FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Systém pro správu ročníkových prací
Student:	Max Hejda
Vedoucí:	Ing. Jiří Hunka
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Webové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Cílem této práce je umožnit pohodlnou a efektivní správu životního cyklu ročníkových prací.

Postupujte v těchto krocích:
Proveďte analýzu konkurence a srovnání s existujícími nástroji, které řeší podobný problém.
Udělejte sběr a analýzu požadavků budoucích uživatelů. Jako referenční zdroj využijte Gymnázium Jiřího Gutha-Jarkovského.
Analyzujte možnosti realizace budoucího systému.
S přihlédnutím k požadavkům uživatelů navrhněte vhodné a udržitelné řešení.
Návrh řádně implementujte (například rozšíření existujících systémů či implementaci vlastního řešení).
Dbejte na univerzálnost, ale také na jednoduchost pro uživatele.
Výsledek řádně otestujte.
Zhodnoťte výsledky a uveďte další možná rozšíření či směřování projektu.

Elektronicky schválil/a Ing. Jaroslav Kuchař, Ph.D. dne 20. ledna 2022 v Praze.

Bakalářská práce

SYSTÉM PRO SPRÁVU ROČNÍKOVÝCH PRACÍ

Max Hejda

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: Ing. Jiří Hunka
11. května 2022

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Chtěl bych poděkovat především svému vedoucímu bakalářské práce Ing. Jiřímu Hunkovi z katedry softwarového inženýrství FIT ČVUT. Rovněž velké díky si zaslouží Bc. Tomáš Pavlůsek, na jehož bakalářské práci jsem stavěl a který ochotně zodpovídal moje dotazy. Dále bych zde rád zmínil Ing. Marka Mojžíše a jeho kolegy ze společnosti IT Profík s.r.o., jež má na starosti IT služby na Gymnáziu Jiřího Gutha-Jarkovského.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této svojí práce, včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, jež vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 11. května 2022 .....

Abstrakt

Bakalářská práce řeší digitalizaci správy ročníkových prací na středních školách. Cílem práce bylo vytvořit systém, který nahradí stávající správu ročníkových prací na Gymnáziu Jiřího-Gutha Jarkovského, jež je závislá na řadě osobních interakcí. Analýzou existujících systémů s podobnými funkcemi jsem došel k rozhodnutí rozšířit SOS – Studentský odevzdávací systém vytvořený pro potřeby Fakulty informačních technologií ČVUT.

Výsledkem je aplikace, do níž není třeba kromě zadání ročníkových prací a dat odevzdání jejich jednotlivých etap vkládat žádné údaje, neboť je hladce importuje ze systému Bakaláři, který gymnázium využívá ke správě údajů o studentech, zaměstnancích, předmětech, rozvrzích apod.

Klíčová slova ročníkové práce, SOS – Studentský odevzdávací systém, odevzdávací systém, systém pro podporu výuky, webová aplikace, Django, Bakaláři

Abstract

The subject matter of this bachelor thesis is digitalisation of seminary works at secondary schools. Its aim was to build a system that would replace the existing administration of seminary works at the Gymnázium Jiřího Gutha-Jarkovského secondary school which relies heavily on interpersonal interactions. Further to analysing existing systems with similar functions I decided to extend the SOS – Student Submission System created for the Faculty of Information Technology of the CTU in Prague. The result is an application into which only seminary work assignments and deadlines for submission of their individual stages need to be entered as all other data are seamlessly imported from Bakaláři – a system the secondary school uses for administration of data on students, staff, subjects, timetables, etc.

Keywords seminary works, SOS – Student Submission System, submission system, teaching support system, web application, Django, Bakaláři

Seznam zkratek

API	Application Programming Interface
CD	Continuous Deployment
CI	Continuous Integration
ČVUT	České vysoké učení technické
FIT	Fakulta informačních technologií
FP	funkční požadavek
FURPS	kategorizace požadavků na software
GJGJ	Gymnázium Jiřího Gutha-Jarkovského
IS	informační systém
IT	informační technologie
KOS	komponenta studium
LMS	Learning Management System
MVC	Model View Controller
MVT	Model View Template
NP	nefunkční požadavek
REST	Representational State Transfer
RP	ročníková/ročníkové práce
SaaS	software jako služba (Software as a Service)
UC	případ užití (use case)
WSGI	Web Server Gateway Interface

Úvod

Používání výpočetní techniky se pro většinu lidí stalo nedílnou součástí života, neboť řadu činností usnadňuje. Neexistuje-li ovšem cenově dostupné a z hlediska uživatelů jednoduché řešení, mnohé záležitosti se řeší stále v osobní rovině. Například na pražském Gymnáziu Jiřího-Gutha Jarkovského, jež mě inspirovalo k této bakalářské práci, se veškerá administrace ohledně ročníkových prací řeší převážně nevýpočetními prostředky. Jednotliví vyučující vedou na papíře agendu ohledně zadaných témat, přičemž kontrola, zda skutečně všichni studenti mají práci zadanou, obnáší koordinaci řady učitelů.

Jakožto absolvent tohoto gymnázia jsem zejména s ohledem na osobní vazby chtěl toto změnit a situaci pedagogům, ale i studentům ulehčit. Po dohodě s vedením školy a vedoucím bakalářské práce jsem se rozhodl vytvořit systém, který správu ročníkových prací zjednoduší a plně digitalizuje.

V rámci své práce analyzuji stávající řešení dané problematiky. Na základě této analýzy bylo mým úkolem buď navrhnout nový systém, nebo rozšířit nějaký již existující. Dalším úkolem je výsledný software implementovat a otestovat. Z důvodů podrobněji rozepsaných dále jsem se rozhodl rozšířit webovou aplikaci, která řadu požadavků kladených na systém pro správu ročníkových prací již splňuje.

Práce se tedy opírá o SOS – Studentský odevzdávací systém, jejž v rámci své bakalářské práce vyvinul v roce 2021 Tomáš Pavlůsek pod vedením Ing. Jiřího Hunky z katedry softwarového inženýrství. Rozšířený SOS je univerzálnější a využijí ho nyní jak vysoké, tak střední školy, přestože se požadavky na správu studentských prací u různých škol značně liší.

Kapitola 1
Analýza

V této kapitole čtenáře podrobněji seznámím se stávajícími postupy Gymnázia Jiřího-Gutha Jarkovského ohledně ročníkových prací a dále vysvětlím, co by měl nový systém pro jejich správu splňovat. Zde jsem vycházel z vlastních zkušeností absolventa tohoto gymnázia, z příručky k ročníkovým pracím, která je k dispozici na webových stránkách školy [1] a ze schůzek s odpovědnými zaměstnanci gymnázia. V části 1.3 analyzuji několik stávajících systémů z hlediska zjištěných požadavků a část 1.4 je věnována podrobnější analýze systému, který vyšel z hodnocení vítězně.

1.1 Současná správa ročníkových prací

Ročníkové práce (dále též RP) na Gymnáziu Jiřího-Gutha Jarkovského jsou odbornějšího či uměleckého rázu. Jedná se o zhruba sedmiměsíční projekt, kdy si student v říjnu vybírá téma, jemuž se bude věnovat až do konce dubna. Cílem je připravit studenty gymnázia na systematickou činnost během studia na vysoké škole, a proto se s postupujícími ročníky nároky na RP zvyšují. Většina vyučujících v daných ročnících vypisuje několik témat, nicméně vítána je také iniciativa samotných studentů. Pakliže navrhnou téma odpovídající náročnosti, vždy je jim vyhověno. Pravidla připouštějí i týmové ročníkové práce, ovšem po dohodě s příslušným vyučujícím. [1]

Celý průběh ročníkových prací podléhá harmonogramu, který stanovuje, do kdy si student musí zvolit téma, resp. vyplnit a předat svému třídnímu učiteli přihlášku k ročníkové práci, absolvovat konzultace a hotovou ročníkovou práci odevzdat.

Na začátku školního roku dostává student od třídního učitele vytištěnou přihlášku k ročníkové práci a tzv. studentský list. Zvolené téma student zapíše do přihlášky, kterou musí podepsat také vyučující, jenž bude téma studentovi konzultovat a práci následně hodnotit. Vyplněnou přihlášku odevzdá student zpět třídnímu učiteli do stanoveného termínu.

V dalších fázích následují dvě konzultace. První konzultace se zaměřuje na anotaci práce, která obsahuje hlavní zaměření práce a její předběžné členění spolu se základní rešerší. Při druhé konzultaci se hodnotí koncept práce, její členění na kapitoly a naznačený obsah příloh.

Nejzazší termíny konzultací jsou pevně stanoveny. Ke každé konzultaci musí student přinést svůj studentský list. Do něj konzultant zaznamenává datum konzultace, připomínky k práci, jakož i závěr konzultace, tj. hodnocení vyhověl/nevyhověl. Pokud se student na konzultaci dostaví, ovšem jeho započatá ročníková práce zatím nesplňuje podmínky pro úspěšné hodnocení konzultace, má nárok na opravný termín. Pakliže student list ztratí, konzultace je hodnocena jako nevyhovující. Stejné je to i v případě, kdy student na konzultaci nepřijde.

Na konci dubna student odevzdává fyzicky hotovou ročníkovou práci spolu se svým studentským listem. Konzultant práci ohodnotí, přičemž tento výsledek se promítá i do výsledné známky z předmětu, jenž se k ročníkové práci váže, za druhé pololetí. Například neodevzdání ročníkové

práce či její hodnocení jako zcela nevyhovující automaticky snižuje známku o stupeň.

Z uvedeného je zřejmé, že administrace ročníkových prací na gymnáziu se dvanácti třídami po cca 30 studentech, které mají v osnovách ročníkové práce, je náročná nejen pro třídní učitele a konzultující vyučující, ale zejména pro koordinátora – vyučujícího, jenž má ročníkové práce globálně na starosti. Navržený systém pro správu ročníkových prací tuto zátěž minimalizuje.

1.2 Požadavky na nový systém

Po provedení analýzy dosavadního postupu, jak je uvedeno v předchozí části, jsem se ve spolupráci s koordinátorem ročníkových prací na Gymnáziu Jiřího Gutha-Jarkovského zaměřil na shromáždění veškerých požadavků na nový systém. V příloze A je uveden zápis z jedné prvotní schůzky, kdy jsem ještě hodlal systém naprogramovat zcela od základu. Od některých požadavků bylo později upuštěno a bylo rozhodnuto soustředit se především na hlavní funkce.

Jedná se například o požadavek na nominaci RP na Zlatou Truhlu. Jde o celoškolní akci, na které jsou prezentovány a následně v závislosti i na hlasování všech studentů oceňovány ročníkové práce mimořádných kvalit. Digitalizovat tuto funkci nemá ovšem valný význam, jelikož v souvislosti se Zlatou Truhlou zatím neprobíhá elektronicky vůbec nic. Dále bylo upuštěno od konfigurace lhůt, do kdy musí student opravit zvolené téma nebo vlastní práci, a automatické přiřazení tématu studentům, kteří si nezvolili téma do vypršení termínu pro volbu tématu, neboť se účastníci schůzek neshodli na kritériích automatického přiřazování. Poslední z požadavků, jež nakonec nebyly implementovány, je kontrola délky odevzdávaného textu, protože bylo rozhodnuto, aby první verze systému pro správu ročníkových prací obsahovala pouze opravdu důležité funkce.

Ze schůzek vzešel následující seznam funkčních a nefunkčních požadavků.

1.2.1 Analýza požadavků

Analýza požadavků představuje první fázi vytváření jakéhokoliv systému. Slouží k vymezení hranice systému a k zachycení omezení, která jsou na systém kladena. V praxi též pomáhá klientovi a zadavateli vyjasnit a upřesnit zadání. U větších komerčních projektů se rovněž využívá ke kalkulaci ceny.

Správný požadavek musí být jednoznačný. Dále musí být splnitelný a nesmí být v rozporu s ostatními požadavky. Také musí být dodavatelem systému zajistitelný. Rovněž je potřeba, aby požadovanou funkcionalitu nebo vlastnost systému bylo možné otestovat. [2]

Požadavky se dělí na dvě obecné kategorie: funkční požadavky a nefunkční požadavky. Funkční požadavky popisují jednotlivé funkce systému, zatímco nefunkční požadavky popisují ostatní náležitosti systému, například user experience, technické parametry atd.

Základní rozdělení je ovšem dost obecné. Přesnější je kupříkladu klasifikace FURPS [3], kterou v práci dále využívám:

- Functionality – funkční požadavky
- Usability – jak systém působí na uživatele, jak se používá, dokumentace atd.
- Reliability – dostupnost systému, četnost a závažnost chyb/výpadků
- Performance – rychlost odezev systému a jeho technické parametry
- Supportability – údržba a podpora aplikace, dále rozšiřitelnost systému nebo přizpůsobení novým potřebám

1.2.1.1 Funkční požadavky

V této části uvádím finální seznam požadavků na nový systém pro správu ročníkových prací, jak byl dohodnut s koordinátorem ročníkových prací na základě řady schůzek. U každého požadavku je uvedena důležitost. Tento parametr určuje, jak je daný požadavek pro zadavatele důležitý. Funkční požadavky s nízkou důležitostí zatím nebudou implementovány.

- **FP1 – Import dat**
  Zásadní ulehčení pro vyučující představuje možnost importace dat uživatelů, předmětů a dalších potřebných údajů pro evidenci a hodnocení ročníkových prací.
  Důležitost: vysoká
  Náročnost: střední

- **FP2 – Konfigurace požadavků pro jednotlivé ročníky**
  Požadavky na ročníkové práce se pro každý ročník trochu liší, proto je potřeba tyto informace nějak konfigurovat.
  Důležitost: vysoká
  Náročnost: vysoká

- **FP3 – Definice témat**
  Každá práce má zadaní, jež vytvoří student nebo vyučující (eventuálně oba ve vzájemné spolupráci).
  Důležitost: vysoká
  Náročnost: nízká

- **FP4 – Konfigurace konzultací**
  Ročníkové práce obnášejí dvě konzultace, přičemž kritéria pro úspěšné splnění každé z nich se liší. Mezi tyto požadavky patří například termín, do kdy musí být splněna, rozsah odevzdaného textu nebo členění práce.
  Důležitost: vysoká
  Náročnost: nízká

- **FP5 – Výběr tématu**
  Student nebo skupina studentů si vybírá téma (buď vlastní, nebo již vytvořené a zadané do systému).
  Důležitost: střední
  Náročnost: střední

- **FP6 – Schvalování zadaní, hodnocení konzultací a práce**
  Konzultující učitel schvaluje zadání, hodnotí konzultace a nakonec i samotnou práci. Tyto kroky musí být možné v systému zaznamenat.
  Důležitost: vysoká
  Náročnost: střední

- **FP7 – Odevzdávání řešení**
  Student bude prostřednictvím systému odevzdávat jednotlivé fáze své práce (podklady pro konzultace a výslednou práci).
  Důležitost: vysoká
  Náročnost: střední

- **FP8 – Nominace na Zlatou Truhlu**
  V tabulce hodnocení bude možné ohodnocenou práci nominovat na Zlatou Truhlu.
  Důležitost: nízká
  Náročnost: nízká

- **FP9 – Kontrola délky odevzdávaného textu a její limitace**
  Důležitost: nízká
  Náročnost: střední

1.2.1.2 Nefunkční požadavky

Nefunkční požadavky vycházejí především z požadavku gymnázia, aby se jednalo o bezpečnou webovou aplikaci nezávislou na době, kdy jsou vyučují a studenti ve škole.

- **NP1 – Dostupnost**
  Systém je dostupný nepřetržitě.
  Důležitost: vysoká
  Náročnost: střední
  Kategorie: spolehlivost

- **NP2 – Zabezpečení**
  Do systému a k jeho datům mají přístup pouze studenti a vyučující dané školy.
  Důležitost: vysoká
  Náročnost: střední
  Kategorie: spolehlivost

- **NP3 – Responzivita**
  Alespoň některé úkony lze provádět na mobilních zařízeních.
  Důležitost: střední
  Náročnost: střední
  Kategorie: použitelnost

Poslední nefunkční požadavek jsem přidal sám, jelikož v dnešní době by mělo být možné každý systém alespoň omezeně ovládat pomocí mobilního zařízení.

1.2.2 Případy užití

Ze schůzek s koordinátorem ročníkových prací a z již zmiňované příručky [1] také vyplynuly role jednotlivých uživatelů a případy užití, jež bude systém muset umožňovat. Případy užití reprezentují, jaké úkony budou jednotliví aktéři potřebovat v systému vykonávat.

1.2.2.1 Seznam aktérů

Uživatelé systému pro správu ročníkových prací se dělí na dvě základní kategorie: vyučující a student. Role vyučujícího se dále dělí na dvě podkategorie, a to na roli správce ročníku a roli konzultanta, přičemž vyučující vůbec nemusí být v daném školním roce konzultantem. Vzhledem k tomu, že takový případ se bude v systému řešit tak, že dotyčný vyučující nevypíše žádné téma, není třeba této variantě přiřazovat samostatnou roli.

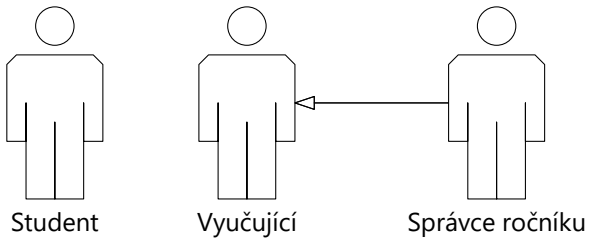
V seznamu níže jsou uvedeny jednotlivé role a jejich stručný popis.

- Student – uživatel studující v některém ročníku gymnázia. Dle konfigurace daného ročníku si vybírá zadaní z daného okruhu předmětů, případně si může téma vymyslet sám a požádat o jeho schválení a konzultování některého z vyučujících, kteří učí tematicky odpovídající předmět.

- Vyučující – uživatel vyučující některý z předmětů povolených pro ročníkové práce. Vymýšlí zadání, konzultuje práci studentům, kteří si zvolili některé z jeho témat, a následně tuto práci hodnotí.

- Správce ročníku – může provádět stejné akce jako vyučující. Dále je zodpovědný za konfiguraci ročníkových prací pro daný ročník, tj. nastavení povolených předmětů, jestli jsou povoleny týmové práce, zda si učitelé mohou upravit některá nastavení apod.

Vazby mezi jednotlivými rolemi jsou znázorněny na obrázku 1.1. Dříve zmiňovaný koordinátor ročníkových prací je jedním ze správců ročníku, přičemž se bude starat o chod celého systému.



Obrázek 1.1 Diagram aktérů

1.2.2.2 Případy užití

- UC1: Zobrazení zadání ročníkové práce a kontrolních bodů
  Aktér: Student nebo Vyučující

- UC2: Odevzdání vypracované ročníkové práce či její dílčí části
  Aktér: Student

- UC3: Úprava řešení (pokud již není ohodnoceno)
  Aktér: Student

- UC4: Zobrazení ohodnocené ročníkové práce či její dílčí části
  Aktér: Student nebo Vyučující

- UC5: Ohodnocení ročníkové práce či její dílčí části
  Aktér: Vyučující

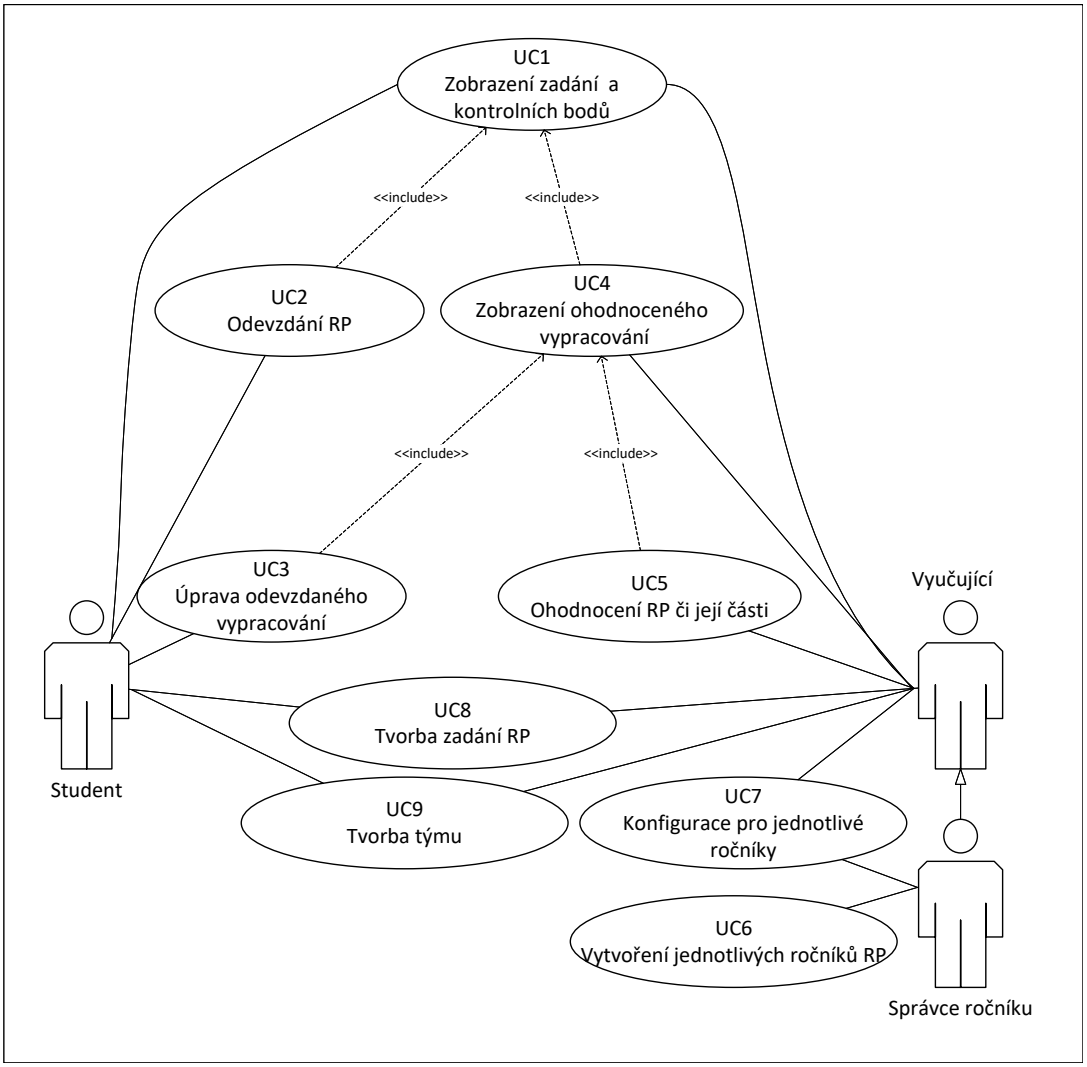
- UC6: Vytvoření jednotlivých ročníků pro ročníkové práce
  Aktér: Správce ročníku

- UC7: Konfigurace jednotlivých ročníků – vyučující (konzultanti) si mohou přizpůsobit některá nastavení pro daný ročník či jeho část
  Aktér: Vyučující, správce ročníku

- UC8: Vytvoření zadání ročníkové práce
  Aktér: Student nebo Vyučující

- UC9: Vytvoření týmu pro skupinovou ročníkovou práci
  Aktér: Student nebo Vyučující

Diagram případů užití je znázorněn na obrázku 1.2 na straně 8.



Obrázek 1.2 Diagram případů užití

1.2.3 Activity diagram

Posledním krokem souvisejícím se stanovením požadavků na systém pro správu ročníkových prací je vytvoření activity diagramu. Popisuje životní cyklus ročníkových prací v průběhu celého školního roku. Čtenář jej nalezne na obrázku 1.3 na straně 10.

1.3 Analýza stávajících řešení

Na základě zjištěných požadavků jsem přistoupil k hledání systému, který by dokázal dané požadavky uspokojit. Zaměřil jsem se na aplikace v kategorii LMS (Learning Management System), v češtině známé jako aplikace pro řízení výuky.

Tyto systémy lze rozdělit na dvě kategorie: open-source a closed-source, přičemž jsem hledal kandidáty z obou těchto kategorií. Do analýzy jsem zahrnul pouze řešení, za která by gymnázium nemuselo platit, jelikož rozpočet školy je napjatý a digitalizace správy ročníkových prací není pro GJGJ prioritou.

Uvádím-li, že řešení splňuje FP1, je tím myšleno, že podporuje hromadný import uživatelů, případně dalších informací. Propojení přímo s aplikací Bakaláři, kterou Gymnázium Jiřího Gutha-Jarkovského používá, žádné systémy nenabízejí.

Z kategorie open-source jsem zvolil asi nejznámější a v Česku alespoň do roku 2017 nejpoužívanější LMS, a to Moodle. [4] Dále do této kategorie patří řešení, které mi doporučil vedoucí bakalářské práce. Jedná se o webovou aplikaci SOS – Studentský odevzdávací systém.

Z kategorie closed-source jsem analyzoval Google Classroom, především protože je již na gymnáziu využíván pro podporu výuky. Jako protipól ke Google Classroom jsem vybral Microsoft Teams.

Poslední zvažovanou možností bylo naprogramovat celý systém od základu sám.

Jako kritéria vhodnosti jednotlivých řešení jsem zvolil následující parametry:

- rozšiřitelnost

- náročnost konfigurace

- náročnost nasazení

- splnění jednotlivých funkčních požadavků

V přehledu splnění jednotlivých kritérií u každého z uvedených řešení používám následující symboly:

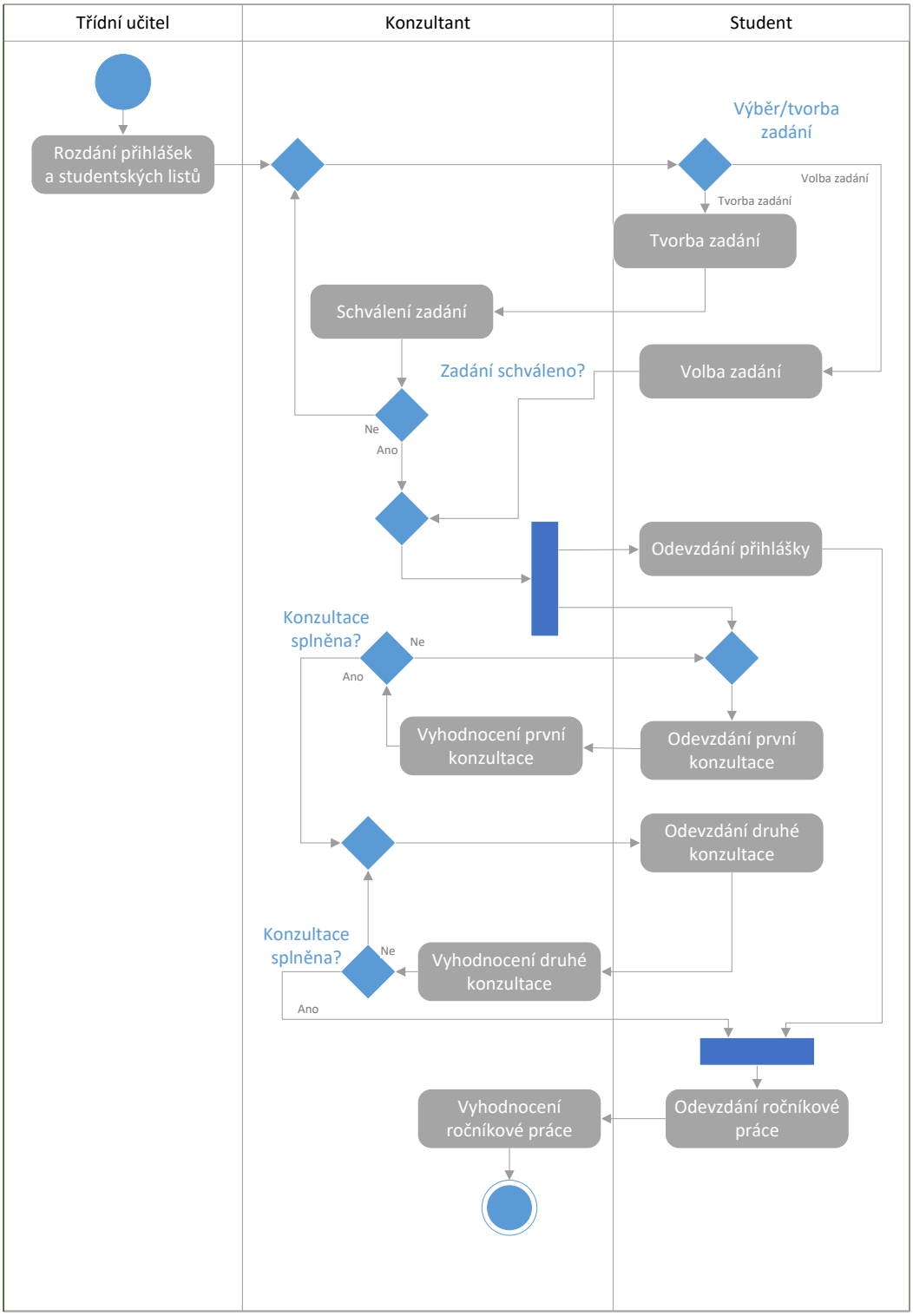
splněno	✓
splněno s výhradami	/
splněno částečně	⚠
nesplněno	◯

1.3.1 Google Classroom

Google Classroom [5] je webová aplikace, která je součástí Google Education Suite, a je tedy pro školy zdarma.

Jelikož se jedná o řešení z kategorie closed-source, je rozšiřitelnost Google Classroom v současné době téměř nulová.

Nasazení Google Classroom je poměrně jednoduché, poněvadž jde o cloudové řešení. Co se konfigurace týče, veškerou administraci ovšem musí dělat správci IT. Učitelé mohou maximálně



Obrázek 1.3 Activity diagram průběhu RP

vytvářet třídy. To by tedy znamenalo, že konfiguraci ročníkových prací by museli řešit IT pracovníci školy, což by v případě GJGJ přípravu dost komplikovalo. IT služby gymnáziu poskytují externisté, kteří do školy docházejí jen některé dny v týdnu a jsou hodně vytížení.

Google Classroom splňuje funkční požadavek FP1 částečně – umožňuje hromadný import uživatelů. Dále lze pomocí API vytvářet jednotlivé třídy a ty naplnit studenty a vyučujícími. Řešením by tedy bylo naprogramovat nástroj, který vytvoří třídy pro jednotlivé ročníky a naplní je správnými daty.

FP2 Google Classroom splňuje, reprezentace jednotlivých ročníků by byla tvořena třídami pro každý ročník, přičemž tyto třídy by šlo nastavit tak, aby obsahovaly požadavky pro každý ročník.

FP3 aplikace splňuje, ovšem s výhradami. Tuto funkci by bylo potřeba spojit s FP4 a FP5, což by bylo z hlediska učitelů dost nepraktické: učitelé by museli vytvořit seznam témat v rámci jednoho úkolu (či více úkolů dělených dle předmětů). Studenti by jako řešení tohoto úkolu napsali, o které téma mají zájem, případně by navrhli vlastní. Vyučující by se těmito „řešeními“ museli probrat a následně vytvořit úkoly pro jednotlivé ročníkové práce a jejich konzultace a přiřadit správně jednotlivé studenty k těmto úkolům. Studenti by poté nahrávali řešení k těmto nově vytvořeným úkolům. Tedy FP3, FP4 a FP5 lze považovat za splněné, ale s výhradami, protože by takové řešení učitele příliš zatěžovalo.

FP6 je splněn. Google Classroom má dobrou podporu pro hodnocení úkolů. Schvalování studentských zadání by se provádělo mimo systém, jelikož student by stejně s vlastním tématem musel zajít za učitelem, aby mu ho do sytému zapsal.

FP7 je splněn.

Shrnu-li využitelnost Google Classroom pro účely správy ročníkových prací, vychází následující hodnocení:

rozšiřitelnost	žádná
náročnost nasazení	jednoduchá
náročnost konfigurace	střední
FP1	/
FP2	✓
FP3	⚠
FP4	⚠
FP5	⚠
FP6	✓
FP7	✓

1.3.2 Microsoft Teams

Microsoft Teams jsou zde uvedeny spíše jako ilustrační příklad, jelikož nepředpokládám, že by gymnázium bylo ochotno přejít na konkurenční řešení, když v současné době používá Google Classroom.

Microsoft Teams [6] jsou součástí Office 365 for Education, tedy škola by je mohla využívat zdarma.

Opět jde o closed-source řešení, proto jej, podobně jako Google Classroom, nelze rozšiřovat.

Protože se jedná o cloudové řešení, nasazení nepředstavuje žádný problém. Konfiguraci by stejně jako v případě Google Classroom museli dělat převážně IT pracovníci školy, což by přípravu komplikovalo.

FP1 je splněn částečně: existuje možnost hromadného importu uživatelských dat. Microsoft Teams stejně jako Google Classroom mají API, které umožňuje hromadně vytvářet i týmy a jednotlivé kanály v nich. Dále umožňuje do těchto týmů rovnou přidávat studenty. Bylo by tedy pouze nutné naprogramovat rozšíření, které by z Bakalářů vyčetlo potřebné informace, importovalo uživatele a vytvořilo týmy a kanály.

FP2 je splněn: jednotlivé ročníky by se reprezentovaly jako jednotlivé týmy. Dělení na jednotlivé povolené předměty pro daný ročník by šlo řešit pomocí kanálů v jednotlivých týmech. Učitelé by tak viděli pouze daný kanál.

FP3, FP4, FP5: Stejně jako u Google Classroom by se tyto tři požadavky musely spojit do většího celku, kdy by učitel vytvořil jeden úkol, který by sloužil pro výběr zadání. Následně by musel vytvořit úkoly pro jednotlivé RP a konzultace a přiřadit k nim příslušné studenty, jelikož studenti se sami přiřazovat nemohou. Tyto tři funkční požadavky jsou tedy splněny s výhradami.

FP6: Microsoft Teams umožňují hodnotit jednotlivé úkoly. Schvalování zadání by bylo součástí prvního úkolu, ve kterém by si studenti vybírali téma, jak bylo vysvětleno v předchozím odstavci. FP6 ze jej tedy považovat za splněný.

FP7 je splněn.

Celkové hodnocení Microsoft Teams vychází následovně:

rozšiřitelnost	žádná
náročnost nasazení	jednoduchá
náročnost konfigurace	střední
FP1	/
FP2	✓
FP3	⚠
FP4	⚠
FP5	⚠
FP6	✓
FP7	✓

1.3.3 Moodle

Moodle [7] je velmi populární open-source systém na podporu výuky, který je vysoce rozšiřitelný.

Nasazení je ovšem dost náročné, protože se jedná o self-hosted řešení. Musel by jej tedy někdo nasadit a systematicky udržovat.

Má řadu funkcí, ale pro účely správy ročníkových prací je velmi složitý. Konfigurací systému by učitelé strávili mnoho času, takže jeho využití by se vyplatilo, leda kdyby již na škole byl používán i k jiným účelům. S ohledem na složitost konfigurace je značně pravděpodobné, že by učitelé gymnázia raději setrvali u stávajícího systému, tj. řízení ročníkových prací s využitím pouze fyzických papírových podkladů.

FP1: Moodle podporuje hromadný import dat, tedy funkční požadavek číslo 1 je splněn částečně. Jednotlivé kurzy by se musely dělat ručně a studenty do nich hromadně importovat. Nepřišel jsem žádný způsob, jak toto provést programaticky.

FP2: Jednotlivé ročníky by se dělaly jako jednotlivé kurzy, které by měly různé požadavky na splnění. Tento funkční požadavek je tedy splněn.

FP3 je splněn s výhradami. Nepodařilo se mi dohledat, zda student může vytvářet úkoly, které by reprezentovaly celou ročníkovou práci, tj. i zadání.

FP4 je splněn částečně, neboť neumožňuje rozdělit úkol na více samostatných částí, do kterých by student odevzdával ročníkové práce v jednotlivých etapách. Moodle sice umožňuje povolit více odevzdání, ale tento způsob by mohl být problematický, jelikož není flexibilní a mohlo by se stát, že by student vyčerpal všechny pokusy a nemohl pokračovat. Dále tento způsob neumožňuje hodnotit jednotlivé části samostatně.

FP5 je splněn, s jedinou výhradou: Úkoly pro jednotlivá zadání by museli vytvářet učitelé, protože Moodle nepodporuje tvorbu úkolů studenty. Studenti však mohou vytvářet skupiny a následně na projektu spolupracovat.

FP6 je splněn. Konzultace by bylo třeba tvořit jako samostatné úkoly, jelikož Moodle nepodporuje úkoly s více částmi. Dále zadání úkolů (ročníkových prací) mohou do Moodle vkládat

pouze učitelé, takže schvalování vlastních zadání studentů by muselo být řešeno částečně mimo Moodle nebo jako další samostatný úkol.

FP7 je splněn.

Veškeré výhrady a nedostatky ohledně funkčních požadavků mají u Moodle menší váhu, protože by bylo možné naprogramovat rozšíření vyhovující potřebám gymnázia.

Celkové hodnocení Moodle je následující:

rozšiřitelnost	vysoká
náročnost nasazení	střední
náročnost konfigurace	složitá
FP1	/
FP2	✓
FP3	⚠
FP4	/
FP5	⚠
FP6	✓
FP7	✓

1.3.4 SOS – Studentský odevzdávací systém

Tato webová aplikace, vyvinutá v rámci loňské bakalářské práce studentem FIT ČVUT, je určena k odevzdávání a hodnocení týmových i individuálních semestrálních prací v předmětech vyučovaných na FIT ČVUT. [8]

SOS FP1 nesplňuje, neboť je vázán přímo na fakultní systémy, a tedy není důvod, aby umožňoval import dat například ve formátu csv.

Funkční požadavky FP2, FP3, FP4, FP5, FP6, FP7 SOS splňuje za předpokladu, že se podaří ročníky reprezentovat jako předměty, které v SOS již dané FP splňují. Takové rozšíření se ukázalo jako realizovatelné.

SOS má vysokou a relativně jednoduchou rozšiřitelnost. Konfigurace je jak pro učitele, tak pro správce IT středně náročná. Mírně složitější je nasazení, jelikož se jedná o self-hosted řešení. Ovšem v porovnání s Moodle je výrazně jednodušší.

Celkové hodnocení SOS:

rozšiřitelnost	vysoká
náročnost nasazení	střední
náročnost konfigurace	střední
FP1	◯
FP2	✓
FP3	✓
FP4	✓
FP5	✓
FP6	✓
FP7	✓

Analýzu zvažovaných řešení shrnuje tabulka 1.1 na straně 14. Z ní je zřejmé, že nejlepším kandidátem je SOS – Studentský odevzdávací systém. Tento systém splňuje nejvíce funkčních požadavků a navíc je rozumně náročný na konfiguraci i nasazení. Je vysoce rozšiřitelný, lze jej tedy upravit tak, aby gymnáziu plně vyhovoval. Proto jsem se rozhodl jej použít jako základ systému pro správu ročníkových prací. Podrobně se mu věnuje následující část.

	Google Classrooom	Micorosoft Teams	Moodle	SOS	Nový systém
rozšiřitelnost	žádná	žádná	vysoká	vysoká	vysoká
náročnost nasazení	jednoduchá	jednoduchá	střední	střední	střední
náročnost konfigurace	střední	střední	složitá	střední	střední
FP1	/	/	/	◯	◯
FP2	✓	✓	✓	✓	◯
FP3	⚠	⚠	⚠	✓	◯
FP4	⚠	⚠	/	✓	◯
FP5	⚠	⚠	⚠	✓	◯
FP6	✓	✓	✓	✓	◯
FP7	✓	✓	✓	✓	◯

Tabulka 1.1 Výsledky analýzy zvažovaných řešení

1.4 Analýza SOS

Studentský odevzdávací systém je vícestránková webová aplikace psaná v Djangu, což je open-source webový framework vytvořený v Pythonu. [9] Využívá architektonický vzor MVT (Model View Template), který je modifikací vzoru MVC (Model View Controller).

SOS byl vytvořen pro účely podpory výuky na FIT ČVUT s tím, že splňuje požadavky vícero různých předmětů. [8]

Základní jednotkou jsou předměty, přičemž každý předmět obsahuje zadání. Zadání mohou vytvářet jak vyučující, tak studenti. Tvorbu zadání studenty lze povolit zvlášť pro jednotlivé paralelky daného předmětu. Pro potřeby středních škol budou třídy a ročníky školy považovány v rámci SOS za paralelky.

Dále má každý předmět nastaven kontrolní body, které reprezentují jednotlivé části procesu tvorby daného projektu. Pro účely správy ročníkových prací by jejich konzultace představovaly kontrolní body v SOS. Kontrolní body mohou mít pro jednotlivé paralelky různé časy odevzdání. Rovněž lze v rámci předmětu povolit týmové projekty, přičemž SOS umožňuje vytvářet týmy napříč paralelkami. Toto nastavení bude pravděpodobně pro většinu ročníků gymnázia povoleno, jelikož každý ročník má dvě paralelky (třídy). Další zajímavé nastavení je možnost povolit pozdní odevzdání.

SOS má také interní notifikace, které uživatele upozorní na důležité události. Rovněž obsahuje hezky zpracovaný správce týmů.

Projekty psané v Djangu se běžně člení do více tzv. aplikací ve formě Python modulů. Každá aplikace je zodpovědná za určitou logickou část systému, přičemž každý modul v sobě obsahuje příslušné modely, formuláře, aplikační logiku, dílčí konfiguraci směrovače požadavků a šablony potřebné pro generování odpovědí. [10, kap. 4]

SOS je tvořen následujícími aplikacemi:

Aplikace Assignments
Řeší zadávání jednotlivých studentských projektů.

Aplikace Attachments
Pracuje s přílohami, které studenti nahrávají ke svým odevzdávaným řešením. Přílohy mohou studenti i vyučující přidávat také přímo k tvořenému zadání.

Aplikace Checkpoints
Řeší implementaci jednotlivých kontrolních bodů. Obsahuje mimo jiné model CustomDeadline, který udržuje informaci o nastaveném termínu odevzdání pro konkrétní paralelku, jelikož jednotlivé paralelky mohou mít odlišné termíny kontrolních bodů.

Aplikace Courses
Jedná se o nejrozsáhlejší aplikaci celého systému. Obsahuje modely, které reprezentují jednotlivé semestry, předměty a paralelky. Dále řeší konfiguraci předmětů pro jednotlivé vyučující.

Aplikace Homepage
Tato aplikace neobsahuje žádné modely a týká se pouze zobrazení domovské stránky.

Aplikace Notifications
Slouží k realizaci upozornění uživatelů na důležité události v systému SOS, například hodnocení kontrolních bodů.

Aplikace OAuth
Od ostatních aplikací SOS se liší tím, že slouží pouze pro autentizaci uživatelů pomocí fakultního OAuth 2.0 serveru a přístup k datům ze systému KOS ČVUT prostřednictvím KOSapi.

Aplikace Submissions
Realizuje odevzdávání vypracovaných řešení a jejich hodnocení.

Aplikace Teams
Řeší správu realizačních týmů.

Aplikace Users
Obsahuje pouze jediný model User, který je potomkem třídy `django.contrib.auth.models.AbstractUser`. Tento model je zodpovědný za reprezentaci uživatele v systému.

Z podrobné analýzy SOS jsem dospěl k závěru, že pro jeho využití na gymnáziu bude potřeba věnovat se následujícím okruhům problémů: vyřešit import dat, vyřešit přihlašování do systému, vyřešit reprezentaci ročníkových prací a rozšířit sytém notifikací jednak o notifikace o blížícím se termínu odevzdání, jednak aby umožňoval zasílání notifikací e-mailem.

1.5 Použité technologie

Tato část je zaměřena na již použité technologie v rámci SOS a zároveň shrnuje, jaké nové technologie jsem v jeho rozšíření použil a proč.

1.5.1 Django

Django [11] je framework pro tvorbu webových aplikací napsaný v programovacím jazyce Python. Implementuje architektonický vzor Model View Template, což je modifikace často využívaného vzoru Model View Controller. Jádro frameworku obsahuje objektově-relační mapper mezi relační databází a datovým modelem, který se definuje pomocí tříd v jazyce Python. Jednotlivé změny v databázi se provádějí pomocí migrací.

Migrace umožňují inkrementální změny v databázi, není tedy potřeba při každé změně modelu upravovat create script, kterým se vytváří databáze. O migracích se dá uvažovat jako o verzovacím systému pro databázi. Příkaz makemigrations je zodpovědný za zabalení změn v modelech do migračních souborů a příkaz migrate je pak aplikuje na databázi. Systém migrací také umožňuje v případě potřeby návrat k některé předchozí migraci. [12]

Framework nabízí i další komponenty, například serializační a validační systém pro formuláře, cachovací systém nebo šablonovací systém.

Pro aplikaci SOS bylo Django zvoleno vzhledem k tomu, že s Djangem měl autor SOS již poměrně rozsáhlé zkušenosti.

1.5.2 Postgresql

Postgresql je open-source relační databáze. [13] Pro aplikaci SOS byla zvolena s ohledem na dobrou kompatibilitu s Djangem.

1.5.3 Django templates a Bootstrap

Pro frontend SOS je využit šablonovací systém nabízený Djangem v kombinaci s volně dostupnou CSS šablonou AdminLTE, která využívá framework Bootstrap napsaný převážně v jazycích HTML a CSS. Ta je dnes základem téměř každé webové aplikace, protože výrazně zjednodušuje psaní responzivních webových aplikací. [14]

1.5.4 GitLab

Pro správu verzí využívá SOS Git a GitLab.

GitLab [15] je jedním ze systémů pro správu repozitářů postavený pro verzovací systém Git [16], přičemž poskytuje i další související funkcionality, například wiki. Dále nabízí nástroje pro CI/CD (Continuos Integration / Continuos Deployment). Je dostupný buď v self-hosted variantě, nebo jako SaaS – software jako služba.

Na FIT je GitLab nejčastěji užívaným nástrojem pro správu verzí, k čemuž přispívá i fakt, že fakulta tento nástroj hostuje na adrese gitlab.fit.cvut.cz.

1.5.5 Gunicorn

Gunicorn je open-source implementace rozhraní WSGI čili Web Server Gateway Interface pro komunikaci webového serveru s aplikacemi psanými v Pythonu. WSGI je definováno standardem PEP 3333. Toto rozhraní zmenšuje provázanost webového serveru a samotné webové aplikace, čímž odpadá nutnost použít konkrétní webový server v důsledku volby aplikačního frameworku nebo obráceně. [17]

Gunicorn se běžně využívá k propojení webového serveru s Djangem, ve kterém je SOS implementován. Gunicorn je využíván i při použití frameworků jako Flask či Bottle. Jeho implementace využívá tzv. worker procesy k paralelnímu obsluhování požadavků. Gunicorn byl zvolen především proto, že jeho integrace s Djangem je velmi jednoduchá a je dobře popsána v dokumentaci. [18]

1.5.6 NGINX

Pro provoz webových aplikací na operačních systémech s unixovým jádrem se nejčastěji používají webservery Apache a NGINX. [19] Obě řešení jsou open-source. NGINX má oproti Apache výhodu, že je minimalističtější: dokáže obsloužit stejný počet požadavků jako Apache, ale s výrazně menšími nároky na paměť. Nevýhodou NGINX oproti Apache je, že nedokáže sám o sobě zpracovávat dynamický obsah, a je tedy potřeba mít externí aplikační server. Tuto nevýhodu ale vyváží jeho úspornost.

1.5.7 Docker a Docker Compose

Pro snadnější nasazování a také z důvodu, že se systém rozrostl o několik komponent, jsem se rozhodl využít virtualizační nástroj Docker [20]. S ním jsem se seznámil během studia na FIT, proto jsem nevolil jinou virtualizační platformu.

Docker umožňuje oddělit aplikaci od infrastruktury tak, že ji zabalí do volně izolovaného prostředí nazvaného kontejner. Díky izolaci jednotlivých kontejnerů lze spustit na jednom hostiteli více aplikací. Každý Docker kontejner obsahuje vše potřebné pro spuštění dané aplikace.

Docker Compose [21] je nástroj pro definici a spouštění více kontejnerových aplikací. Je založen na konfiguračním souboru v jazyce YAML, ve kterém se konfigurují jednotlivé služby. Ty lze pak jednoduše spravovat jako celek.

Použití Dockeru také přispěje ke zjednodušení vývoje a testování systému pro správu ročníkových prací.

1.5.8 Cron

Do systému SOS bylo nutné přidat notifikace o blížícím se odevzdání práce či jejího kontrolního bodu. Bylo tedy potřeba analyzovat možnosti spouštění periodických událostí.

Často využívanou možností je Cron [22]. Jedná se o nástroj pro nastavování periodických událostí v unixových systémech. Pokud by však aplikace byla hostována bez přístupu k operačnímu systému, bylo by použití Cronu nemožné. Další problém představuje vlastní propojení Cronu s aplikací.

Hledal jsem proto alternativu, která by byla jednoduše integrovatelná s Pythonem a Djangem. Z průzkumu vyšel vítězně Celery [23], což je jednoduchý a flexibilní distribuovaný systém pro zpracování dat v reálném čase. Je již napsaný v Pythonu a má doplněk, který jej integruje přímo do Djanga.

Celery funguje na bázi producent-konzument. Existují tedy dva procesy: Jeden se nazývá beat a slouží k posílání úkolů workerům, kteří pak spouštějí jednotlivé úkoly. Pro přenos zpráv mezi producentem a konzumentem slouží fronta úkolů.

Celery má více plánovačů událostí. Nejjednodušším plánovačem je intervalový, který spouští úkoly v pevně zadaných časových intervalech. Plánovač, jenž je flexibilnější a pro SOS zajímavější, je postavený na crontabech, tedy tabulkách, které cronu určují spouštění událostí v přesně zadaných časech nebo časových intervalech, například každý první den v měsíci. Třetí plánovač se nazývá solar a slouží ke spouštění úkolů v časech vázaných na astronomické jevy, jako je východ nebo západ slunce. [24]

Jako frontu úkolů lze použít řadu nástrojů. S ohledem na jednoduchost jsem pro SOS zvolil redis.

1.5.9 Google OAuth

Od samého počátku jsem na základě konzultací s koordinátorem ročníkových prací na Gymnáziu Jiřího-Gutha Jarkovského počítal s tím, že pro přihlašování do systému pro správu ročníkových prací se bude používat Google OAuth. Protože na škole se již využívá výše zmíněná aplikace Google Classroom, toto řešení se jeví jako nejrozumnější.

1.6 Propojení s informačním systémem školy

Na Gymnáziu Jiřího-Gutha Jarkovského se pro správu školní agendy používá software Bakaláři. Jedná se o nejrozšířenější školní systém v České republice [25], který slouží k administraci a komunikaci mezi školou a studenty i jejich rodiči. Jsou v něm zaznamenáni všichni studenti, vyučující, kontaktní údaje, vyučované předměty, rozvrhy, známky, domácí úkoly atd. Vzhledem k jeho rozšířenosti i využívání gymnáziem, jež poskytlo požadavky na systém, bylo od začátku zřejmé, že pro potřeby importu dat bude nutné systém pro správu ročníkových prací s touto aplikací propojit.

Společnost Bakaláři software s.r.o. vytvořila pro přístup aplikací třetích stran k datům v systému Bakaláři API pod názvem Datový konektor. [26] Použití Datového konektoru je podmíněno

uzavřením smlouvy mezi školou a společností. Musel jsem se tedy obrátit na školu, abych k API získal přístup a mohl jej v rámci SOS použít. Tento proces byl velmi zdlouhavý.

1.6.1 Datový konektor

Datový konektor používá pro většinu endpointů zabezpečení pomocí http basic authentication. [26] Tento typ zabezpečení není v současnosti příliš oblíbený, především proto, že v hlavičce dotazů se posílají přihlašovací údaje kódované pomocí base64. To je bezpečné pouze v kombinaci s ssl a při použití na backendu.

Po zkušenostech s dokumentováním REST API pomocí OpenAPI specifikace [27], se kterou jsem se seznámil v rámci studia předmětu Technologie Java, mě dokumentace poskytnutá společností Bakaláři software dost zklamala. Dokumentace rozhraní je značně nekonsistentní, sice převážně v drobnostech, ale od společnosti s takovým postavením na trhu bych očekával vyšší úroveň. Stalo se mi například, že jsem postrádal informace o endpointu, o kterém jsem se domníval, že by měl být součástí API. Zkusil jsem jeho název odhadnout a testováním jsem zjistil, že skutečně existuje.

V API bohužel neexistuje endpoint pro získání všech uživatelů, a je tedy potřeba pracovat zvlášť se studenty a zvlášť se zaměstnanci. Dále jsem postrádal endpoint, který by zprostředkoval data jednoho uživatele určeného jedinečným identifikátorem. Rovněž jsem nenašel endpointy vracející informace, které předměty student studuje a učitel učí. Dodnes nevím, jestli tyto informace v Datovém konektoru chybí, nebo jen nejsou v dokumentaci. Nicméně vzhledem k tomu, že Datový konektor údajně využívá i mobilní aplikace Bakalářů [26], bych očekával, že tyto informace v Datovém konektoru k dispozici jsou. Obrátil jsem se s tímto dotazem opakovaně na technickou podporu společnosti Bakaláři software, ale dosud jsem neobdržel žádnou reakci.

Kapitola 2
Návrh

V této kapitole je podrobněji rozebráno, jakými způsoby by bylo možné realizovat rozšíření SOS, která vyplynula z analýzy požadavků, a jaká řešení jsem zvolil.

Z analýzy vzešly následující požadavky, které bylo potřeba navrhnout a implementovat. Pro připomenutí je zde znovu uvádím:

- Import dat

- Přihlašování do systému

- Reprezentace ročníkových prací

- Rozšíření systému notifikací

- Jednodušší způsob nasazení

Veškerá rozšíření SOS jsem prováděl tak, aby nebyla omezena původní funkcionalita. Také jsem pamatoval na budoucí vývoj. Všechny modifikace jsem programoval s ohledem na co nejjednodušší rozšiřování v budoucnu.

2.1 Import dat

Jak již bylo zmíněno v části 1.6 kapitoly Analýza, Gymnázium Jiřího-Gutha Jarkovského využívá pro správu školní agendy software Bakaláři. SOS jakožto systém vyvinutý pro potřeby FIT je provázaný s aplikací KOS využívanou v rámci celého ČVUT.

Bylo tedy nutné navrhnout takové rozšíření, aby SOS dokázal pracovat s oběma zdroji dat s velmi rozdílnými API.

V původní verzi SOS se import dat řeší v aplikaci (ve smyslu Djanga) `oauth`. V ní se nachází modul `utils`, který obsahoval modul `kos`. V něm bylo několik souborů, které obsahovaly veškeré funkce potřebné k importování dat pomocí KOSapi. Tento modul bylo tedy potřeba refaktorovat, aby SOS dokázal importovat data více způsoby.

První možností by bylo vytvořit soubor, v němž by byly potřebné funkce pro import dat z Bakalářů. Přepínání mezi způsoby importování dat by se řešilo v místech použití jednotlivých funkcí. Tento návrh jsem ovšem zavrhl s ohledem na možná další rozšiřování v budoucnu. Při každém přidání nového způsobu importování dat by se musela projít všechna místa, kde se funkce používají, a přidat do nich další rozhodování, které by zajistilo použití správné funkce pro danou hodnotu nastavení způsobu importu.

První zmiňovanou možnost rozšíření lze ale zdokonalit: rozhodování o tom, která funkce pro import dat se má použít, by se přesunulo do speciálních funkcí, jež by vracely příslušná data podle zvoleného importu dat. Tyto speciální funkce by pak byly volány místo původních funkcí. Ani tento návrh ovšem není z hlediska rozšiřitelnosti ideální.

Proto jsem nakonec modul `kos` nahradil nově vytvořeným modulem `data_provider`, jenž obsahuje pro každý způsob importace dat jednu třídu. Metody v každé z těchto tříd mají jednotné rozhraní a pojmenování, a lze je tedy zaměňovat. Dále jsem vytvořil funkci, která v závislosti na nastavení rozšířeného SOS rozhoduje, jaký objekt vrátí. Použitím navráceného objektu se pak volají správné funkce.

V souvislosti s importem dat bylo ještě třeba navrhnout způsob, jak přiřadit uživatele v SOS k uživatelům v Bakalářích. Datový konektor Bakalářů neposkytuje kromě interního ID žádný unikátní identifikátor, jenž by mohl sloužit pro jednoznačnou identifikaci uživatele. Interní ID k tomuto účelu nelze použít, neboť tento údaj uživatelé neznají a nemohou si ho v Bakalářích zobrazit. Proto jsem jako identifikátor zvolil školní e-mailovou adresu. Tu mají přiřazenou všichni studenti i vyučující a využívají ji v souvislosti s Google Classroom. Bohužel v době tvorby této bakalářské práce měli školní e-mailovou adresu v Bakalářích zapsanou pouze dva studenti gymnázia a žádný vyučující. Dle vyjádření vedení školy se to bude řešit až v létě.

2.2 Přihlašování do systému

Jak bylo již řečeno v části 1.5.9 kapitoly Analýza, k přihlašování do systému se počítá s Google OAuth. Nebylo tedy nutné nic navrhovat, jelikož přihlašování do Google OAuth je vcelku přímočaré a relativně jednoduché na implementaci.

Alternativou k přihlašování pomocí Google OAuth by bylo přihlašování pomocí lokálních účtů. Tuto možnost jsem s gymnáziem konzultoval, ovšem na základě poukázaných nevýhod o ni projevilo jen vlažný zájem. Proti hovoří například známý fakt, že uživatelé rádi používají stejná hesla k různým účtům, což představuje bezpečnostní riziko [28]. Lze sice vyžadovat silná hesla, ale ani silné heslo není bezpečné, pokud je již použito někde jinde. Google má nastavena pravidla, která tomuto do jisté míry předcházejí. Navíc má-li uživatel spárováno s Googlem nějaká mobilní zařízení, při každém přihlášení z nového zařízení dostane uživatel na tato zařízení (a na záložní e-mailovou adresu, jestliže je v Google účtu nastavena) dotaz, zda se opravdu přihlašuje on. Zareaguje-li uživatel na tuto výzvu v případě, že nově přihlašovaným není on sám, Google pokus o přihlášení z nového zařízení zamítne a vyzve oprávněného uživatele k vytvoření nového hesla. [29] Tedy pravděpodobnost prolomení Google účtu i s opakovaně použitým heslem je menší než prolomení SOS s lokálními účty.

S lokálními účty souvisí také otázka generování hesel. Jako nejvhodnější se jeví umožnit uživateli se do systému registrovat, takže by si uživatel volil heslo sám. Opět ale narážíme na skutečnost, že v současné době nejsou v Bakalářích u všech uživatelů zapsané údaje, podle kterých by je bylo možné jednoznačně identifikovat.

Dalším problémem lokálních účtů je změna hesla, pokud jej uživatel zapomene. Standardně se uživateli, který o změnu požádal, pošle e-mail s odkazem na stránku obnovení hesla. Tento způsob by na ale gymnáziu nemusel dobře fungovat, jelikož zatím v Bakalářích nemají e-mailovou adresu uvedenu všichni uživatelé. Bylo by tedy nutné, aby uživatelé změnu hesla řešili osobně se správcem systému, což představuje zejména pro správce systému nežádoucí zátěž. Případně by mohl příslušný odkaz na stránku obnovení hesla obdržet zákonný zástupce – pro chod aplikace Bakaláři je e-mailová adresa na alespoň jednoho z rodičů studentů vyžadována, a tedy je v aplikaci uvedena.

Pevně věřím, že do začátku příštího školního roku budou školní e-mailové adresy do aplikace Bakaláři doplněny, a nebude tedy třeba alternativní způsob přihlašování řešit.

2.3 Reprezentace ročníkových prací

Nejzásadnější rozšíření SOS pro potřeby správy ročníkových prací se týká jejich reprezentace v SOS.

RP není přímo vyučovaný předmět, ale spíše kolekce zadání z více předmětů, které se pro jednotlivé ročníky mohou lišit. Dále může mít každý ročník trochu jiné další parametry, například zda jsou povoleny týmové práce.

První možnost, jak toto realizovat, by bylo reprezentovat ročníkové práce jako předmět a jednotlivé ročníky jako paralelky. Tento způsob s sebou ale nese několik problémů. Především není dopředu známo, kteří učitelé jsou potřební v kterých ročnících: zda má být daný učitel přiřazen k danému ročníku, je dáno tím, jestli je jím vyučovaný předmět povolen pro daný ročník.

To nás přivádí k druhému problému a tím je otázka, jak v jednotlivých ročnících povolit jednotlivé předměty. Jistě by šlo pro každý ročník vytvořit nějaké nastavení, ve kterém by správce ročníku povolil jednotlivé předměty, a tento úkon by pak importoval učitele do daného ročníku.

Ovšem tento návrh reprezentace není příliš efektivní a jeho implementace by značně bořila již fungující procesy SOS.

Druhý návrh vychází z myšlenky, že ročníková práce je kolekcí zadání z různých vyučovaných předmětů. V tomto návrhu tedy jednotlivé ročníky gymnázia představují samostatné předměty v SOS. Znamená to, že vyučující v SOS vytvoří 6 samostatných předmětů pro ročníky, které mají RP v osnovách (sexty a oktávy ročníkové práce nedělají, protože se připravují na malou, resp. skutečnou maturitu).

Fakt, že RP je kolekcí zadání z jiných předmětů, bude v SOS reprezentovat možnost přidat v předmětu tzv. zdroje zadání úloh. Zdroj zadání v daném předmětu vlastně slouží ke zkopírování zadání (přenesení zadání) z jiného předmětu. Existují-li v SOS již předměty angličtina a anglická konverzace, při vytváření předmětu seminář z anglického jazyka zvolí vyučující jako zdroje zadání úloh v tomto předmětu angličtinu a anglickou konverzaci. Díky tomu může při vytváření zadání úlohy v předmětu angličtina současně vytvořit totéž zadání i v semináři z anglického jazyka (a následně ho v případě potřeby upravit). Může také zvolit možnost toto zadání přidat pouze do semináře a nemít jej v angličtině. Tím vlastně indikuje, že zadání pochází z angličtiny a není nativním zadáním semináře.

Taková reprezentace je sice komplikovanější, ale skýtá větší flexibilitu. Pokud si studenti více ročníků mohou vybírat zadání RP z nějakého vyučovaného předmětu (pro ilustraci například z matematiky), může učitel matematiky jednoduše vytvořit zadání, které bude dostupné pro každý ročník, v němž je matematika zahrnuta mezi předměty, ze kterých může student daného ročníku tvořit RP. Dále tato reprezentace rovnou zobrazuje, k jakému předmětu se zadání váže. Další výhodou je, že učitel může mít jednodušší verzi zadání, kterou použije jako úkol v daném předmětu, a zároveň náročnější verzi téhož zadání pro účely RP. Navíc při vytváření předmětu pro daný ročník jsou do tohoto předmětu rovnou automaticky importováni učitelé, kteří daný předmět vyučují.

Jelikož se jedná o návrh zvolený k implementaci, popíšu podrobněji všechny změny v SOS, které bylo v souvislosti s tímto návrhem potřeba navrhnout.

Především šlo o způsob rozšíření databáze, aby podporovala zdroje zadání. Samotné zdroje zadání byly vcelku jednoduché, jednalo se pouze o přidání jedné M:N vazby k tabulce *Course*. Dále bylo nutno nějak reprezentovat, do jaké „skupiny“ zadání dané zadání patří. „Skupina“ zadání reprezentuje, z jakého předmětu zadání pochází a do jakých všech předmětů se má rozkopírovat. Do databáze jsem tedy přidal vazbu typu 1:N mezi tabulkou *Assignment* a novou tabulkou *AssignmentGroup*, která obsahuje 1:N vazbu s tabulkou *Course*, reprezentující zdroj (předmět, v němž zadání vzniklo), a M:N vazbu s tabulkou *Course*, jež reprezentuje, do jakých předmětů bylo zadání rozkopírováno. Všechny změny v databázovém modelu jsou znázorněny červenou barvou na obrázku 2.2 na straně 25.

Dále bylo potřeba vyřešit, jak do těchto „skupinových“ předmětů importovat studenty a paralelky ze zdrojů zadaní. Navrhl jsem novou stránku, na niž se lze dostat ze stránky upravující

nastavení daného předmětu. Na této stránce jsem vytvořil formulář, který zobrazuje všechny paralelky ze všech předmětů, jež má předmět jako zdroje zadání. Tyto paralelky jsou seskupeny podle předmětů. Uživatel, který spravuje daný předmět, zde má možnost vybrat paralelky, jež mají být do předmětu importovány. Snímek stránky je uveden na obrázku 3.4 na straně 32. Tato obrazovka vznikla až v rámci implementace, nevytvářel jsem k ní tedy drátěný model. Formulář je poměrně velký, takže na základě zpětné vazby od učitelů jej pravděpodobně časem přepracuji.

Nabízí se upravit formulář tak, aby zobrazoval seznam studentů, kteří studují daný předmět (s nějakým filtrováním), a dále učitele tohoto předmětu, do něhož budou importovány paralelky, přičemž tyto paralelky sestaví pověřený učitel sám.

Dále jsem provedl drobnější modifikace formulářů pro tvorbu předmětů a zadání, které jsou podrobněji popsány v části 3.3 na straně 30.

2.4 Rozšíření systému notifikací

V původním SOS notifikace fungovaly pouze v rámci webové aplikace. Uživatel se tedy musel přihlásit, aby zjistil, zda nenastala nějaká nová událost. Pro vysokoškolské použití to není příliš na závadu, jelikož uživatelé jsou disciplinovanější a dokážou si vytvořit návyk čas od času se do aplikace přihlásit. Pro středoškoláky to však přestavuje zásadní nevýhodu. Bylo tedy třeba navrhnout externí zasílání notifikací.

Nejprve jsem notifikace SOS kategorizoval. Rozhodl jsem se zavést čtyři typy notifikací: notifikace ohledně zadání (zadání je ke schválení, zadání je schváleno), notifikace týkající se odevzdání (řešení ke kontrole, řešení ohodnoceno), notifikace týkající se týmů (např. informace o přidání studenta do týmu, o pozvání do týmu atd.), notifikace o blížícím se termínu kontrolního bodu (či jiné časově citlivé události v SOS).

2.4.1 Tabulka notifikací

V dalším kroku jsem navrhl tabulku s informacemi o tom, jaké notifikace chce uživatel dostávat e-mailem. Jednou možností by byla tabulka pouze se dvěma sloupci, přičemž v prvním sloupci by byl primární klíč uživatele a v druhém typ notifikace, kterou chce uživatel dostávat.

Tento návrh však není rozumně flexibilní, zejména s ohledem na poslední typ notifikací. Nejedná se totiž o boolean, ale zahrnuje několik možností, s jakým časovým předstihem uživatele informovat. Bylo by sice možné vytvořit tabulku, která by reprezentovala jednotlivé typy notifikací a jejich stavy, a tabulku nastavení využívat pouze jako vazební mezi tabulkou studentů a tabulkou stavů jednotlivých notifikací, ale takové řešení rozhodně není elegantní.

Druhou možností by bylo reprezentovat každý typ notifikace jako samostatný sloupec. Tento návrh je flexibilnější. Jediná potíž spočívá v tom, že při eventuálním přidání nového typu notifikace je třeba vytvořit v databázi nový sloupec a pro všechny již existující záznamy do něj přidat nějakou defaultní hodnotu. Pro implementaci jsem zvolil tuto možnost.

Dále bylo potřeba navrhnout stránku, která uživateli umožní toto nastavení měnit. Rozhodl jsem se pro jednoduchý formulář, ve kterém má každý typ notifikace své místo, přičemž jsem dbal na soulad nové stránky s původním designem SOS.

2.4.2 Notifikace o blížícím se termínu odevzdání

Notifikace o blížícím se termínu odevzdání jsou specifické v tom, že nemají spouštěč přímo v SOS. Jsou závislé na čase, a je tedy potřeba mít samostatnou komponentu, která určí čas odeslání notifikace konkrétnímu uživateli. Jednou z variant řešení tohoto problému by bylo použití klasického cronu, ale tento návrh jsem zavrhl z důvodu horší flexibility (nemožnosti jednoduše upravit frekvenci kontroly). Navíc by mohly být potíže s integrací cronu do SOS.

Obrázek 2.1 Obrazovka nastavení notifikací

Hledal jsem proto řešení, které by bylo dobře integrovatelné do Djanga. To jsem nalezl v aplikaci Celery a jejího doplňku django-celery-beat, který zajišťuje přímou integraci Celery do Djanga. Více se čtenář o této aplikaci dočte v části 1.5.8 na straně 17. V případě potřeby lze tedy nastavení frekvence spouštění funkce zajišťující případné odesílání notifikací upravovat rovnou v Djangu. Navíc se veškeré nastavení Celery provádí přímo v Djangu, není tedy při nasazení potřeba konfigurovat další nástroj. Podrobněji je tato funkcionalita popsána v části 3.4 na straně 33.

2.4.3 Nastavení notifikací

Rovněž bylo potřeba navrhnout obrazovku, na které si uživatel bude moci nastavit, jaké notifikace chce dostávat. Za tímto účelem jsem vytvořil drátěný model (viz obrázek 2.1), který jsem následně implementoval do systému.

2.5 Jednodušší způsob nasazení

Vzhledem k tomu, že se projekt celkově zvětšil, jsem se rozhodl navrhnout jednodušší způsob nasazení. Zvětšení je způsobeno především přidáním aplikace Celery, která má dva samostatné procesy a navíc potřebuje externí aplikaci pro přenos zpráv. K tomu jsem zvolil nástroje Docker [20] a Docker Compose [21]. Díky nim nebude potřeba na serveru provádět tolik nastavování. Odpadne například konfigurace překladů a kompilace statických souborů. Nasazení se tedy zjed-

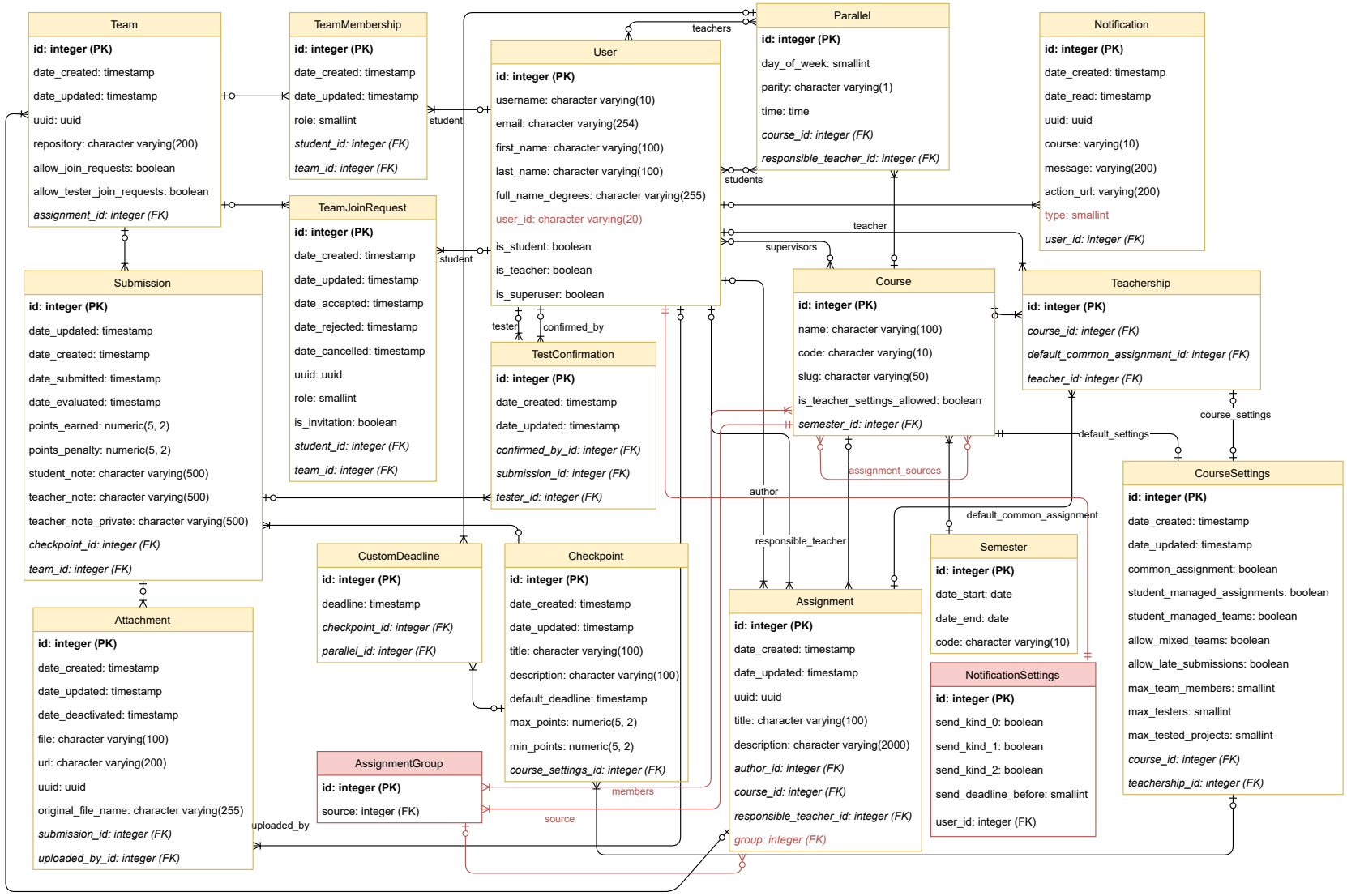
noduší z původního skriptu na pullnutí nového kontejneru a spuštění migrací. Výjimkou je první nasazení, které je stále poměrně komplikované, jelikož se musí nastavit podpůrné služby jako třeba NGINX.

2.6 Databázový model

Výše popsaná rozšíření SOS obnášela úpravu databázového schématu aplikace. V této části jsou shrnuty veškeré změny v tomto schématu.

Bylo potřeba přidat tabulku pro nastavení notifikací, do tabulky notifikací přidat typ notifikace a navrhnout reprezentaci skupiny předmětů. Poslední změnou v databázovém schématu bylo vytvoření tabulky reprezentující skupinu zadání. Důvody pro všechny tyto změny jsou popsány v příslušných předchozích sekcích této kapitoly.

Na obrázku 2.2 na straně 25 je databázové schéma, v němž jsou jednotlivé změny zvýrazněny červenou barvou.



Obrázek 2.2 ER diagram rozšířené databáze systému SOS

Kapitola 3
Implementace

V této kapitole je podrobněji popsána implementace jednotlivých požadavků na rozšíření, jak vyplynuly z podrobné analýzy a návrhu.

Předchozí kapitola byla věnována požadavkům na rozšíření SOS a popsal jsem v ní možné přístupy k jejich implementaci. Jedná se o tyto oblasti:

- Import dat
- Přihlašování do systému
- Reprezentace ročníkových prací
- Rozšíření systému notifikací

3.1 Import dat

Import dat řeší SOS aplikací `oauth`, přičemž původní struktura této aplikace je znázorněna na obrázku 3.1 na straně 28.

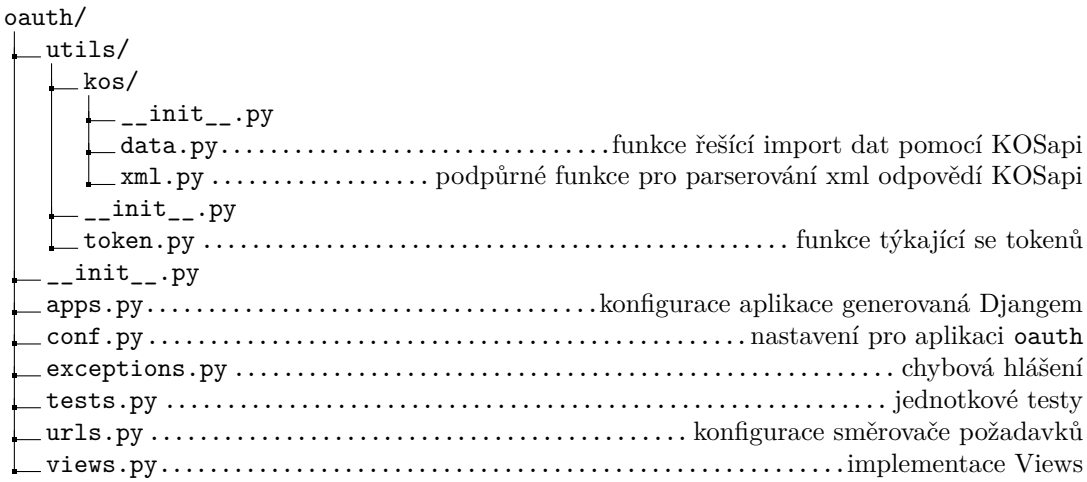
V rámci přidání nového způsobu importace dat jsem tuto aplikaci refaktoroval následujím způsobem. Místo modulu `kos` nacházejícího se v modulu `utils` jsem vytvořil modul `data_provider`, ve kterém jsem implementoval dvě třídy `BakalariApi` a `KosApi`. Dále jsem do tohoto modulu přesunul soubor xml.py, jenž slouží k jednoduššímu parserování xml dat vracených z rozhraní KOSapi. Rovněž jsem v tomto modulu vytvořil modul `tests` pro uložení jednotkových testů jednotlivých způsobů importu dat. Struktura aplikace `oauth` po popsaných modifikacích je na obrázku 3.2 na straně 28.

Jednotlivé třídy v modulu `data_provider` mají jednotné rozhraní a veškerá logika potřebná pro změnu importu je na jednom místě. Konkrétně v nově vzniklé třídě `DataProviderMixin`, která slouží pro výběr správného objektu pro import dat. Mixin obecně je třída, jež zprostředkovává své metody jiným třídám, aniž by musela být jejich nadtřídou. Implementovaný mixin se pak importuje do Views všech aplikací, v nichž je potřeba pracovat s importem dat. Na výpisu kódu 3.1 na straně 29 je tento mixin uveden.

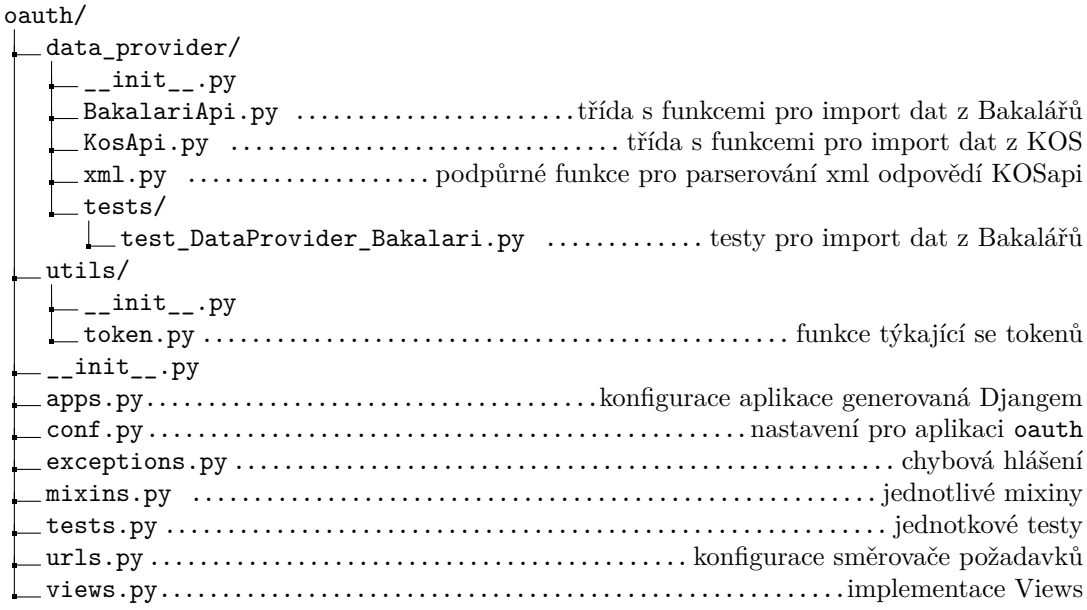
3.1.1 Funkce pro import dat z Bakalářů

V této části jsou podrobněji popsány jednotlivé funkce v třídě `BakalariApi`.

Do SOS bylo třeba naimplementovat funkci pro získání uživatelských dat. Oproti analogické funkci v třídě zajišťující import dat z KOS je o něco složitější. Důvodem je, že Datový konektor



Obrázek 3.1 Původní adresářová struktura aplikace `oauth`



Obrázek 3.2 Nová adresářová struktura aplikace `oauth`

Výpis kódu 3.1 Mixin určující třídu pro import dat

```
class DataProviderMixin:
dataProvider = None

def setup(self, request, *args, **kwargs):
    super().setup(request, *args, **kwargs)
    self.dataProvider = self.switch(settings.DATA_PROVIDER)

def switch(self, value):
    providers = {
        'Kosapi': KosApi.KosApi,
        'Bakalari': BakalariApi.BakalariApi
    }
    provider = providers.get(value)
    return provider
```

Výpis kódu 3.2 Získání předmětů vyučovaných konkrétním učitelem

```
#gets the current user's timetable
teacher = request.user.user_id
timetables = self.
_retrieve_resource(f"timetable/permanent/teacher/{teacher}")
.get('Cells')

df = pd.json_normalize(timetables, ['Atoms'])
#dropping information from timetable that isn't needed
df.drop(['Cycles', 'Stamps', 'Class.Name', 'Teacher.Abbrev',
        'Room.Id', 'Room.Abbrev', 'Room.Name', 'Group.Id',
        'Group.Abbrev', 'Group.Name', 'Teacher.Id', 'Teacher.Name',
        'Class.Id', 'Class.Abbrev', 'Subject.Id'],
        axis=1, inplace=True)

teaches = df.groupby(['Subject.Abbrev', 'Subject.Name'])
            .count()
            .reset_index()
            .to_dict('tight')
supervised_courses = {l[0]: l[1] for l in teaches.get('data')}
```

Bakalářů není navržen příliš šťastně. Chybí endpoint, který by vrátil konkrétního uživatele, a má rozdělené endpointy pro zaměstnance a studenty. Pro získání informací o konkrétním uživateli je tedy třeba načíst informace o všech studentech a všech zaměstnancích a pak v těchto seznamech hledat, zda se v jednom z nich daný uživatel vyskytuje. Pokud se požadovaný uživatel nenachází ani v jednom seznamu, funkce vrací informaci o tom, že uživatel neexistuje. V případě nalezení vrací funkce údaje potřebné k vytvoření či aktualizaci uživatele. Tato funkce byla na implementaci nejjednodušší, protože potřebné endpointy byly k dispozici.

Další funkce jsem musel vzhledem k podivnému návrhu Datového konektoru řešit méně elegantně. Kdyby totiž škola měla řádově tisíce studentů, systém by zpomalovaly. V těchto funkcích bylo potřeba znát vztah mezi konkrétními předměty a uživateli. Tuto informaci poskytuje Datový konektor pouze v rámci rozvrhu, nebo je v jeho dokumentaci zatajená. Je tedy třeba načíst studentské nebo učitelské rozvrhy a z nich získat potřebné údaje. Ukázka tohoto postupu je na výpisu kódu 3.2.

3.2 Přihlašování do systému

V původním SOS je Django aplikace `oauth`, která mimo jiné řeší přihlašování do systému pomocí fakultního OAuth serveru. Tuto aplikaci bylo potřeba upravit, aby systém podporoval více způsobů přihlášení. Jedním z požadavků Gymnázia Jiřího Gutha-Jarkovského bylo vyřešit přihlašování do systému pro správu ročníkových prací co nejjednodušeji. Vzhledem k tomu, že všichni studenti i vyučující mají školní adresy v rámci Google Workspace for Education, které používají pro přístup do Google Classroom, jsem k přihlašování využil službu Google OAuth. Na implementaci Google OAuth do různých typů aplikací má Google pěkně vypracovaný návod. [30]

V aplikaci `oauth` se nachází soubor views.py, který řeší logiku přihlašování. Do něj jsem přidal dvě třídy pro jednotlivé fáze přihlašování.

První fáze je jednodušší, jedná se o připravení položky state, která ověřuje, že při procesu přihlášení nedošlo k podvržení odpovědi. V položce state je také uložena informace, kam uživatele po přihlášení přesměrovat. Po vygenerování položky state je potřeba připravit odkaz, na který je uživatel následně přesměrován. To v SOS zajišťují metody z knihovny *google_auth_oauthlib*.

Druhá fáze spočívá v ověření informací zaslaných Googlem a přihlášení uživatele do SOS. Nejdříve je tedy potřeba získat access token: SOS odešle Googlu autorizační kód a Google mu vrátí access token. Na základě access tokenu získá SOS z Google API informace o přihlašovaném uživateli (což je v tuto chvíli e-mailová adresa), které jsou potřeba pro kontrolu, zda uživatel existuje v Bakalářích. Pokud v Bakalářích existuje, je přihlášen a je mu vytvořen uživatelský účet, případně je mu uživatelský účet aktualizován. Pro ilustraci je na obrázku 3.3 znázorněn diagram procesu autorizace Google OAuth.

Dále jsem do souboru urls.py přidal dvě nové cesty. První uživatele přesměruje na přihlašovací stránku Googlu a druhou používá Google pro přesměrování uživatele zpět do SOS.

V souvislosti s přihlašováním bylo potřeba upravit soubor views.py v adresáři core/. Tento adresář obsahuje pomocné funkce a další položky, které přímo nesouvisejí s žádnou logickou částí systému. Nejsou proto zařazeny do žádné z aplikací. V tomto souboru jsem refaktoroval metodu `get_login_url`, aby v závislosti na importu dat přepínala způsoby přihlašování uživatelů. Momentálně se ke každému způsobu importování dat váže pouze jeden způsob přihlašování, nepočítám-li přihlašování pomocí lokálního účtu, které je dostupné pouze v rámci testovacího provozu.

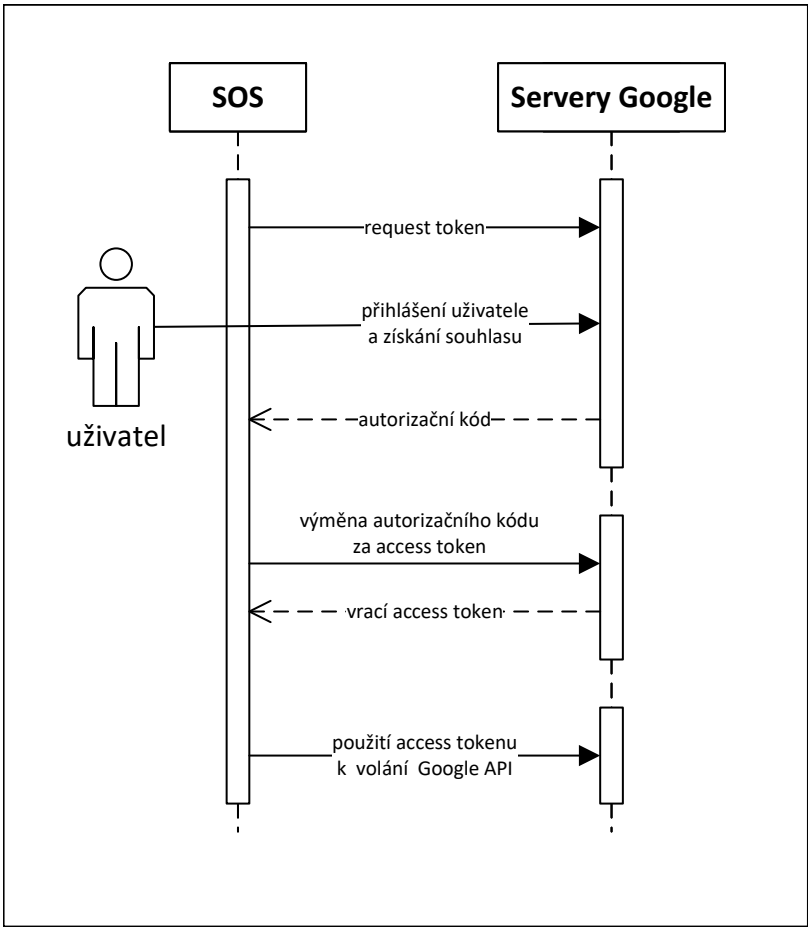
3.3 Reprezentace ročníkových prací

Z hlediska implementace byla reprezentace ročníkových prací nejobtížnější. Jak jsem uvedl v části 2.3, rozhodl jsem se reprezentovat ročníkové práce pro jednotlivé ročníky gymnázia jako jednotlivé předměty v SOS. To znamená, že bylo potřeba naprogramovat, jak předmětu umožnit zobrazení zadání z více předmětů.

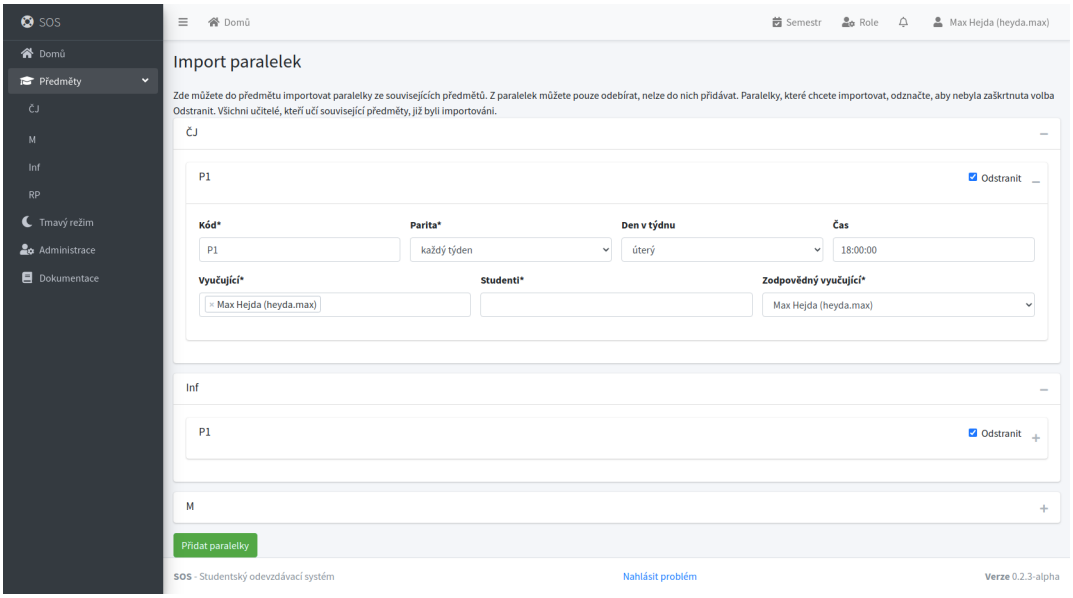
3.3.1 Změny v aplikaci Courses

V této aplikaci jsem upravil soubor base.py v modulu `views`, konkrétně třídu `CourseCreateView`, a dále v něm vytvořil novou třídu `ParallelImportView`, která zajišťuje import paralelek pro předměty, jež mají zdroje zadání.

V třídě `CourseCreateView` jsem do metody `form_valid` přidal ukládání zdrojů zadání pro nově vznikající předmět. Dále jsem tuto metodu rozšířil o kód, kterým se importují vyučující do vytvářeného předmětu z jednotlivých zdrojů zadání. Import vyučujících může proběhnout hromadně, protože je potřeba, aby všichni učitelé, kteří vyučují předmět, jenž je zdrojem zadání, mohli vytvářet a hodnotit zadání i ve vytvářeném předmětu.



Obrázek 3.3 Proces autorizace Google OAuth



Obrázek 3.4 Obrazovka importu paralelek

Nově vzniklá třída `ParallelImportView` zajišťuje import paralelek ze zdrojů zadání do vytvářeného předmětu. Jedná se o prototyp, který vznikl kvůli zajištění funkčnosti. Třída zobrazuje správci předmětu formulář, v němž si vybírá paralelky z jednotlivých zdrojů zadání, které chce do předmětu importovat. Zobrazení tohoto formuláře je poněkud uživatelsky nepřívětivé, některé funkce nejsou úplně doladěny, ale při správném použití je import studentů celkem jednoduchý.

Prozatím si správce předmětu, který importuje studenty, musí dát pozor, aby neimportoval nějakého studenta do více paralelek. Dále by měl hlídat, aby se nepokusil importovat nějakou paralelku víckrát. Zatím jsem ošetřil, že se v tomto formuláři nezobrazují studenti, kteří v předmětu, do něhož správce předmětu paralelky importuje, již jsou. Tuto stránku mám v plánu přes léto vylepšit a zmíněné problémy odstranit. V tuto chvíli jde jen o prototyp, na němž mohu předvést funkčnost rozšířeného SOS včetně vytváření předmětů a importu paralelek.

Na stránku s tímto formulářem se správce předmětu dostane ze stránky obsahující nastavení předmětu. Čtenář její ukázku nalezne na obrázku 3.4.

3.3.2 Změny v aplikaci Assignments

V této aplikaci bylo potřeba upravit tvorbu, editaci a zobrazování zadání, aby mohla pracovat s novými možnostmi v předmětu. Učitel, jenž vytváří zadání, by měl být schopen vytvářené zadání přenést i do předmětů, které mají tento předmět jako zdroj zadání.

Jedná se o docela komplexní operaci. Nejprve je potřeba vytvořit skupinu zadání, která bude určovat, z jakého předmětu zadání pochází a v jakých všech předmětech je zadání uloženo. Skupinu zadání reprezentuje nová třída `AssignmentGroup` v souboru models.py v této aplikaci. Tato třída neobsahuje žádná databázová pole, nýbrž pouze dvě relace, obě odkazující na model `Course`. První relace je typu 1:N a reprezentuje, z jakého předmětu zadání pochází, druhá je typu M:N a reprezentuje všechny předměty, ve kterých se dané zadání nachází.

Po vytvoření nebo nalezení skupiny zadání lze přistoupit k vytvoření samotného zadání. Pokud učitel nezaškrtl možnost, aby zadání vzniklo pouze v souvisejících předmětech, vytvoří se zadání nejprve v tomto předmětu a následně i v souvisejících předmětech. V opačném případě se zadání vytvoří pouze v souvisejících předmětech. Poté je potřeba vytvořit a rozkopírovat přílohy k jednotlivým zadáním. Každé zadání může mít nula nebo více příloh. Příloha k zadání je soubor

nebo webová stránka a slouží k přidání extra materiálů k zadání. Toto přiřazení a uložení má na starosti `AssignmentUpdateMixin`. Postupoval jsem tak, že nejprve se uloží přílohy k prvnímu zadání a poté se rozkopírují i do ostatních zadání.

Při úpravě zadání je potřeba rozhodnout, jestli se mají aktualizovat všechna zadání, nebo jen upravované zadání. Do formuláře pro aktualizaci zadání jsem pro tento účel přidal nové zaškrtávací pole. Pokud jej učitel zaškrtne, zadání se aktualizuje pro všechny předměty, ve kterých se dané zadání nachází. Vlastní postup aktualizace zadání je velmi podobný tvorbě zadání. Nejprve se aktualizují jednotlivá zadání, následně se aktualizují přílohy k prvnímu zadání a poté se rozkopírují do ostatních zadání.

Zobrazení zadání bylo potřeba upravit tak, aby bylo zřejmé, z jakého předmětu zadání pochází. Změnil jsem tedy soubor views.py a v něm metodu, jež řeší zobrazování zadání uživateli. Vytvořil jsem nový atribut, který je předáván šablonovacímu systému. Atribut obsahuje zadání seskupená podle jejich příslušnosti k jednotlivým předmětů. Na frontendu jsou pak všechna zadání zobrazena jak v souhrnném seznamu, tak v samostatných seznamech podle předmětu, z něhož zadání pochází. V souhrnném seznamu je u každého zadání, jež přísluší k nějaké skupině zadání, uveden i předmět, ze kterého zadání pochází. Příslušnost k předmětu se zobrazuje i na detailu daného zadání.

3.4 Rozšíření systému notifikací

Rozšíření aplikace `notifications` byl nejlehčí úkol. Do souboru models.py této aplikace přibyla třída `NotificationSettings`, která reprezentuje nastavení e-mailových upozornění pro uživatele.

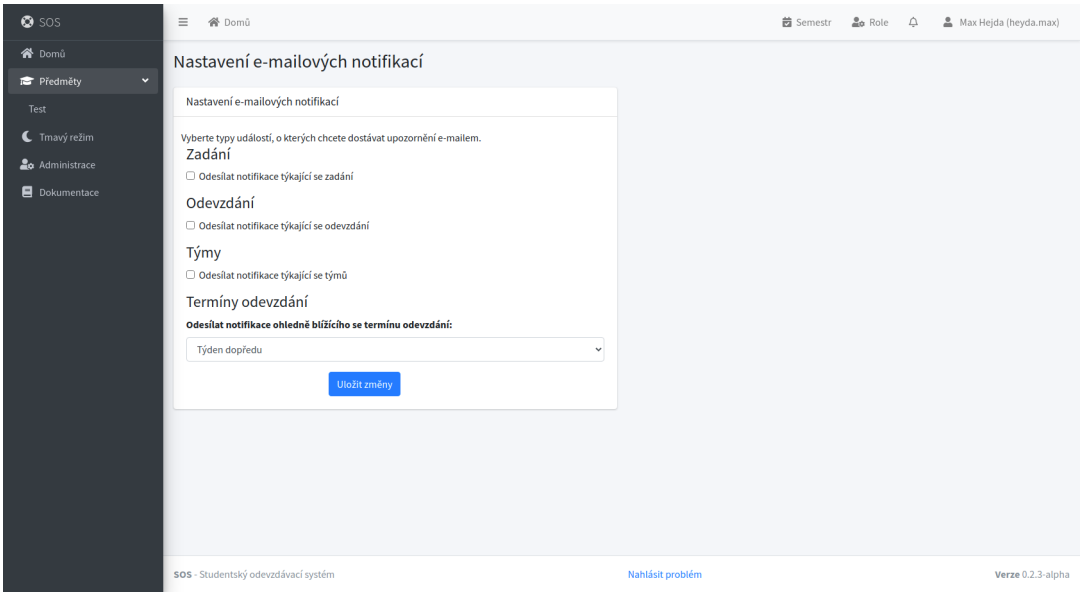
V této třídě jsou jako jednotlivé atributy uvedeny jednotlivé typy notifikací, jak bylo zmíněno v části 2.4 Rozšíření systému notifikací na straně 22. Dále tato třída obsahuje subtřídu `DeadlineChoices`, která představuje, s jakým předstihem mají být zasílána upozornění na blížící se termín odevzdání. Momentálně nabízí rozšířený SOS zasílání s předstihem dva týdny, týden, pět dnů, tři dny, den, případně vůbec. Jako defaultní hodnota je vypnuto, tj. uživatel nedostane žádné upozornění na blížící se termín odevzdání, pokud ho neaktivuje. Pro správu nastavení jsem vytvořil novou obrazovku, na kterou se uživatel dostane odkudkoliv a na níž si může vybrat, jaké notifikace chce dostávat e-mailem. Upozornění týkající se zadání (ke schválení, schváleno) se zasílají defaultně, přičemž defaultní nastavení lze změnit velmi jednoduše. Výsledná implementace této obrazovky je uvedena na obrázku 3.5 na straně 34.

Jedním z typů notifikací jsou upozornění na blížící se termín odevzdání. To jsem realizoval pomocí doplňku django-celery-beat. Do aplikace `notifications` jsem přidal soubor tasks.py s funkcí, kterou spouští celery worker v předem nastavené časy. Tato funkce projde všechny týmy v aktuálním semestru a vypočte zbývající počet dní do data nejbližšího nadcházejícího kontrolního bodu.

Pokud je zbývající počet dní roven některému nastavení, funkce `check_deadline` provede kontrolu, v jakém stavu je příslušná práce. Je-li ve stavu rozpracována nebo vrácena k dopracování, tatáž funkce zkontroluje všechny studenty v daném týmu. Jestliže někteří z členů týmu mají nastaveno, že chtějí dostat upozornění o blížícím se termínu odevzdání s tímto předstihem, je jim odeslán e-mail s upozorněním.

Pakliže je jejich práce ve stavu čekající na ohodnocení a učitel si nastavil upozornění právě s tímto předstihem, odešle se e-mail učiteli. Odesílání e-mailů vyučujícím je ošetřeno tak, aby dostali maximálně jeden e-mail z každého předmětu, který učí.

Veškerá konfigurace Celery se nachází v aplikaci `celery`. Jedná se v podstatě o načtení nastavení z modulu `settings`. K nim patří, jakou frontu úloh má Celery použít a jak se k ní připojí. Dále jsem do této aplikace přidal administrační příkaz, který slouží k vytvoření záznamu v databázi pro nastavení Celery crontab plánovače. Tento cron je pak využit při vytváření periodické události (ta je také uložena v databázi), která pak spouští samotnou událost řešící odesílání upozornění na blížící se termín odevzdání.



Obrázek 3.5 Obrazovka nastavení notifikací

Tvorba tohoto administračního příkazu je vidět na výpisu kódu 3.3 na straně 35.

Výpis kódu 3.3 Administrační příkaz pro vytvoření periodické události

```
class Command(BaseCommand):
help = "Creates␣the␣cron␣task␣to␣send␣deadline␣notification␣emails"

def handle(self, *args, **options):
    schedule, _ = CrontabSchedule.objects.get_or_create(
        minute='0',
        hour='3'
    )

    obj, created = PeriodicTask.objects.get_or_create(
        crontab=schedule,
        name='Deadline␣email␣notifications',
        task="apps.notifications.tasks.check_deadlines",
    )

    if not created:
        self.stdout.write(
            self.style.WARNING("Task␣already␣exists.")
        )
        return
```

Kapitola 4
Testování

Nezbytnou součástí tvorby jakéhokoli složitějšího softwaru je návrh a realizace testů jeho jednotlivých funkcí. V části 4.1 jsem podrobněji popsal jednotkové testy a rozsah kódu, který pokrývají. Část 4.4 se více věnuje uživatelským testům – scénářům, jež jsem pro tento účel připravil, jakož i zpětné vazbě od uživatelů.

4.1 Jednotkové testy

Jednou ze základních metod testování jsou jednotkové testy, které testují jednotlivé metody dané aplikace. Píše je většinou samotný vývojář, přičemž se jedná převážně o white box testy. Tedy je znám kód jednotlivých metod a testuje se přímo jejich chování. Tyto testy se také využívají ke kontrole, zda nějaká nová změna v systému nerozbila již existující a fungující část.

V SOS téměř každá aplikace (ve smyslu Django aplikace) obsahuje modul `tests` s alespoň dvěma soubory. Jeden soubor se jmenuje factories.py a obsahuje factory, která vytváří testovací data pro jednotlivé modely. V dalších souborech jsou samotné testy. V každém souboru s testy se nachází třída, která testy v tomto souboru sdružuje a obsahuje také podpůrné metody, jako například přípravu různých testovacích dat, nebo metody, které vracejí data použitá při mockování.

Pro ulehčení přípravy testovacích dat je v SOS použita knihovna *factory_boy*. Umožňuje využít stávající modely a vytvářené objekty těchto modelů doplnit náhodnými daty tak, aby objekty byly validní. [31]

Pro testy pokrývající aplikační logiku uloženou ve Views bylo potřeba simulovat chování aplikace, tedy odesílání požadavků a testování jejich odpovědí. Na tyto typy testů jsem využil testovací klient, jenž umožňuje simulovat požadavky a vrací objekty s odpověďmi, který je součástí frameworku Diango. Lze tedy například testovat, jak se požadavek přesměrovává, jaká šablona je použita pro renderování daného požadavku a jaké objekty jsou této šabloně předány. Testovací klient neslouží jako náhrada za nástroje typu Selenium, které testují vyrenderované HTML a chování webové stránky.

Jelikož jednotkové testy mají řešit co nejmenší části kódu, pro testy v aplikaci `oauth`, jež pokrývají chování nového způsobu importu dat, jsem zvolil mockování. Jedná se o nahrazení závislostí dané metody jejich zjednodušenou reprezentací, se kterou se pak pracuje v rámci testů. Dalším důvodem této volby je fakt, že i s ohledem na množství osobních údajů nebylo vhodné vázat testy na Datový konektor aplikace Bakaláři. Pro mockování dat jsem zvolil knihovnu *mock*, jež je součástí standardní knihovny *unittest* programovacího jazyka Python, a tudíž nebylo třeba instalovat nic dalšího. Pro testování importu dat z Bakalářů jsem mockoval metodu `_retrieve_resource`, která slouží k získání odpovědi z Datového konektoru a vrací tuto odpověď ve formátu JSON.

V rámci implementace jsem vytvořil 15 jednotkových testů. Ty pokrývají veškerá rozšíření, jež jsem v systému implementoval. V budoucnu je potřeba jednotkové testy doplnit tak, aby jimi byla pokryta většina systému – v tuto chvíli je testy pokryto 62 % kódu (v původním SOS bylo vytvořeno celkem 30 jednotkových testů a pokrytí kódu dosahovalo 59 %).

4.2 Manuální testování

Manuální testování je proces, při němž někdo projde aplikaci a zjišťuje, jestli vše funguje. Tento způsob testování jsem prováděl během implementace, abych si ověřil spolehlivou funkci všech implementovaných rozšíření.

4.3 Akceptační testování

Během vývoje aplikace jsem byl v kontaktu s některými vyučujícími i studenty gymnázia. Jsem si tedy téměř jistý, že s funkcionalitou systému nebudou mít žádné problémy. Bohužel jsem nemohl provést akceptační testování, protože koordinátor ročníkových prací je toto pololetí v dlouhodobé pracovní neschopnosti a vedení GJGJ trvá na tom, že k novému systému pro správu ročníkových prací se má vyjádřit především on. Akceptační testování proto proběhne co nejdřív po jeho návratu do školy.

4.4 Uživatelské testování

Uživatelské testování většinou bývá posledním testováním před nasazením produktu do prostředí klienta. Jde o testování přímo klientem podle scénářů vytvořených ve spolupráci s klientem. V uživatelském testování se testuje, zda chování produktu odpovídá požadavkům a je pro uživatele komfortní.

4.4.1 Testovací scénáře

Pro uživatelské testování jsem vytvořil tři scénáře, které pokrývají většinu důležitých rozšíření. Nezahrnul jsem do něj tvorbu předmětu.

První dva scénáře jsou určeny studentům. Třetí scénář se týká vyučujících.

4.4.1.1 První scénář

Tento scénář testuje přibližně, jak budou studenti pracovat s SOS v kontextu ročníkových prací. Jde tedy o volbu zadaní z více předmětů, vytvořených učiteli. Dále se scénář zabývá odevzdáváním prací a náhledem hodnocení. Rovněž testuje funkčnost editace nastavení notifikací.

Kroky jsou následující:

1. Uživatel T1 se přihlásí pomocí dodaných přihlašovacích údajů.
2. Vypne si notifikace na e-mail. (Na serveru není zatím smtp server a případné odesílání notifikací by skončilo chybovým hlášením.)
3. Prohlédne si zadání v předmětu RP a jedno z nich si vybere.
4. Při vytváření týmu pro práci na daném zadání pozve studenta s uživatelským jménem T2.
5. Zkontroluje, že uživatel T2 pozvánku přijal.

6. Prohlédne si požadavky na konzultace a jiné kontrolní body.
7. Odevzdá řešení kontrolního bodu s názvem „Vybrat zadání a potvrdit“.
8. Zkontroluje jeho hodnocení.
9. Odevzdá první konzultaci a jako přílohu přidá nějaké PDF.
10. Zkontroluje hodnocení (s nulovým počtem bodů).
11. Odevzdá opravu první konzultace a jako přílohu přidá nějaké jiné PDF.
12. Zkontroluje hodnocení.
13. Odevzdá finální řešení RP a jako přílohu přidá PDF a webovou stránku.
14. Uvědomí si, že přidal špatné URL a toto URL nahradí nějakým jiným.
15. Zkontroluje výsledné hodnocení RP.

Administrátor, který test provádí, se musí po 4. kroku přihlásit jako student T2 a přijmout pozvání do týmu od studenta T1. Dále musí jako učitel po 7. kroku vyhodnotit kontrolní bod s komentářem ve smyslu, že zadání studentovi schválil. Také musí ohodnotit první konzultaci po 9. kroku nulovým počtem bodů a požádat studenta o opravu. Tuto konzultaci pak opět vyhodnotí, tentokrát s nenulovým počtem bodů. Po 14. kroku ohodnotí kontrolní bod s názvem „Odevzdání RP“ nenulovým počtem bodů.

4.4.1.2 Druhý scénář

Tento scénář testuje práci jednotlivce v rámci RP a tvorbu zadání samotným uživatelem.

Kroky jsou následující:

1. Uživatel se přihlásí pomocí dodaných přihlašovacích údajů.
2. Vypne si notifikace na e-mail. (Na serveru není zatím smtp server a případné odesílání notifikací by skončilo chybovým hlášením.)
3. Prohlédne si zadání v předmětu RP.
4. Rozhodne se vytvořit si vlastní.
5. Prohlédne si, že mu bylo zadání schváleno.
6. Prohlédne si požadavky na konzultace a jiné kontrolní body.

Administrátor, který test provádí, musí jako učitel před 5. krokem zadání studentovi schválit.

Testovací data pro oba studentské scénáře jsem vytvořil ručně. Jedná se o vytvoření dvou studentů v jedné paralelce a jednoho učitele. Dále bylo také potřeba vytvořit 4 předměty, přičemž čtvrtý z nich s názvem RP má jako zdroje zadání zbylé tři předměty. V předmětu RP jsem pak vytvořil tři kontrolní body. První kontrolní bod má název „Vybrat zadání a potvrdit“. Druhý se jmenuje „1. konzultace“ a třetí „Odevzdání RP“. Nastavení tohoto předmětu jsem předem upravil tak, aby studenti směli vytvářet týmy. Pro druhý testovací scénář je potřeba povolit tvorbu zadání studenty.

Studentské uživatelské scénáře testovali studentka septimy gymnázia, která v letošním školním roce vypracovávala RP naposledy, a bývalý student gymnázia, jenž loni maturoval a principy vedení ročníkových prací si ještě dobře pamatuje. Testovací scénáře byly testovány jak na počítači, tak na mobilním telefonu.

4.4.1.3 Třetí scénář

Tento scénář slouží k otestování vytváření a editace zadání do předmětu, který má zdroje zadání.

Kroky jsou následující:

1. Přihlaste se dodanými přihlašovacími údaji.
2. V předmětu se zkratkou M vytvořte zadání, které bude také v předmětu se zkratkou RP.
3. Zobrazte si zadání v obou předmětech.
4. V popisu zadání jste nalezli nějakou chybu, upravte tedy obě zadání.
5. Chcete rozšířit zadání v předmětu se zkratkou RP, upravte tedy pouze toto zadání.

Testovací data pro učitelský scénář jsou následující. Je potřeba vytvořit minimálně dva předměty se zkratkami M a RP. Dále je potřeba jako zdroj zadání RP nastavit předmět M. Rovněž je nutné vytvořit alespoň jednoho učitele, který učí oba předměty.

4.4.2 Výsledky testování

První testovací subjekt se v SOS dobře orientoval. Mírné potíže se vyskytly se zobrazením si nastavení notifikací. Na mobilním zařízení se při testování nezobrazovaly popisky u menu, kde jsou uloženy odkazy na některá uživatelská nastavení. Tuto chybu jsem obratem opravil. Další problém testovacímu subjektu působilo nahrávání souborů v důsledku dvou závad, které ale nesouvisely s implementací, nýbrž s konfigurací serveru, na němž je aplikace nasazena. Obě tyto závady jsem rovněž opravil. Dále by uvítal při použití lokálního přihlášení možnost náhledu hesla. Vzhledem k tomu, že se s touto metodou přihlášení do budoucna nepočítá, nezadával jsem tuto úpravu jako issue do GitLabu. Pokud by přece jen měl být používán tento způsob přihlašování, bylo by velmi jednoduché navrženou úpravu provést.

Druhý testovací subjekt se také v SOS bez potíží orientoval. Mírně zmatečně působí tvorba zadání studentem, jelikož je skryta pod tlačítkem tvorby týmu. Do budoucna proto na toto tlačítko přidám popisek, který tuto skutečnost ozřejmí. Jako výtku uvedl, že při zmenšení postranního panelu se neskrývá text u volby tmavého a světlého motivu. Dále si povšiml, že na stránce Správa týmů se v tmavém motivu nesprávně vybarvuje selectbox pro zvaní členů týmů, tj. není vidět volby. Tuto chybu jsem opravil přidáním stejného CSS jako například na stránce tvorba předmětu. Jinak se mu aplikace vcelku líbila.

Kapitola 5
Závěr

Cílem této bakalářské práce bylo vytvořit systém pro správu ročníkových prací na středních školách. V rámci analýzy problematiky jsem se blíže seznámil se třemi komerčními řešeními, u nichž jsem vyhodnotil možnosti jejich využití pro tento účel. Dalším zvažovaným řešením byl SOS – Studentský odevzdávací systém, vícestránková webová aplikace vyvinutá pro účely FIT ČVUT v rámci loňské bakalářské práce Tomáše Pavlůska.

Tato aplikace splňovala řadu funkčních i nefunkčních požadavků na systém pro správu ročníkových prací a stačilo ji rozšířit o implementaci čtyř nových požadavků. Konkrétně jsem přidal podporu pro import dat z aplikace Bakaláři, dále jsem rozšířil posílání notifikací z SOS na e-mail a vytvořil novou kategorii notifikací ohledně blížícího se termínu časově citlivých událostí. Také jsem rozšířil možnost tvorby předmětů, aby bylo možné rozumně reprezentovat ročníkové práce. Rovněž jsem přidal způsob přihlašování do systému pomocí Google OAuth.

V rámci testování jsem se soustředil převážně na automatické testování, uživatelské testování bylo provedeno v minimální míře. SOS jsem již nasadil na server a je dostupný na adrese sos.truhla.cz, ovšem před jeho ostrým nasazením do provozu je v plánu rozsáhlejší uživatelské testování.

Cíl práce jsem tedy splnil. Systém pro správu ročníkových prací plně odpovídá požadavkům na digitalizaci správy ročníkových prací. V budoucnu ho lze samozřejmě ještě vylepšit o implementaci funkčních požadavků s nízkou důležitostí, které byly v první verzi vynechány. Rozšíření SOS je přínosné i pro samotný systém SOS, jejž má Bc. Tomáš Pavlůsek zájem rozvíjet v rámci svého magisterského studia.

Osobně jsem si ze své bakalářské práce odnesl řadu zkušeností, které jistě zužitkuji v praxi. Vyzkoušel jsem si mimo jiné integraci s komerčním softwarem Bakaláři, práci s cizím kódem i chvílemi trnité vyjednávání s budoucími uživateli systému.

Pravděpodobně během léta roku 2022 budou do aplikace Bakaláři doplněny školní e-mailové adresy studentů Gymnázia Jiřího Gutha-Jarkovského a vyučujících, což mi umožní provést uživatelské testování částí SOS závislých na přihlašování a importu dat z Bakalářů. V příštím školním roce by se měl systém začít využívat pro správu ročníkových prací.

Příloha A

Zápis ze schůzky

Datum: 31. srpna 2021

Systém pro správu ročníkových prací

Na začátku roku administrátor systému naimportuje nové žáky a učitele (a jimi vyučované předměty) z vybraného zdroje (mohlo by jít importovat z Bakalářů – Datový konektor). Dále nastaví v systému důležitá kalendářní data (nejzazší vybrání tématu, konec první konzultace, konec druhé konzultace, datum odevzdání RP). Také se zde budou konfigurovat další věci.

Příslušní učitelé do systému doplní jednotlivá témata. Každé téma bude mít několik možností: počet lidí, kteří si dané téma mohou zvolit; týmový projekt a jeho maximální počet členů (teoreticky i minimum). Téma se skládá z názvu, popisu zadání (témata mohou být klidně velmi široká – pak bude na studentovi, aby v rámci výběru upřesnil zadání) a výše uvedených možností. Učitelé budou mít možnost u daného předmětu přijímat i studentem vymyšlená témata (zde učitel specifikuje, kolik maximálně takových témat přijme). Student u takto vymyšleného tématu bude muset stanovit název a podrobné zadání.

Student si následně z témat vybere (u širokých témat napíše specifické zadání) a předá učiteli ke schválení. V případě týmových prací student doplní do systému i e-mailové adresy ostatních členů. Tým tedy zastupuje pouze jeden člen. Učitel mu buď téma schválí, nebo může téma odmítnout s připomínkami a možností téma např. do týdne (tato lhůta se nejspíše bude konfigurovat centrálně) opravit. Třetí možností učitele v této fázi je téma zamítnout. V případě zamítnutí tématu (i s možností opravy) studentovi opět běží lhůta. Pokud nedodá opravené téma do stanovené lhůty, „rezervované“ téma mu propadá. Jestliže si student nevybere téma včas, bude mu systémem přiděleno nějaké téma ze seznamu témat (tedy z databáze všech volných témat od všech učitelů – kritériem by mohla být vytíženost učitelů z hlediska počtu ročníkových prací).

Konzultace a následné odevzdání prací by probíhaly zhruba takto:

Student nahraje dokument (např. ve formátu pdf) s rozpracovanou prací. Systém následně u první konzultace rovnou zkontroluje délku anotace (toto se bude nastavovat nejspíš pro každý ročník zvlášť). Pokud bude příliš krátká, nedovolí systém práci předat konzultantovi; bude-li anotace moc dlouhá, systém pouze studenta upozorní.

Po předání konzultantovi (jemuž systém automaticky o odevzdání práce odešle e-mail dle nastavených preferencí) si konzultant stáhne nahraný dokument, ohodnotí ho a výsledný komentář zapíše do systému a předá studentovi spolu s výsledkem konzultace/hodnocení. Výsledek konzultace je buď uspěl (student má splněno a komentář slouží k posunutí práce vpřed), nebo zamítnuto (zde může být komentářem pouze sada připomínek nebo zde konzultant může požádat studenta o osobní setkání). V případě zamítnutí má student lhůtu na opravu a předání konzultantovi (toto předání musí proběhnout do deadlinu daného kroku). Učitel se k práci může vyjádřit i po deadlinu (ale v případě zamítnutí má již student smůlu).

Hodnocení práce by bylo dobré rozmyslet do několika sekcí (tyto by měly nějaké maximální ohodnocení, které by zase bylo možné nastavit). Učitel by tedy vyplnil bodové ohodnocení a textový komentář.

Systém bude umožňovat danou ohodnocenou práci nominovat na Zlatou Truhlu. Jaké notifikace od systému uživatelé dostanou, si budou moci nastavit (jestli chtějí info o blížícím se deadlinu, reakci učitele/studenta na práci atd.).

Do webové aplikace se uživatelé budou přihlašovat pomocí Google účtu (stejný jako u Google Classroom).

Bibliografie

1. GYMNÁZIUM JIŘÍHO-GUTHA JARKOVSKÉHO. *Příručka k ročníkovým pracím* [online]. 2019 [cit. 2022-03-15]. Dostupné z: http://new.truhla.cz/gymnazium/wp-content/uploads/priruckaRP19.pdf.
2. IEEE Recommended Practice for Software Requirements Specifications. *IEEE Std 830-1998*. 1998, s. 1–40. Dostupné z DOI: 10.1109/IEEESTD.1998.88286.
3. PETER, Eeles. Capturing Architectural Requirements [online]. 2005 [cit. 2022-03-15]. Dostupné z: https://web.archive.org/web/20201112020231/http://www.ibm.com/developerworks/rational/library/4706.html.
4. NEUMAJER, Ondřej. Platformy a systémy pro školní komunikace a spolupráci. *Řízení školy* [online]. 2020, roč. 17, č. 9, s. 38–41 [cit. 2022-05-01]. ISSN 1214-8679. Dostupné z: https://ondrej.neumajer.cz/platformy-a-systemy-pro-skolni-komunikaci-a-spolupraci/.
5. GOOGLE LLC. *Classroom Help* [online]. 2022 [cit. 2022-05-01]. Dostupné z: https://support.google.com/edu/classroom/#topic=10298088.
6. MICROSOFT. *Teams for education* [online]. 2022 [cit. 2022-05-01]. Dostupné z: https://docs.microsoft.com/en-us/microsoftteams/expand-teams-across-your-org/teams-for-education-landing-page.
7. MOODLE. *Documentation* [online]. 2022 [cit. 2022-05-01]. Dostupné z: https://docs.moodle.org/400/en/Main_page.
8. PAVLŮSEK, Tomáš. *SOS - Studentský odevzdávací systém* [online]. 2021 [cit. 2022-04-15]. Dostupné z: https://dspace.cvut.cz/handle/10467/95107. Bak. pr. FIT ČVUT.
9. PYTHON SOFTWARE FOUNDATION. *Python reference manual* [soft.]. 2021. Ver. 3.9.6 [cit. 2022-05-09]. Dostupné z: https://docs.python.org/3.9/.
10. FELDROY, Daniel; FELDROY, Audrey. *Two Scoops of Django 3.x: Best Practices for the Django Web Framework* [online]. Two Scoops Press, 2020. Alpha [cit. 2022-04-15]. Dostupné z: https://www.feldroy.com/products/two-scoops-of-django-3-x.
11. DJANGO SOFTWARE FOUNDATION. *Django* [soft.]. 2021. Ver. 3.2 [cit. 2022-05-04]. Dostupné z: https://djangoproject.com.
12. DJANGO SOFTWARE FOUNDATION. *Migrations* [online]. 2022 [cit. 2022-05-04]. Dostupné z: https://docs.djangoproject.com/en/3.2/topics/migrations/.
13. THE POSTGRESQL GLOBAL DEVELOPMENT GROUP [online]. 2022 [cit. 2022-05-04]. Dostupné z: https://www.postgresql.org/about/.
14. BOOTSTRAP [online]. [B.r.] [cit. 2022-05-01]. Dostupné z: https://getbootstrap.com/.

15. GITLAB B.V. [online]. 2022 [cit. 2022-05-02]. Dostupné z: https://about.gitlab.com/.
16. CHACON, Scott; STRAUB, Ben. *Pro git*. Apress, 2014. ISBN 1484200772. Dostupné také z: https://git-scm.com/book/en/v2.
17. EBY, P.J. *PEP 3333* [online] [cit. 2022-05-01]. Tech. zpr. Dostupné z: https://peps.python.org/pep-3333.
18. DJANGO. *How to use Django with Gunicorn* [online]. 2022 [cit. 2022-05-01]. Dostupné z: https://docs.djangoproject.com/en/3.2/howto/deployment/wsgi/gunicorn/.
19. DREAMHOST. *Web server performance comparison* [online]. 2022 [cit. 2022-05-01]. Dostupné z: https://w3techs.com/technologies/comparison/ws-apache,ws-microsoftiis,ws-nginx.
20. DOCKER INC. [online]. 2021 [cit. 2022-05-02]. Dostupné z: https://www.docker.com/.
21. DOCKER INC. [online]. 2021 [cit. 2022-05-02]. Dostupné z: https://docs.docker.com/compose/.
22. VIXIE, Paul. *cron(8) - Linux manual page* [online]. 2013 [cit. 2022-05-07]. Dostupné z: https://man7.org/linux/man-pages/man8/cron.8.html.
23. SOLEM, Ask. *Celery User Manual* [online]. 2021 [cit. 2022-04-15]. Dostupné z: https://docs.celeryq.dev/en/latest/index.html.
24. SOLEM, Ask. *Celery schedulers* [online]. 2021 [cit. 2022-05-08]. Dostupné z: https://docs.celeryq.dev/en/stable/userguide/periodic-tasks.html.
25. BAKALÁŘI SOFTWARE S.R.O. [online]. 2022 [cit. 2022-04-07]. Dostupné z: https://bakalari.cz.
26. BAKALÁŘI SOFTWARE S.R.O. [onine]. 2022 [cit. 2022-04-07]. Dostupné z: https://www.bakalari.cz/Static/konektor.
27. LINUX FOUNDATION [onine]. 2020 [cit. 2022-04-07]. Dostupné z: https://github.com/OAI/OpenAPI-Specification/blob/main/versions/3.0.3.md#specification.
28. PANDA SECURITY. *Password reuse* [online]. 2018 [cit. 2022-05-07]. Dostupné z: https://www.pandasecurity.com/en/mediacenter/security/password-reuse.
29. GOOGLE LLC. *Create a strong password and a more secure account* [online]. 2022 [cit. 2022-05-04]. Dostupné z: https://support.google.com/accounts/answer/32040?hl=en&ref_topic=3382296.
30. GOOGLE LLC. *Using OAuth 2.0 for Web Server Applications* [online]. 2022 [cit. 2022-05-04]. Dostupné z: https://developers.google.com/identity/protocols/oauth2/web-server.
31. RAPHAËL BARROIS, Mark Sandstrom. *factory_boy documentation* [online]. 2015 [cit. 2022-05-10]. Dostupné z: %https://factoryboy.readthedocs.io/en/stable/.

Obsah přiloženého média

Mimo paměťové médium se veškeré zdrojové kódy také nacházejí v GitLab repozitáři dostupném na adrese https://gitlab.fit.cvut.cz/pavlutom/sos. Do tohoto repozitáře má přístup vedoucí práce Ing. Jiří Hunka. Na požádání zařídím přístup i dalším osobám.

```
readme.txt ..... stručný popis obsahu média
src
    impl ..... zdrojové kódy implementace
    thesis ..... zdrojová forma práce ve formátu LaTeX
text ..... text práce
    thesis.pdf ..... text práce ve formátu PDF
```