů vložených do atributu `id` daných elementů.

Robor framework díky tomu je schopen nalézt požadovaný prvek stránky. Takto jednoznačně identifikovatelné není potřeba mít pouze akční prvky typu odkazu nebo tlačítka, ale také i dalších prvků stránky. Ty Robot framework potřebuje nalézt, aby z nich mohl číst data.

Navržené automatické testování se na tuto vlastnost spoléhá např. při orientaci v aplikaci. Každá stránka s obsahem má svůj nadpis. Tento nadpis má jednoznačný identifikátor v celé aplikaci. Na základě této značky se při testování zjišťuje, zda je otevřena žádaná stránka. Obdobným způsobem dokáže také získat data z jednotlivých elementů, ke kterým se dostane, a může provést jejich kontrolu.

Jednotkové testování

Princip jednotkového testování je ověření nejelementárnějších procedur aplikace. V tomhle ohledu se práce vyvíjené webové aplikace se skládá z většinové části pouze z interpretace dat z a do databáze voláním funkcí Nette frameworku a tedy se nedostáváme na elementární úroveň, kterou můžeme jednotkově testovat. Avšak při vývoji byly vytvořeny pomocné procedury, které pomáhají se zpracováním dat (např. parsování). Tyto funkce jsou oddělené od logiky aplikace a jsou závislé pouze na svých vstupech. Pro těchto pár funkcí byly vytvořeny jednotkové testy, které ověřují jejich výstupy.

Logování

Logování je určené pro dlouhodobé sledování činnosti programu. V sekvenci záznamů je možné dohledat souvislosti zpracovaných akcí a stavu systému. Pomocí logování sledujeme zejména běžnou činnost programu, výskyt chyb a výjimek, konfiguraci programu a její změny. Tato technika lze být použita i pro ladění, ale v našem případě máme jiné ladící nástroje [3].

Pro PHP existuje profesionálně používaná knihovna Monolog[3]. Ta již integrována do známých frameworků jako Symfony, Laravel i Nette. Logovací záznam je pole o jednotlivých položkách např. úroveň, časové razítko, jméno loggeru, zaznamenávaná zpráva, dodatečn informace, atd. Tyto údaje však negeneruje programátor přímo, ale zpravidla pomocí logovacích funkcí loggeru.

Záznamy loggerů ukládáme do adresáře `/log/syslog/` v textovém formátu `.log`. Záznamy jsou rozděleny do jednotlivých souborů dle dne a soubory s logy starší 30 dní jsou automaticky mazány. Veškeré tyto parametry lze změnit v konfiguraci aplikace a upravit tak logování dle zjištěných reálných potřeb systému.

Ladící nástroj Tracy

4.3.7 Responzivita

Responzivní desing spočívá v přizpůsobení zobrazení obsahu šířce okna webového prohlížeče (příp. velikosti obrazovky zařízení). To znamená, že uživatel navštěvující naší stránku ze stolního počítače nebo mobilního zařízení uvidí uspořádaný obsah tak, aby byl na daném zařízení dostatečně čitelný a ovladatelný [4].

Front-end aplikace je implementován frameworkem AdminLTE, který staví na základech technologie Bootstrap. Díky tomu je snadné udržovat aplikaci responzivní pokud je dodržena správná HTML struktura a jsou důsledně aplikovány styly použitím tříd stylů.

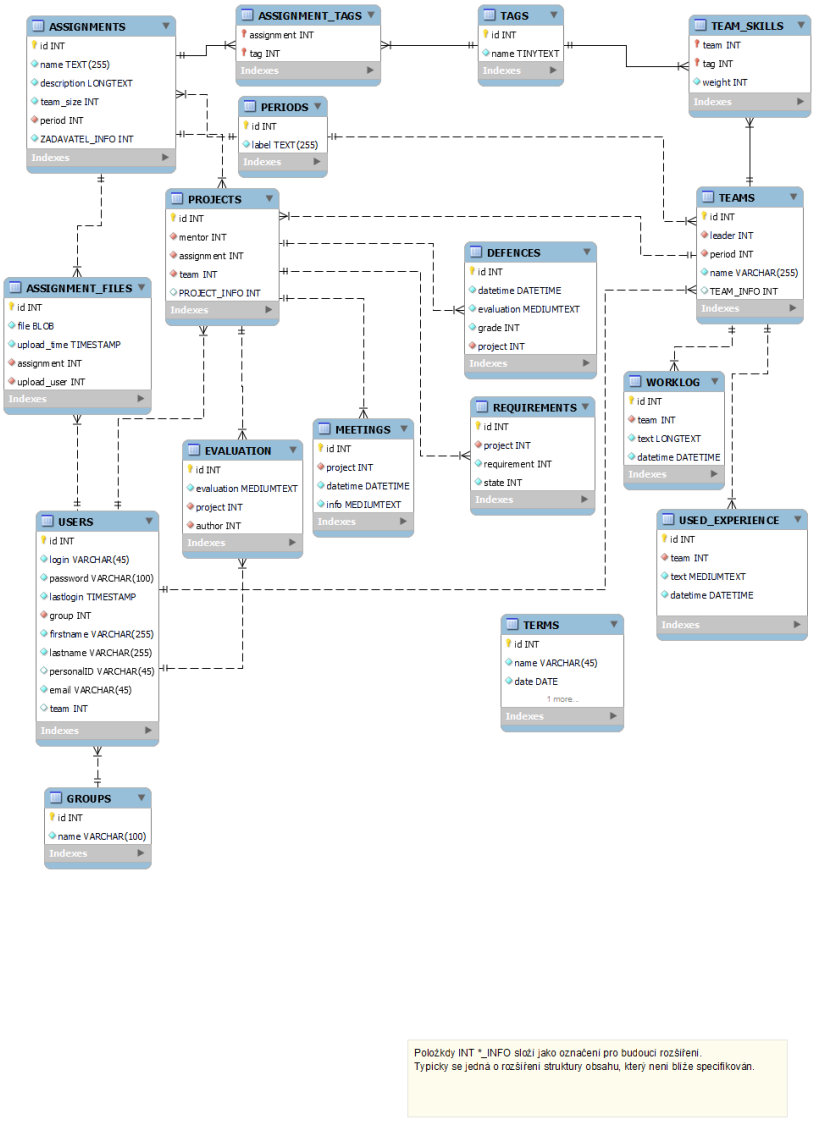
4.4 Databázová struktura

Struktura databáze musí obsahovat veškeré informace použitelné v softwarové podpoře a umožnit jejich snadné propojení. Dalších aspektem pro tvorbu struktury je použitý PHP framework, který využivá službu pro správu

[3]https://github.com/Seldaek/monolog

databázových dotazů a jejich optimalizaci. Pro jeho správnou funkci je potřeba mít strukturu databáze řádně připravenou včetně vytvořených indexů a propojení pomocí cizích klíčů.

Databázová struktura musí respektovat, že požadavkem na aplikaci je prohlížení historie. To nás přivádí na strategii, že každá tabulka u které to bude potřeba bude provázána s tabulkou `PERIOD` jejíž řádky představují jednotlivé cykly TSP. Také se však vyskytuje problém u odstraňování záznamů. Prakticky nebude možné z databáze cokoli odstraňovat, aby nebyl porušen konzistentní stav již proběhlých projektů. To je následně ve struktuře zapracováno v různých tabulkách sloupcem s příznakem, který určuje zda je prvek odstraněn nebo znepřístupněn pro další užití. Tím prvek stále existuje, není porušeno integritní omezení a potřebné změny pro nové projekty jsou upraveny.



Obrázek 4.2: Schéma databázové struktury (zdroj: vlastní)

4.4.1 Základní tabulky

- USER – Tabulka pro ukládání uživatelů, kteří mají/měli přístup do aplikace. Součástí této tabulky jsou přihlašovací údaje, role uživatele a následné propojení na další informace dle role uživatele.
- STUDENT – Uživatelé s rolí „Vedoucí“ vycházejí ze záznamu tabulky „STUDENT“, která obsahuje seznam všech studentů studujících předměty TSP.
- TEAM – Tabulka pro evidenci týmů s jejich vedoucích.

- PROJECT – Tabulka propojuje zadání z tabulky „ASSIGNMENT“ a týmu z tabulky „TEAM“, tím vzniká projekt.
- CONTACT – Tabulka pro evidenci informací a kontaktů na mentory a zadavatele.
- ASSIGNMENT – Tabulka pro ukládání jednotlivých zadání.
- ASSIGNMENT_INTEREST – Tabulka pro evidenci zájmů o témata.

4.4.2 Číselníkové tabulky

Tabulky sloužící jako číselník.

- TEAM_ROLE – Tabulka dostupných rolí v týmu.
- TAG – Tabulka profilů pro použití v týmu a zadání.
- PROGRESS – Tabulka se seznamem bodů, které tým v projektu plní.
- REQUIREMENT – Tabulka se seznamem kritérií, která musí tým splnit.
- EXPERIENCE – Tabulka využitých zkušeností
- PERIOD – Tabulka jednotlivých cyklů aplikace a předmětů TSP.

4.4.3 Spojovací tabulky

Následující tabulky jsou pomocné tabulky pro realizaci N:M relace. Propojují typicky dvě tabulky s případným ohodnocením nebo informacemi dané relace.

- TEAM_ROLE_STUDENT – Propojení mezi tabulkami „TEAM_ROLE“ a „STUDENT“. Tabulka slouží pro zaznamenání přiřazení rolí jednotlivým členům týmu.
- TEAM_SKILL – Propojení mezi tabulkami „TAG“ a „TEAM“. Tabulka specifikuje jednotlivé profily týmů .
- ASSIGNMENT_TAG – Propojení mezi tabulkami „TAG“ a „ASSIGNMENT“. Tabulka specifikuje jaký profil týmu je předpokládán pro tým, který bude zadání řešit.

- PROGRESS_STATE – Propojení mezi tabulkami „PROGRESS“ a „PROJECT“. Tabulka eviduje postup v řešení projektu. Relace uchovává stavovou hodnotu daného přepínače.
- REQUIREMENT_STATE – Propojení mezi tabulkami „REQUIREMENT“ a „PROJECT“ pro realizaci N:M relace. Tabulka eviduje plnění kritérií při řešení projektu. Relace uchovává stavovou hodnotu daného přepínače.
- EXPERIENCE_STATE – Propojení mezi tabulkami „EXPERIENCE“ a „PROJECT“ pro realizaci N:M relace. Tabulka eviduje stav potvrzení využitých zkušeností. Relace uchovává stavovou hodnotu daného přepínače.

4.4.4 Další tabulky

- ASSIGNMENT_FILE – Tabulka sloužící pro uložení PDF souborů s příslušnými informacemi.
- SETTING – Tabulka s nastavením aplikace. Každý záznam představuje typ nastavení a jeho hodnotu.

4.5 Uživatelské rozhraní

4.6 Návrh loga

5 Realizace

5.1 Adresářová struktura

Adresářová struktura vychází z třívrstvé MVP architektury a „best practices“ použití Nette Frameworku.

Struktura adresářů je naznačena v následujícím stromu:

- `/`
  - `app`
    - `components` – třídy znovupoužitelných komponent
    - `locale` – překlady
    - `Model` – třídy modelů
    - `Presenters` – třídy presenterů
      - `templates` – soubory šablon a pohledů
    - `Router` – konfigurace URL adres
    - `Boostrap.php` – zaváděcí třída Boostrap
  - `config` – konfigurační soubory
  - `log` – logovací soubory
  - `temp` – dočasné soubory, cache
  - `tests` – třídy s jednotkovými testy
  - `vendor` – knihovny instalované Composerem
    - `autoload.php` – automatické načítání nainstalovaných balíčků
  - `www` – veřejný adresář
    - `src` – CSS a JavaScript soubory
    - `index.php` – prvotní soubor, kterým se aplikace spouští

5.2 Konfigurace aplikace

Framework Nette používá konfiguraci pomocí externích souborů formátu NEON. Zápis konfigurace je velmi podobný zápisu formátu YAML.

5.2.1 Konfigurace databázového spojení

Na databázi se aplikace připojuje nativním připojením pomocí balíčku `nette/database`, kdy spojení získáme jako službu z DI kontejneru.

```
1 database:
2     dsn: 'mysql:host=127.0.0.1;dbname=test'
3     user: root
4     password: password
```

Ukázka kódu 2: Konfigurace databázového spojení

V nastavení databázové struktury (viz ukázka kódu 2) nastavujeme položku `dns` jako textový řetězec obsahující typ připojované databáze, adresu serveru (případně s číslem služby) a název výchozí zvoleného schématu v databázi. Dále se nastavuje uživatelské jméno a heslo pro připojení.

5.2.2 Vlastní nastavení aplikace

Do konfigurace aplikace lze vkládat vlastní konfigurovatelné možnosti, které lze následně v aplikaci načítat a upravovat tak chování systému. V konkrétním případě tohoto využíváme pro nastavení režimu pro testování aplikace, kdy veškerá emailová komunikace se zasílá na předem definovaný email.

5.3 Vylepšování aplikace

5.3.1 Oprava nalezených defektů

5.3.2 Úpravy dle aktualizování specifikace

V průběhu vývoje byla specifikace a požadavky několikrát upravovány.

5.3.3 Připomínky budoucích uživatelů

Od počátku vývoje jsou přijímány podměty pro vylepšování uživatelského komfortu z používání aplikace. Nejčastěji se jednalo o úpravy vzhledu, popisků nebo změnu pozicování prvků stránek. Vyskytly se však návrhy na přidání zcela nových funkcionalit, které zcela vytváří nový modul aplikace nebo pouze rozšiřuje již existující funkce.

Na začátku prosince roku 2021 bylo uskutečněno představení vyvíjeného softwaru budoucím mentorům předmětů TSP za účelem získání zpětné vazby od reálných budoucích uživatelů tohoto systému.

Veškeré návrhy a podměty byly projednány a případně zaneseny do nástrojů pro řízení projektu.

5.4 Požadavky na zavedení aplikace

6 Testování

7 Závěr

Seznam ukázek kódu

1 Ukázka tovární metody v presenteru 24
2 Konfigurace databázového spojení 32

Literatura

[1] Callum, H. *PHP Okamžitě*. Computer Press, 2014. ISBN 978-80-251-4196-0.

[2] Castro Elizabeth, H. B. *HTML5 a CSS3*. Computer Press, 2012. ISBN 978-80-251-3733-8.

[3] Herout, P. *Přednášky z OKS* [online]. Pavel Herout, 2022. [cit. 2022/04/10]. Dostupné z: `https://www.kiv.zcu.cz/~herout/vyuka/oks/prednasky/oks-2022.pdf`.

[4] Lazarit, L. *CSS Okamžitě*. Computer Press, 2014. ISBN 978-80-251-4176-2.

[5] Napoli, M. *Understanding Dependency Injection* [online]. Matthieu Napoli, 2022. [cit. 2022/03/29]. Dostupné z: `https://php-di.org/doc/understanding-di.html`.

[6] *Dependency Injection* [online]. Nette Foundation, 2022. [cit. 2022/03/29]. Dostupné z: `https://doc.nette.org/cs/dependency-injection`.

[7] *Interactive Components* [online]. Nette Foundation, 2022. [cit. 2022/03/27]. Dostupné z: `https://doc.nette.org/en/application/components`.

[8] Čápka, D. *Třívrstvá architektura* [online]. David Čápka, 2021. [cit. 2022/04/02]. Dostupné z: `https://www.itnetwork.cz/navrh/architektury-a-dependency-injection/trivrstva-architektura-a-dalsi-vicevrstve-architektury`.