

# GYMNASIUM JANA KEPLERA

Parléřova 2/118, 169 00 Praha 6



## Odevzdávací systém maturitních prací

Maturitní práce

Autor: Lukáš Veškrna

Třída: R8.A

Školní rok: 2020/2021

Předmět: Informatika

Vedoucí práce: Ing. Šimon Schierreich

Praha, 22. 03. 2022





## GYMNASIUM JANA KEPLERA *Kabinet informatiky*

### ZADÁNÍ MATURITNÍ PRÁCE

*Student:* Lukáš Veškrna  
*Třída:* R8.A  
*Školní rok:* 2021/2022  
*Platnost zadání:* 30. 9. 2022  
*Vedoucí práce:* Šimon Schierreich  
*Název práce:* Webová aplikace pro správu maturitních prací

#### *Pokyny pro vypracování:*

Cílem práce je vytvořit webovou aplikaci zjednodušující agendu okolo zadávání a odevzdávání maturitních prací. Řešitel zanalyzuje současné procesy odevzdávání maturitních prací v různých předmětech a na základě toho zformuluje požadavky na novou podpůrnou aplikaci a vybere vhodné technologie pro naplnění těchto požadavků. Aplikace tak bude umožňovat minimálně vypisování nových témat, zamlouvání vypsaných témat studenty, odevzdávání hotových prací a v neposlední řadě také hodnocení úspěšně odevzdaných prací. Součástí aplikace budou také notifikace o důležitých událostech souvisejících s maturitními pracemi. Výsledná aplikace bude řádně otestována, zdokumentována a při odevzdání bude nasazena na školním serveru.

#### *Doporučená literatura:*

- [1] MARTIN, Robert C. *Design Principles and Design Patterns*. [www.objectmentor.com](http://www.objectmentor.com), 2000. Dostupné z: [https://fi.ort.edu.uy/innovaportal/file/2032/1/design\\_principles.pdf](https://fi.ort.edu.uy/innovaportal/file/2032/1/design_principles.pdf).
- [2] FOWLER, Martin. *Patterns of Enterprise Application Architecture*. Boston, Massachusetts, USA: Addison-Wesley Professional, 2003. The Addison-Wesley Signature Series. ISBN 978-0-321-12742-6.
- [3] EVANS, Eric. *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Boston, Massachusetts, USA: Addison-Wesley Professional, 2003. ISBN 978-0-321-11252-7.
- [4] ARLOW, Jim a Ila NEUSTADT. *UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky*. 2., aktualiz. a dopl. vyd. Brno: Computer Press, 2007. ISBN 978-80-251-1503-9.

#### *URL repozitáře:*

<https://github.com/lesves/acceptor>

---

*student*

---

*vedoucí práce*

*V Praze dne 21. 10. 2021*



## **Prohlášení**

Prohlašuji, že jsem svou práci vypracoval samostatně a použil jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů. Nemám žádné námitky proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších předpisů.

V Praze dne 23. března 2022

Lukáš Veškrna



## **Poděkování**

Děkuji celému vesmíru, že se vše vždy stane, jak se stát má.





## **Abstrakt**

Práce se věnuje návrhu, implementaci a dalším vlastnostem systému odevzdávání maturitních prací. V první části popisuje získávání požadavků, jejich analýzu a postup při návrhu systému. Druhá část je věnována zvoleným technologiím a implementačním detailům. Poslední část popisuje instalaci a způsob nasazení systému a poté se věnuje návodu použití.

## **Klíčová slova**

webová aplikace, django, odevzdávací systém

## **Abstract**

The thesis deals with the design, implementation and other features of the maturita thesis submission system. The first part describes the gathering of requirements, their analysis and the process of designing the system. The second part is devoted to the choice of used technologies and implementation details. The last part describes the installation and deployment instructions and also a user manual.

## **Keywords**

web application, django, submission system



# Obsah

<b>1</b>	<b>Teoretická část</b>	<b>3</b>
1.1	Motivace . . . . .	3
1.2	Sběr požadavků . . . . .	3
1.3	Analýza požadavků . . . . .	4
1.4	Návrh systému . . . . .	5
1.4.1	Analýza aktivit . . . . .	5
1.4.2	Stav systému . . . . .	7
1.4.3	Návrh databáze . . . . .	8
<b>2</b>	<b>Implementace</b>	<b>11</b>
2.1	Výběr vhodných technologií . . . . .	11
2.1.1	Python . . . . .	11
2.1.2	Django . . . . .	11
2.1.3	PostgreSQL . . . . .	11
2.1.4	Gunicorn . . . . .	12
2.1.5	nginx . . . . .	12
2.1.6	Django Q . . . . .	12
2.1.7	OAuth 2 a django-allauth . . . . .	12
2.1.8	Docker . . . . .	12
2.1.9	PicoCSS . . . . .	13
2.2	Detaily implementace . . . . .	13
2.2.1	Django administrace . . . . .	13
2.2.2	UUID pole jako primární klíče . . . . .	13
2.2.3	Odevzdávání velkých příloh . . . . .	13
2.2.4	Rich text v zadání, posudcích atd. . . . .	13
2.2.5	Rozdělení na učitele a studenty při registraci Google účtem . . . . .	14
2.2.6	UserPassesTestMixin a .get_object() . . . . .	14
2.2.7	Datová migrace . . . . .	14
<b>3</b>	<b>Technická dokumentace</b>	<b>15</b>
3.1	Instalace . . . . .	15
3.1.1	Nastavení . . . . .	15
3.1.2	Spuštění . . . . .	15
3.1.3	Vytvoření administrátorského uživatelského účtu . . . . .	16
3.2	Návod k použití . . . . .	16
3.2.1	Administrační prostředí . . . . .	16
3.2.2	Uživatelské použití . . . . .	18
	<b>Závěr</b>	<b>25</b>
	<b>Seznam použité literatury</b>	<b>27</b>
	<b>Seznam obrázků</b>	<b>29</b>



# 1. Teoretická část

## 1.1 Motivace

V současnosti je na Gymnáziu Jana Keplera způsob odevzdávání a vedení maturitních prací poněkud komplikovaný. Neexistuje žádný přehled o zadání prací ani o proběhlých konzultacích. V některých předmětech se průběžné kontroly práce studentů a domlouvání konzultací řeší pomocí Google Classroom, což ale dosti obecně zameřené řešení a tudíž není ideální.

Jednou z výhod, které by zavedení systému správy maturitních prací přineslo, by byla možnost jejich archivování a zveřejňování. K podobným účelům dříve sloužil systém Chytré palice, ale ten je v současnosti již zastaralý a vesměs nefunkční. Navíc byl určen pouze na literární eseje.

Také neexistuje konkrétní řešení, které by bylo možné snadno použít. Různé univerzity sice mají své systémy pro správu prací (např. theses.cz, is.cuni.cz), avšak ty jsou určené specificky pro tyto instituce a není možné jich využít. Existují i systémy určené přímo na maturitní práce, avšak opět se jedná o systémy pro konkrétní instituce. Před návrhem jsem si prohlédl IS Karlovy Univerzity a částečně jsem (z veřejného archivu) inspiroval. Výhodou vlastního systému je možnost přispůsobení, kterou nám poskytuje.

## 1.2 Sběr požadavků

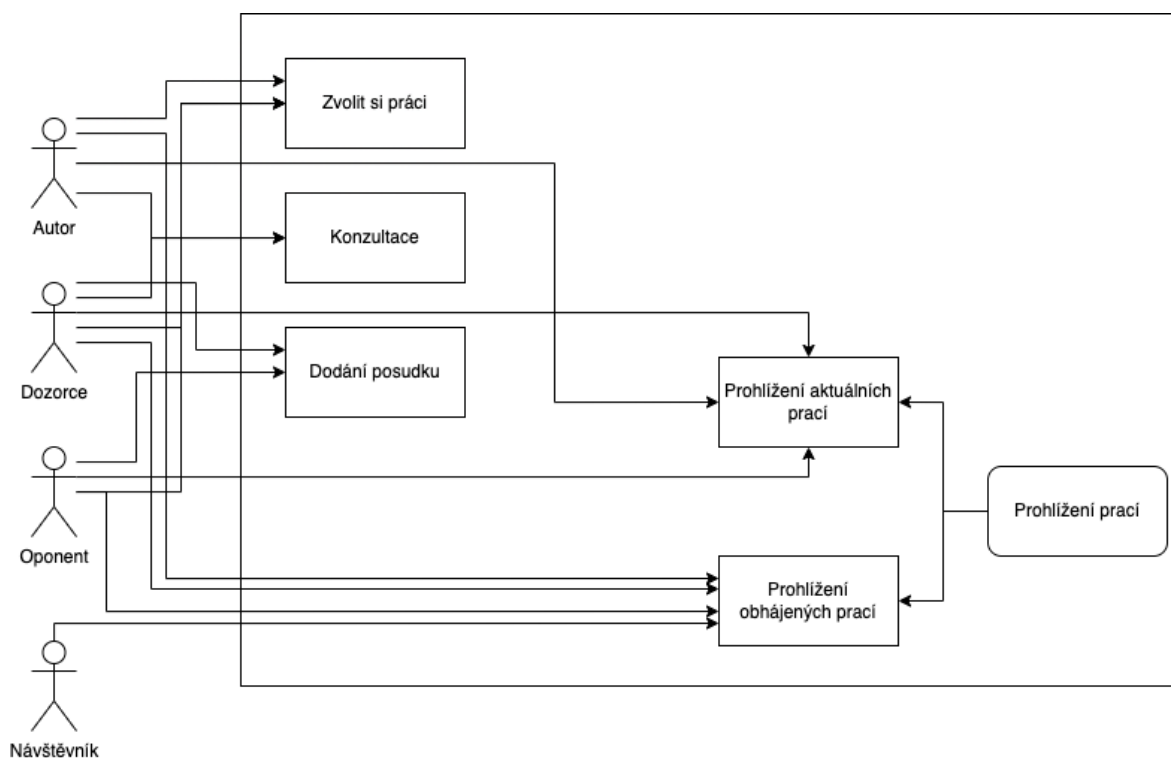
Obešel jsem tedy učitele, kteří jsou součástí maturitních komisí u předmětů, kde je součástí maturity také vypracování maturitní práce, a sestavil jsem následující seznam požadavků.

- Základní funkčnost, informatika (Šimon Schierreich)
  - Vypsání témat studenty a vedoucími
  - Přiřazení studentů/vedoucích k tématu
  - Přiřazení oponenta práce
  - Proces schvalování zadání
  - Kontrola povinných konzultací (notifikace o nich) a odevzdávání posudků
- Humanitní studia (Jan Kolář)
  - Témata rozdělená podle kategorií (sociologie, filosofie, antropologie...)
  - Možnost nahrát anotaci/resumé
  - Vyhledávání podle klíčových slov, vizí je, že systém by se mohl postupně stát jakousi databází prací
  - Nahrání posudků hodnotitelů
  - Zobrazení náhledu k práci
  - Propojení s databází CHYTRÁ PALICE (pouze navíc)
- Deskriptivní geometrie (Jakub Šebek)
  - Odevzdání libovolného typu souboru
  - Podpora nahrávání většího množství dat (např. 100MB fotek)
  - Podpora externistů
  - Větší možnosti zásahu do databáze
- Český jazyk a tvůrčí psaní (Filip Horák)

- Stejně požadavky jako HSt
- Výtvarná výchova (Alfred Filipík)
  - O systém není zájem, výtvarná výchova je o materiální tvorbě (ze stejného důvodu jsem vyloučil hudební výchovu, kde se jedná o živou ukázkou hudby)
- Technické požadavky (František Šimorda)
  - Docker (pro snadný deployment a správu)

### 1.3 Analýza požadavků

Na základě těchto požadavků jsem sestavil tzv. UML2 use case diagram (viz obrázek 1.1), který slouží k průzkumu využití, které mohou různí aktéři mít pro náš systém. Use case diagram neřeší detaily použití, soustředí se pouze na požadavky na nejvyšší úrovni (důvody pro použití systému). Také se do něj zachycují pouze funkční požadavky (nefunkčních požadavků však máme pouze minimum)[6, sekce 4.3.3].



Obrázek 1.1: Usecase diagram systému

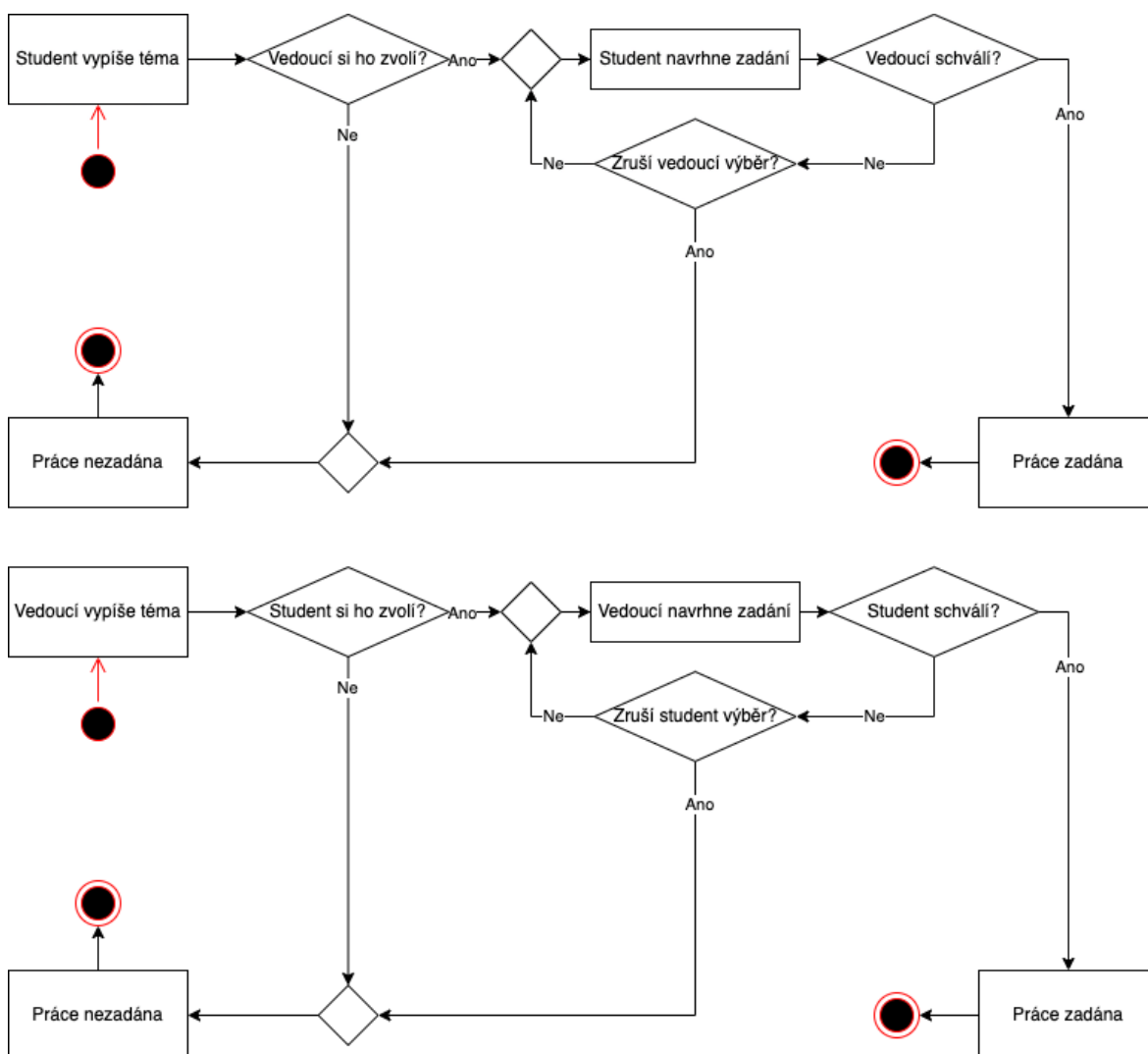
Autor a vedoucí využívají systém k prohlížení aktuálních prací, volbě práce a konzultacím. Vedoucí a oponent dodávají posudky. Oponent si také prohlíží aktuální práce (ke kterým se může přiřadit). Návštěvníci pak mohou prohlížet obhájené práce v archivu.

Bonusový požadavek na propojení s databází Chytré palice jsem vyhodnotil jako nereálný, ale nevylučuji budoucí možnost importu dat ze starého systému.

## 1.4 Návrh systému

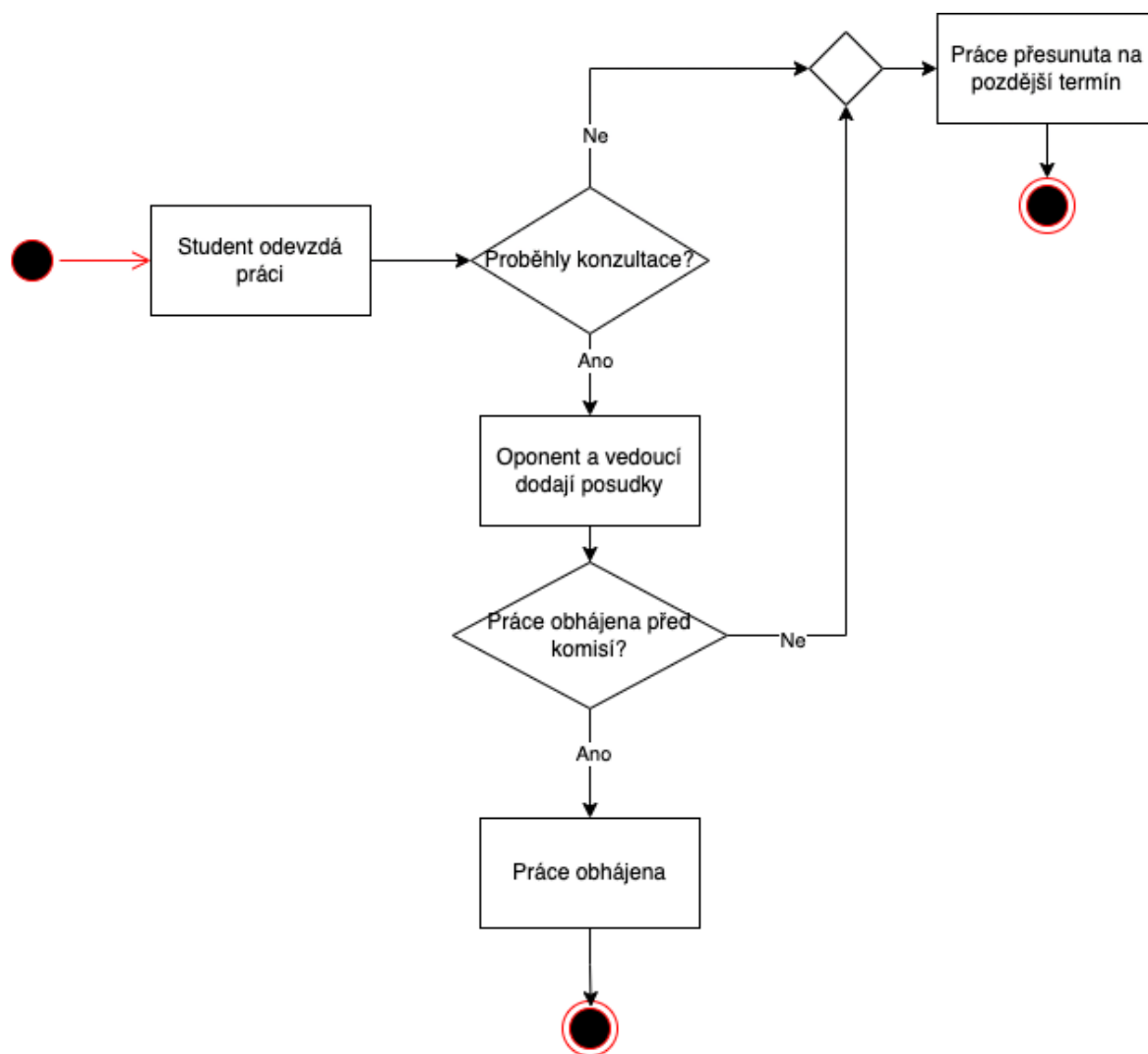
### 1.4.1 Analýza aktivit

Prvním krokem při návrhu systému po získání požadavků bylo rozpoznání činností, které budou uživatelé při používání systému provádět, a vytvoření odpovídajících diagramů aktivit, které se tyto činnosti snaží popisovat. Kromě softwaru se používají např. i pro modelování obchodních procesů a pracovních postupů[6, sekce 13.2]. Pro tyto účely jsem vypracoval diagramy na obrázcích 1.2, 1.3 a 1.4., které se zabývají výběrem tématu práce, odevzdáním a obhajobou práce a kontrolou konzultací.



Obrázek 1.2: Activity diagram pro výběr tématu práce

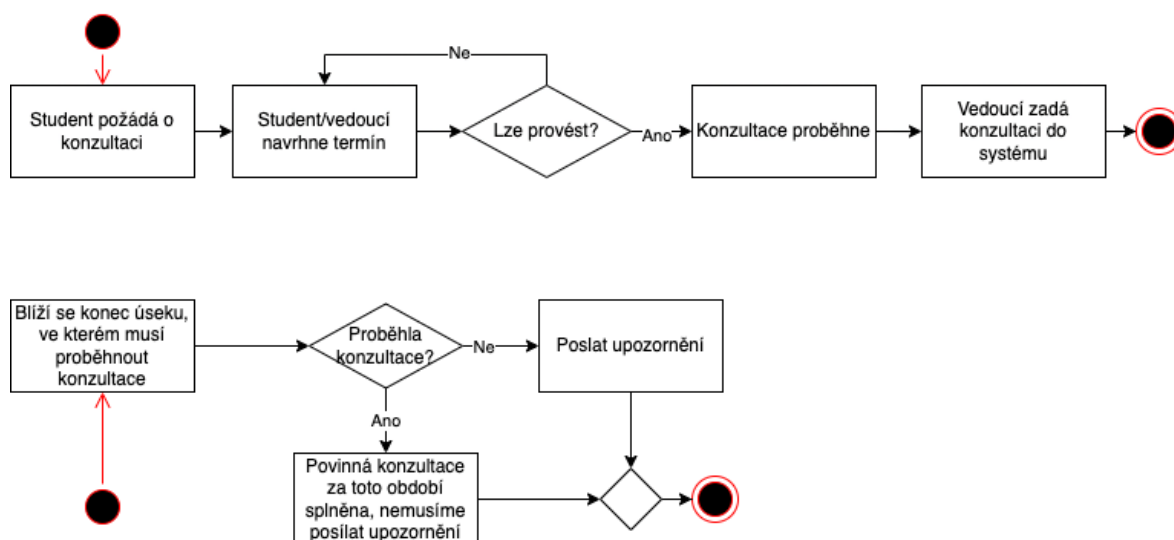
Diagram popisuje dvě symetrické aktivity. Student/vedoucí nejprve vypíše téma, které si jeho student/vedoucí zvolí. Poté se opakovaně navrhuje zadání, dokud není schváleno. V tom momentě dojde k zadání práce.



Obrázek 1.3: Activity diagram pro proces odevzdání a obhájení práce

Diagram začíná tím, že student odevzdá svou práci. V případě, že konzultace neproběhly, je práce přesunuta na pozdější termín. V opačném případě se počká na dodání posudků. Pokud je práce úspěšně obhájena před komisí, dojde k jejímu označení za obhájenou.

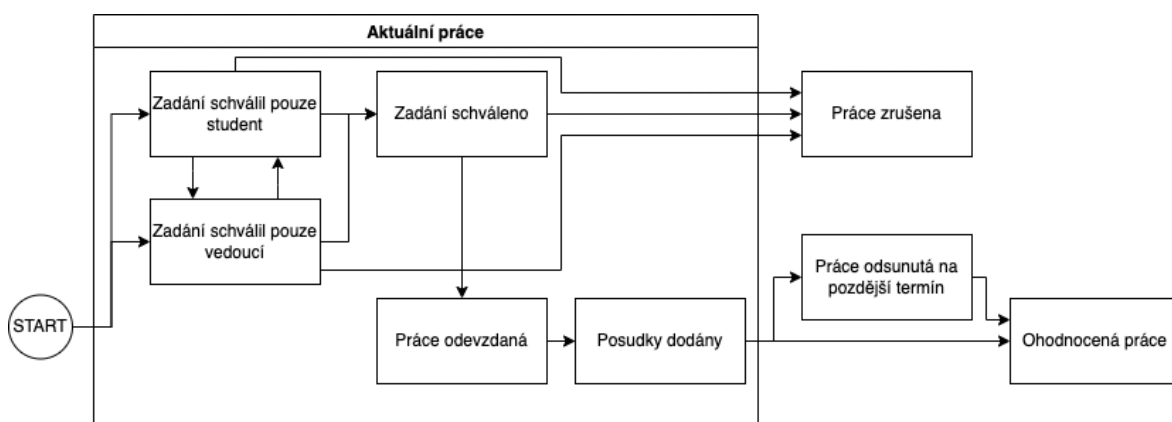




Obrázek 1.4: Activity diagram pro kontrolu konzultací

Dva diagramy popisují fungování povinných konzultací. Student a vedoucí se domluví na konzultaci, kterou po jejím proběhnutí zaneše vedoucí do systému. Na konci úseku, ve kterém student nemá splněné povinné konzultace, jsou studentu odeslány upozornění.

#### 1.4.2 Stav systému



Obrázek 1.5: Stavový diagram

Práce se dle tohoto diagramu nachází nejprve ve stavech student/vedoucí schválil zadání, mezi kterými se přesouvá podle úprav zadání, dokud i druhý neschválí nějaký konkrétní návrh. Poté se může práce odevzdávat. Po odevzdání práce následuje dodání posudků. Pak je možné odložit práci ohodnotit.

V návrhu softwaru je jedním z ústředních bodů modelování stavu. V UML2 se sestavují tzv. stavové diagramy, které nejdůležitější stavy v systému zachycují jako konečný stavový automat, což nám zjednodušuje orientaci např. v tom jaké operace mají být v nějakém stavu možné[6, sekce 19.2]. Na obrázku 1.5. je stavový diagram maturitní práce v našem systému. Ve stavovém diagramu jsou uvedeny pouze ty stavy, na které jsem pomyslel v průběhu fáze návrhu, upozorňuji, že v pozdějších fázích vývoje přibýly i nějaké další.

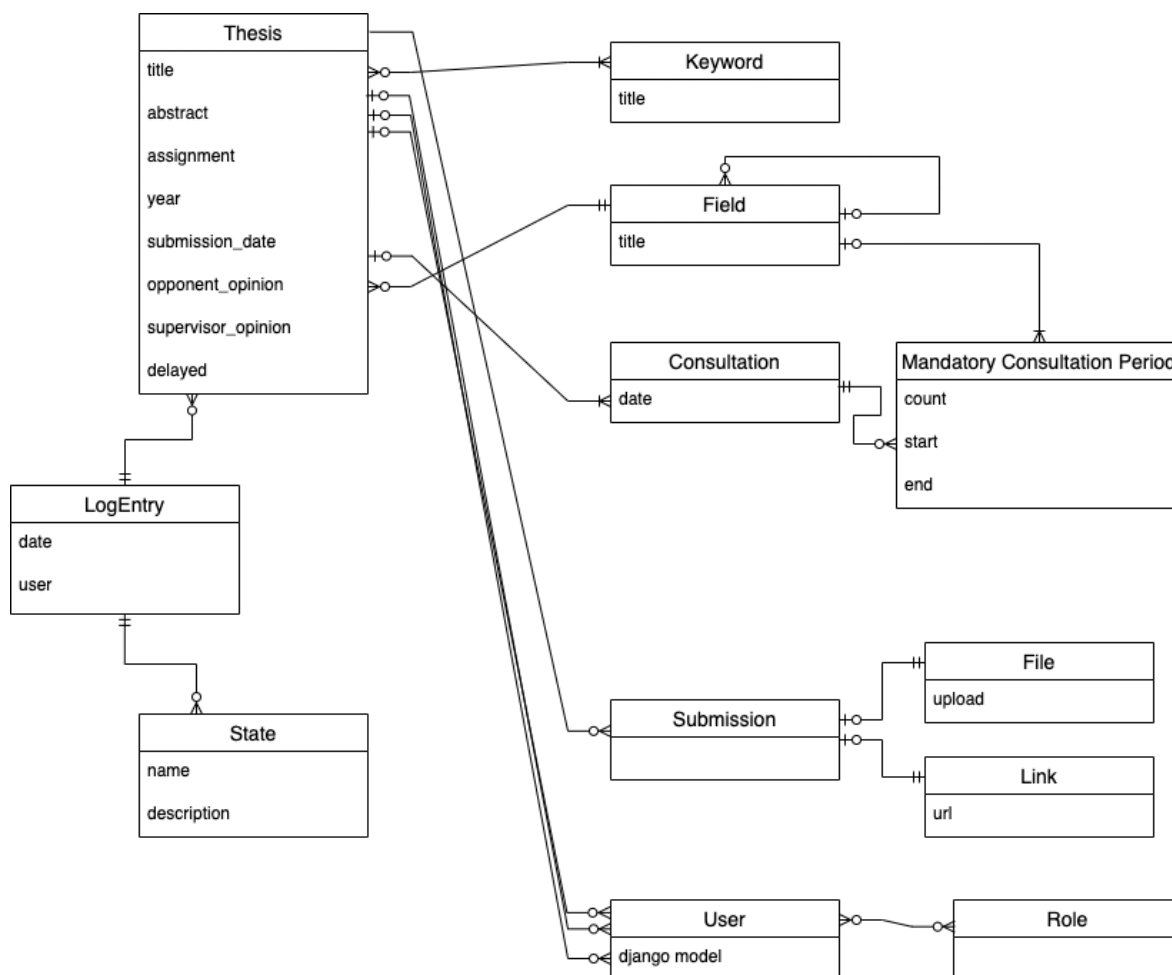
### 1.4.3 Návrh databáze

Po obecném rozmyšlení stavů, ve kterých se práce může nacházet, bylo na čase již detailněji rozmyslet, jak budeme všechna důležitá data k pracím i ke zbytku systému ukládat. Vytvořil jsem tedy konceptuální schéma návrhu databáze. Po poradě s mým vedoucím práce jsem oddělil stav práce do separátní entity, což umožňuje do budoucna lepší rozšiřitelnost systému s příchodem nových požadavků. Také jsme s vedoucím přidali log stavů práce, ve kterém je zachyceno kdo a kdy změnil stav práce.

Pro reprezentaci uživatele a skupiny uživatelů využívám dvě entity, uživatel a role. Dynamické role oproti případnému jednoduchému boolovskému poli `is_teacher` opět lépe umožňují pozdější rozšiřitelnost systému a další možnosti rozdělení uživatelů. Pro implementaci uživatelů, rolí a oprávnění jsem využil vestavěné funkce frameworku django (o tom však později v sekci Implementace).

Entita předmětu obsahuje odkaz sama na sebe. Ten slouží k umožnění libovolně hlubokých předmětových hierarchií a tedy splnění požadavku učitelů HSt. Předměty dědí povinná konzultační období svých nadpředmětů.

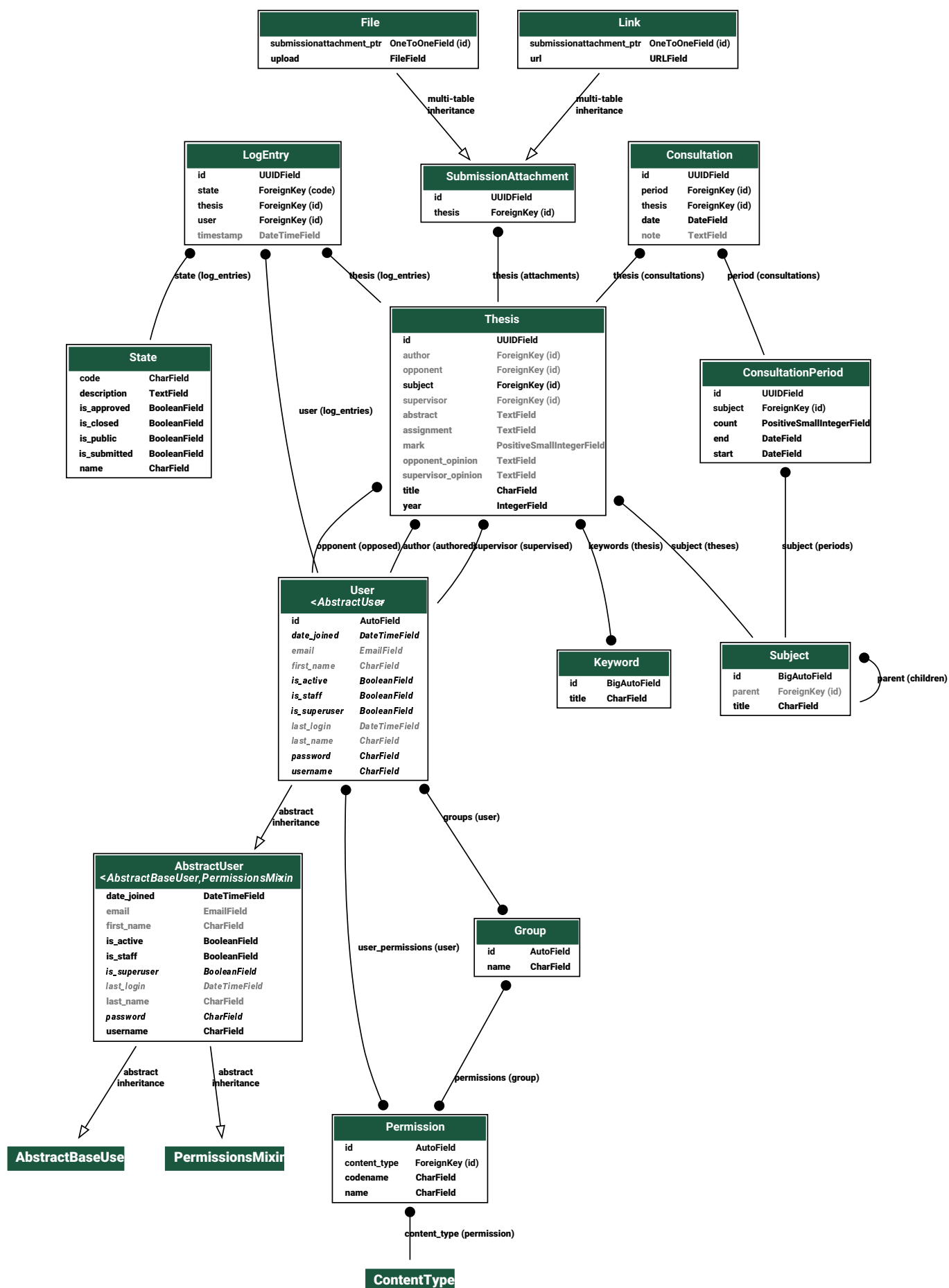
Co jsem nakonec do systému nezahrnul, ale bylo by dobrým dalším krokem v jeho zobecnění, by byla i možnost dynamicky upravovat přechody mezi stavy, které jsou zatím definovány pouze v kódu.



Obrázek 1.6: Konceptuální schéma návrhu databáze

Schéma popisuje základní entity v systému, jejich relace a některé jejich vlastnosti. Základní je práce, která má určitý počet záznamů v logu, z nichž každý obsahuje stav práce, datum změny a kdo stav změnil. Každá práce má také předmět, který může odkazovat na libovolný jiný předmět.

Předmět může mít vícero povinných konzultačních období. Práce může mít vícero konzultací, vícero odevzdaných příloh a klíčových slov. Také je zde znázorněn vztah práce s djangoovým uživatelem (tři propojení, autor, vedoucí a oponent).



## 2. Implementace

### 2.1 Výběr vhodných technologií

Už od začátku jsem měl představu systému v podobě webové aplikace. Webové aplikace mají zaslouženou popularitu právě proto, že umožňují velice snadný přístup k centralizované databázi velkému množství klientů a to s minimálními požadavky na klienty (stačí pouze webový prohlížeč).

#### 2.1.1 Python

Rozhodl jsem se systém naprogramovat v programovacím jazyce Python. Mezi jeho výhody patří zejména dobrá čitelnost a rychlost vývoje. Nehodí se však na tvorbu obrovských systémů, zvláště kvůli dynamickému typování (v posledních letech statické typování Python do jisté míry získal, avšak pro tuto práci jsem jej nepoužil) a také kvůli jeho omezené rychlosti. Náš systém však není zdaleka tak komplikovaný, aby Python mohl způsobit jakékoli problémy v jeho rozšiřování. Existují dokonce i případy velmi náročných systémů běžících na Pythonu (např. Instagram na kombinaci Python+Django[4])

Také mé rozhodnutí určitě ovlivnilo, že jsem s jazykem Python již delší dobu seznámen a dokážu tudíž dobře využít jeho předností a vyvarovat se jeho skrytých nástrah.

#### 2.1.2 Django

Rozhodl jsem se využít open source webový framework Django. Mezi jeho hlavní výhody patří snadnost použití a rychlost vývoje. Dává také důraz na bezpečnost a tudíž mezi jeho funkce patří např. robustní ochrana proti CSRF útokům, XSS útokům nebo SQL injection. Není s ním také třeba řešit kryptografii a hashování hesel, což jsou věci, které bych zejména z důvodů bezpečnosti nechtěl sám implementovat[3] Je nezávislé na zvolené databázové technologii, což se může při pozdější údržbě softwaru hodit.[1]. Také má dobrou komunitní podporu a existuje k němu mnoho knihoven a rozšíření, některé z nichž jsem použil. Užitečné je také vestavěné administrační prostředí.

V neposlední řadě mé rozhodnutí opět ovlivnilo to, že už jsem s tímto frameworkem seznámen, a také správce školní sítě s ním má zkušenosti.

#### 2.1.3 PostgreSQL

PostgreSQL se prosadila jako jedna z nejrobustnějších open source databází. Podporuje mnoho pokročilých rozšíření, které se mohou do budoucna hodit (např. fulltextové vyhledávání)[8].

### 2.1.4 Gunicorn

Gunicorn je webový server, který pomocí WSGI (standardní interface Pythonu pro webové servery) spouští Django. Gunicorn jsem zvolil na základě toho, že je jednoduché ho spustit a že je jedním ze zmíněných WSGI serverů v dokumentaci Django[2].

### 2.1.5 nginx

nginx slouží jako hlavní server, který se stará o statické soubory, nahraná média a přesměrovává requesty na gunicorn. V dokumentaci gunicornu i djanga je uvedeno[2][5], že ideální způsob hostování je právě za serverem jako je nginx (jinak je gunicorn náchylný na DoS útoky). Nginx jsem zvolil nad konkurečními servery (jako např. Apache) na základě dobré osobní zkušenosti, a také je s ním seznámen správce školní sítě.

### 2.1.6 Django Q

Pro odesílání e-mailových notifikací jsem potřeboval použít nějaký systém, který dokáže periodicky spouštět nějaké úlohy. Jednoduchým systémem, který toto řeší je např. klasický unixový daemon cron, avšak u něj mi chyběla větší integrace s djangem. Nakonec jsem se rozhodl pro Django Q, jejíž hlavní výhoda je, že narozdíl od jiných podobných systémů, nevyžaduje použití další služby jako message broker, což by mi pro mé účely přišlo jako přehnané. Důležitou výhodou Django Q je také to, že se pokouší o zopakování selhaných odesláních a opakuje úlohy, které měly běžet, když byl systém vypnutý (což by řešení využívající cron tak snadno nedokázalo).

### 2.1.7 OAuth 2 a django-allauth

Rozhodl jsem se, že bych rád v systému podporoval přihlašování a registraci pomocí Googlu, protože škola využívá služby Googlu, takže by systém byl lépe integrován se zbylými školními službami, což povede k snadnější údržbě a k méně heslům, které si uživatelé musí pamatovat. Volbu této technologie jsem konzultoval se správcem sítě, který s jejím použitím souhlasil, a má s nastavením této služby předchozí zkušenosti. Na implementaci tohoto protokolu jsem použil knihovnu django-allauth.

### 2.1.8 Docker

Od dockeru si slibuji to, že software bude fungovat stejně jak na mém počítači, tak i při běhu na školním serveru. Obrovskou výhodou je také to, že je s použitím docker-compose velmi snadné celý systém zprovoznit. Systém totiž vyžaduje čtyři běžící procesy (postgres, gunicorn+django, nginx, django-q workery), které je pomocí dockerových kontejnerů velmi snadné sesynchronizovat, aniž by musel správce nějakým způsobem řešit konkrétní procesy.

### 2.1.9 PicoCSS

Primárním cílem práce nebylo zabývat se designem webu, proto jsem použil minimalistický CSS framework PicoCSS, jehož velikost je pouze kolem 10kB, je navržený v souladu s mobilními zařízeními, dále je snadno upravitelný pomocí CSS proměnných, vede k jednoduchému HTML kódu, a má podporu dark módu, což mí vrstevníci milují[7].

## 2.2 Detaily implementace

### 2.2.1 Django administrace

Protože si (nejenom) učitelé deskriptivní geometrie přáli nějaký způsob detailnějšího zásahu do systému, rozhodl jsem se, že bude nejlepší variantou využít kromě hlavního uživatelského prostředí, určeného pro studenty a většinu učitelů, také vestavěné automaticky vygenerované administrační prostředí frameworku Django, který budou využívat technicky zdatnější uživatelé, popř. správci. Toto prostředí je velmi dobře rozšiřitelné a nastavitelné s pomocí oprávnění. Také jeho použití je vcelku intuitivní.

### 2.2.2 UUID pole jako primární klíče

Na doporučení svého vedoucího využívám jako primární klíče k většině tabulek UUID, jejichž hlavní výhodou je, že při případné chybě zabezpečení nejdou lehce uhodnout. U některých jsem však zůstal u běžných autoinkrementačních, konkrétně u předmětů a klíčových slov, o kterých jsem předpokládal, že neobsahují citlivé údaje, a mohou být veřejné, a jejich zkrácení zjednodušuje URL k vyhledávacímu formuláři.

### 2.2.3 Odevzdávání velkých příloh

Jedním z požadavků učitelů deskriptivní geometrie bylo i odevzdávání velkých příloh, což jsem po konzultaci se správcem sítě vyhodnotil jako velmi nesnadné, takže jsem se rozhodl, že odevzdávání velkých příloh se bude řešit nahráním odkazu na Google disk, který škola v současnosti již stejně využívá. V systému je implementovaná plná podpora odkazových příloh.

### 2.2.4 Rich text v zadání, posudcích atd.

Chtěl jsem uživatelům umožnit psát některé texty s dodatečným formátováním, což jsem implementoval knihovnou na editaci textu na frontendu a ověřenou knihovnou od Mozilly, bleach, na backendu, která zbavuje text všech HTML tagů vyjma povolených a tudíž zabraňuje možným XSS útokům.

### 2.2.5 Rozdělení na učitele a studenty při registraci Google účtem

Při prvním přihlášení google účtem je třeba určit zda je uživatel student anebo učitel. Domnívám se, že existuje způsob, jak tyto role odlišit pomocí API Googlu, ale nebyl jsem si jistý, jestli je naše škola používá, takže jsem zvolil o něco jednodušší řešení, kdy jsem prohlásil, že uživatelé, jejichž e-mailové uživatelské jméno začíná x a končí číslem, jsou studenti. Tento způsob asi není úplně robustní, avšak správci sítě to nevadí. Teoreticky dva učitelé s příjmením na x rozbijí systém, popř. při změně jmeného systému bude potřeba tuto logiku pozměnit (je ale izolována v nastavení projektu).

### 2.2.6 UserPassesTestMixin a .get\_object()

V Django jsem většinu views implementoval jako class based views. Pokud jsem potřeboval, aby view nedovolovalo přístup uživatelům bez nějakých pravomocí, použil jsem doporučený UserPassesTestMixin. Avšak občas se stalo, že jsem chtěl prošetřovat pravomoce uživatele podle konkrétních vlastností objektu, např. nepovolit uživatelům, kteří nemají oprávnění zobrazení práce, zobrazit práce, které nejsou zveřejněné v archivu. Toto řeším tak, že v rozhodovací metodě UserPassesTestMixin zavolám metodu `.get_object()` a tím získám objekt z databáze. Avšak obávám se, že view v takovém případě udělá dva požadavky na databázi. Dalo by se to vyřešit nepoužitím UserPassesTestMixin anebo implementací vlastní upravené verze této třídy, avšak přišlo mi to jako předčasná optimalizace.

### 2.2.7 Datová migrace

Protože v systému využíváme dynamické stavy a uživatelské role, je potřeba je vytvořit v databázi před uvedením systému do provozu. Protože jsem však nechtěl komplikovat život správcům softwaru, použil jsem tzv. datovou migraci, jejíž spuštění vytvoří v databázi potřebná data. Využívám k tomu vestavěné možnosti Djangových databázových migrací.



## 3. Technická dokumentace

### 3.1 Instalace

Nejprve si naklonujeme repozitář.

```
git clone https://github.com/lesves/acceptor.git
cd acceptor
touch .env
```

#### 3.1.1 Nastavení

Nyní je třeba vyplnit soubor `.env`, který obsahuje tajné a na provozu specifické informace (proto jej není možné dopředu připravit). Tento soubor obsahuje libovolné množství řádků ve formátu `PROMĚNNÁ="HODNOTA"`. Tyto deklarace jsou poté načteny jako environmentální proměnné, které si systém sám přebere (je tedy teoreticky možné informace předávat i jinak, právě přes environmentální proměnné). Soubor `.env` musí pro běh systému obsahovat následující proměnné:

- `DEBUG` – v productionu nesmí být `True`
- `HOST` – doména, na které webová aplikace poběží (např. `acceptor.gjk.cz` nebo `localhost`), pro nastavení vícero domén je třeba upravit soubor s detailnějšími nastaveními `acceptor/settings.py`.
- `ORIGIN` – totéž jako doména, ale obsahující i protokol a volitelně port
- `SECRET_KEY` – tajný klíč, který je použit např. pro hashování hesel. Lze vygenerovat příkazem

```
python -c 'from django.core.management.utils import get_random_secret_key;
print(get_random_secret_key())'
```

- `GOOGLE_AUTH_CLIENT_ID` – ID klienta pro přihlášení Googlem (lze nastavit na prázdný řetězec, ale Google přihlášení nebude fungovat)
- `GOOGLE_AUTH_SECRET` – tajný klíč pro přihlášení Googlem
- `EMAIL_USER` – uživatelské jméno e-mailového účtu (v defaultním nastavení je systém připraven na posílání e-mailů přes gmail (který používá škola), pokud chceme jinak, je třeba zasáhnout do detailnějších nastavení v souboru `acceptor/settings.py`)<sup>1</sup>
- `EMAIL_PASS` – přihlašovací token e-mailového účtu

#### 3.1.2 Spuštění

Tento krok vyžaduje nainstalovaný a spuštěný docker. Díky dockeru není potřeba žádná komplikovaná instalace.

```
docker-compose up
```

---

<sup>1</sup>Nakonec jsme při instalaci systému tyto nastavení upravili, protože škola používá *smtp-relay* od googlu, který vůbec nekontroluje přihlášení z určitých adres.

`docker-compose` připraví a spustí následující čtyři kontejnery:

- `db` – PostgreSQL databáze
- `web` – Django a gunicorn
- `taskqueue` – Django Q (pro spouštění pravidelných úloh)
- `nginx` – Hlavní server

Dále také následující tři sdílené oddíly (volumes):

- `static` – Obsahuje statické soubory (nutné pro poskytnutí přístupu k nim nginxu)
- `media` – Obsahuje nahrané soubory (pozor: uživatelská data)
- `postgres_data` – Obsahuje databázi (pozor: uživatelská data)

Kontejner s databází má triviální uživatelské jméno a heslo, avšak není přístupný z venčí, veškerá komunikace z těchto čtyřech kontejnerů je otevřená pouze na portu 1337 (tento port je možné upravit v `docker-compose.yml`). Pro údržbu je také možné přistupovat k jednotlivým datovým oddílům.

### 3.1.3 Vytvoření administrátorského uživatelského účtu

Pro používání systému je ještě nutné vytvořit alespoň jeden administrátorský uživatelský účet (přes který poté dokážete vytvořit veškeré další účty). Toho lze docílit spuštěním následujícího příkazu (za předpokladu, že dockerové kontejnery běží):

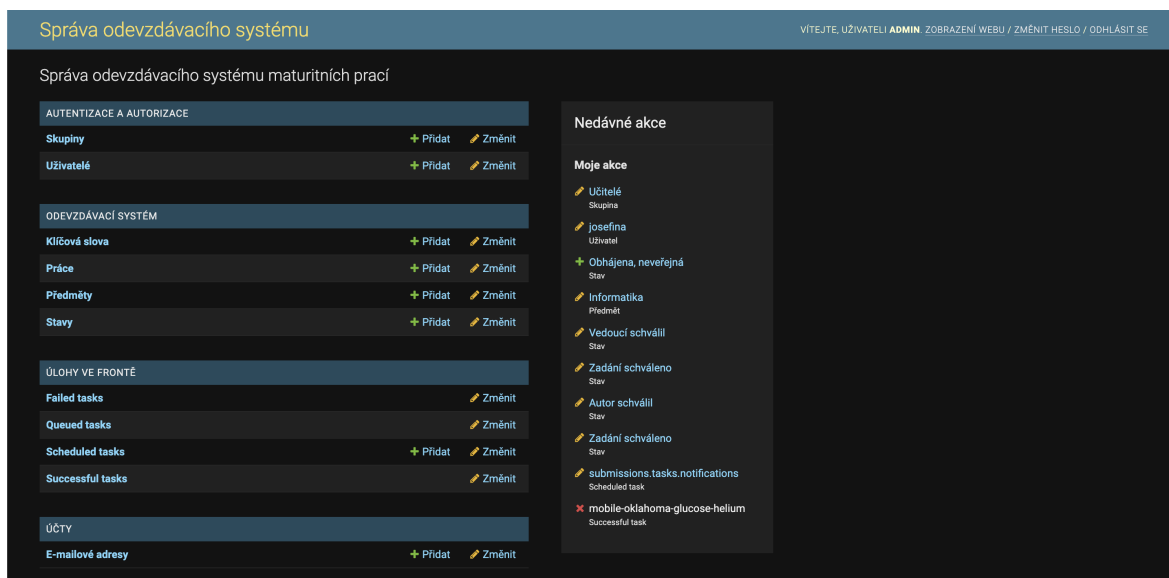
```
docker exec -it acceptor_web_1 /usr/bin/env python3 manage.py createsuperuser
```

Dojde ke vstupu do dockerového kontejneru a spuštění djangového administračního příkazu. Ten se sám zeptá na uživatelské jméno, e-mail (ten není vůbec nutné vyplnit) a heslo administrátorského účtu. Takto vytvořený účet je superuživatel, což znamená, že má automaticky všechna oprávnění bez jejich explicitního přiřazení.

## 3.2 Návod k použití

### 3.2.1 Administrační prostředí

Po spuštění systému a vytvoření superuživatele je možné se přihlásit do administračního prostředí na adrese `/admin/`.



Obrázek 3.1: Administrační prostředí, když je přihlášen superuživatel

Když je přihlášen superuživatel, ukazuje administrační prostředí i vesměs nepotřebné údaje (šlo by je skrýt úplně, ale říkal jsem si, že to někdy přecejenom může být užitečné). Pokud je však přihlášen uživatel, který není superuživatel a je například pouze členem skupiny Učitelé, uvidí pouze informace relevantní k maturitním pracím.

V sekci **Autentizace a autorizace** je možné přidávat nové uživatele (např. externisty, kteří se budou přihlašovat heslem), upravovat existující uživatele, přidávat a upravovat skupiny a nastavovat jejich oprávnění.

V sekci **Odevzdávací systém** je možné zobrazit, upravit a smazat maturitní práce, klíčová slova, předměty a stavy maturitních prací.

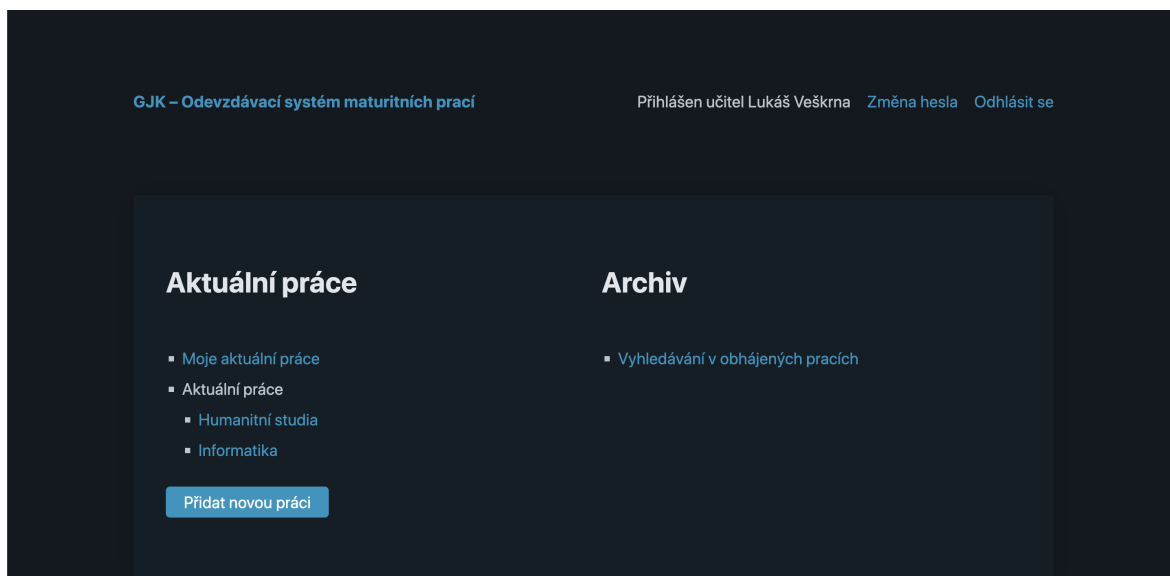
V sekci **Úlohy ve frontě** je možné zobrazit právě probíhající úlohy Django Q, tedy aktuálně odesílané e-maily, také je možné např. deaktivovat periodické odesílání e-mailů. Tato sekce je však poměrně uživatelsky nepřívětivá (neočekává se, že jí bude používat technicky nezdatný uživatel), navíc není přeložená do češtiny.

Sekce **Účty** a **Účty sociálních aplikací** jsou implementační detaily knihovny django-allauth. *E-mailové adresy* v sekci **Účty** nejsou vůbec relevantní pro fungování aplikace a šlo by je schovat (jedná se o místo, kde tato poměrně obsáhlá knihovna uchovává zdali jsou e-mailové adresy ověřené, což není důležité pro naši aplikaci), opět však nevidím důvod proč je schovávat před technicky zdatným uživatelem. *Účty sociálních sítí* jsou propojení Django účtů s Google účty.

Vyvarujte se však několika operací, které systém bez úprav v kódu rozruší. Těmi je smazání některé z dvou základních skupin uživatelů (to jest *Studenti* a *Učitelé*), dále také smazání některého ze základních stavů prací (je však možné přidávat a mazat nové stavy, také je možné že smazání určitého základního stavu systém nerozbije, např. stav *Obhájena, neveřejná* je takto redundantní).

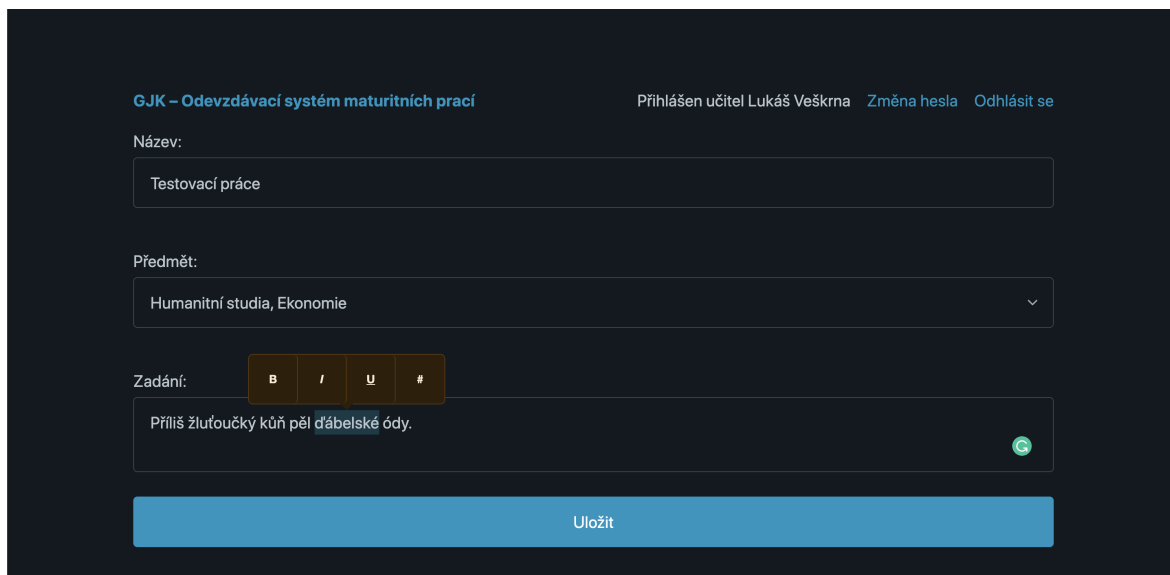
### 3.2.2 Uživatelské použití

Běžný uživatel implicitně nemá přístup do administračního prostředí. Může se přihlásit pomocí Googlu (což jej i automaticky zaregistruje) anebo pomocí hesla (což však vyžaduje vytvoření účtu administrátorem).



Obrázek 3.2: Hlavní strana – takto je přivítán přihlášený učitel.

Po přihlášení se zobrazí hlavní strana, kde je možné přejít na zobrazení prací v aktuálním ročníku (aktuálních prací) a také přidání nové práce.



Obrázek 3.3: Založení nové práce. Při zakládání práce je třeba vyplnit její název, předmět a poskytnout návrh zadání.

Po založení práce se zobrazuje v kategorii **Moje aktuální práce** a je možné upravovat její detaily. Uživatel, který ji vytvořil, je u ní zapsán buďto jako autor nebo jako vedoucí (podle jeho role v sys-

tému) Možnosti úprav závisí na přihlášeném uživateli a jeho oprávněních. Pokud má uživatel právo libovolné úpravy práce (implicitně jej mají všichni ve skupině *Učitelé*), může libovolně aktualizovat aktuální stav práce (tato skutečnost je zachycena v příslušejícím logu). Např. možnost odevzdávat soubory má pouze autor práce, možnost dodat posudek mají pouze oponent a vedoucí a tak dále.

Poté co je nastaven stav na *Obhájena* (to je možné jak s pomocí možnosti **Upravit stav**, tak i v případě práce připravené na obhajobu s pomocí **Nastavit hodnocení**), zobrazuje se práce v archivu, ve kterém může vyhledávat veřejnost.

The screenshot shows the user interface of the 'GJK - Odevzdávací systém maturitních prací'. At the top, the user is logged in as 'Josefina Abcdeová' with links for 'Změna hesla' and 'Odhlásit se'. The main title of the work is 'Abcdef', with an 'Upravit název' button. The 'Základní informace' section contains the following data:

Autor:	nepřiřazen
Vedoucí:	Josefina Abcdeová <a href="#">Smazat návrh práce</a>
Oponent:	nepřiřazen <a href="#">Přiřadit se</a>
Ročník:	2022
Předmět:	Humanitní studia, Psychologie
Známka:	-
Stav:	Vedoucí schválil
Popis stavu:	Zadání práce bylo schválené vedoucím. Čeká se na schválení autorem. (poslední změna stavu: Josefina Abcdeová 21. března 2022 22:59)

At the bottom of the section is an 'Upravit stav' button.

Obrázek 3.4: Přehled informací o práci z pohledu vedoucího.

Práce ještě nemá přiřazeného autora (ani oponenta), tudíž se vedoucímu zobrazuje pouze možnost vymazat návrh práce (místo odřazení). Pod tabulkou s informacemi se nachází tlačítko upravit stav, kterým je možné změnit stav na libovolný z databáze.

**GJK – Odevzdávací systém maturitních prací**

Přihlášen student Lukáš Veškrna [Změna hesla](#) [Odhlásit se](#)

# Abcdef

## Základní informace

Autor:	nepřiřazen
	<a href="#">Přiřadit se</a>
Vedoucí:	Josefina Abcdeová
Oponent:	nepřiřazen
Ročník:	2022
Předmět:	Humanitní studia, Psychologie
Známka:	-
Stav:	Vedoucí schválil
Popis stavu:	Zadání práce bylo schválené vedoucím. Čeká se na schválení autorem. (poslední změna stavu: Josefina Abcdeová 21. března 2022 22:59)

Obrázek 3.5: Přehled informací o práci z pohledu studenta, který k ní není přiřazen. Je zde zobrazena pouze možnost přiřazení k této práci do role autora.

## Zadání

And as we wind on down the road

Our shadows taller than our soul

There walks a lady we all know

Who shines white light and wants to show

How everything still turns to gold

And if you listen very hard

The tune will come to you at last

When all are one and one is all

To be a rock and not to roll

Upravit zadání

Schválit

Obrázek 3.6: Pohled na neschválené zadání

Po připsání může student upravit zadání nebo jej schválit. Úprava zadání změní stav práce z *Vedoucí schválil* na *Student schválil*, naopak schválení přesune práci do stavu *Zadání schváleno*, protože současnou verzi zadání schválily obě strany.



Obrázek 3.7: Seznam konzultací z pohledu vedoucího.

Za každé období se zobrazuje kolik konzultací v něm bylo splněno. Po kliknutí na konkrétní období konzultací se zobrazí seznam konzultací v něm i s jejich poznámkami (poznámky vidí pouze učitelé). Po kliknutí na **Přidat konzultaci** viz obrázek 3.8.



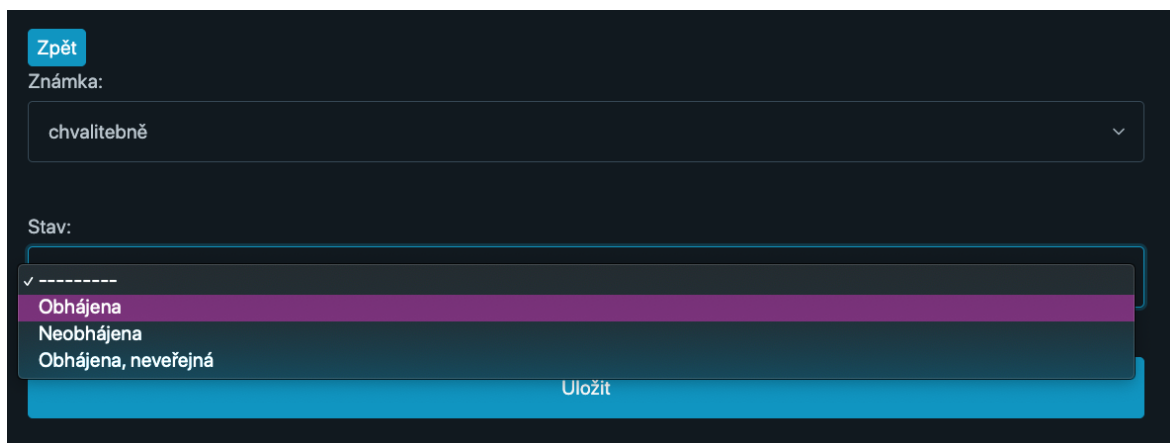
The screenshot shows the 'GJK – Odevzdávací systém maturitních prací' interface. At the top, it indicates the user is logged in as 'Přihlášen učitel Josefina Abcdeová' with links for 'Změna hesla' and 'Odhlásit se'. A 'Zpět' button is in the top left. The main form includes: a date range selector for 'Období:' with a dropdown menu showing 'Humanitní studia: 2021-10-01 - 2022-03-22'; a 'Date:' section with two dropdowns for 'březen' and '2022'; a large text area for 'Poznámky:'; and a blue 'Přidat' button at the bottom.

Obrázek 3.8: Přidávání nové konzultace.

Je třeba zvolit období a datum konzultace. Volitelně lze ke konzultaci přidat poznámku, která je určena pouze pro učitele.

The screenshot shows the 'Odevzdání' (Submission) section. It features three buttons: 'Přidat soubor' (Add file), 'Přidat odkaz' (Add link), and 'Odevzdat' (Submit), arranged in a grid-like fashion.

Obrázek 3.9: Možnosti nahrání souboru/odkaz a odevzdání práce (pohled studenta)



Obrázek 3.10: Nastavení hodnocení práce (pohled vedoucího).

Vedoucí zvolí známku a přesune práci do jednoho z *uzavřených* stavů – *Obhájena*, *Neobhájena* popř. *Obhájena, neveřejná*. V případě, že chce učitel raději např. odložit práci na pozdější termín, může toho docílit pomocí tlačítka **Změnit stav**, viz obrázek 3.4.

# Závěr

Odvážil bych se tvrdit, že původní zadání jsem splnil v plné míře. Implementoval jsem i drtivou většinu dalších požadavků, které mi byly navrženy učiteli v maturitních komisích, a s vedoucím jsem postup pravidelně konzultoval. Samozřejmě je ale možné nalézt spoustu prostoru pro různé vylepšení a další funkce, ku příkladu již zmíněné dynamické přechody mezi stavy.

V tomto projektu jsem se naučil něco o detailnějším navrhování projektů ještě před fází implementace. Zcela nové pro mě byly UML diagramy, které určitě mohou být dobrým způsobem vizualizace vnitřní logiky navrhovaného systému, avšak úplně jsem nepochopil, proč u těchto diagramů tolik záleží na jejich formální stránce. Jsem zvědav zda se budu schopen s touto důkladnější přípravou dostat dále u svého příštího většího projektu.

Přestože se mi nejprve příliš nezdál, ukázal se postup mého vedoucího s dynamickými stavy jako velmi užitečný. Vyplatil se např. když jsem později přidával stav pro obhájenou práci, která však není zveřejněná v archivu.

Byl bych rád, kdyby tento systém opravdu maturantům i učitelům o něco zpřehlednil maturity, jejichž některé části zpřehlednění potřebují jako sůl.



# Seznam použité literatury

- [Dja22a] Django. *Django dokumentace | Databases*. <https://docs.djangoproject.com/en/4.0/ref/databases/>. [online; cit. 2022-03-15]. 2022.
- [Dja22b] Django. *Django dokumentace | How to use Django with Gunicorn*. <https://docs.djangoproject.com/en/4.0/howto/deployment/wsgi/gunicorn/>. [online; cit. 2022-03-15]. 2022.
- [Dja22c] Django. *Django dokumentace | Security in Django*. <https://docs.djangoproject.com/en/4.0/topics/security/>. [online; cit. 2022-03-15]. 2022.
- [Eng22] Instagram Engineering. *Types for Python HTTP APIs: An Instagram Story*. <https://instagram-engineering.com/types-for-python-http-apis-an-instagram-story-d3c3a207fdb7>. [online; cit. 2022-03-15]. 2022.
- [Gun22] Gunicorn. *Deploying Gunicorn*. <https://docs.gunicorn.org/en/latest/deploy.html>. [online; cit. 2022-03-15]. 2022.
- [Neu07] Jim Arlow & Ila Neustadt. *UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky. 2., aktualiz. a dopl. vyd.* Brno: Computer Press, 2007. ISBN: 978-80-251-1503-9.
- [Pic22] PicoCSS. *Minimal CSS Framework for semantic HTML*. <https://picocss.com/>. [online; cit. 2022-03-15]. 2022.
- [Wik22] Wikipedia. *PostgreSQL — Wikipedia, The Free Encyclopedia*. <http://en.wikipedia.org/w/index.php?title=PostgreSQL&oldid=1076044937>. [online; cit. 2022-03-15]. 2022.



# Seznam obrázků

1.1	Usecase diagram systému . . . . .	4
1.2	Activity diagram pro výběr tématu práce . . . . .	5
1.3	Activity diagram pro proces odevzdání a obhájení práce . . . . .	6
1.4	Activity diagram pro kontrolu konzultací . . . . .	7
1.5	Stavový diagram . . . . .	7
1.6	Konceptuální schéma návrhu databáze . . . . .	9
1.7	Relační schéma návrhu databáze (vygenerováno podle modelů v djangu) . . . . .	10
3.1	Administrační prostředí, když je přihlášen superuživatel . . . . .	17
3.2	Hlavní strana – takto je přivítán přihlášený učitel. . . . .	18
3.3	Založení nové práce. Při zakládání práce je třeba vyplnit její název, předmět a poskytnout návrh zadání. . . . .	18
3.4	Přehled informací o práci z pohledu vedoucího. . . . .	19
3.5	Přehled informací o práci z pohledu studenta, který k ní není připsán. Je zde zobrazena pouze možnost přiřazení k této práci do role autora. . . . .	20
3.6	Pohled na neschválené zadání . . . . .	21
3.7	Seznam konzultací z pohledu vedoucího. . . . .	22
3.8	Přidávání nové konzultace. . . . .	23
3.9	Možnosti nahrání souboru/odkaz a odevzdání práce (pohled studenta) . . . . .	23
3.10	Nastavení hodnocení práce (pohled vedoucího). . . . .	24