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

Bakalářská práce

Softwarová podpora organizace předmětů TSP

Plzeň 2022 Jan Hinterholzinger

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Jan HINTERHOLZINGER**
Osobní číslo:	**A19B0052P**
Studijní program:	**B0613A140015 Informatika a výpočetní technika**
Specializace:	**Informatika**
Téma práce:	**Softwarová podpora organizace předmětů TSP**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Seznamte se s komerčně úspěšnými Php frameworky. Dále se seznamte s požadavky kladenými na organizaci předmětů KIV/TSP 1,2. Prostudujte též alespoň jeden nástroj pro řízení projektů.
2. Navrhněte webovou aplikaci pro softwarovou podporu organizace předmětů TSP. V návrhu klaďte důraz na modularitu, responzivitu, testovatelnost a možnosti budoucích rozšíření.
3. Podle návrhu vytvořte s pomocí zvoleného frameworku webovou aplikaci. Při realizaci zohledňujte průběžné návrhy na zvyšování její kvality.
4. Vytvořenou aplikaci důkladně zdokumentujte a ověřte její funkcionalitu připravenými testy.

Rozsah bakalářské práce: **doporuč. 30 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce.

Vedoucí bakalářské práce: **Doc. Ing. Pavel Herout, Ph.D.**
Katedra informatiky a výpočetní techniky

Datum zadání bakalářské práce: **4. října 2021**
Termín odevzdání bakalářské práce: **5. května 2022**

Západočeská univerzita v Plzni
Fakulta aplikovaných věd
L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 14. října 2021

Prohlášení

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

V Plzni dne 3. května 2022

Jan Hinterholzinger

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

Bakalářská práce se zabývá návrhem systému pro podporu organizace předmětů TSP. Cílem je navrhnout a implementovat robustní webovou aplikaci s řízeným přístupem, která vychází ze specifikace předmětu. Součástí práce je zvolení vhodného frameworku, popsání případů užití a sestavení struktury modelu databáze. Návrh aplikace zohledňuje požadavky na modularitu, responzivitu a možnosti budoucího rozšíření. Při vývoji aplikace je pohlíženo zejména na použitelnost, spolehlivost a bezpečnost. Výsledná aplikace je důkladně otestována připravenými testy.

Obsah

1 Úvod

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.

Na obou předmětech je doba pro řešení projektu, dva semestry rozdělené do dvou předmětů a ročníků. Budou zapojeny čtyři skupiny lidí garant, zadavatelé, mentoři a studenti, což bude klást poměrně značné nároky na organizaci předmětu. Bylo rozhodnuto o vytvoření webové aplikace, která všem zúčastněným stranám zjednoduší zadávání témat a evidenci postupu prací projektu. Cílem aplikace je umožnit lepší informovanost a zapojení studentů, řešení evidence zadavatelů a možnosti sdílení informací mezi mentorem, garantem a vedoucími týmů.

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, modulární a důkladně otestována.

Pro tak značný rozsah časový i personální by již pravděpodobně nestačila evidence pomocí spreadsheet[1] tabulky, byť potenciálně sdílené.

[1] tabulkový procesor

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. Z tohoto důvodu je potřeba aplikaci psát s dostatečným odstupem a přiměřeně obecně, aby odolávala drobným změnám specifikace. Aplikaci bylo stanoveno jméno PSTSP (Podpůrný software TSP).

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.

Životní cyklus bude nejčastěji probíhat následovně.

- Garant přijímá od domluvených zadavatelů jejich zadání, která mají studentské týmy řešit. Tato zadání společně s PDF souborem vloží do PSTSP pro zvolené období předmětu.
- Před začátkem TSP budoucí vedoucí týmu zažádá garanta o založení účtu v PSTSP s rolí „Vedoucí“. Vedoucí týmu společně s dalšími studenty sestaví tým a vloží informace o týmu.
- Sbor mentorů vyhodnotí zadání z hledisek náročnosti, vhodnosti, technologické proveditelnosti a případně požádá zadavatele o jejich korekci či upřesnění. Z opravených zadání se stanou témata.
- Poté co jsou témata projektů zveřejněna, vedoucí týmu (na základě rozhodnutí v týmu) projeví zájem o jedno či více zvolených témat.
- Mentor zodpovědný za téma ohodnotí vhodnost zvoleného tématu pro daný tým na základě zájmu a deklarovaných schopností týmu.
- Garant následně podle rozhodnutí mentora a zájmu týmu definitivně přiřadí některé z témat týmu. Přiřazením tématu vzniká projekt.
- Tým řeší projekt a postup řešení vkládá do PSTSP.

- Mentor na základě postupu řešení týmu kontroluje a potvrzuje postup týmu.
- Na závěru předmětu TSP se koná obhajoba projektu, jejíž výsledek se zapíše do PSTSP.

2.2 Role v PSTSP

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 již existující 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 témata, které následně ohodnocuje mentor a na základě zájmu je garantem téma přiděleno. Také má povinnost evidovat v projektu splnění kritérií, postupu a využitých zkušeností.

2.2.3 Mentor

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

2.2.4 Garant

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

3 Dostupné technologie

3.1 PHP

PHP je jedním z nejoblíbenějších jazyků zpracovávaných na straně serveru v oblasti webového vývoje. Jedná se volně dostupný skriptovací jazyk, který je spravován a vyvíjen skupinou vývojářů s názvem The PHP Group.

Jazyk PHP se většinou používá na straně serveru (backend). Po jeho zpracování jsou výsledky typicky poskytovány webovým serverem. Takový server musí mít nainstalovaný modul PHP. Poté lze vkládat kód jazyka do značkovaného kódu jazyka HTML. Soubory s PHP kódem mají koncovku .php. Jazyk je možné provozovat na velké škále operačních systémů na všech platformách.

Díky své jednoduchosti bývá PHP častá volba pro vývoj webových aplikací. Poskytuje dostatečné množství funkcí, aby programátor nepotřeboval přecházet na jiný jazyk a vystačil si čistě s jazykem PHP. Jazyk je konkurentem pro jazyky jako ASP.NET, Java a Node.js, které se také zabývají webovým vývojem.

Díky velké popularitě jazyka vznikla velká komunita vývojářů, kteří převážně zdarma nabízejí pomoc na různých fórech [1].

3.2 Srovnání PHP Frameworků

Existuje několik cest jak webovou aplikaci vyvíjet. V základu se jedná o dva směry. Vyvíjení aplikace z již vytvořených komponent a tvorba od úplného počátku. Oba směry mají své výhody a nevýhody. Pro rozsáhlé systémy se však posledních několik let rozmáhá použití frameworků.

3.2.1 PHP Framework

PHP framework je platforma pro tvorbu webových aplikací. Obsahuje již vytvořené knihovny pro často používané činnosti. Tvoří tak základ, na kterém mohou vývojáři stavět svůj produkt. Tento přístup je často vítaný, protože umožňuje urychlení a sjednocení vývoje.

PHP framework pro vývojáře totiž řeší celou řadu problémů. Příkladem mohou být různé optimalizační nástroje, DI[1] kontejner nebo komunikace s

[1]Dependency Injection – technologie vkládání závislostí

databází. Vývojář se tak může zaměřit na vývoj samotné logiky aplikace a nezabývat se příliš repetetivními a nízkoúrovňovými záležitostmi.

Bezpečnost je také jedním z hlavních bodů, kterým se PHP Frameworky věnují. Poskytují totiž obalovací objekty nad nízkoúrovňovými funkcemi a v nich sami vytváří bezpečností vrstvu. Pomocí frameworku prakticky bez námahy chráníte svoji aplikaci proti různým typům útokům.

Součástí frameworků bývají šablonovací sytém, který je možno při vývoji použít. Případně může vývojář zvolit jiný, který mu vyhovuje. Šablonovací enginy slouží pro prezentaci dat z backendové části aplikace a tvoří tak HTML stránku, kterou vidí uživatel. Některé šablonovací systémy obsahují vlastní funkce například pro úpravu textu, podmíněné vykreslování nebo cykly, tj. struktury, které známe z jiných programovacích jazyků. Zde však slouží pro zmíněnou prezentaci dat.

3.2.2 Vybrané frameworky ke srovnání

V dnešní době již několik spousta spolehlivých frameworků, které mohou být použity. Celosvětové žebříčky popularity dlouhodobě vedou frameworky Symfony a Laravel. Oba frameworky jsou mírně rozdílné a rozhodnutí, který nasadit spočívá v druhu použití. V českém prostředí jim konkuruje Nette.

Laravel

Framework Laravel je možné považovat za 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.

Symfony

Tento framework se vyznačuje zakládáním si na striktním dodržování PHP standardů a snaží se maximálně využívat různé návrhové vzory. Díky tomu jsou komponenty frameworku robustnější, což může znamenat větší časovou náročnost, na druhou stranu 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 je 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 potvrzuje jeho flexibilitu.

Nette

Český projekt Nette není tolik světově známý, ale v České republice je velice populární. Kombinuje totiž výhody obou předchozích frameworků a dosahuje velmi dobrých výsledků. Je distribuován po různých volitelných modulech. Součástí těchto modulů je i ladička[2] Tracy, která poskytuje rozsáhlé ladící možnosti. Svoji popularitu si nadále udržuje i díky velmi aktivní komunitě.

3.2.3 Srovnání

Jedná se o frameworky, které jsou si velmi podobné a prostý výčet nabízených funkcí je takřka podobný. V určitých drobnostech se však jejich použití liší.

Podobnosti

Všechny zmíněné frameworky používají při komunikaci s databází objektově relační mapování. To znamená, že výsledky z databáze nejsou vráceny jako pole hodnot, ale jako objekty, nad kterými lze volat další funkce. Díky tomu je možné dotazy na databázi i postupně optimalizovat a tím zrychlovat činnost aplikace.

V otázkách poskytování závislostí nabízejí všechny vlastní formu DI kontejneru.

Přikládané šablonovací enginy poskytují formu dědičnosti šablon. Umožňují rozšíření vlastními funkcemi a tak upravovat výpisy.

Rozdílnosti

Světově jsou frameworky Symfony a Laravel velice oblíbené a značně zastoupené. Z perspektivy České republiky je však Nette nejužívanější a v tomto ohledu značně překonává i zbylé dva porovnávané frameworky.

Učební křivka frameworků se liší. Zatímco Symfony se řadí mezi ty náročnější, Laravel si zakládá na své jednoduchosti a Nette tvoří pomyslný střed.

[2]Ladící nástroj pro vylepšování aplikace

3.2.4 Výběr

Na vyvíjený projekt, který je rozsahem střední velikosti lze vhodně použít jakýkoli ze zmíněných frameworků. Výsledný výběr tak stojí na zkušenostech a osobních preferencí vývojáře.

Z důvodu toho, že vývojář i tester se již setkali s frameworkem Nette a mají s ním alespoň začátečnické zkušenosti, byl vybrán framework Nette.

3.3 Front-end

3.3.1 HTML a CSS

Základem webových aplikací je způsob zobrazování. Webové technologie nabízejí tvorbu elementů pomocí formátu HTML[3] založeného na SGML[4]. Vlastností tohoto formátu je tvorba webových elementů pomocí tagů, kterými se specifikují jejich vlastnosti. Dnes již naprostá většina prohlížečů zobrazuje webové stránky pomocí tohoto formátu a je prakticky synonymem pro internetové stránky [2].

Další obdobně oblíbenou technologií je stylovací kaskádový jazyk CSS[5]. 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 společného 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 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]Hyper Text Markup Language – Značkovací jazyk pro tvorbu webových technologií
[4]Standard Generalized Markup Language – Obecně standardizovaný značkovací jazyk
[5]Cascading Style Sheet – technologie pro úpravu vzhledu nejen webových stránek

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 Bootstrap 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 již vytvořených stylů.

3.4 Databáze MariaDB

Databáze je důležitá součást aplikace, 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 projektu MySQL poté co jej společnost Oracle odkoupila. MariaDB staví na své značné kompatibilitě s databází MySQL, která je považována podle 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 nástroj MySQL Workbench, který pro většinu případů poskytuje stejné funkce jako pro databázi MySQL.

Byla zvolena databáze MariaDB díky tomu, že se jedná o otevřený projekt spravovaný nezávislou skupinou. Oproti tomu distribuování MySQL zdarma může být z důvodu korporátních vlivů v budoucnu omezeno. Je však nutno podotknout, že přechod z MariaDB na MySQL je většinou velmi snadný a bezproblémový. Je tak možno prohlásit podporu aplikace i pro 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 a spolehlivost.

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

Další důležitou funkcí Git je větvení. V základu je v repozitáři hlavní větev „main“ nebo „master“, která obsahuje centrální vývoj projektu. Od této větve 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 účty, nemusí se tedy organizačně řešit vstup na novou platformu.

Z těchto důvodů jsme se v naší volbě verzovacího systému rozhodli zvolit právě variantu GitLab 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, který navíc zajišťuje základní formátování.

V této dokumentační 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.

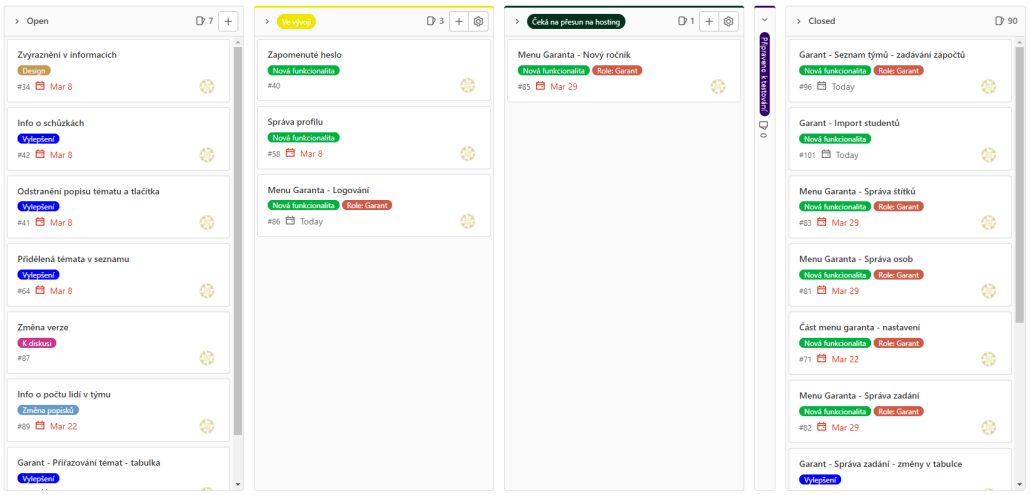
V systému GitLab tyto issues používáme také ve funkci Board, který využívá principy kanbanu.

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) nalezených 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 toto zadání k mentorování. Po zvolení se automaticky přidělí danému mentorovi.

Po odsouhlasení zadání sborem mentorů se zadání stává tématem, které je možno zveřejnit studentům. Je tedy zobrazováno ve veřejném seznamu témat. Vedoucí týmů mohou o takové zadání projevit téma hodnotou na pětibodové stupnici vážnosti.

Zadání se skládá z částí:

- název zadání
- stručný popis zadání
- PDF soubor s podrobným zadáním
- doporučovaná velikost týmu (3–7 lidí)
- časová náročnost řešení v člověkohodinách
- 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 vytvořen 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 zejména vedoucí týmu včetně přidávání studentů do týmu ze seznamu volných studentů zapsaných na předmět.

Vedoucí týmu je zodpovědný za obsah, který do PSTSP vkládá.

4.1.3 Projekt

Projekt je propojením instance tématu a týmu. Obsahuje dvě základní části. Data o postupu řešení projektu týmem a informace o projektu samotném.

Data o postupu řešení projektu jsou rozděleny na tři sekce. Plnění minimálních kritérií, plnění postupu projektu a záznamy o využitých zkušenostech. Všechny záznamy v těchto sekcích mají dva stavy. Stav, který určil vedoucí týmu a stav, který potvrdil mentor projektu. Na základě potvrzení mentora se určuje opravdový oficiální stav řešení projektu.

Druhá část ohledně údajů o projektu obsahuje kromě informací o týmu a samotném tématu vedoucím definované a vyplněné odkazy například na různá úložiště nebo stránky řízení projektu. Dalšími daty jsou údaje o udělení zápočtů a proběhlých obhajobách.

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 dílčí specifikace požadavků byly sestaveny případy užití, které popisují práci s výslednou aplikací. Rozdělení případů užití je ilustrováno v use-case diagramu (viz obrázek 4.1). Následuje seznam popsaných případů užití s popisem funkcí, které pokrývají.

4.2.1 Nepřihlášený uživatel

Přihlášení (UC.01)

Většina aplikace je přístupná pouze uživatelům dle jejich role. To vytváří potřebu autentizace v aplikaci. Uživatel má možnost se přihlásit využitím přihlašovací stránky pomocí registrované kombinace přihlašovacího jména a hesla. Pokud je uživatel autentizován, je přesměrován na domovskou obrazovku podle jeho konkrétní role.

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.

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

Neúplné týmy (UC.04)

Nepřihlášený uživatel má k dispozici seznam neúplných týmů, které hledají nové členy. U každého týmu lze zobrazit 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 obsahovat 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. Jedná se o stavy splněno, nesplněno, částečně splněno a nevhodné (pro konkrétní řešení není použitelné).

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 volných témat 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 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 „Témata bez mentora“ vedoucí na stránku s tabulkou témat, která nemají mentora. Tato 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, kdy 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 (UC.32)

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

Seznam týmů (UC.33–36)

Obdobná tabulka jako u pohledu mentora „Mentorovaná témata“, obsahuje tedy seznam projektů s jejich stavem řešení. 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 (UC.37–38)

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 uživatelů (UC.39–41)

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 (UC.42–43)

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 (UC.44–48)

Používané štítky může garant také upravovat, vytvářet nebo znemožnit další použití. V aplikace jsou dva druhy štítků. Týmové role, které vedoucí přiřazuje členům jeho týmu, a štítky pro profil týmu a tématu. Pokud garant pro jakýkoliv štítek nastaví, 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 (UC.49–50)

Speciální tabulka, kde každý sloupec reprezentuje tým bez přiřazeného tématu a každý řádek jedno volné téma. V tabulce jsou vyplněné jen 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 téma jednomu z přihlášených týmů.

Správa studentů (UC.51–55)

Na stránce správy studentů má garant k dispozici tabulku všech přidaných studentů. Z těchto studentů vedoucí vybírají 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 získaného exportem ze STAGu[2].

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 (UC.56–57)

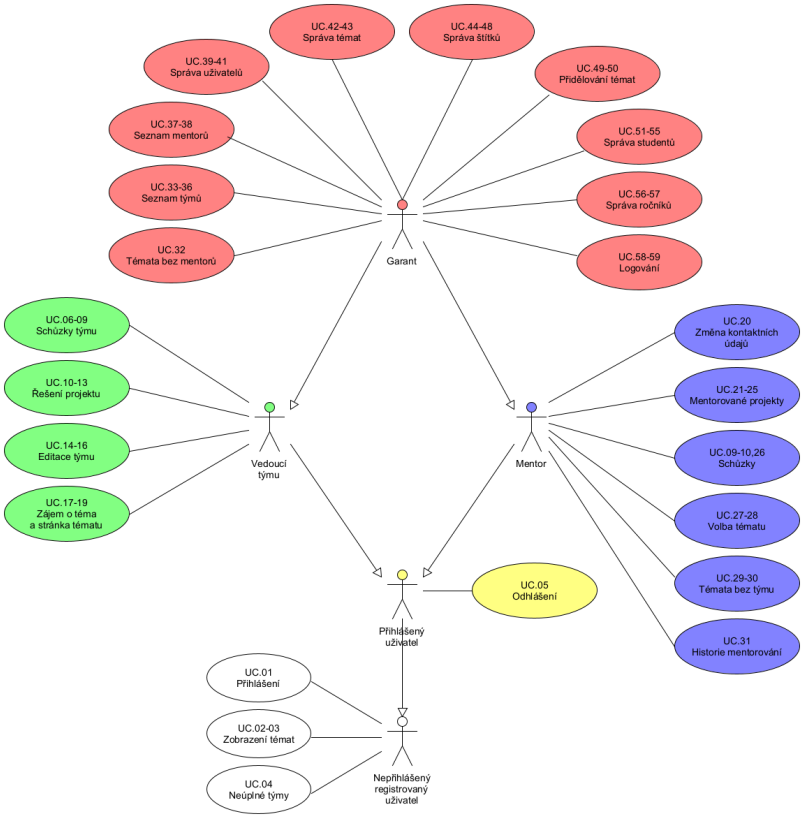
Garant ve správě ročníků může vidět informace o jednotlivých proběhlých ročnících. Zároveň může tvořit nové ročníky a nastavovat aktuální ročník TSP. Po zvolení nového aktuální cyklu TSP se původní cyklus považuje za historický a je tak i zobrazován ve zbytku aplikace. Tímto způsobem je řešeno zachování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

Logování (UC.58–59)

Pro sledování správné 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 a provádět s ním další operace.

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

[2] Informační systém studijní agendy



Obrázek 4.1: Schéma případů užití (zdroj: vlastní)

5 Architektura aplikace

5.1 Třívrstvá architektura

Architektura MVP[1] 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 umožňuje 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 (view). Ta se skládá z jednotlivých šablon a prezentace dat. Představuje tak rozhraní, přes které uživatel komunikuje s aplikací.

Mezi další velice rozšířenou architekturou patří MVC architektura, ze které MVP struktura vychází. Hlavním rozdílem mezi MVP a MVC architekturami je způsob zpracovávání vstupů od uživatele [8].

Rozdělení aplikace do několik vrstev je výhodné z důvodu nezávislosti jednotlivých částí. To zvyšuje efektivnost vývoje, kdy jednotlivé části aplikace mohou být vytvářeny téměř nezávisle.

5.2 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 představuje výhodu zejména při zpracování AJAX požadavků, kde je typicky výsledkem operace pouze část stránky, kdy není komponenta využita. Díky tomu se vůbec nevytváří a tím se šetří výkon serveru.

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

[1] Model-View-Presenter – třívrstvá architektura

5.2.1 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`.

5.3 Dependency Injection v Nette

Vkládání závislostí je technika programování vedoucí k čistému a přehlednému kódu. Princip tohoto způsobu programování spočívá v předávání zdrojů (služeb), které konkrétní funkce nebo třída vyžaduje, aby splnila svoji úlohu. Cílem je zamezit získávání zdrojů přímo v daném objektu, kdy programátor bez dodatečné informace neví s jakými zdroji informací systém pracuje a jak je případně získává. 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. Dependency 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].

5.4 Formuláře v Nette

Formuláře jsou důležitou součástí webového systému. Jedná se o jednu z možností jak získat uživatelský vstup. Zároveň ale představují bezpečnostní riziko, které je nutno ošetřit. Typů útoků skrz uživatelský vstup přes formuláře je velké množství a jejich ošetření klade nemalé časové a znalostní nároky na programátora. Také i proto je tvorba a validace formulářů 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. Výhoda 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 použití obyčejné komponenty. 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].

5.5 Database Explorer

Vyvíjená aplikace je postavená na předávání dat mezi aplikací a databází. Proto je využita služba Nette Database Exploreru, který umožňuje vytvářet dotazy databáze bez potřeby psát kompletní SQL příkazy. Je zde využíváno techniky objektově relačního mapování, kdy není výsledkem dotazu pole hodnot, ale objekty, nad kterými lze volat další funkce (viz ukázka kódu 2).

```
$user = $explorer->table('USER')->get($id);
$address = $explorer->table('ADDRESS')->where('id', $user->id)
    ->fetch();
$address->update([
    'street' => 'New Street',
    'city' => 'City',
]);
```

Ukázka kódu 2: Ukázka použití Nette Database Explorer

Databáze nám vrací objekt typu `ActiveRow` reprezentující záznam výsledku dotazu nebo `Selection` představující výběr. Pomocí instancí typu `ActiveRow` lze přistupovat k datům jednotlivých sloupců záznamu. Naproti tomu u objektu `Selection` je jeho hlavním účelem možnost iterování přes záznamy a získat tak instance `ActiveRow`.

Použití Database Exploreru poskytuje nativně i ochranu proti různým typům útokům zejména před SQL Injection. Zároveň se za programátora snaží prvotně optimalizovat dotazy pro zvýšení efektivnosti a tím i rychlosti.

5.6 Testovatelnost aplikace

5.6.1 Testování pomocí Selenia

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.

Selenium je díky tomu schopno nalézt požadovaný prvek stránky. Takto jednoznačně identifikovatelné není potřeba mít pouze akční prvky typu odkazu nebo tlačítka, ale také i další prvky stránky. Ty Selenium 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.

5.6.2 Jednotkové testování

Princip jednotkového testování je ověření nejelementárnějších procedur aplikace. V tomhle ohledu se činnost vyvíjené webové aplikace skládá z většinové části pouze z interpretace dat z a do databáze voláním funkcí Nette frameworku. Tak 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 několik funkcí byly vytvořeny jednotkové testy, které ověřují jejich výstupy.

5.6.3 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 může být použita i pro ladění, ale v našem případě máme jiné ladící nástroje [3].

Pro PHP existuje profesionálně používaná knihovna Monolog[2]. Ta je 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, dodatečné 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. Veškeré tyto a další parametry lze změnit v konfiguraci aplikace a upravit tak logování dle zjištěných reálných potřeb systému.

5.7 Vícejazyčnost

Mezi požadavky na aplikaci je podpora vícejazyčného prostředí. Zejména byla vyžádána možnost přepnout rozhraní do anglického jazyka.

Pro splnění tohoto požadavku byl implementován balíček pro implementaci překladů `kdyby/translation`. Balíček umožňuje vkládat do šablon a kódu řetězce dle jejich identifikátoru ze souborů s překlady. Vždy se tak použije řetězec ze souboru dle aktuálně nastaveného jazyka.

Soubory s překlady jsou uloženy ve formátu NEON ve vlastním adresáři. Každé jazykové podání je obsaženo v samostatném souboru. Hodnoty jed-

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

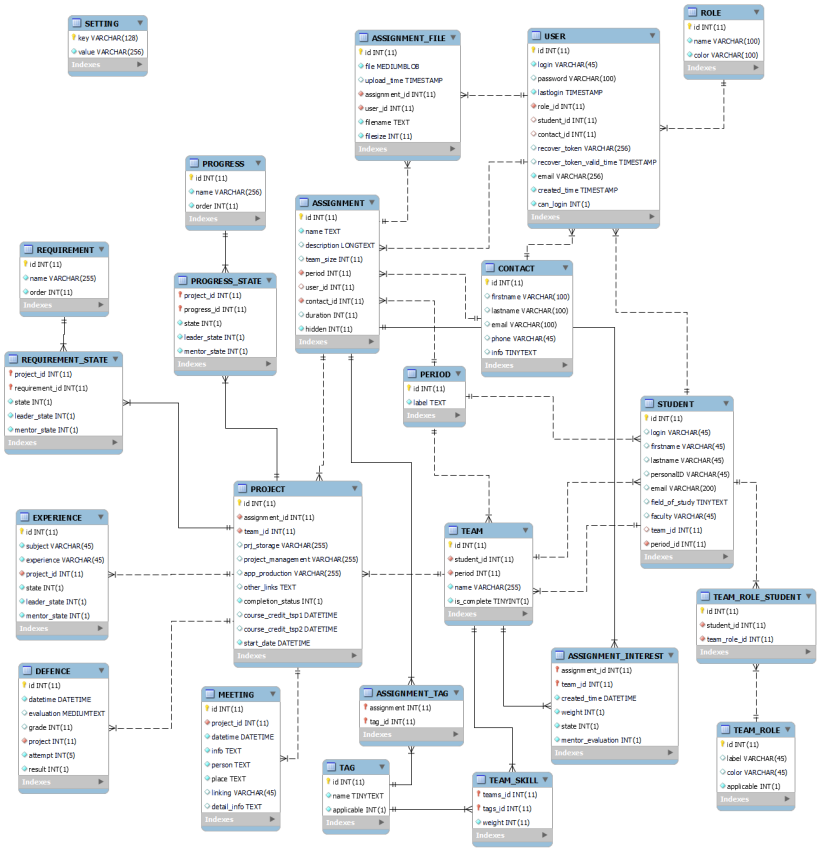
notlivých překladů jsou uspořádány v hierarchické struktuře. Pro vizuální i logické oddělení tak lze větvit strukturu dle částí aplikace, ve kterých jsou překlady řetězce obsaženy.

5.8 Databázová struktura

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

Podle požadavků byl sestaven následující model 23 tabulek, jejichž strukturu lze vidět v diagramu (viz obrázek 5.1).



Obrázek 5.1: Schéma databázové struktury (zdroj: vlastní)

5.8.1 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.
- `MEETING` – tabulka pro evidenci schůzek týmů

5.8.2 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 ročníků aplikace a předmětů TSP
- `ROLE` – tabulka dostupných rolí uživatelů aplikace

5.8.3 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 tabulek `TEAM_ROLE` a `STUDENT`. Tabulka slouží pro zaznamenání přiřazení rolí jednotlivým členům týmu.
- `TEAM_SKILL` – propojení tabulek `TAG` a `TEAM`. Tabulka specifikuje jednotlivé profily týmů.
- `ASSIGNMENT_TAG` – propojení tabulek `TAG` a `ASSIGNMENT`. Tabulka specifikuje jaký profil týmu je předpokládán pro tým, který bude zadání řešit.
- `PROGRESS_STATE` – propojení tabulek `PROGRESS` a `PROJECT`. Tabulka eviduje postup v řešení projektu. Relace uchovává stavovou hodnotu daného přepínače.
- `REQUIREMENT_STATE` – propojení tabulek `REQUIREMENT` a `PROJECT`. Tabulka eviduje plnění kritérií při řešení projektu. Relace uchovává stavovou hodnotu daného přepínače.

- `EXPERIENCE_STATE` – propojení tabulek `EXPERIENCE` a `PROJECT`. Tabulka eviduje stav potvrzení využitých zkušeností. Relace uchovává stavovou hodnotu daného přepínače.

5.8.4 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.

5.9 Uživatelské rozhraní

Důležitou částí aplikace je uživatelské rozhraní. Pomocí něj je totiž aplikace ovládána a jedná se o pohled, kterým uživatel pracuje s aplikací. Zjednodušeně lze říct, že nekvalitně vytvořené uživatelské rozhraní může ohrozit příjemný uživatelský zážitek z používání aplikace a znepříjemnit práci se systémem. Návrhu uživatelského prostředí je tedy potřebné věnovat značnou pozornost.

Návrh rozhraní rozděluje zobrazovací prostor na tři hlavní části. Horizontální navigační lišta, která obsahuje hlavní rozcestník na důležité stránky vztahující se k předmětu (např. CourseWarová stránka předmětu), údaje o přihlášeném uživateli a nastavení jazykové předvolby. Tato navigace je navržená tak, že je vždy přístupná. Pod ní se zobrazují hlášky aplikace například o úspěšně provedených akcích. Na levé straně straně zobrazovací plochy je umístěna hlavní (vertikální) navigace systému. Obsahuje odkazy na stránky, ke kterým má aktuální uživatel přístup. Poslední a hlavní částí je část obsahu, která zobrazuje obsah jednotlivých stránek. Tato stránka pro orientaci v aplikaci vždy obsahuje i název stránky.

Při návrhu je potřeba klást důraz na jednotnost. Uživatel by měl mít shodný způsob používání napříč aplikací.

5.9.1 Responzivita

Responzivní desing spočívá v přizpůsobení zobrazení obsahu šířce okna webového prohlížeče (příp. velikosti obrazovky zařízení). To znamená, že uživatel navštěvující 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 v aplikaci responzivní design, pokud je dodržena správná HTML struktura a jsou důsledně aplikovány styly použitím tříd stylů.

V aplikaci jsou jednotlivé celky umístěné do tzv. karet. Díky tomu je možné je snadno přemisťovat jako ucelené balíčky uspořádané do sloupců a řádků. Například při zobrazení na malém mobilním zařízení jsou veškeré karty uspořádány do jednoho sloupce, kdy je podstatné, aby ty důležitější karty byly zobrazovány navrchu.

5.10 Návrh loga

Aplikace pro svoji vizuální komunikaci a rozeznatelnost potřebuje logo.

Vývoj loga vychází z názvu projektu „Podpůrný software TSP“ respektive z jeho zkratky „PSTSP“. Bylo využito skutečnosti, že zkratka názvu je palindromem[3], a vlastnosti osové souměrnosti písmena „T“.

Počáteční návrh loga (viz obrázek 5.2) spočívá v zrcadlení části „PS“ resp. „SP“, které jsou „zastřešeny“ velkým písmenem „T“, kdy jeho vertikální čára působí jako zrcadlo.

Finální verze (viz obrázek 5.3) vychází z prvotního návrhu. Celé logo bylo vloženo do šestiúhelníkového rámu s přidanou perspektivou. Písmena k sobě přiléhají a kvůli tvarům písmen a rámu jsou hranaté. Barevná paleta byla vybírána s ohledem na barvy univerzity (modrá) a fakulty (zlatá). V celkovém pojetí logo připomíná výraznou šipku symbolující posun se zlatavým ohonem a bílým vzduchem. Logo tedy symbolizuje rychlý pokrok vpřed.

[3]Palindrom je slovo, které lze číst zleva i zprava stejně

Obrázek 5.2: Prvotní návrh loga (zdroj: vlastní)

Obrázek 5.3: Finální verze loga aplikace (zdroj: vlastní)

5.10.1 Použití loga

Logo je plánováno pro použití v samotném systému a hlavně pro použití jako favicon webové aplikace.

Otázka ikon webových stránek není úplně triviální, protože je potřeba definovat jednotlivé obrázky a další údaje pro různé velikosti zobrazení na různých platformách. Typický příklad je ikonka v prohlížeči, kde se použije rozlišení 64×64 px, na rozdíl od mobilního telefonu s moderním operačním systémem s ikonkou stránky na ploše, kdy se použije podstatně větší rozlišení s definovanou barvou pozadí.

Pro vyřešení této otázky byl nalezen a použit nástroj RealFaviconGenerator[4], který z jednoho vstupního obrázku umožní sestavit si vlastní nastavení faviconů pro různé platformy. Výsledkem tohoto procesu jsou prvky HTML hlavičky, které stačí implementovat do práce.

Díky tomuto nástroji máme rychle zajištěnou kompatibilitu zobrazení ikonky webu.

[4]Dostupné na https://realfavicongenerator.net/

6 Realizace

Během realizace byly využívány postupy a technologie uvedené v předchozích kapitolách.

6.1 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
    - `Bootstrap.php` – zaváděcí třída Bootstrap
  - `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

6.2 Konfigurace aplikace

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

6.2.1 Konfigurace databázového spojení

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

```
database:
   dsn: 'mysql:host=127.0.0.1;dbname=test'
   user: root
   password: password
```

Ukázka kódu 3: Konfigurace databázového spojení

V nastavení databázové struktury (viz ukázka kódu 3) nastavujeme položku `dns` jako textový řetězec obsahující typ připojované databáze, adresu serveru (případně s číslem služby) a název výchozího zvoleného schématu v databázi. Dále se nastavuje uživatelské jméno a heslo pro připojení.

6.2.2 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.

6.3 Vylepšování aplikace

Již implementované funkce jsou v aplikaci stále vylepšovány. Nejčastěji z důvodu nalezených defektů a následujících případech.

6.3.1 Úpravy dle aktualizování specifikace

V průběhu vývoje byla specifikace a požadavky několikrát upravovány. To vyžadovalo obvykle úpravy na již dokončených částí aplikace. Aktualizace specifikace pramenily většinou z diskusí garantů předmětu a budoucích mentorů, kteří následně přehodnotili postoje k původně požadovaných funkcím.

6.3.2 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 vytváří zcela nový modul aplikace nebo pouze rozšiřují 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.

6.4 Požadavky na zavedení aplikace

Pro úspěšnou instalaci projektu na webovém serveru je potřeba splnit požadavky pro běh programu. Nejdůležitější požadavek pro spuštění je mít nainstalovaný webový server s modulem PHP podporující verzi PHP minimálně 7.2. Další požadavky se shodují s požadavkami frameworku Nette ve verzi 3.1. Nette však již spolehlivě běží na každé běžné PHP instalaci a zrušil tedy nástroj Requirement Checker.

Případné balíčky aplikací jsou pr̈ibalené s aplikací nebo je lze získat pomocí nástroje Composer a přiložené definice závislostí.

Dále je také požadována přístupná databáze MySQL nebo MariaDB pro trvalé úložiště dat.

Pro korektní odesílání emailových zpráv je potřeba také definovat SMTP server, přes který se má korespondence odesílat.

7 Testování

Jedním z hlavních požadavků na výslednou aplikaci je její spolehlivost a robustnost. Tomu je podřízeno provedené rozsáhlé testování. Tvorba testů však není součástí této práce a je prováděna paralelní bakalářskou prací „Komplexní testy webové aplikace“ psanou Davidem Kůtou.

7.1 Testování aplikace

Aplikace byla připravována na různé druhy testů (viz Testovatelnost aplikace). Tyto testy byly vytvářeny současně při tvorbě aplikace, přičemž byl systém průběžně testován.

7.1.1 Jednotkové testování

Ačkoli jednotkové testování v případě naší aplikace je velice omezené, je vytvořeno množství testů pro funkce pomocné třídy `Utils`. Tyto testy ověřují správnost výsledků funkcí na základě různých vstupních dat.

7.1.2 Automatické testování

Byly sestaveny automatické testy aplikace pomocí Selenium WebDriver Pro testování je automatickými testy kompletně dokončeno pokrytí rolí Vedoucí a Mentor. Pro roli Garant jsou dokončeny významné části přímo souvisejících s organizací předmětů.

Pro účely automatického testování byl připraven stav aplikace, kdy se v datech systému vyskytuje většina možných stavů aplikace. Tento stav lze nahrát pomocí tlačítka „ResetDB“ umístěného v liště horizontální navigace. Toto tlačítko přístupné pouze v testovacím režimu aplikace.

7.1.3 End-to-end testy

Součástí testovací strategie jsou i testy, které otestují činnost aplikace v sekvencí scénářů. Scénáře jsou vytvořeny dle předpokládaného způsobu užití aplikace.

7.1.4 Logování

Aplikace si také pro účely testování zaznamenává data o provedených akcích. Při testování tedy možné sledovat posloupnost akcí, které se prováděly a které selhaly. Díky tomu lze jednoduše vysledovat příčinu nebo její přibližný výskyt.

7.1.5 Manuální testování

Některé funkce systému nebylo možné nebo přínosné testovat automaticky pomocí Selenium WebDriver. Typickým příkladem těchto částí aplikace je stahování a odesílání emailů. Proto tyto činnosti byly otestovány manuálně v různých případech v různých částech aplikace.

7.2 Výsledky testování

Po zahájení testování se objevila celá řada selhání. Většina nalezených defektů byla způsobena nedokončeným ošetřením přístupu na jednotlivé stránky nebo dat při práci s databází. Veškeré defekty však byly původem implementačního rázu. Jednalo se tak o odklad implementace od návrhu a dosud nebyl nalezen defekt, který by byl způsoben chybou návrhu aplikace.

Nalezené odchylky od specifikace byly okamžitě po nalezení testerem komunikovány s vývojářem a buď byla chyba na místě opravena nebo byl vytvořen záznam v nástroji pro řízení projektu za účelem prošetření a případné opravy.

8 Závěr

V bakalářské práci byla navržena a implementována webová aplikace, pro jejíž vývoj byl využity frameworky Nette a AdminLTE. Aplikace je středního rozsahu a svojí komplexností poměrně výrazně přesáhla původní odhady. Bylo vytvořeno ?? PHP souborů a dalších ?? pomocných souborů, o celkové velikosti ??? KB zdrojového kódu. Aplikace dává k dispozici celkem ?? různých webových stránek. Při práci byly dodržovány všechny dobré postupy běžné v současném SW vývoji, jako např. použití verzovacího systému, použití systému pro řízení projektu, použití systému pro správu incidentů. Při vývoji byl kladen zvýšený důraz na robustnost, modularitu a testovatelnost aplikace. Vlastní testy ale byly prováděny v rámci jiné bakalářské práce. Pozitivní význam důrazu na výše zmíněné vlastnosti se plně projevil již v průběhu vývoje, neboť prototyp aplikace byl již v prosinci 2021 podroben oponentuře budoucích uživatelů. A na jejím základě vznikly četné požadavky na úpravy, změny a doplnění. Díky výše zmíněným vlastnostem aplikace bylo možné tyto požadavky beze zbytku provést a otestovat. Vyvinutá aplikace svojí realizovanou funkcionalitou splňuje všechny požadavky na nasazení v předmětu KIV/TSP1.

V současné době je aplikace nasazena na vývojovém prostředí. Úplné dokončení aplikace, tj. její nasazení na katedrální server a doladění všech přístupových práv, je plánováno v období srpen až září 2022 tak, aby již v ZS akademického roku 2022/23 mohli aplikaci využívat vyučující i studenti předmětu KIV/TSP1.

8.1 Další možný vývoj

Aplikace má mnoho potenciálu jak v univerzitním prostředí růst. Jedním z další možných rozšíření je implementace přihlášení skrz protokol Kerberos univerzitní sítě a umožnit tak jednotné přihlášení do aplikace. To by mohlo znamenat i možnost nové role Student, jejíž držitelé by si mohli prohlížet aktuální stav svého týmu a projektu a případně kontrolovat akce svého vedoucího.

Dalším rozšířením aplikace je přidat podporu pro netypické případy studia předmětů TSP (např. studium pouze jednoho z předmětů).

V neposlední řadě také byl předložen nápad do budoucna rozšířit software i do dalších předmětů podobného zaměření (KIV/ASWI, KIV/ZSWI, atd.).

Seznam obrázků

Seznam ukázek kódu

Seznam použitých zkratek

PHP Hypertextový preprocesor

HTML Hyper Text Markup Language

SGML Standard Generalized Markup Language

CSS Cascading Style Sheet

PDF Portable Document Format

CSV Comma-separated values

UC Use Case

DI Dependency Injection

MVC Třívrstvá architektura Model-View-Controller

MVP Třívrstvá architektura Model-View-Presenter

SQL Structured Query Language

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`.

A Případy užití

Případy užití jsou sestaveny v originální XML struktuře následně vygenerované do čitelné HTML podoby. Následující obrázek ?? představuje ukázku těchto vygenerovaných výstupů. Veškeré sepsané případy užití jsou součástí obsahu odevzdané práce. Případy jsou očíslovány dle tabulky ??.

Číslo UC	Popis	Role
UC.01	Přihlášení	
UC.02		
UC.02		
UC.03		
UC.04		
UC.05		
UC.06		
UC.07		
UC.08		
UC.09		
UC.10		
UC.11		
UC.12		
UC.13		
UC.14		
UC.15		
UC.16		
UC.17		
UC.18		
UC.19		
UC.20		
UC.21		
UC.22		
UC.23		
UC.24		
UC.25		
UC.26		
UC.27		
UC.28		
UC.29		
UC.30		
UC.31		
UC.32		
UC.33		
UC.34		
UC.35		
UC.36		
UC.37		
UC.38		
UC.39		
UC.40		
UC.41		
UC.42		
UC.43		
UC.44		
UC.45		

B Obsah CD