Návrh aplikace pro tvorbu úvazků akademických pracovníků

Bc. Jan Hána

Diplomová práce
2022

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2021/2022

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Jan Hána**
Osobní číslo:	**A20602**
Studijní program:	**N0613A140022 Informační technologie**
Specializace:	**Softwarové inženýrství**
Forma studia:	**Kombinovaná**
Téma práce:	**Návrh aplikace pro tvorbu úvazků akademických pracovníků**
Téma práce anglicky:	**Application Design for the Evidence of Academic Staff Working Time**

Zásady pro vypracování

1. Proveďte rešerši existujících řešení.
2. Vypracujte stručný rozbor technologií, které budou použity k návrhu.
3. Proveďte rozbor a analýzu požadavků na zvolené řešení.
4. Zpracujte aplikaci na základě výsledků analýzy.
5. Věnujte pozornost zabezpečení aplikace.

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. LOCKHAR, Josh. Modern PHP: New Features and Good Practices. O'Reilly Media, Inc, USA, 2015. ISBN 1491905018.
2. POTENCIER, Fabien. Symfony 5: The Fast Track. Symfony SAS, 2019. ISBN 2918390372.
3. SIKORA, Martin. PHP Reactive Programming. Packt Publishing, 2017. ISBN 9781786461575.
4. AJZELE, Branko. Modular Programming with PHP 7. Packt Publishing Limited, 2016. ISBN 1786462958.
5. SKLAR, David. PHP 7: praktický průvodce nejrozšířenějším skriptovacím jazykem pro web. Brno: Zoner Press, 2018. Encyklopedie Zoner Press. ISBN 978-80-7413-363-3.
6. WELLING, Luke a Laura THOMSON. Mistrovství PHP a MySQL. Brno: Computer Press, 2017. ISBN 978-80-251-4892-1.
7. DYER, Russell J. T. Learning MySQL and MariaDB: Heading in the Right Direction with MySQL and MariaDB. O'Reilly Media, Inc, USA, 2015. ISBN 1449362907.
8. ŘEZÁČ, Jan. Web ostrý jako břitva: návrh fungujícího webu pro webdesignery a zadavatele projektů. Jihlava: Baroque Partners, 2014. ISBN 978-80-87923-01-6.

Vedoucí diplomové práce: **doc. Ing. Petr Šilhavý, Ph.D.**
Ústav počítačových a komunikačních systémů

Datum zadání diplomové práce: **3. prosince 2021**
Termín odevzdání diplomové práce: **23. května 2022**

L.S.

doc. Mgr. Milan Adámek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 24. ledna 2022

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne

Jan Hána, v.r.
podpis studenta

ABSTRAKT

Tato diplomová práce se zabývá vývojem webové aplikace pro tvorbu úvazků. Primárně je určena pro tvorbu úvazků akademických pracovníků na akademické půdě. Jejím účelem je přiřadit dílčí úvazky akademickým pracovníkům a rozdělit úvazky rovnoměrně mezi jednotlivé lektory. Na základě rešerše existujících řešení a podrobné analýzy požadavků bylo navrženo vhodné řešení pro tvorbu aplikace. Se správným počtem entit a efektivními funkcemi bylo docíleno celkového požadavku na tvorbu úvazků akademických pracovníků.

Klíčová slova: aplikace; Bootstrap; MariaDB; PHP; Symfony; tvorba úvazků; úvazkový systém;

ABSTRACT

This thesis deals with the development of a web application for creating jobs. It is primarily intended for the creation of academic staff's jobs on campus. Its purpose is to assign part-time work to academic staff and to distribute work evenly among individual lecturers. A suitable solution for creating an application was designed based on a search of existing solutions and a detailed analysis of requirements. With the right number of entities and efficient functions, the overall demand for the creation of academic staff was achieved.

Keywords: application; Bootstrap; MariaDB; PHP; Symfony; work creation; working time system;

Touto cestou bych rád poděkoval svému vedoucímu diplomové práce panu doc. Ing. Petru Šilhavému, Ph.D. za podporu, odborné vedení, věnovaný čas a cenné rady při psaní této diplomové práce.

Dále bych chtěl poděkovat Mgr. Haně Vojáčkové, Ph.D. z Vysoké školy polytechnické Jihlava za spolupráci při tvorbě diplomové práce.

Poděkování rovněž patří mé rodině a nejbližším za podporu při realizaci práce.

Prohlašuji, že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Při výběru vedoucího práce jsem oslovil pana doc. Ing. Petra Šilhavého, Ph.D., nabídl mi, zda bych se nechtěl zabývat návrhem aplikace pro tvorbu úvazků akademických pracovníků.

Při zamyšlení nad daným tématem mě osobně ani žádný takový úvazkový systém, který řeší tvorbu úvazků akademických pracovníků, nenapadl. Pouze mě napadly úvazkové systémy, které slouží pro účely základních a středních škol, například software Bakaláři. Dále mě napadaly spíše úvazkové systémy, které řeší pouze personální záležitost, ke kterým je přidán například docházkový systém zaměstnanců do zaměstnání nebo mzdová agenda.

Proto mě toto navržené téma velice zaujalo, protože tvorba úvazků akademických pracovníků není pouze o tom, že v systému jsou zadané osobní informace o zaměstnanci, jako je jméno, příjmení, tituly, rodné číslo, adresa bydliště a přidělená pozice, ale že se například navíc jedná o administrativního pracovníka.

Zajímavým shledávám také řešení úvazků akademických lektorů, kromě toho, že máme osobu, které dáme učit například elektrotechniku, tak samotný předmět je rozdělen na jednotlivé bloky, které se skládají z přednášky a cvičení, popřípadě i ze seminářů. Do daného předmětu může zasahovat i druhý lektor, který bude vyučovat pouze přednášky a první bude mít cvičení nebo si cvičení oba profesoři budou dělit. Cílem je analyzovat celou situaci a rovnoměrně přidělit jednotlivé předměty tak, aby nenastala situace, kdy je jeden lektor příliš zahlcen povinnostmi a druhý lektor by měl v rozpise pouze několik hodin.

Přínos práce shledávám v několika bodech, ať už se jedná o rešerši existujících řešení anebo se jedná o podrobný rozbor a analýzu požadavků na funkčnost celého systému a jeho celistvost a propracovanost, tak tím nejdůležitějším bodem je podle mého názoru samotný návrh tohoto uceleného systému.

Pro realizaci projektu je zapotřebí provést podrobnou analýzu potřebných požadavků pro funkčnost minimálních požadavků systému. Dále je potřeba použít vhodnou technologii na celý systém. Proto na správné technologii závisí samotný vývoj aplikace a popřípadě rozšíření systému do budoucna nebo napojení systému na jiné informační systémy.

Z analýzy a rozboru požadavků zjistím, jaká bude nejvhodnější technologie na aplikaci, zda bude stačit pouze desktopová aplikace nebo webová aplikace. Dále zjistím, zda bude nutné použít na uchování dat nějaký databázový systém, či nikoliv.

Hlavní motivací k řešení daného tématu je samotný problém s tvorbou úvazků akademických lektorů.

Cílem práce je navrhnout a zpracovat aplikaci, jež by vytvářela úvazky akademických pracovníků, spravovala jednotlivé předměty a přidělovala jednotlivé akce lektorům. Důležitým prvkem je analyzovat jednotlivé pracovní postupy při rozdělování a přidělování jednotlivých úvazků a na jaké všechny náležitosti musíme brát zřetel.

I. TEORETICKÁ ČÁST

1 SOUČASNÝ STAV

Se současným vývojem moderní technologie se úvazky pracovníků uchovávají už jen převážně v elektronické podobě a papírová podoba už se tolik nezachovává. Hlavním důvodem, proč se úvazky zaměstnanců řeší už jen v elektronické podobě je ulehčení práce personální agendě a v jednoduchosti s uchováním dat. V dnešní době je navíc rozšířené uchovávat vše v digitální podobě, popřípadě v digitalizované podobě.

Pod pojmem úvazek si můžeme představit vztah mezi zaměstnancem a zaměstnavatelem. Tento vztah můžeme nazvat také pracovní poměr. [1]

Úvazek můžeme členit v základu na:

- Plný
- Částečný

Plný úvaze je úvazek, při kterém vybraná osoba pracuje 40 hodin týdně, 5 dní v týdnu, 8 hodin denně. [2] Rozvržení dnů a hodin může být upraveno individuálně zaměstnavatelem, pokud se například jedná o směnný provoz nebo o střídání dlouhého a krátkého týdne.

Částečný úvazek je úvazek, který je poloviční neboli zkrácený, není plnohodnotný. Většinou se jedná o polovinu hodin, než má plný úvazek, tedy 20 hodin týdně není-li upraveno jinak.

Při pracovním úvazku zaměstnavatel se zaměstnancem uzavírá pracovní smlouvou, ať už na dobu určitou nebo na dobu neurčitou.

V pracovní smlouvě musí být uvedeno:

- Druh práce
- Místo výkonu práce
- Den nástupu do zaměstnání

Pro nás z pohledu úvazkového systému je důležité, co bude zaměstnanec dělat za pracovní pozici, tedy jestli bude administrativním pracovníkem nebo bude dělat vrátného.

Dále obsahuje pro nás důležité informace jako je jméno a příjmení a další osobní údaje o zaměstnanci z pracovní smlouvy. Dokonce můžeme mít v úvazkovém systému zaznamenán nástup zaměstnance do zaměstnání, z toho důvodu, abychom mu mohli práci přidělit až bude v zaměstnání.

1.1 Úvazkové systémy

Samotné úvazkové systémy, které by se zaměřovaly pouze na úvazky zaměstnanců, neexistují. Většina těchto systémů má další rozšiřující moduly pro řízení lidských zdrojů nebo personální agendu. Je to systém s přidanou hodnotou a tím je to i výhoda pro administrativu podniku.

Například tyto systémy umí sledovat docházku zaměstnanců do zaměstnání. Zaměstnanec, který přijde do práce si musí u vchodu potvrdit vstup čipem, aby mohl pokračovat dále do budovy.

Další modulem může být i mzdová agenda pro mzdu či plat zaměstnanců. Pak se takový systém stává komplexnější. Například k takovým aplikacím patří PAMICA, VEMA a další.

1.1.1 PAMICA

Program PAMICA je ucelený personální a mzdový systém, jenž pomáhá personálnímu oddělení a mzdové agendě. [3] Software je vyvíjen společností STORMWARE s.r.o., její centrální sídlo se nachází v Jihlavě v Kraji Vysočina. Od této společnosti je také známý ekonomický a informační software POHODA, jenž slouží pro komplexní účetnictví, fakturaci, daňovou evidenci a další záležitosti týkající se účetnictví.

Aplikace PAMICA má několik agend, které jsou rozděleny na jednotlivé části, například na: personalistiku, pracovní poměry, evidenci uchazečů, řízení lidských zdrojů, evidenci upomínek, mzdy a další body týkající se pracovních záležitostí. [3]

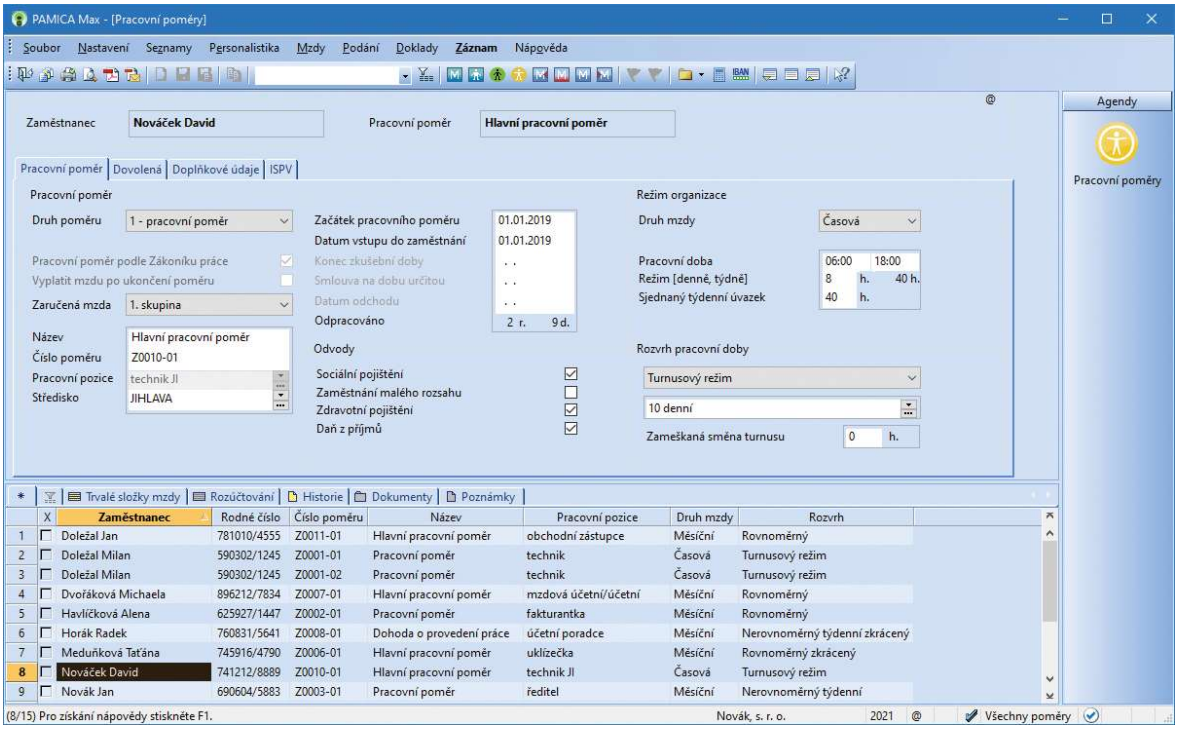
Jeden z důležitých modulů je modul personalistika. Zde jsou evidovány všechny osobní údaje o jednotlivých zaměstnancích společnosti. [4] V této agendě se vypočítává mzda zaměstnanců, uplatnění slev na dani a jiné.

Další z důležitých modulů je pracovní poměr, ve kterém jsou definované jednotlivé pracovní poměry pro jednotlivé zaměstnance. Zde můžeme definovat, zda se v pracovním poměru jedná o dohodu o pracovní činnosti, dohodu o provedení práce nebo hlavní pracovní poměr. [5]

Třetí důležitou částí je modul řízení lidských zdrojů, kde si nadefinujeme organizační jednotky do stromové struktury pro naši organizaci. A tyto organizační jednotky si potom můžeme přiřadit k jednotlivým pracovním pozicím, které dále můžeme upravit v agendě

pracovní pozice. [5] Pro každou pracovní pozici lze definovat pracovní poměr, dále obsahuje důležité informace o výpočtu mezd.

Na obrázku níže je znázorněno uživatelské grafické rozhraní programu PAMICA v agendě Pracovní poměr, kde lze definovat pracovní poměr zaměstnanců a další důležité informace pro práci v této agendě.



Obrázek 1 – Uživatelské rozhraní aplikace PAMICA [3]

1.1.2 Vema Cloud

Vema Cloud představuje profesionální řešení zabývající se lidskými zdroji (HR, Human Resources). Aplikace je vyvíjena softwarovou společností Vema, a.s., jedná se o českého dodavatele, jenž řeší podnikové informační systémy. [6] Hlavní pobočka sídlí v Brně.

Jak už bylo zmíněno na začátku kapitoly, tak Vema Cloud je HR systém. [7] Jedná se o komplexní řešení, které řeší jak úvazky zaměstnanců, lidské zdroje, tak i jednotlivé příchody a odchody zaměstnanců. Dále je řešena i mzdová agenda.

Aplikace pracuje online. Výhodou aplikace je, že všechny data jsou uloženy v cloudu, a tak k daným datům může personalistka/personalista přistupovat odkudkoliv, ať už osobně z kanceláře, z domova nebo odkudkoliv, kde bude možnost se připojit na internet. [7] Výhodou je i to, že nemůžeme o data přijít. Kdybychom měli data uložené u sebe ve firmě

na serveru a daný server by selhal, tak bychom o cenná data mohli přijít, kdybychom současně neřešili zálohování.

Do aplikace se správci daných agent přihlašují za pomoci komerčního certifikátu. Tím je zabezpečena situace, že se do aplikace přihlašuje příslušný člověk, který má oprávnění tuto aplikaci používat.

1.2 Tvorba úvazků učitelů

Pro tvorbu úvazků učitelů základních a středních škol existuje několik aplikací. Jednou z rozšířených a nejznámějších je aplikace Bakaláři.

Dalšími aplikacemi jsou například:

- DM Software
- Škola OnLine
- iŠkola

1.2.1 Bakaláři

Bakaláři jsou nejrozšířenější školní systém, který zahrnuje komplexní řešení. Aplikace se dělí na dvě části, a to na studentskou část a učitelskou část. Studentská část slouží k zobrazení známek studentů a rozvrh předmětů. [8] Učitelská část už je více rozsáhlá, viz níže.

Učitelská část obsahuje:

- Rozvrhové akce
- Plán akcí školy
- Zapsání známek studentům
- Třídní knihy
- Tematické plány
- Přijímací zkoušky
- Školní matriku
- A další

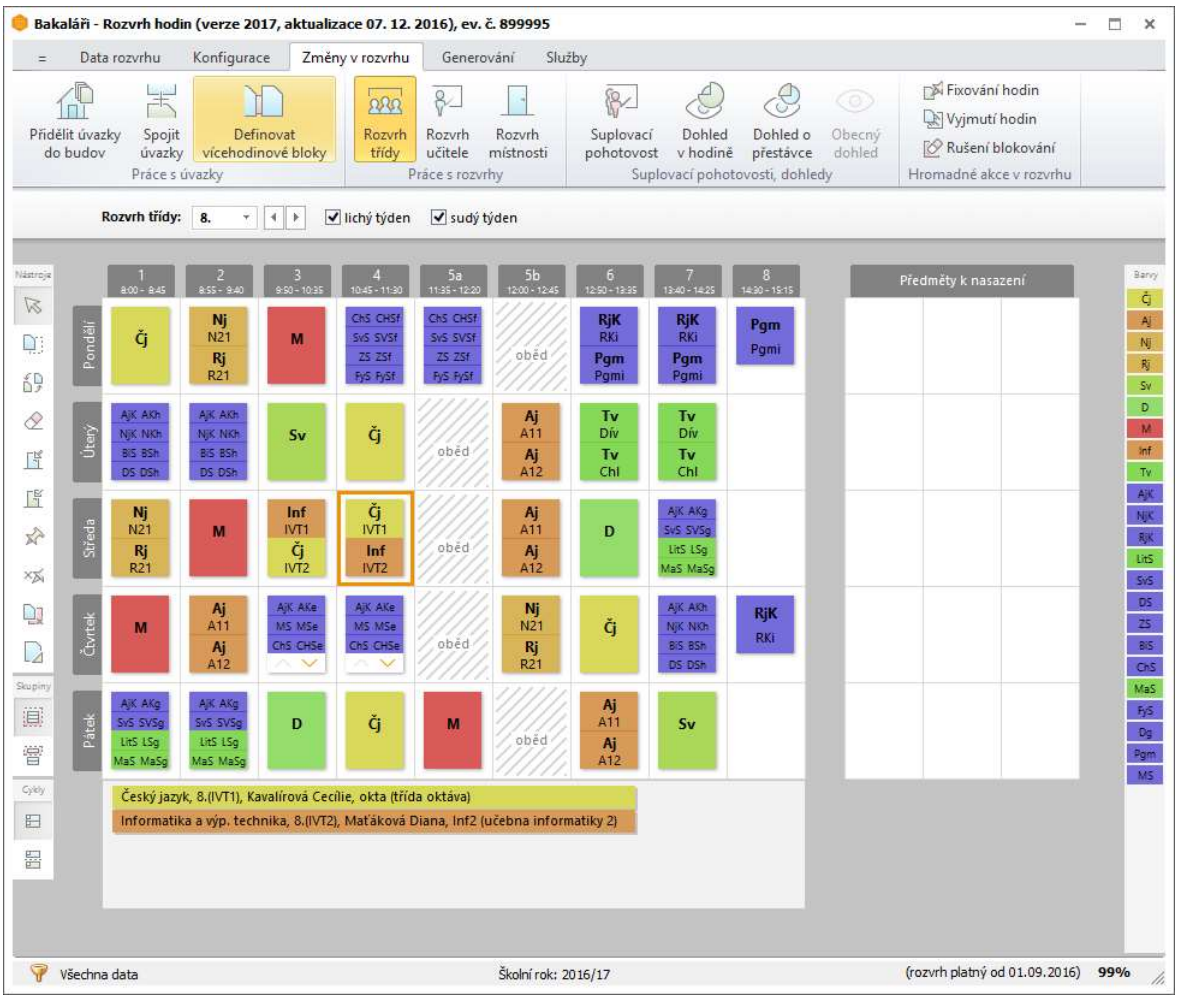
Dále je aplikace rozdělena na webovou část a lokální část. Do webové části mají možnost se přihlásit studenti, aby si mohli prohlédnout svůj prospěch a rozvrh. Lokální část aplikace slouží pro práci s jednotlivými daty.

Pro tvorbu rozvrhů a úvazků existuje v aplikaci modul nazvaný Evidence a Rozvrh, tvorba úvazků. [8] V tomto modulu lze vytvořit nový školní rok, do kterého je možné přidat skupinu učitelů a úvazků z minulého školního roku. Při vytváření nového školního roku vytváříme i nové úvazky pro daný školní rok. [9] Do nového školního roku se překlopí i jednotlivé třídy, například ze třídy 1.A vznikne v dalším roce 2.A a závěrečný ročník už se do nového školního roku nepřeklopí. [10] Dále úvazky můžeme upravit dle vlastních potřeb a potřeb školy.

V dalším modulu nazvaný Evidence žáků a zaměstnanců evidujeme jednotlivé osobní údaje, jako je jméno, příjmení, rodné číslo, adresu bydliště a popřípadě kontakty. [8]

Každý učitel do systému Bakaláři přistupuje přes svoje přístupové údaje, může si zobrazit svůj plán úvazků, tj. jaký předmět vyučuje a v jaké učebně výuka probíhá, může si zobrazit seznam jednotlivých studentů, rozvrh učeben a další.

Na obrázku níže je znázorněno uživatelské rozhraní lokální aplikace v agendě rozvrh hodin.



Obrázek 2 – Rozhraní aplikace Bakaláři – rozvrh hodin [8]

1.2.2 DM Software

DM Software je další informační systém, který je zaměřen na evidenci školních záležitostí. Aplikace je rozdělena do několika modulů, tak jako aplikace předchozí. Aplikace funguje obdobně jako Bakaláři, kdy ve školní matrice evidujeme jednotlivé osoby, o kterých uchováváme osobní informace.

Dále máme modul Rozvrh a suplování, kde jsme schopni vytvářet nové rozvrhy na nový školní rok, dále můžeme upravit rozvrh již existující, můžeme udělat různé výpisy pro své potřeby a další. [11] [12]

Aplikace obsahuje další moduly jako:

- Elektronická třídní kniha
- Elektronická žákovská knížka
- Tisk vysvědčení
- Výuka

1.3 Tvorba úvazků akademických pracovníků

Aplikace pro tvorbu úvazků akademických pracovníků je už specifický systém a tolik řešení na trhu neexistuje, protože systém je přímo specifikován a od úvazků základních a středních škol se velice liší.

Většina vysokých škol používá pro rozvrhy a předměty informační systém IS/STAG. Systém slouží pro administraci studijní agendy. [13] Následně tento systém může být napojen na informační systém pro hodnocení pracovníků (IS HAP). Tento systém slouží k hodnocení akademických pracovníků. Hodnotí se dle velikosti úvazku, pracovní pozice, vzdělávací oblasti a tvůrčí činnosti. [14] Tento systém také může být napojen na personální systém.

1.3.1 IS VŠPJ

Při rešerši stávajících existujících řešení tvorby úvazků akademických pracovníků jsem oslovil svou bývalou Vysokou školu polytechnickou v Jihlavě. V této škole využívají svůj vlastní systém na tvorbu úvazků akademických pracovníků. Jedná se o komplexní informační systém, který slouží jak pro akademické pracovníky, tak i pro studenty. Systém je neustále vyvíjen a zdokonalován, dle potřeb školy.

Studenti si v informačním systému můžou zapisovat předměty na jednotlivé semestry, zobrazit si rozvrh na daný semestr, zobrazit své hodnocení z předmětů a jiné.

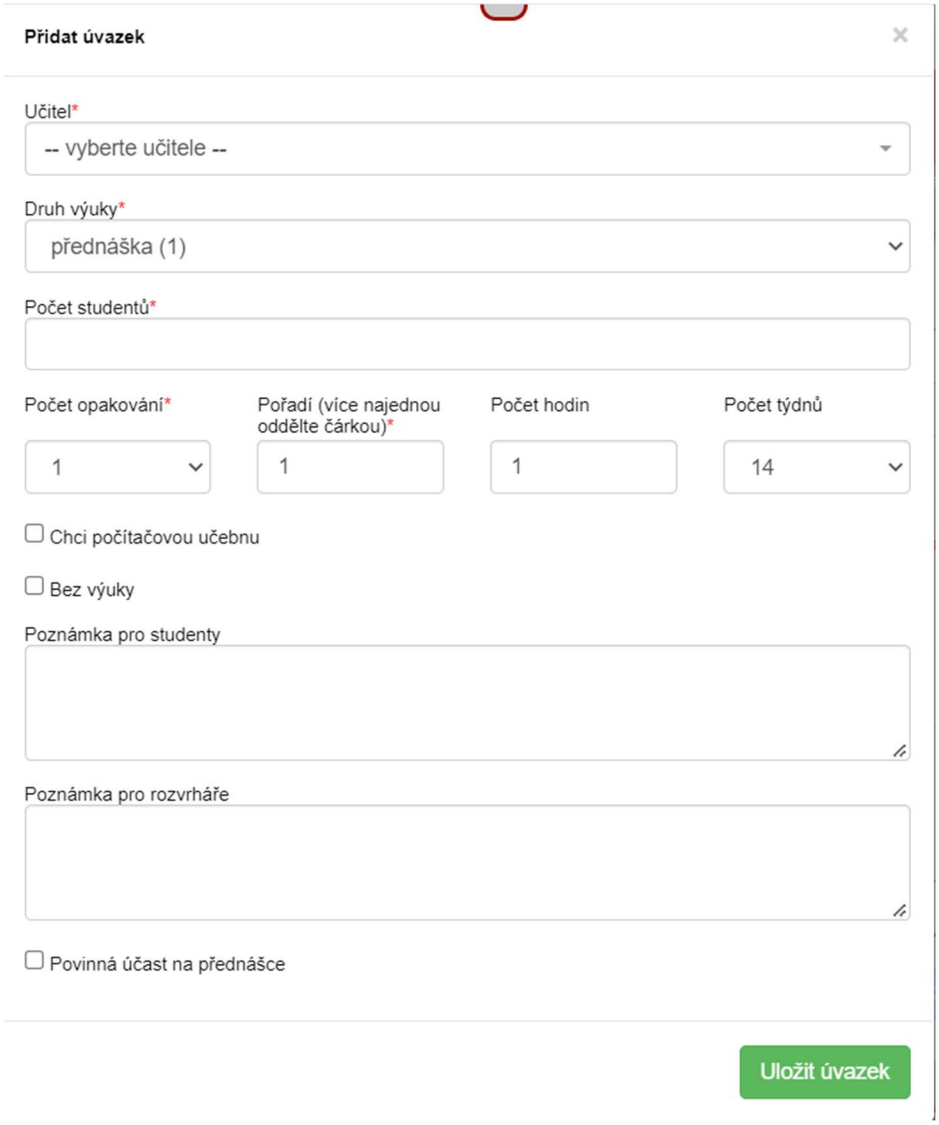
Akademickým pracovníkům systém umožnuje zadávat klasifikace studentů, akce školy, jednotlivé úvazky na daný akademický rok a mnoho dalšího.

Systém umožnuje vytvářet úvazky pro akademické pracovníky. Úvazky jsou rozděleny do prezenční a kombinované formy studia. Předtím než se vytvoří úvazek, tak se ze studijního plánu vybere daný předmět, pro který chceme vytvářet úvazek. Systém ukáže počet studentů na tento předmět.

Obrázek 3 – IS VŠPJ – Prezenční forma

Obrázek 4 – IS VŠPJ – Kombi forma

Pro vytvoření úvazku je v systému formulář, do kterého se zadají jednotlivé parametry pro vygenerování úvazku. Vybírá se akademický pracovník, který bude mít daný úvazek. Poté je vybrán druh výuky, zda přednáška či cvičení nebo tutoriál. Následně je zadán počet studentů pro dané cvičení. Dále se nastavuje počet hodin a počet týdnů. Na závěr se může ještě vybrat, zda chceme počítačovou učebnu, popřípadě můžeme zadat poznámku a další.



Obrázek 5 – IS VŠPJ – Formulář pro vytvoření úvazku

2 ROZBOR POUŽITÉ TECHNOLOGIE

Pro návrh a vývoj aplikací existuje velké množství navrhovacích a programovacích aplikací, které mají uživatelské grafické rozhraní. Nejprve je potřeba určit parametry: jak má aplikace fungovat, co vše má aplikace dělat a umět, jak bude uchovávat data, jaká data bude uchovávat, pro jakou skupinu osob je aplikace určena, kde bude využívána a zda má být aplikace desktopová nebo webová, případně mobilní.

Většinou u návrhu aplikace řešíme její strukturu, jaké bude obsahovat třídy, entity a funkce, které můžeme navrhnout přes grafický jazyk pro vizualizaci Unified Modeling Language (UML). Popřípadě můžeme rovnou navrhnout Entity-relationship (ER) model toho, jak bude vypadat přesně databáze a jaké bude obsahovat entity a jaké vztahy budou mezi entitami.

Dalším bodem návrhu je výběr technologie. Pokud bychom vybrali špatnou technologii, mohlo by se stát, že bychom nebyli schopni aplikaci naprogramovat, mohli bychom také narazit na problém daného programovacího jazyka, celý vývoj by se tak vrátil na začátek, anebo bychom projekt ukončili jako neúspěšný. Vybrat správný jazyk není tak snadné. [15] Proto je potřeba důkladně rozmyslet platformu, na které aplikace bude působit a kde ji uživatelé budou využívat a poté zbývá vybrat nejvhodnější programovací jazyk.

2.1 Návrhové prostředí pro návrh systému

Jak už bylo řečeno na začátku této kapitoly pro vizualizaci systémů používáme UML jazyk, který zobrazuje vývojové diagramy neboli flowcart. Tyto vývojové grafy nám mohou znázorňovat například případy užití daného systému, diagram tříd nebo model chování. [16]

Vývojový graf je složen z jednotlivých grafických obrazců a šipek, které slouží ke grafickému znázornění jednotlivých kroků algoritmu.

Aplikace, které umí vytvářet grafické diagramy, jsou komerčního použití nebo jsou bezplatné. Zmíněné 2 druhy aplikací se liší pouze ve vzhledu grafického uživatelského rozhraní, jinak jsou si podobné. Můžou se lišit v grafickém vzhledu jednotlivých tvarů objektů, ale na čitelnost diagramů odlišnost nemá vliv.

Mezi nejznámější programy, jenž pracují s UML jazykem patří například:

- Microsoft Visio
- Lucidchart
- Diagrams.net (Draw.io)

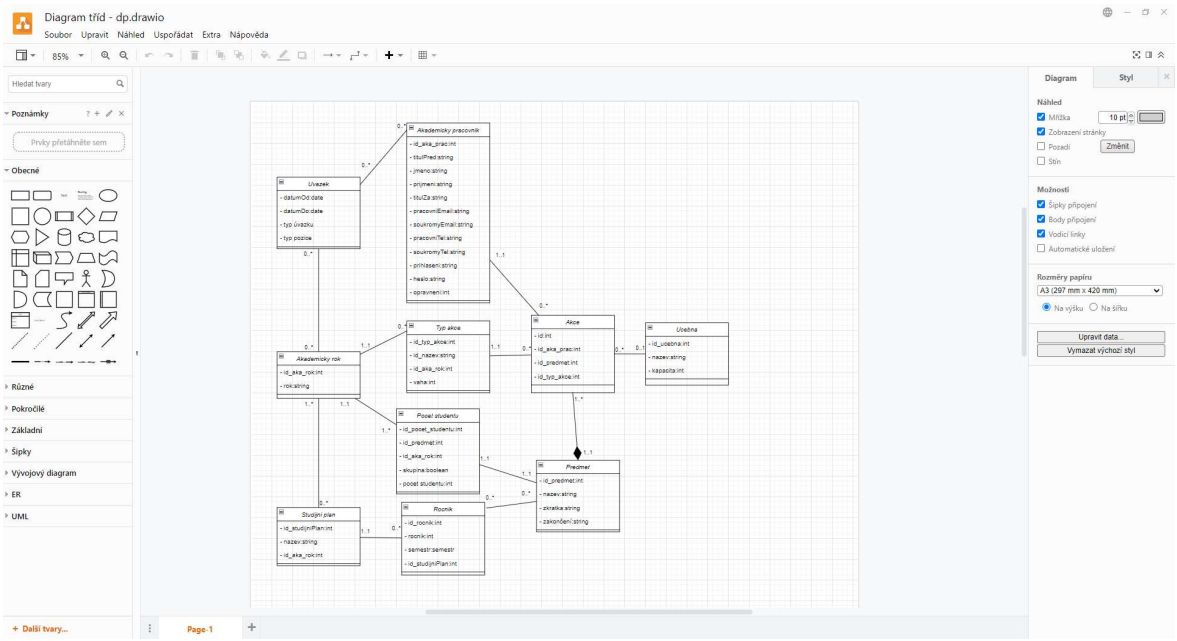
- Edraw Max
- Archi
- StarUML

Tyto návrhy nám pomáhají při realizaci vývoje dané aplikace k lepšímu pochopení toho, jak daná aplikace má fungovat, jaké typy uživatelů budou k aplikaci přistupovat a jaká budou mít oprávnění k určitým činnostem. Dále z grafů můžeme vyčíst i s jakými třídami nebo entitami bude navrhovaná aplikace pracovat.

K návrhu určitých diagramů jsme si vybrali aplikaci Draw.io, neboť nám přijde, že má přívětivé uživatelské prostředí.

2.2 Diagrams.net (draw.io)

Diagrams.net neboli draw.io je aplikace na tvorbu vývojových diagramů. [17] Jedná se o multiplatformní software, můžeme jej mít spuštěný pouze v internetovém prohlížeči, ale také si můžeme stáhnout desktopovou verzi aplikace. Jedná se o aplikace open source. Rozhraní aplikace obsahuje různé objekty, jenž umožňují vytvářet drátové modely, UML diagramy, vývojové diagramy, organizační diagramy a síťové diagramy.



Obrázek 6 – Prostředí aplikace Draw.io

Projekt z tohoto programu lze uložit ve formátu souboru .drawio nebo lze daný výsledek vyexportovat do klasických obrázkových formátů jako .jpg, .png a další. [18]

2.3 Vývojové prostředí pro tvorbu kódu

Pro vývoj jak mobilních, tak desktopových nebo webových aplikací existuje mnoho různých vývojových prostředí. Pokud se, ale zaměříme na jeden segment, tak se výběr vývojových prostředí zúží.

Pro naší aplikaci na tvorbu úvazků akademických pracovníků jsme zvolili webovou aplikaci a programovací jazyk PHP.

Pro vývoj PHP aplikací existují vývojová prostředí jako:

- PSPad
- Notepad++
- PHPStorm
- NetBeans IDE
- Geany
- Visual Studio Code

Naše preferované vývojové prostředí je PHPStorm pro vývoj webových stránek v PHP, protože nám je sympatické grafické uživatelské rozhraní.

2.4 Vývojové prostředí PHPStorm

PHPStorm je vývojové prostředí (IDE, Integrated Development Environment) pro programovací jazyk PHP. Jedná se o software od společnosti JetBrains. Dříve známa pod názvem IntelliJ Software. Tato vývojářská společnost má zastoupení i v České republice, a to v Praze, kde se nachází její pobočka. [19]

JetBrains nabízí další produkty, například:

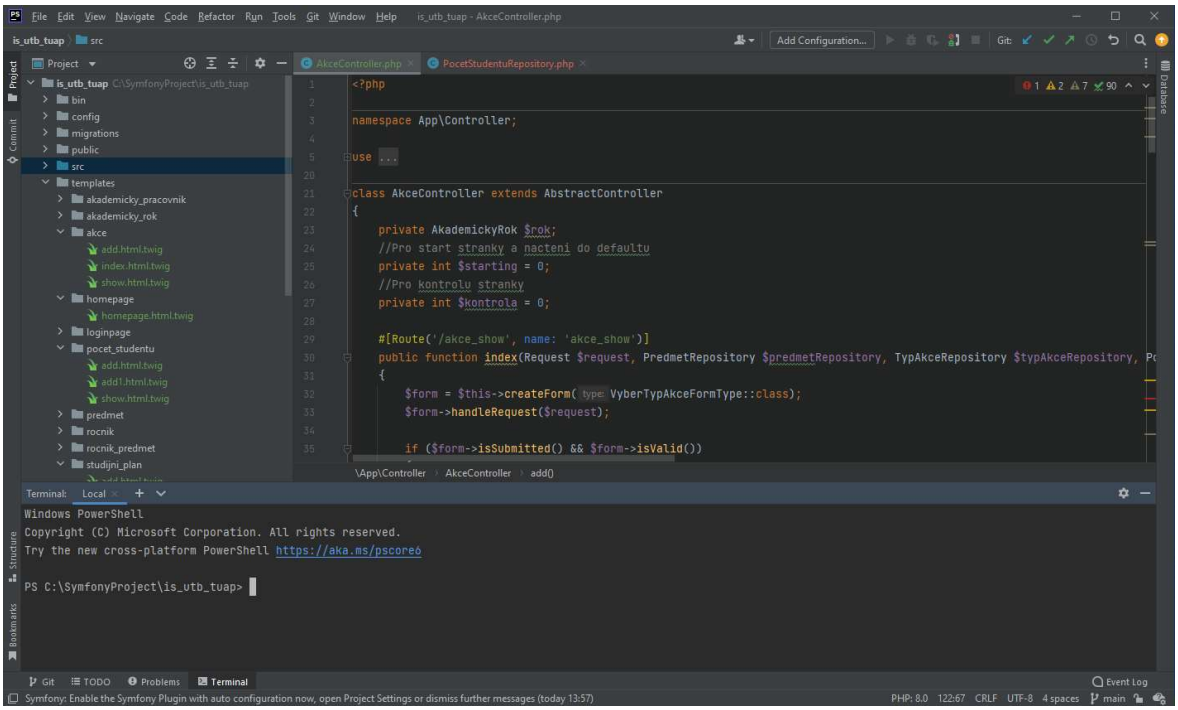
- CLion – pro programovací jazyk C a C++
- PyCharm – pro programovací jazyk Python
- Rider – pro platformu .NET
- RubyMine – pro programovací jazyk Ruby, Rails, JavaScript a další
- WebStorm – pro programovací jazyk JavaScript

Vývojová prostředí od JetBrains jsou komerční produkty. Tedy i PHPStorm je komerční produkt a je nutné zakoupit licenční klíč. Společnost JetBrains umožnuje bezplatnou roční licenci pro studenty, učitele a akademickou oblast. [20]

Aktuální verze programu je z roku 2022 pod označením 2022.1. Vývojové prostředí aplikace PHPStorm je napsané v jazyce Java. [20] Jedná se o multiplatformní aplikaci, kterou lze spustit například v operačních systémech Windows, MacOS a Linux.

Hlavními vlastnostmi vývojového prostředí jsou například kompatibilita s různými verzemi PHP, automatické doplňování kódu tříd, funkcí, názvu proměnných a dalších důležitých slov. Dalšími vlastnosti jsou detekce duplicitního kódu, refaktorování – přejmenování třídy a jiné.

PHPStorm je naprogramovaný v jazyce Java, díky tomu umožnuje přidání dalších rozšiřujících pluginů do IDE, ať už těch, co nabízí samotný PHPStorm, popřípadě můžeme naprogramovat a použít vlastní pluginy. PHPStorm obsahuje i editor pro JavaScript a značkovací jazyk HyperText Markup Language (HTML) a kaskádové styly (CSS, Cascading Style Sheets). [21]



Obrázek 7 – Vývojové prostředí PHPStorm

2.5 PHP

PHP se řadí do open source nejrozšířenějších skriptovacích programovacích jazyků. PHP jazyk je převážně navržen pro dynamické internetové stránky a webové aplikace. [22] Pro správné fungování a vykreslení daného výsledku jazyka je potřeba webový server, na kterém běží PHP interpret, který zajišťuje překlad jazyka a kontroluje syntaxi jazyka. PHP příkazy

jsou převáděny na straně serveru a klientovi webových stránek už je pouze zobrazen daný výsledek příslušného skriptu.

PHP kód může být vkládán rovnou do HTML souboru.

Jazyk v prvních verzích vznikl v roce 1994 dánsko-kanadským programátorem Rasmusem Lerdorfem. [23] Tyto verze byly ovšem pouze testovací. Jednalo se o prvotní skripty pro účely autora, který sledoval návštěvnost svého online životopisu umístěného na internetu. [24] První oficiální verze pro vývojáře vyšla v roce 1995. [23] Nyní se na vývoji skriptovacího jazyka podílí The PHP Group.

Od doby vzniku PHP jazyk prošel velkou proměnou, ze sad skriptů se stal plnohodnotným jazykem s danou syntaxí. [23] Za tuto dobu vyšlo několik dalších verzí, například PHP 2, 3, 4 a 5. Verze 6 tohoto skriptovacího jazyka nikdy nevyšla, protože při projektování této verze s implementací nového kódu vznikly komplikace, a proto se od této verze upustilo.

Rozšířenou a nejznámější verzí PHP je verze 7, která vyšla v listopadu roku 2015 a stala se nejrozšířenější verzí tohoto skriptovacího jazyka. Tato verze se rozšířila také do dalších podverzí, které obdržely další vylepšení a nyní je finální verze 7.4.

Nejnovější PHP verze je 8. Verze byla vydaná v listopadu roku 2020. Tato verze opět přináší spoustu novinek a vylepšení, například opravuje problémy z minulé verze, vylepšena je optimalizace, pole může začínat záporným indexem, lepší pojmenování argumentů a další. [25]

Syntaxe jazyka je inspirovaná několika programovacími jazyky, jako je jazyk C, Java a Perl. [23] Jazyk je nezávislý na platformě, protože jej překládá webový server. PHP podporuje různé internetové protokoly, knihovny pro zpracování textu, pro práci se soubory, pro ukládání dat do souborů a hlavní částí je přístup k databázovým systémům. Databázové systémy, které známe jako například MySQL, MariaDB, Oracle, PostgreSQL a mnoho dalších.

Pro vývoj profesionálních webových aplikací PHP existují frameworky, které mohou obsahovat strukturu projektu, podpůrné knihovny, podpůrné podprogramy, vzory pro vývoj a mnoho dalších možností. Frameworky umožnují programátorům ulehčit práci při vývoji webových aplikací.

Hlavními výhodami, proč se používají frameworky, jsou:

- Rychlejší rozvoj projektu

- Méně kódu k zápisu
- Knihovny pro běžné úlohy
- Dodržování správného postupu při kódování
- Bezpečnější
- Snadnější údržba

Pro PHP existuje velké množství frameworků. Nejvíce používané frameworky jsou:

- Symfony
- Lareval
- CodeIgniter
- Phalcon
- CakePHP
- Yii 2
- Zend Framework
- Nette

2.6 Symfony

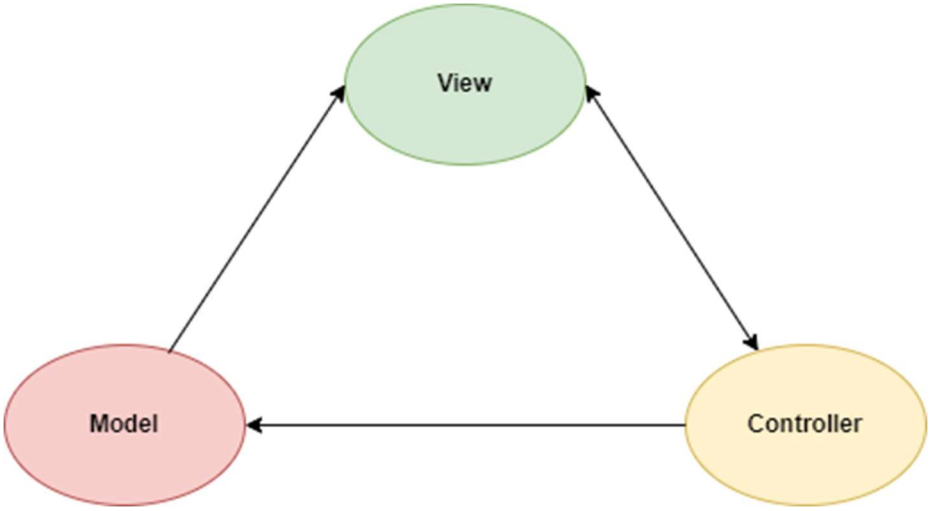
Symfony je webový framework stanovený pro skriptovací jazyk PHP. Jedná se o jeden z nejúspěšnějších frameworků. [26] Symfony slouží pro vývoj webových aplikací. Struktura projektu vychází z architektury Model-View-Controller (MVC). MVC je občas označován jako návrhový vzor, ale jedná se o architekturu, jenž rozděluje logiku od výstupu.

Architektura se skládá ze 3 částí:

- Model (Model)
- Pohled (View)
- Řadič (Controller)

První část model zahrnuje vše související s databázemi, údržbou a uchováním samotných dat. Druhá část pohled značí komponenty zodpovědné za prezentaci dat uživateli, jenž navštíví danou webovou aplikaci. Třetí a zároveň poslední část řadič je přechodem mezi pohledem a modelem. Propojuje je odesíláním dat z modelu do zobrazení, zpracovává data a v případě potřeby je i aktualizuje.

Na obrázku níže je znázorněna architektura MVC.



Obrázek 8 – Architektura MVC

První verze tohoto frameworku vyšla v roce 2005. Od této doby framework prošel velkou změnou. Symfony má open source licenci a díky tomu jej lze přizpůsobit specifickým potřebám každého jedince. K tomuto frameworku existuje komunita, která se skládá z více než 600 000 vývojářů z různých zemí a díky tomu je Symfony rozmanité na tvorbu webových aplikací. [27]

Poslední verze Symfony je označena 6.0. Jedná se o verzi, jež vyšla v listopadu roku 2021. Tato konkrétní verze už pro své správné fungování vyžaduje verzi PHP 8 a vyšší.

Symfony je inspirováno jinými frameworky jako jsou Spring, Django a Ruby on Rails. [28]

Hlavními výhodami Symfony frameworku jsou:

- Flexibilita
- Rychlý výkon
- Uživatelsky přívětivé
- Jednoduché použití
- Podpora
- Rozšíření dle sebe
- Inovace
- Reference

2.7 MariaDB

MariaDB je relační databázový systém. Jedná se o kompatibilní náhradu za široce využívaný databázový systém MySQL, který odkoupila veřejná obchodní společnost Oracle Corporation. [29] Zakladatel MySQL bázového systému je Michael Widenius, přezdívaný „Monty“. [30]

Díky obavám z toho, že MySQL směřuje ke komerční licenci, tak vznikla MariaDB z důvodu, aby byla udržena svobodná licence. [30] MariaDB je vyvíjena odnoží vývojářů, kteří pracovali na MySQL, hlavním vývojářem byl Michael Widenius. [30] První verze databázového systému byla vydána v lednu 2009.

MariaDB a MySQL nejsou dokonalé – žádná databáze taková není, ale jsou užitečné pro své uživatele. [31]

Jedná se o multiplatformní bázový systém. Převážné použití je jako databáze pro webové aplikace nebo informační systémy, ale můžeme je také použít pro samostatné aplikace vestavěné v lokální síti.

Samotná databáze ve výchozím nastavení funguje už ve velmi dobře zabezpečeném režimu. [31]

Pro grafické zobrazení a management databáze existují různé aplikace, například:

- Adminer
- DBeaver
- DBVisualizer
- Firebird
- SQL Manager

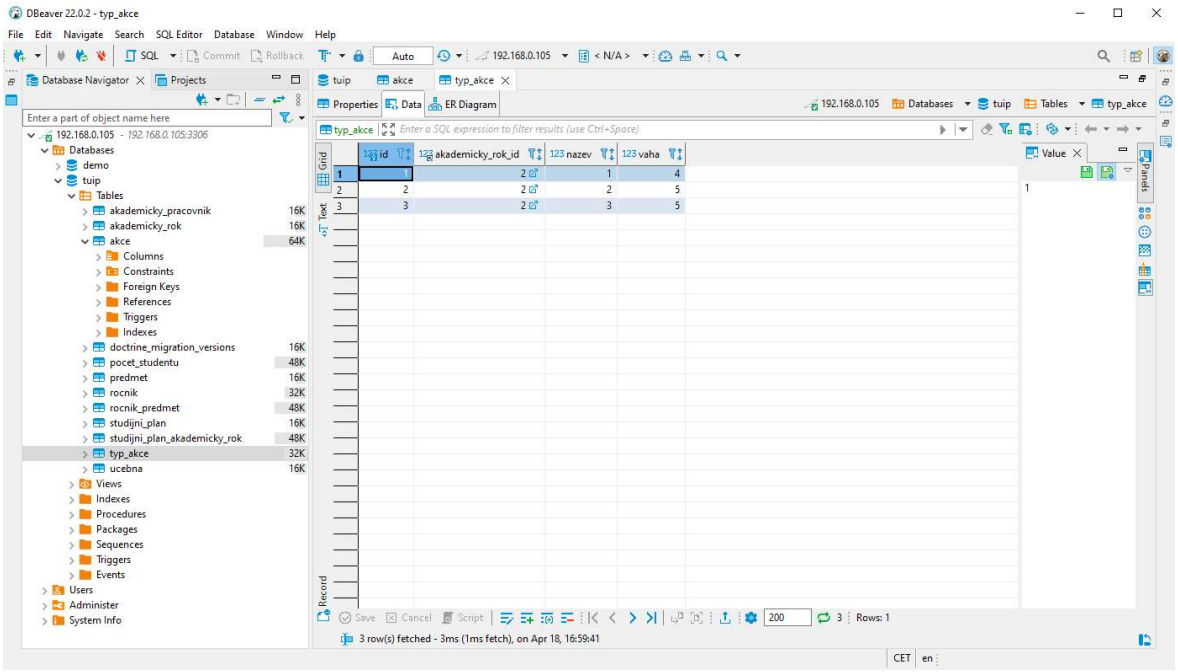
Pro naší práci jsme si pro kontrolu a lepší orientaci v databázi vybrali nástroj DBeaver, ve kterém jsme mohli zkontrolovat, zda je správně navržena a vytvořena databáze s entitami pro aplikaci.

2.8 DBeaver

DBeaver je univerzální nástroj pro správu databází, který je postaven na profesionální úrovni. [32] Jedná se o multiplatformní aplikaci, která má přívětivé uživatelské grafické rozhraní, které je jednoduché a intuitivní. Aplikace má open source licenci a je napsaná v programovacím jazyce Java. [32]

Nástroj umožňuje manipulovat s daty, vytvářet SQL dotazy, vytvářet databáze a tabulky a vytvářet analytické sestavy a další. Samotný program umí i zobrazit ER model dané databáze. Tato funkce se dá využít, pokud potřebujeme k projektu doložit ER model databáze, popřípadě nám tento model může sloužit k porovnání s navrženým ER modelem, zda jsme databázi správně navrhli a provedli správnou implementaci.

Aplikace komunikuje s jakoukoliv databází, například s MariaDB, MySQL, MongoDB, PostgreSQL, Oracle, SQLite a jiné.



Obrázek 9 – Prostředí aplikace DBeaver

II. PRAKTICKÁ ČÁST

3 ROZBOR A ANALÝZA POŽADAVKŮ

Při konzultaci s panem doc. Ing. Petrem Šilhavým, Ph.D. jsem byl seznámen se stávající situací a dle toho byly určeny minimální požadavky na tvorbu úvazků akademických pracovníků.

Jak už bylo řečeno v první kapitole, tak úvazkový systém uchovává minimálně jméno a příjmení o zaměstnanci, tedy základní informace o akademickém pracovníkovi. Důležité je zahrnout i tituly před jménem a za jménem, kvůli odbornosti. Dále by systém měl uchovávat určité kontakty na dané pracovníky, tedy pracovní e-mail a pracovní telefonní číslo, popřípadě soukromý e-mail a soukromé telefonní číslo. Většina zaměstnavatelů v dnešní době již poskytuje svým zaměstnancům jak pracovní e-maily, tak služební telefonní čísla.

Při analýze požadavků bylo zmíněno, že daný lektor může mít částečný či plný úvazek a dále, že se může lišit typ pozice.

Typy pozicí jsou následující:

- Asistent
- Odborný asistent
- Docent
- Profesor

Každý akademický pracovník by měl mít přidělené jednotlivé akce předmětů, které učí za aktuální akademický rok. Předměty se dělí na 3 části.

Předmět se dělí na:

- Přednáška
- Cvičení
- Seminář

Daný předmět má určitý počet studentů. Na každý předmět v daném ročníku může chodit odlišný počet studentů, protože někteří předmět můžou opakovat podruhé, anebo si předmět zapsali až v dalším ročníku. Učebny volíme pro plný počet studentů tak, aby se počet studentů do dané učebny vešel. Uvedeme příklad, na daný předmět může být zapsáno 40 studentů, ale také 150 studentů. Pro takto velký počet studentů jsou určené velké přednáškové místnosti s dostatečnou kapacitou míst.

Ale cvičení, kdy potřebujeme například počítačovou učebnu, tak velkou kapacitu pojmout nemůže, proto se cvičení dělí na menší skupiny pro lepší práci akademických lektorů. Cvičení dělíme na malé či středně větší skupiny, například na 12 nebo 24 studentů.

Vysoké školy se dělí na univerzity nebo na neuniverzitní vysoké školy. Univerzita se dále dělí na fakulty, například Fakulta aplikované informatiky. Dále se fakulty dělí na katedry, užívá se také označení vysokoškolský ústav. [33]

Každá fakulta se skládá ze studijních oborů pro studenty. Obor se skládá z ročníků. Bakalářský obor má převážně 3 roky studia. Navazující magisterské studium trvá většinou 2 roky, tedy má 2 ročníky. Délka trvání studia jednotlivých studentů je také ovlivněna individuálně, tj. student může opakovat ročník například z důvodu nesplnění některého z povinných předmětů. Ročníky se dále dělí na jednotlivé semestry.

Školní rok se na vysokoškolské půdě nazývá akademický rok, ten trvá 12 měsíců a dělí se na 2 semestry.

A to na:

- Zimní semestr
- Letní semestr

Do semestru se zahrnuje výuka, zkoušky a jiné akce, například praxe.

Akademický rok je přelom mezi 2 roky a může vypadat například takto: 2020/2021.

Každý studijní obor má studijní plán, podle kterého se vyučuje. Studijní plán se dělí na ročníky dle typu oboru. A ročníky se dělí na jednotlivé semestry. Do jednotlivých semestrů jsou přiděleny předměty. Předměty nemusí, ale můžou na sebe také navazovat z 1. semestru do 2. semestru, například v 1. semestru se budou vyučovat Základy počítačových sítí a v 2. semestru se bude konat navazující předmět Pokročilé počítačové sítě. Pokud budeme mít náročnější předmět jako je matematika, tak tento předmět může být vyučován jak v zimním semestru, tak v letním semestru.

Předmět má 2 možné zakončení:

- Zápočet
- Zkouška

Každý předmět je ohodnocený určitým počtem kreditů. Počet kreditů se může lišit dle náročnosti předmětu.

Studijní obor v daném akademickém roce může být nový a teprve začínat, ale také může studijní obor za 5 let skončit a už nebude znovu vypsán, pouze se dokončí studium zbylých studentů, kteří jsou v pokročilejších ročnících než v 1. ročníku. Z toho vyplývá, že v akademickém roce může být studijní obor například Softwarové inženýrství vypsáno, ale už v dalším akademickém roce nemusí být vypsáno.

V daném akademickém roce může být určitý počet studentů na určitý studijní obor, v daném ročníku je zapsaný určitý počet studentů, ale z hlediska předmětů se situace liší. Protože můžou být studenti, co mohou předmět opakovat a tedy na daný předmět se může zapsat více studentů než ti studenti, kteří jsou v daném ročníku.

Z úvazkového hlediska pro tvorbu úvazku pro akademického pracovníka nám stačí pouze znát předměty. Pro lepší orientaci a rozdělování předmětů pro systém bude lepší znát studijní plán, který bude rozdělen do jednotlivých ročníků a semestrů.

Jednotlivé obory se dělí na:

- Prezenční formu studia
- Kombinovanou formu studia

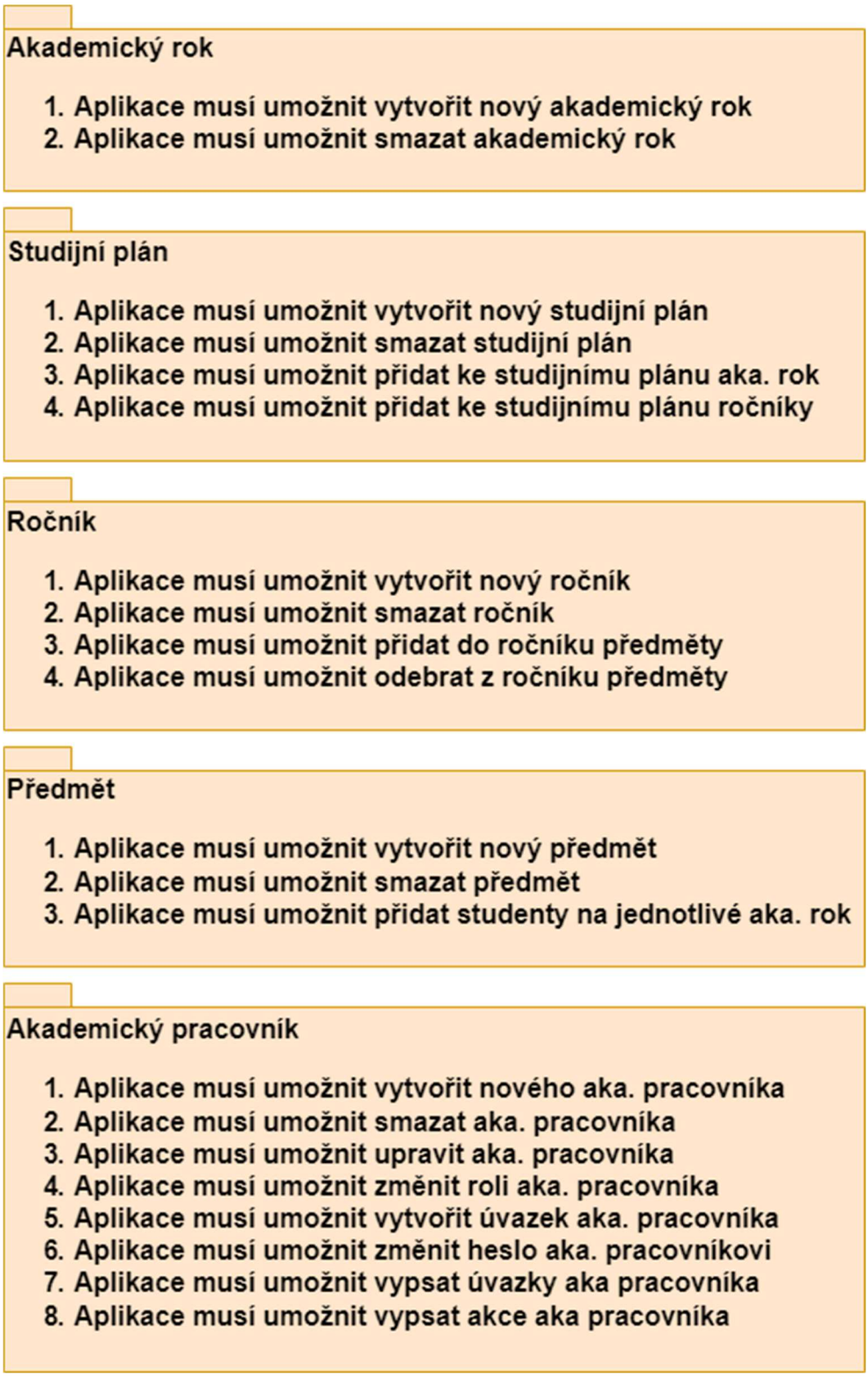
Prezenční forma studia je forma studia, při které studenti dochází osobně během týdne do školy, převážně každý den od pondělí do pátku dle rozvrhu. Výuka trvá většinou 14 týdnů. V této formě studia jsou předměty většinou rozděleny na přednášku a cvičení.

Kombinovaná forma studia je forma studia, při které studenti dojíždí do školy buď každou sobotu, popřípadě jednou za 14 dní na výuku v pátek a sobotu, záleží na dané škole. Délka výuky je stejná jako u prezenční formy studia, tedy 14 týdnů. Výuka se od prezenční formy studia liší v tom, že předměty jsou zkrácené nebo vyučované jako vícehodinové bloky. Například předmět, který je v prezenční formě studia vyučován ve 14 dnech po jedné hodině, tak v kombinované formě studia může být vyučován jen ve 3 dnech po 4hodinových blocích. Předmět je převážně v této formě studia rozdělen jen na přednášky, občas může obsahovat také cvičení.

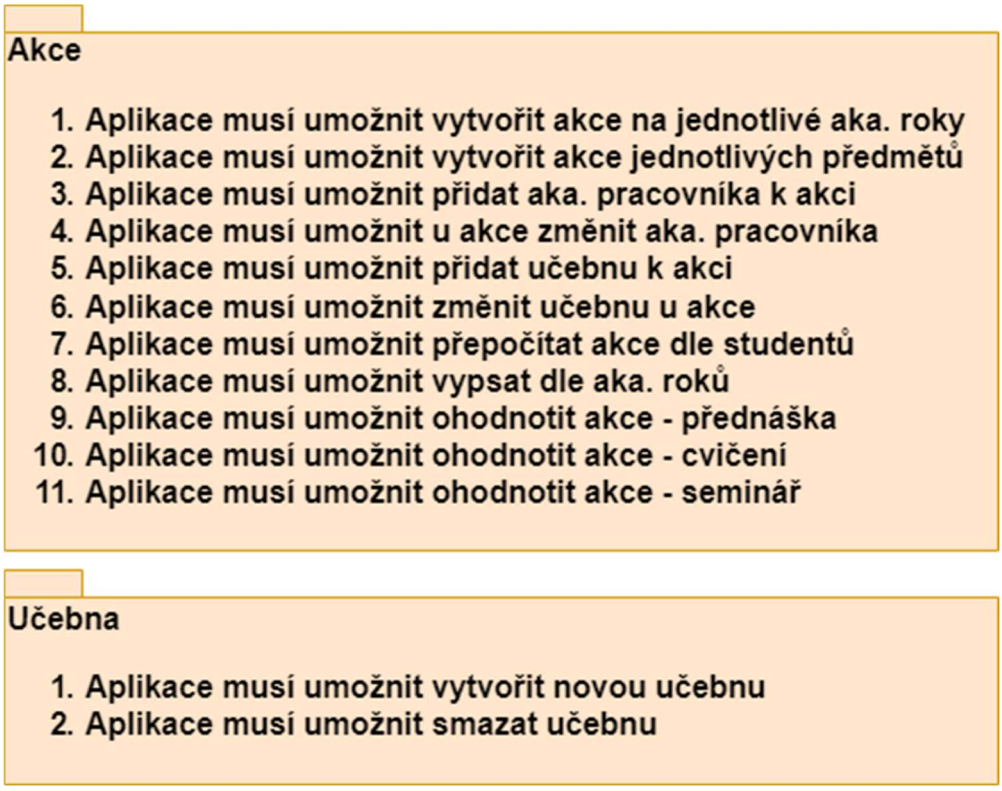
Pro minimální funkčnost tvorby úvazků nám stačí, aby systém uměl generovat akce jednotlivých předmětů dle potřeby na přednášky, cvičení, popřípadě semináře. A aby se u daných akcí dal nastavit počet týdnů a počet hodin.

3.1 Funkční požadavky

Mezi funkční požadavky patří jednotlivé aktivity a akce, které musí být zpracovány systémem pro tvorbu úvazků akademických pracovníků.



Obrázek 10 – Funkční požadavky – část 1



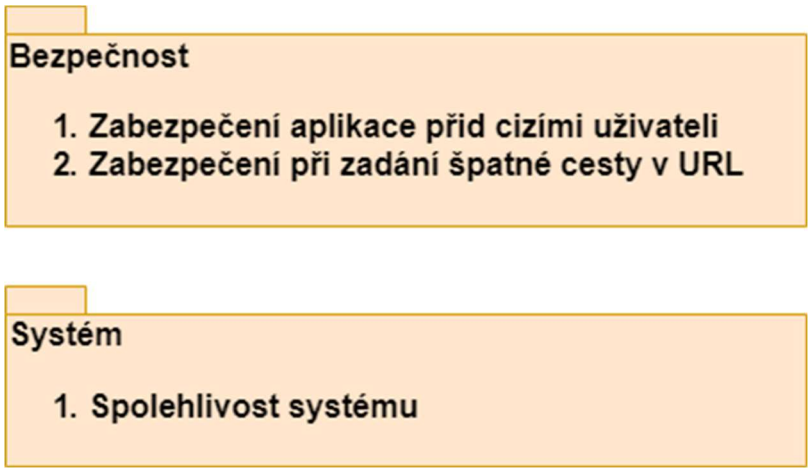
Obrázek 11 – Funkční požadavky – část 2

Na obrázcích výše jsou popsány jednotlivé funkční požadavky, které by aplikace měla umožňovat.

3.2 Nefunkční požadavky

Mezi nefunkční požadavky patří požadavky na zabezpečení, výkon, spolehlivost, efektivitu a přenositelnost aplikace.

Z analýzy požadavků vznikly nefunkční požadavky, viz obrázek níže.



Obrázek 12 – Nefunkční požadavky – Bezpečnost a Systém

Hlavním nefunkčním požadavkem je bezpečnost. Pokud by se k systému dostal uživatel, který nemá přístupové údaje, tak je nutné zajistit, aby se uživatel nedostal dále do systému. A pokud by napsal do *Uniform Resource Locator* (URL) například https://nasweb.cz/uzivatele tak, aby byl odkázán na přihlašovací okno. Dále pokud by uživatel zadal špatnou URL, tak by systém uživatele neměl nikam pustit a měl by se vrátit zpět na hlavní stránku.

Druhou částí na požadavky je spolehlivost. Jedná se o to, aby systém fungoval bez problému.

3.3 Případy užití

Při analýze požadavků jsme dále zjistili, že by systém měl umožnit přihlášení. S touto problematikou souvisí jeho zabezpečení. Z toho vychází několik uživatelských oprávnění a několik pohledů na daný systém, protože uživatelé mají různé pravomoci v systému.

Množstvím požadavků z analýzy vznikl souhrnný diagram případů užití neboli use case diagram.

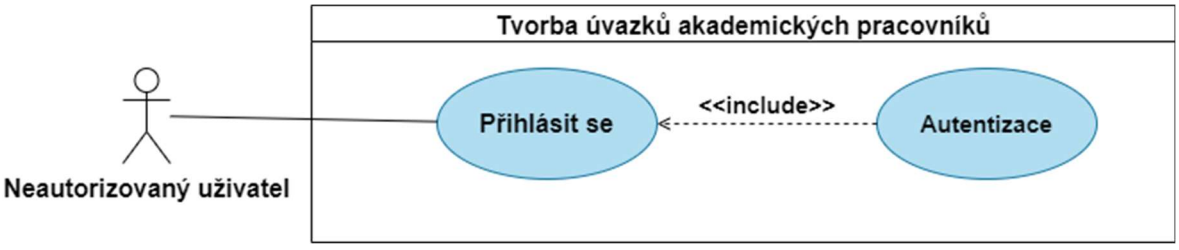
Pohledy na systém nazýváme v diagramu případu užití jako aktéry.

Aktéři:

- Neautorizovaný uživatel
- Akademický pracovník
- Správce úvazku
- Administrátor

3.3.1 Neautorizovaný uživatel

Prvním pohledem na systém je neautorizovaný uživatel neboli nepřihlášený uživatel, který přistupuje k aplikaci. Na obrázku je znázorněno, jaké možnosti by měl mít nepřihlášený uživatel.

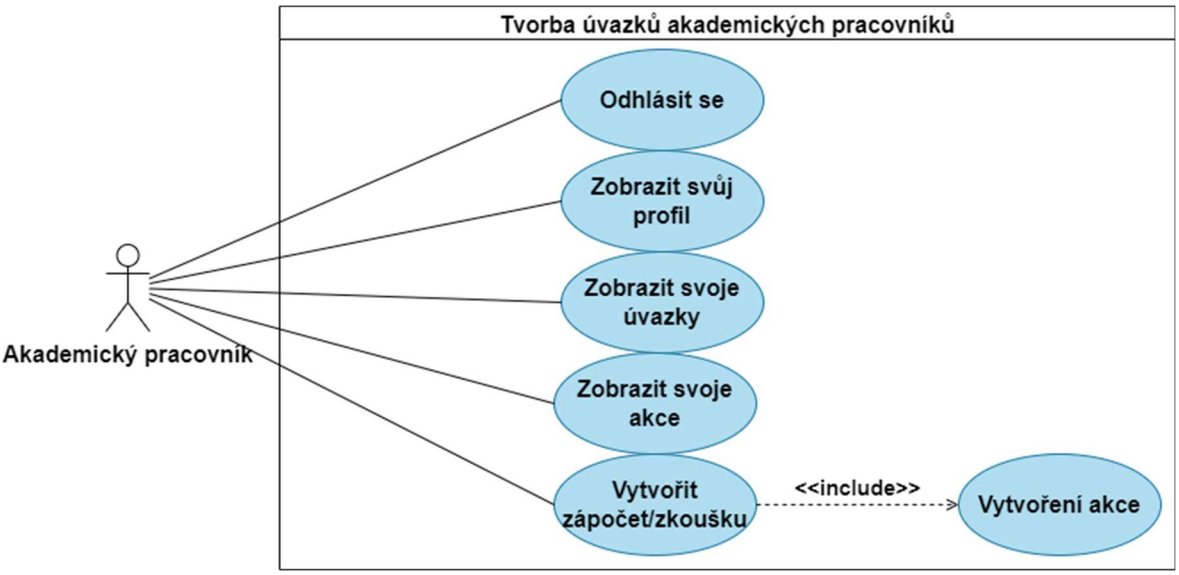


Obrázek 13 – Případ užití – neautorizovaný uživatel

Nepřihlášený uživatel se může pouze do systému přihlásit, aby dále mohl pracovat a využít potenciál systému. Při přihlášení dochází ke komunikaci se systémem, přesněji dochází k autentizaci, která ověří jeho přihlašovací údaje v databázi uživatelů.

3.3.2 Akademický pracovník

Akademický pracovník je aktér, který bude mít pouze omezené možnosti.

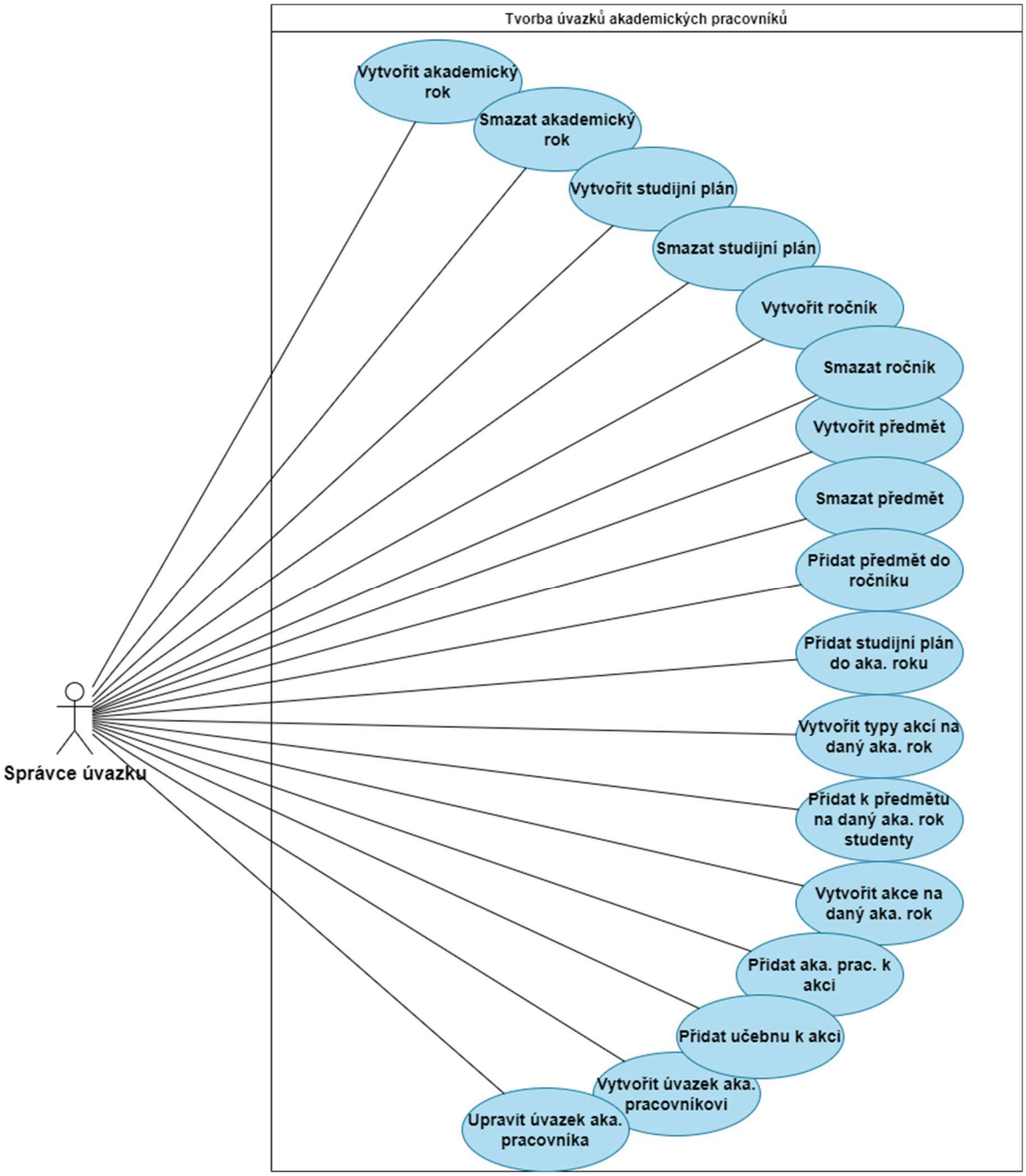


Obrázek 14 – Případ užití – akademický pracovník

Na obrázku je zobrazen aktér akademický pracovník a jednotlivé buňky, které prezentují jednotlivé činnosti, které smí aktér vykonávat. Daný aktér dědí možnosti od aktéra neautorizovaný uživatel – tedy aktér akademický pracovník se může do systému přihlásit, ale má i možnost se odhlásit ze systému. Dále si akademických pracovník může zobrazit svůj osobní profil, jedná se o jeho osobní údaje. Dále si může zobrazit své vlastní úvazky, například zda má poloviční úvazek nebo plný úvazek a jaký má typ pozice. Také si může zobrazit své vlastní akce, tedy jaké předměty učí. V neposlední řadě si může vytvořit vlastní akce na zápočet nebo zkoušku z předmětu, který vyučuje, ale musí z daného předmětu mít přednášku.

3.3.3 Správce úvazku

Aktér správce úvazků má už větší pravomoc pracovat v systému. Dědí funkce po aktérovi akademický pracovník a předchozí funkce od aktéra neautorizovaný uživatel. Aktér správce úvazků má širokou nabídku možností, co vše v systému může vykonávat, jeho oprávnění jsou taková, že může 2/3 celkových dat v systému upravovat.



Obrázek 15 – Případ užití – správce úvazku

Na obrázku je zobrazený aktér správce úvazku. Aktér správce úvazku může vytvářet nové akademické roky, zároveň je může i odstranit.

Správce úvazku může spravovat a vytvářet nové studijní plány. Následně může vytvářet nové ročníky do studijních plánů a do těchto ročníků může správce úvazku přidávat jednotlivé předměty, které si musí nejdříve vytvořit do seznamu předmětů. Ve správě předmětů může odstranit i konkrétní předmět.

Další jeho pravomocí je přidat studijní plán do akademického roku. Dále správce úvazku může vytvořit typy akcí na daný akademický rok. A poté může vytvořit akce předmětů na daný akademický rok. Pokračující možností je přidat akademického pracovníka k akci a přidat učebnu k akci.

Před závěrečnými funkcemi jsou funkce vytvářet učebny a odstraňovat učebny. Správce úvazku může tyto učebny přidávat k jednotlivým akcím.

Správce úvazku může také přidávat úvazky akademickým pracovníkům a ukončovat úvazky akademických pracovníků.

3.3.4 Administrátor

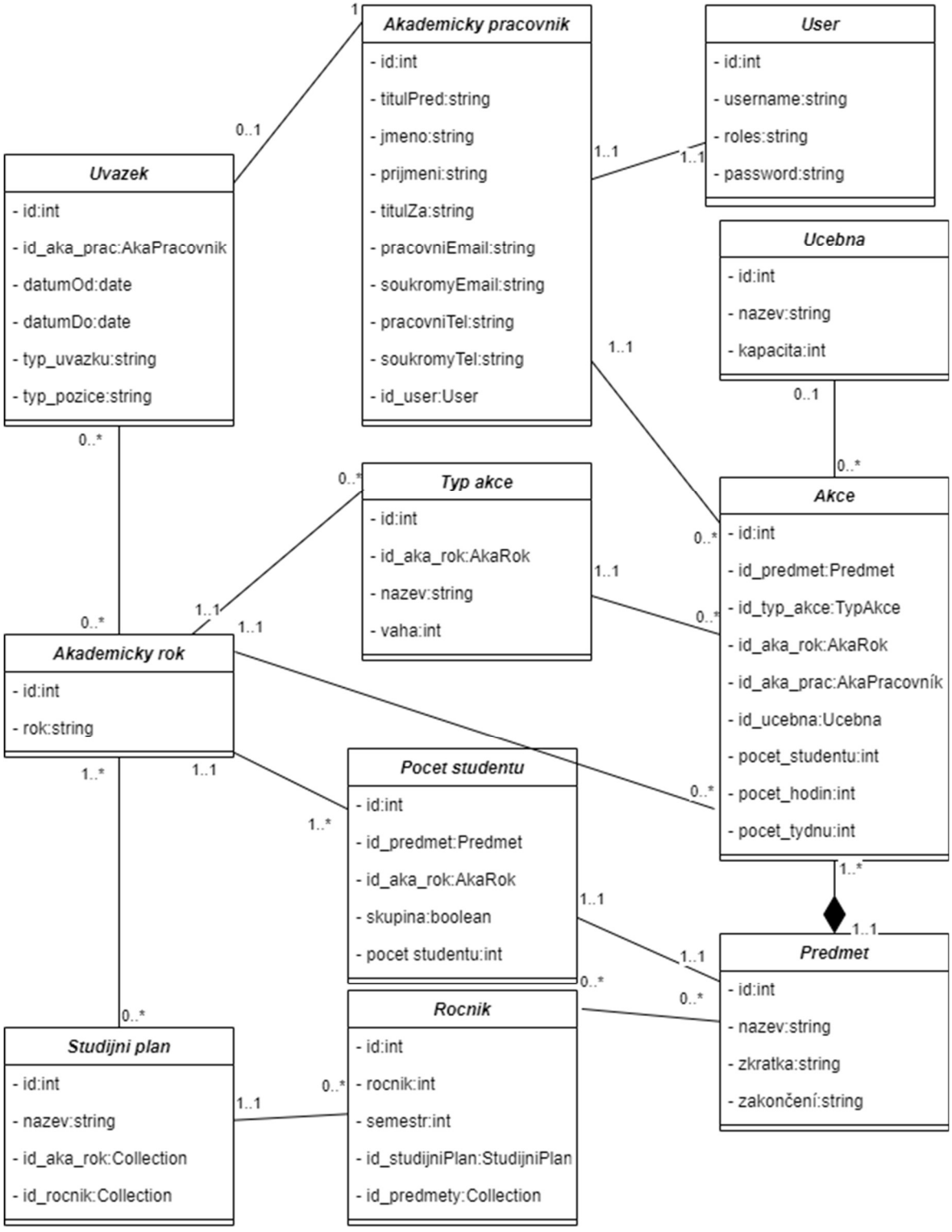
Aktér administrátor dědí všechny funkce po aktéru správce úvazku a po aktéru akademický pracovník. Na obrázku níže jsou zobrazeny všechny aktivity, které může aktér administrátor vykonávat.

Obrázek 16 – Případ užití – administrátor

Administrátor může vytvářet nové akademické pracovníky. Může uživatele upravovat nebo smazat. Dále může uživateli změnit oprávnění z akademického pracovníka na správce úvazku, popřípadě na administrátora systému. Poslední pravomocí je možnost resetovat uživateli heslo.

3.4 Diagram tříd

V diagramu tříd jsou znázorněny jednotlivé třídy a vztahy mezi nimi. Z rozboru požadavků vznikl diagram tříd, který popisuje objekty, které musí být v systému.



Obrázek 17 – Diagram tříd – tvorby úvazku akademických pracovníků

Na obrázku jsou zobrazeny jednotlivé třídy pro minimální funkčnost systému, která dosahuje tvorby úvazků akademických pracovníků. Jedná se o entity. Na diagramu nejsou zobrazeny třídy řadičů, které budou ovládat jednotlivé pohledy. Názvy tříd a jednotlivých atributů jsou v aplikaci zaznamenány bez diakritických znamének, v této práci budeme používat názvy s diakritickými znaménky.

Diagram se skládá z 11 tříd, které mají několik atributů a funkcí pro jednotlivé ovládání. Atributy jsou označené znakem - „mínus“, protože se jedná o privátní položky. A většinou funkce jako veřejné jsou označeny znakem + „plus“.

Třídy jsou následující:

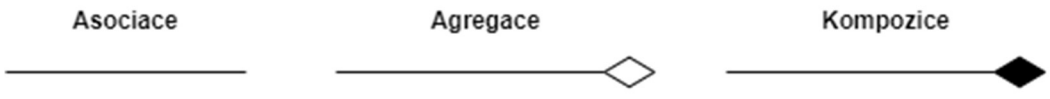
- Akademický rok
- Studijní plán
- Ročník
- Předmět
- Počet studentů
- Učebna
- Typ akce
- Akce
- Akademický pracovník
- User
- Úvazek

Mezi jednotlivými třídami jsou vztahy a mohutnost vztahu. Mohutnost vztahu značí to, kolik instancí třídy 1 může být svázáno s třídou 2.

Tabulka 1 – Mohutnost vztahu mezi třídami

Žádná instance	Žádná nebo právě jedna	Právě jedna instance	Žádná nebo více instancí	Jedna nebo více instancí
0	0..1	1	0..*	1..*

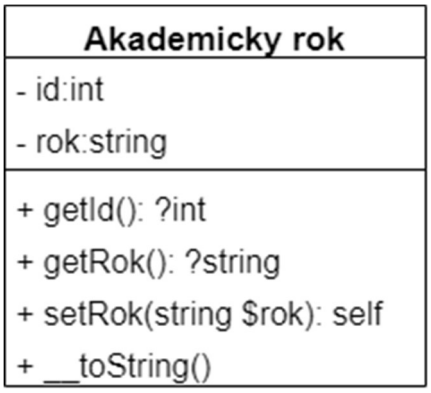
Asociace je vztah mezi 2 třídami. Agregace je vztah, kdy 1 objekt může existovat bez 2. objektu. Kompozice je vztah kdy 1 objekt nemůže existovat bez 2. objektu.



Obrázek 18 – Vztahy mezi třídami

3.4.1 Akademický rok

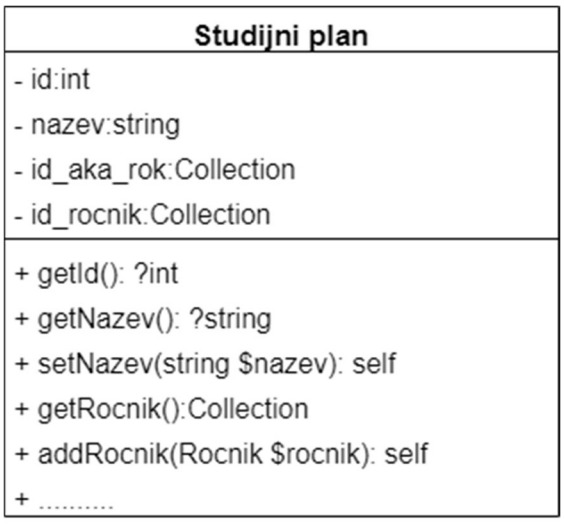
První hlavní třídou je třída Akademický rok, jedná se o třídu, do které se budou načítat jednotlivé řádky z databáze. Třída obsahuje atribut objektu id v datovém typu integer a poté atribut rok v datovém typu string. A jednotlivé metody pro vrácení hodnot z jednotlivých proměnných a nastavení hodnot. A funkce __toString() pro vrácení názvu objektu.



Obrázek 19 – Třída – Akademický rok

3.4.2 Studijní plán

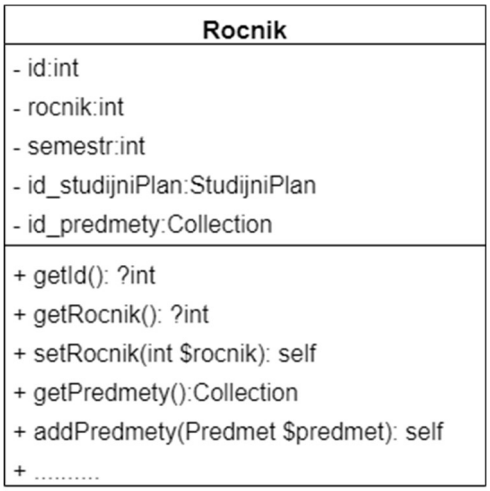
Druhou třídou, která navazuje na akademický rok, je třída s názvem Studijní plán. Třída obsahuje atribut id v datovém typu integer, název v datovém typu string – pro název studijního plánu. A dále obsahuje kolekce pro akademické roky a ročníky. A následně obsahuje jednotlivé metody pro nastavení a vkládání a odebírání objektů z kolekcí.



Obrázek 20 – Třída – Studijní plán

3.4.3 Ročník

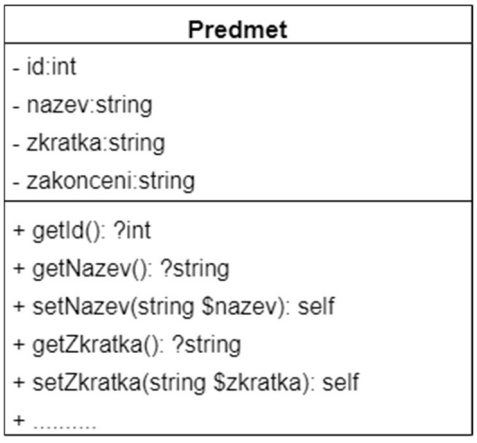
Třetí třída se nazývá Ročník. Ročník obsahuje atribut id v datovém typu integer, dále obsahuje atribut ročník, jenž je v datovém typu integer, který slouží pro číslo ročníku, zda se jedná o 1. ročník či 2. ročník. Poté obsahuje atribut semestr, jenž je v datovém typu string. Tento atribut slouží k rozpoznání zimního a letního semestru. A předposlední proměnná odkazuje na id objektu studijní plán, ke kterému ročník patří. Na závěr ročník má kolekci s přiřazenými předměty. Následně obsahuje jednotlivé funkce pro ovládání jednotlivých proměnných.



Obrázek 21 – Třída – Ročník

3.4.4 Předmět

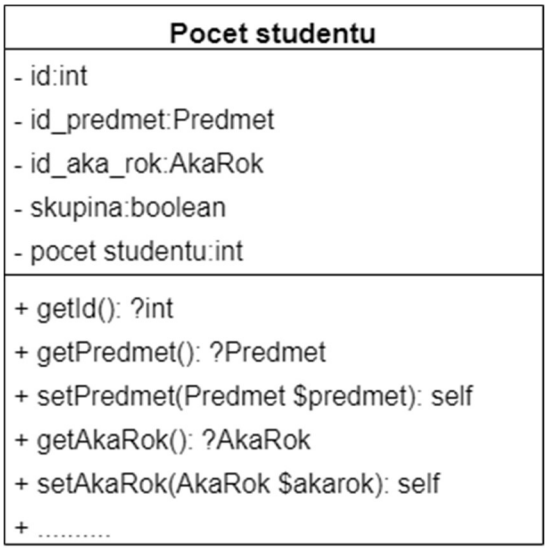
Další důležitou třídou je třída Předmět. Předmět obsahuje atribut proměnné id typu integer, název v typu string, zkratka předmětu ve formátu string a na závěr zakončení předmětu v datovém formátu string.



Obrázek 22 – Třída – Předmět

3.4.5 Počet studentů

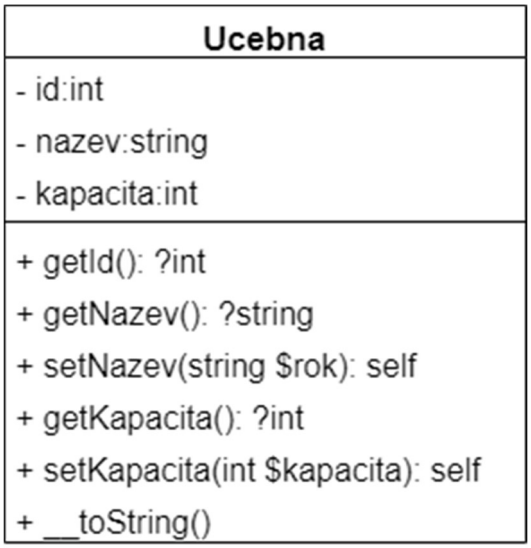
Třída pro objekt Počet studentů obsahuje atribut id typu integer. Dále obsahuje odkaz na id předmětu, ke kterému patří, poté i id na akademický rok. Tím bude zajištěno, že pokud bude daný předmět v akademickém roce například 2021/2022, tak se pro tento rok musí zadat počet studentů, kteří budou docházet na tento předmět. V neposlední řadě je proměnná s názvem skupina, jenž je tvaru boolean. Datový typ boolean nabývá hodnoty true a false. Pro hodnotu true je velká skupina s počtem 24 studentů a pro hodnotu false je malá skupinu o 12 studentech. A posledním hlavním atributem je počet studentů. Tento atribut je typu integer.



Obrázek 23 – Třída – Počet studentů

3.4.6 Učebna

Třída Učebna obsahuje proměnnou id, následující proměnnou je proměnná typu textový řetězec string pod názvem název. Tato proměnná slouží pro název učebny. Poslední proměnnou je proměnná celočíselného typu integer pod názvem kapacita. Jedná se o kapacitu učebny, tj. kolik studentů se do dané učebny vejde. Následně třída obsahuje funkce pro nastavení id, názvu a kapacity.



Obrázek 24 – Třída – Učebna

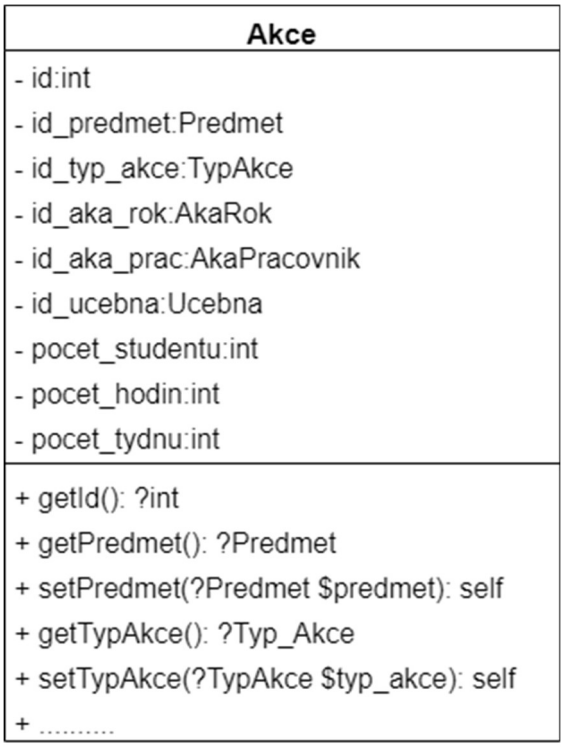
3.4.7 Typ akce

Typ akce je následující třída pro objekt. Typ akce obsahuje atribut pod názvem id typu integer. Druhým atributem je id odkazující se na akademický rok. Třetím atributem je název, do názvu bude k vybrání, zda se jedná o přednášku, cvičení nebo seminář. Jedná se o textový řetězec string. A poslední atribut je váha a je celočíselného typu integer.

Obrázek 25 – Třída – Typ akce

3.4.8 Akce

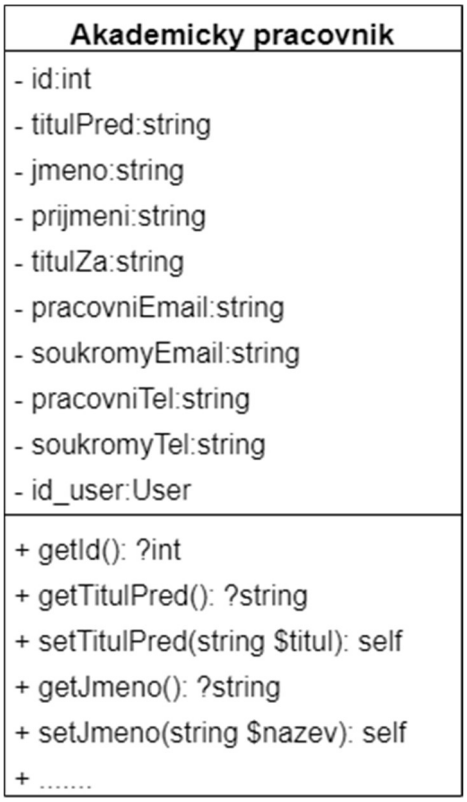
Jedná se o jednu z nejdůležitějších tříd pro objekt Akce. Tato třída Akce obsahuje jako první atribut id objektu. Jako druhý atribut je id odkazující se na id předmětu. Třetí proměnnou je proměnná, jež se odkazuje na id objektu typ akce. Následující proměnné se odkazují na daný akademický rok a akademického pracovníka, který tuto akci bude mít. Důležitými atributy jsou počet hodin a počet týdnů. Jedná se o atributy, z kterých budeme znát, kolik hodin se předmět v daném týdnu bude vyučovat a z druhého atributu bude vycházet, kolik týdnů se bude daná akce konat, zda 14 týdnů jako je tomu u prezenční formy studia, nebo zda bude výuka trvat třeba jen 4 týdny.



Obrázek 26 – Třída – Akce

3.4.9 Akademický pracovník

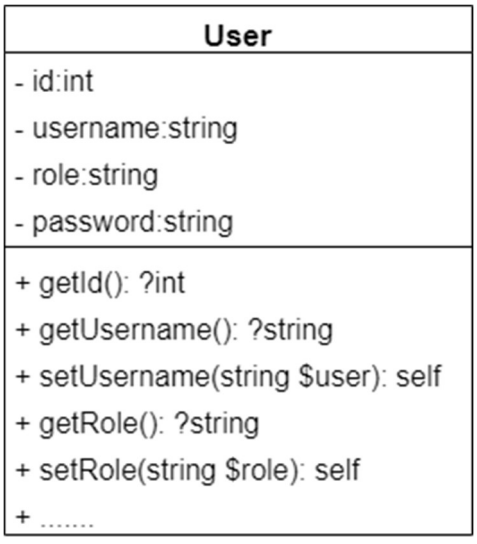
Akademický pracovník je druhá nejdůležitější třída. Tato třída bude mít své univerzální id. Dále atributy typu titul před jménem, titul za jménem, jméno a příjmení. Tyto atributy jsou všechny formátu textový řetězec string. Dále akademický pracovník bude mít pracovní e-mail a pracovní telefonní číslo, obě proměnné budou typu textový řetězec string. Akademický pracovník má další 2 atributy, a to typu textový řetězec string, ten se liší v názvu, kdy se jedná jen o soukromý e-mail a soukromé telefonní číslo.



Obrázek 27 – Třída – Akademický pracovník

3.4.10 User

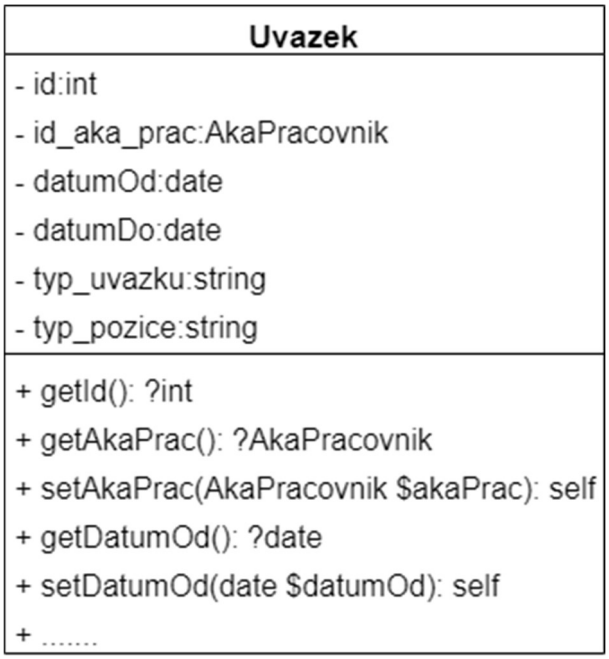
User je další důležitá třída, která řeší uživatele neboli users, nazvaná User. První proměnnou je id. Druhou proměnou je username, jenž je typu textový řetězec string. Poté je atribut password. Předposledním atributem je role, do kterého se budou ukládat role akademicky pracovník, správce úvazku, administrátor. Následně je atribut password pro heslo.



Obrázek 28 – Třída – User

3.4.11 Úvazek

Úvazek je poslední třída. Tato třída slouží pro to, abychom zjistili, na jaké pozici se nachází daný lektor. Prvním atributem je id v datovém typu integer, následující atribut odkazuje na akademického pracovníka. Dalším atributem je datumOd, zde se nastaví datum daného úvazku. Čtvrtým atribut je datumDo, který znamená dokdy platí daný úvazek. Pátým atributem je typ úvazku ve formátu string. Posledním atributem je typ pozice.



Obrázek 29 – Třída – Úvazek

3.5 Závěr z analýzy

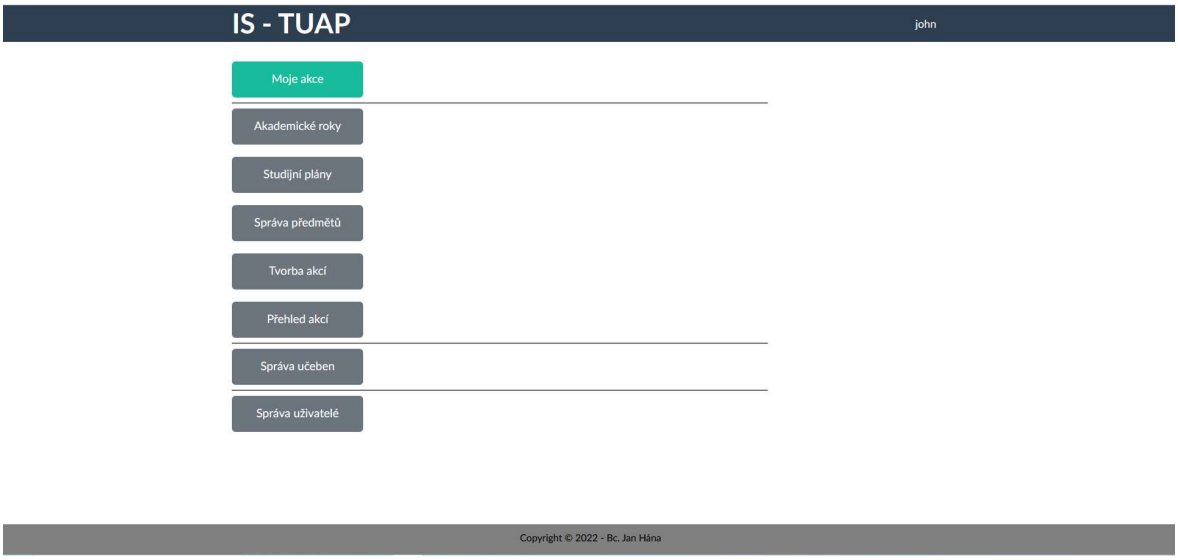
Díky analýze máme vyhodnocené funkční a nefunkční požadavky na daný systém. Zjistili jsme, že nejvhodnější řešení pro tvorbu úvazků akademických pracovníku je webová aplikace alias informační systém.

4 ŘEŠENÍ

Tato kapitola se zabývá řešením části naprogramované webové aplikace. Popisuje, z čeho se celá aplikace skládá.

4.1 Realizace aplikace

Webovou aplikaci pro tvorbu úvazků akademických pracovníků jsem pojmenoval pod zkratkou IS – TUAP. Tedy zkratka IS značí informační systém a zkratka TUAP značí tvorbu úvazků akademických pracovníků. Celá aplikace obsahuje několik tříd, které jsou komplexní pro jednotlivé funkce aplikace.



Obrázek 30 – Uživatelské rozhraní webové aplikace IS - TUAP

Aplikace je založena na skriptovacím jazyce PHP ve verzi 8 a frameworku Symfony 6.

4.2 Struktura

Projekt je rozvržen do stromové struktury, kterému při generování projektu napomáhá Symfony framework.

Tabulka 2 – Struktura projektu

Projekt	Adresář	Podadresář
is_tuap		
	assets	
	bin	
	config	
	migrations	
	public	
	src	
		Controller
		Entity
		Form
		Repository
		Security
		…
	teplates	
		homepage
		ucebna
		…
	var	
	vendor	

Projekt se skládá z hlavní složky, ve které jsou další adresáře. Assets je adresář pro vzhled webové aplikace, jsou zde jednotlivé soubory pro CSS styly.

Adresář bin obsahuje PHP konzoli.

Config adresář obsahuje jednotlivé konfigurační soubory pro celý projekt. Obsahuje například jednotlivé konfigurační soubory pro služby, které běží s aplikací. Dále obsahuje i konfigurační soubory pro zabezpečení aplikace.

Migrations je adresář, do kterého se vytváří konfigurace pro databázi. Když přes příkazový řádek zadáme *php bin/console make:migration* vytvoří se konfigurační soubor pro databázi. Další příkaz *php bin/console doctrine:migrations:migrate* nám vytvoří přesné tabulky v databázi, popřípadě pokud už dané tabulky existují, tak nám je upraví.

Public obsahuje zkompilované soubory pro CSS styly.

Adresář src obsahuje další podadresáře Controller, Entity, Form, Repository, Security a může obsahovat další. Názvy jednotlivých adresářů označují jejich obsah, tedy v podadresáři Controller jsou jednotlivé řadiče pro pohledy. Adresář Entity slouží pro naše jednotlivé tabulky. Podadresář Form je pro jednotlivé formuláře, když chceme například vytvořit nový objekt akademický pracovník, popřípadě chceme daného pracovníka upravit.

V podadresáři Repository jsou jednotlivé dotazy na databázi. A v posledním podadresáři Security se nachází ovladač pro přihlašování do aplikace.

V Teplates se ukrývají jednotlivé pohledy, které jsou roztříděny dále do podadresářů.

V adresáři var jsou umístěny jednotlivé vygenerované soubory jako je mezipaměť, protokoly a další.

Vendor adresář obsahuje jednotlivé knihovny, které aplikace vyžaduje pro svoji funkčnost.

4.3 Controller

Controller řídí jednotlivé pohledy, může do těchto pohledů posílat i data. Controller může obsahovat několik funkcí pro vykreslení různých pohledů.

```php
/**
 * Funkce pro zobrazeni vsech uceben
 */
#[Route('/ucebna_show', name: 'ucebna_show')]
#[IsGranted('ROLE_UVAZEK_ADMINS')]
public function index(UcebnaRepository $ucebnaRepository)
{
    return $this->render( view: 'ucebna/show.html.twig', [
        'ucebny' =>  $ucebnaRepository->findAll(),
    ]);
}
```

Obrázek 31 – Controller Učebna – Funkce pro zobrazení všech učeben

Na obrázku je znázorněna funkce pro vypsání všech učeben z řadiče Učebna. Funkce má nad sebou nastavenou cestu, která má svůj název pro zavolání této funkce.

Následně je na danou funkci nastaveno oprávnění, které umožňuje, kdo k dané funkci může přistupovat a kdo danou funkci může zavolat.

Funkce vrací zavolání pohled pro zobrazení všech učeben. Do tohoto pohledu je pak přidán ještě repositář, který předává všechny učebny.

```php
/**
 * Funkce pro pridani ucebny
 */
#[Route('/ucebna_add', name: 'ucebna_add')]
#[IsGranted('ROLE_UVAZEK_ADMINS')]
public function add(Environment $twig, Request $request, EntityManagerInterface $entityManager)
{
    $ucebna = new Ucebna();

    $form = $this->createForm( type: UcebnaFormType::class, $ucebna);
    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()){
        $entityManager->persist($ucebna);
        $entityManager->flush();

        return $this->redirectToRoute( route: 'ucebna_show');
    }

    return new Response($twig->render( name: 'ucebna/add.html.twig', [
        'form' => $form->createView()
    ]));
}
```

Obrázek 32 – Controller Učebna – Pro přidání učebny

Na obrázku je znázorněna funkce pro přidání nové učebny. Samotná metoda má opět svoji cestu a název. Každá cesta a název musí být unikátní a jedinečný.

Funkce vytváří nový objekt učebna. A pomocí správných příkazů vytváří formulář pro učebnu, který již existuje zvlášť ve své třídě. Následně je podmínka, která čeká na to, až ve formuláři někdo zmáčkne tlačítko Vytvořit a provede se naplnění nového objektu, které se přidá do databáze a následně je uživatel přesměrován na výpis všech učeben.

4.4 Template

Template neboli pohled, jedná se o HTML soubory, které mají dvě koncovky, například ucebna.html.twig. První koncovka je klasické HTML a druhá je Twig. Twig koncovka je pro šablonový systém pro programovací jazyk PHP.

```
{% extends 'base.html.twig' %}

{% block body %}
    <div class="body-cointainer">
        <div class="main">
            <h1>Seznam učeben</h1>
            {% if ucebny|length > 0 %}
            <table class="table table-striped">
                <thead>
                    <tr>
                        <th class="th-left">Název</th>
                        <th class="th-left">Kapacita</th>
                        <th class="th-center">Akce</th>
                    </tr>
                </thead>
                <tbody>
                    {% for ucebna in ucebny %}
                        <tr>
                            <td>{{ ucebna.nazev }}</td>
                            <td>{{ ucebna.kapacita}}</td>
                            <td>
                                <a href="" class="btn btn-sm btn-danger" role="button">Odstranit</a>
                            </td>
                        </tr>
                    {% endfor %}
                </tbody>
            </table>
            {% else %}
                Zatím nebyla vytvořena žádná učebna!
            {% endif %}
        </div>
        <div class="sidebar">
            <a  href="{{ path("ucebna_add") }}" class="btn btn-lg btn-success" role="button"><b>+</b> Vytvořit učebnu</a><br>
            <a href="{{ path('home_index') }}" class="btn btn-lg btn-primary" role="button"><b><-</b> Zpět</a><br>
        </div>
    </div>
{% endblock %}
```

Obrázek 33 – Template – Učebna – vypsání všech učeben

HTML soubor zahrnuje klasické HTML tagy a obsahuje jednotlivé příkazy pro vypsání předaných dat z řadiče.

Soubor dědí ze souboru base.html.twig hlavičku a patičku pro šablonu. A v této šabloně pro vypsání učeben je nastaveno pouze tělo šablony. Tělo šablony má jednotlivé HTML tagy, které mají nastavené své třídy pro získání stylů, jak budou vypadat a jak prvky budou umístěny.

Dále pohled po vypsání učeben má podmínku, jež zjišťuje z předaných dat, zda jí byly předány nějaká data o učebnách, anebo předání bylo prázdné. Pokud předání bylo prázdné, tak se v šabloně uživateli vypíše, že zatím nebyla vytvořena žádná učebna. Pokud ale existuje alespoň jedna učebna, tak podmínka je splněna a přejde na vypsání tabulky a jednotlivých dat pomocí for cyklu.

Šablona obsahuje i tlačítko, které se odkazuje na další šablonu pro vytvoření nové učebny a tlačítko na vrácení zpět na domovskou stránku aplikace.

4.5 Entity

Entity jsou jednotlivé třídy, ve kterých jsou nastavené jednotlivé atributy a funkce pro nastavení těchto entit.

```php
#[ORM\Entity(repositoryClass: UcebnaRepository::class)]
class Ucebna
{
    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column(type: 'integer')]
    private $id;

    #[ORM\Column(type: 'string', length: 255)]
    private $nazev;

    #[ORM\Column(type: 'integer')]
    private $kapacita;

    public function getId(): ?int
    {
        return $this->id;
    }

    public function getNazev(): ?string
    {
        return $this->nazev;
    }

    public function setNazev(string $nazev): self
    {
        $this->nazev = $nazev;

        return $this;
    }
```

Obrázek 34 – Entita – Učebna

Na obrázku je znázorněna entita učebna. Z diagramu tříd víme, jaké má mít přesné atributy a funkce pro správu proměnných. U jednotlivých atributů jsou předdefinované datové typy, například zda se jedná o integer nebo string. U datového typu string je i nastavena délka na 255 znaků.

4.6 Formuláře

Pro každé vytvoření nového prvku existují třídy, ve kterých je předdefinováno, jaké prvky má mít formulář.

```
class UcebnaFormType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options): void
    {
        $builder
            ->add( child: 'nazev')
            ->add( child: 'kapacita', type: IntegerType::class, [
                'attr' => [
                    'min' => 1,
                    'max' => 250
                ]
            ])
            ->add( child: 'save', type: SubmitType::class, ['label' => 'Vytvořit'])
        ;
    }

    public function configureOptions(OptionsResolver $resolver): void
    {
        $resolver->setDefaults([
            'data_class' => Ucebna::class,
            'csrf_protection' => true,
            'csrf_field_name' => '_token',
            'csrf_token_id' => 'ucebna_item'
        ]);
    }
}
```

Obrázek 35 – Formulář – Pro vytváření učebny

Na obrázku výše je formulář pro vytvoření nové učebny. První funkce je pro vytvoření formuláře. V této metodě se nastavují jednotlivé prvky. U učebny je to název, dále kapacita. U kapacity je nastaveno omezení pro minimální počet kapacity a pro maximální počet kapacity. Dále u prvku můžeme nastavit například, zda splňuje regulární výraz. Situace je například využito u akademického pracovníka, ke kterému zadáváme e-mail nebo telefonní číslo.

V druhé metodě je nastaven vhodný mapovač dat pro vytváření objektů, které má formulář používat. Tedy zde je to učebna. Dále je v metodě nastaveno zabezpečení formuláře, které si popíšeme v další kapitole.

4.7 Repositáře

Repositáře jsou třídy pro dotazy z jednotlivých entit. Defaultně po vytvoření repositáře třída obsahuje defaultní dotazy, tedy dotazy, abychom dostali všechny objekty daného typu z databáze. Dále dotaz dle id objektu.

```php
/**
 * @return TypAkce[] Returns an array of TypAkce objects
 */
public function findAllTypAkce($value)
{
    return $this->createQueryBuilder( alias: 'ta')
        ->innerJoin( join: AkademickyRok::class, alias: 'ar')
        ->where( predicates: 'ta.AkademickyRok = ar.id')
        ->andWhere('ar.rok = :val')
        ->setParameter( key: 'val', $value)
        ->getQuery()
        ->getResult()
        ;
}
```

Obrázek 36 – Repositář – Typ akce

Na obrázku je znázorněn už vlastní dotaz z repositáře pro typ akcí. Tento dotaz vrací všechny typy akcí dle hledaného akademického roku. Dotaz se liší od klasického SQL jazyka tím, že je v příkazech a poté se při samotném dotazování databáze změní na daný SQL jazyk.

V první řadě se dotaz ptá tabulky typ akcí v databázi. Dále přes *INNER JOIN* přidáváme další tabulku, a to akademický rok a v dotazu nejdříve porovnáváme, zda se rovná id od typu akce s id akademického roku a poté se dotazujeme akademického roku pomocí hodnoty na daný rok například, zda se jedná o akademický rok 2021/2022.

Následně nám dotaz vrátí výsledky z hledání.

5 ZABEZPEČENÍ APLIKACE

V dnešní době se klade velký důraz na zabezpečení aplikací, protože může nastat situace, kdy uživatel přijde o své data, když aplikace není zabezpečená. A proto při vývoji aplikace byl kladen důraz na její zabezpečení.

5.1 Útoky na webové aplikace

Dnes existují různé způsoby útoků na webové stránky a webové aplikace.

Typy útoků jsou:

- SQL injection
- Cross-Site Scripting (XSS)
- DOS
- Brute force

5.1.1 SQL injection

SQL injection neboli SQL injekce je typ útoku, který cílí na databázi, která je svázána s webem. Škodlivý kód je vložen do SQL příkazu prostředním vstupu na webové stránce. [34]

Útočník tímto způsobem může získat z databáze citlivá data jako jsou osobní informace, ať už se jedná o rodná čísla nebo čísla kreditních karet, která může dále zneužít pro svůj osobní zisk.

K útoku dochází z toho důvodu, že na webu je umístěný neošetřený vstup, který komunikuje s databází. [35]

Škodlivý kód může zničit celou databázi tím, že smaže všechna data.

5.1.2 Cross-Site Scripting (XSS)

Cross-Site Scripting je další typ útoku na webové stránky, při kterém je do důvěryhodného webu vložen škodlivý skript. Útočníkovi umožnuje ohrozit interakce uživatelů se zranitelnou aplikací. A k útoku dojde, když útočník použije webovou aplikaci k odeslání škodlivého kódu obvykle ve formě skriptu na straně prohlížeče jinému koncovému uživateli. [36]

Prohlížeč koncového uživatele většinou skript spustí, protože skript vypadá, že je v pořádku. Pokud se skript spustí u oběti, tak útočník může ohrozit interakci s aplikací. [37]

5.1.3 DoS

Denial of Services (DoS) je typ útoku, který je cílený na webové služby jako jsou například informační systémy, e-shopy a další.

Cílem útoku je stránku znepřístupnit pro ostatní uživatele. Dochází k zahlcení webového serveru obrovským množstvím požadavků na daný web a server nestíhá požadavky vyřizovat. [38] Tak začne být nedostupný pro ostatní uživatele.

DoS útok je veden z 1 počítače a varianta Distributed Denial of Service (DDoS) vede útok až z několika set tisíc počítačů z různých zemí, aniž by o tom majitelé počítačů věděli, protože jejich počítač mohl být infiltrován a využit právě pro útok na daný cíl. [39] Počítače generují několik set tisíc požadavků na danou službu. Služba se ze začátku může zpomalit, ale nápor požadavků nemusí vydržet a přestane fungovat.

Při takovém útoku většinou nedochází k odcizení dat. [38]

5.1.4 Brute force

Brute force je útok hrubou silou na web. Při kterém se útočník za pomoci generátoru a databáze uživatelských jmen a hesel zkouší přihlásit do systému, dokud nenarazí na ty správné údaje. [40]

Nevýhoda tohoto útoku je ta, že útok je časově náročný na nalezení správných přihlašovacích údajů.

5.2 Přihlášení do aplikace

V první řadě je aplikace zabezpečena přihlašováním. Tato stránka se zobrazí při zadání jakékoliv URL adresy, když uživatel není přihlášen. Uživatel se tak nedostane dál do systému. Přihlašovací formulář je zabezpečený metodou CSRF token, kterou si popíšeme dále v práci.

Obrázek 37 – IS – TUAP – Přihlašovací okno

5.3 Hashovaní hesel

Pro uchovávání hesel uživatelů je potřeba v dnešní době, aby hesla byla uchovávána v bezpečné formě. Proto existují hashovací funkce, které heslo převedou do zašifrované formy. Existuje několik hashovacích funkcí, například MD5, SHA-1, SHA-2 a další.

V aplikaci je využita hashovací funkce Bcrypt pro zašifrování hesel uživatelů. Bcrypt k prostému textu přidá ještě kryptografickou sůl – jedná se o náhodné bity. A potom je celé heslo zašifrováno.

5.4 Role uživatelů

Uživatelé jsou rozděleni do rolí, viz kapitola Rozbor a analýza požadavků. Stejně tak je udělaná i aplikace, ve které jsou práva rozdělená tak, že akademický pracovník má roli uživatele. Správce úvazků má roli správce úvazků a zároveň roli uživatele. A administrátor má roli administrátora a zbylé 2 předchozí role.

Za pomoci rolí uživatel vidí pouze to na co má oprávnění, ale aby byla zajištěna větší bezpečnost, tak jsou jednotlivé funkce v ovladačích zabezpečené příkazem *IsGranted*. *IsGranted* je příkaz, jenž omezuje přístup k funkcím. Do příkazu se píšou názvy rolí, které můžou s funkcí pracovat, tím je zabezpečeno, že k funkci, popřípadě k ovladači můžou přistupovat pouze oprávnění uživatelé. Uživatel, který je přihlášený a snažil by se dostat k URL, ke které nemá oprávnění, tak jej systém vrátí na hlavní stránku. Uživatel, který není přihlášený a snažil by zadávat URL adresy, tak vždy bude vrácen na stránku s přihlášením.

5.5 Zfalšování požadavků mezi stránkami

Zfalšování požadavků mezi stránkami neboli Cross-Site Request Forgery (CSRF) je metoda pomocí které se uživatel pokouší přimět legitimní uživatele, aby nevědomě odeslali data, která nemají v úmyslu odeslat.

CSRF ochrana funguje tak, že do formuláře se přidá skryté pole. Toto pole obsahuje hodnotu, kterou známe pouze my a náš uživatel. Tím je zajištěno, že dané údaje zasílá uživatel – nikoli jiný subjekt. Při každém obnovení stránky se hodnota tokenu mění.

```
<form name="akademicky_rok_form" method="post">
    Akademický rok:
    <input type="text" id="akademicky_rok_form_rok" name="akademicky_rok_form[rok]" required="required" />
    <div>
        <button type="submit" id="akademicky_rok_form_save" name="akademicky_rok_form[save]">
            Vytvořit
        </button>
    </div>
</form>
```

Obrázek 38 – Kód HTML – formulář bez CSRF tokenu

```
<form name="akademicky_rok_form" method="post">
    Akademický rok:
    <input type="text" id="akademicky_rok_form_rok" name="akademicky_rok_form[rok]" required="required" />
    <div>
        <button type="submit" id="akademicky_rok_form_save" name="akademicky_rok_form[save]">
            Vytvořit
        </button>
    </div>
    <input type="hidden" id="akademicky_rok_form__token" name="akademicky_rok_form[_token]" value="
    46926886deeedd9e1bca2ddd309def8a.n24gCCm4erVrMztOfKovsr12chFUOV1E9hGfURnj_NE.6FxFW2DgG90FZFodH_
    BL-dI6C2gVa212kybXZXCQj5bxIEFFYoAo1Blyeg" />
</form>
```

Obrázek 39 – Kód HTML – formulář s CSRF tokenem

Na obrázcích výše je zobrazen rozdíl mezitím, když není použit CSRF token a když je použit.

5.6 Připojování k databázi

Konfigurační soubor k připojení k databázi je schovaný v nitru projektu a uživatelé k němu nemají přístup. Pro jednotlivé dotazy na databázi jsou využity repositáře, které byly popsány v předchozí kapitole.

ZÁVĚR

Cílem diplomové práce bylo navrhnout a zpracovat aplikaci pro tvorbu úvazků akademických pracovníků, která by sloužila ke tvorbě úvazků. Nejprve proběhl rozbor a analýza požadavků, které byly rozděleny na funkční a nefunkční požadavky, z kterých poté vznikla aplikace IS - TUAP. Dále bylo za úkol, aby aplikace umožnovala spravovat jednotlivé předměty a přidělovat jednotlivé akce předmětů lektorům.

Aplikace má využití pro vysoké školy pro tvorbu úvazků akademických pracovníků a měla by ulehčit práci při rozhodování a tvorbě úvazků na jednotlivé předměty.

Dalším rozšířením aplikace by mohlo být napojení na adresářovou službu Lightweight Directory Access Protocol (LDAP) nebo Active Directory (AD) pro ověření uživatelů, kteří se chtějí do aplikace přihlásit a s aplikací pracovat. Následně by aplikace mohla být ještě propojena s informačním systémem STAG, ze kterého by čerpala jednotlivé předměty.

Aplikace by mohla být rozdělena podle toho, kde bude použita, tedy na jednotlivé fakulty, popřípadě ústavy. Následně by se aplikace dále vyvíjela dle potřeb školy.

SEZNAM POUŽITÉ LITERATURY

[1] Pracovní smlouvy a druhy úvazků. In: *Jobs.cz* [online]. Praha: LMC, c1996–2022 [cit. 2022-03-20]. Dostupné z: https://www.jobs.cz/poradna/rady/moznosti-uplatneni/pracovni-smlouvy-a-druhy-uvazku/

[2] ZILVAR, Tomáš. Trvalý pracovní poměr, nebo práce na dohodu? Základní porovnání. In: *Měšec.cz: váš průvodce finančním světem* [online]. Praha: Internet Info, 2021 [cit. 2022-05-15]. Dostupné z: https://www.mesec.cz/clanky/trvaly-pracovni-pomer-nebo-prace-na-dohodu-zakladni-porovnani/

[3] 7.1 Pracovní poměry: Příručka Pamica Online. In: *Stormware: software development* [online]. Jihlava: STORMWARE, 2022 [cit. 2022-03-20]. Dostupné z: https://www.stormware.cz/prirucka-pamica-online/Pracovni_pomery/Pracovni_pomery/

[4] 6.1 Personalistika: Příručka Pamica Online. In: *Stormware: software development* [online]. Jihlava: STORMWARE, 2022 [cit. 2022-03-20]. Dostupné z: https://www.stormware.cz/prirucka-pamica-online/Personalistika/Personalistika/

[5] 9.2 Pracovní pozice: Příručka Pamica Online. In: Stormware: software development [online]. Jihlava: STORMWARE, 2022 [cit. 2022-05-15]. Dostupné z: https://www.stormware.cz/prirucka-pamica-online/Rizeni_lidskych_zdroju/Pracovni_pozice/

[6] KLČOVÁ, Hana a Dagmar ŠULOVÁ. Vema: Specialista na řízení lidských zdrojů, ekonomiku a logistiku. In: *Centrum pro Výzkum informačních Systémů* [online]. Zlín: CVIS, 2010 [cit. 2022-03-21]. Dostupné z: http://www.cvis.cz/hlavni.php?stranka=novinky/clanek.php&id=991

[7] Pracujte s nejpoužívanějším HR systémem v Česku. Odkudkoli. In: *Vema* [online]. Brno: Solitea, 2022 [cit. 2022-03-21]. Dostupné z: https://www.vema.cz/cs/vema-cloud

[8] Vybrané moduly systému. In: *BAKALÁŘI: Mezi školou a rodinou* [online]. Příbram: BAKALÁŘI software, 2022 [cit. 2022-03-21]. Dostupné z: https://www.bakalari.cz/Home/Modules

[9] Nastavení školního roku: Nápověda. In: *BAKALÁŘI: Mezi školou a rodinou* [online]. Příbram: BAKALÁŘI software, 2022 [cit. 2022-03-21]. Dostupné z: https://napoveda.bakalari.cz/sp_definice_novych_uvazku_v_okne_ucitelu.htm

[10] Definice nových úvazků v okně učitelů: Nápověda. In: BAKALÁŘI: Mezi školou a rodinou [online]. Příbram: BAKALÁŘI software, 2022 [cit. 2022-03-21]. Dostupné z: https://napoveda.bakalari.cz/sp_definice_novych_uvazku_v_okne_ucitelu.htm

[11] Tvorba rozvrhu v aplikaci dm Software. In: Dm Software [online]. BAKALÁŘI software, 2016 [cit. 2022-03-21]. Dostupné z: https://portal.dmsoftware.cz/%C4%8Cl%C3%A1nky/tabid/327/articleType/ArticleView/articleId/2778/Tvorba-rozvrhu-v-aplikaci-dm-Software.aspx

[12] Rozvrh a suplování. In: Dm Software [online]. BAKALÁŘI software, 2022 [cit. 2022-03-21]. Dostupné z: https://portal.dmsoftware.cz/Moduly/Rozvrhasuplov%C3%A1n%C3%AD.aspx

[13] Studijní agenda IS/STAG. In: Univerzita Tomáše Bati ve Zlíně [online]. Zlín: Univerzita Tomáše Bati ve Zlíně, 2022 [cit. 2022-03-28]. Dostupné z: https://www.utb.cz/cvt/is-stag/

[14] IS HAP - informační systém pro hodnocení pracovníků. In: Univerzita Palackého v Olomouci: Wiki [online]. Olomouc: Univerzita Palackého v Olomouci, 2021 [cit. 2022-05-19]. Dostupné z: https://wiki.upol.cz/upwiki/IS_HAP

[15] BEDNAŘÍK, Mgr. Vladislav. Základy programování. In: Střední škola elektrotechnická [online]. Ostrava: SŠE, 2014 [cit. 2022-03-28]. Dostupné z: https://www.sse-najizdarne.cz/projekty/roboti/dokumenty/u_text_zp.pdf?fbclid=IwAR2gZwcxN146jf67xUju-SaoVNaPyiftS-qiyzUrqNreTvRMHsvG47NGx7Q

[16] UML. In: Wikisofia [online]. 2013, s. 1 [cit. 2022-03-30]. ISSN 2336-5897. Dostupné z: https://wikisofia.cz/wiki/UML

[17] About diagrams.net. In: Diagrams.net [online]. Northampton: JGraph Ltd, c2005-2021 [cit. 2022-03-28]. Dostupné z: https://www.diagrams.net/about

[18] Diagramly: A Free Online Tool for Creating Diagrams and Charts. In: Old GigaOm: A repository of old GigaOm and paidContent posts [online]. 2011 [cit. 2022-03-28]. Dostupné z: https://old.gigaom.com/2011/04/28/diagramly-a-free-online-alternative-to-visio/

[19] Powering teams Powering dreams. In: JetBrains [online]. JetBrains, c2000-2022 [cit. 2022-05-19]. Dostupné z: https://www.jetbrains.com/company/people/

[20] DARKCRIZT. PhpStorm, vynikající IDE pro multiplatformní PHP. In: LinuxAddicts [online]. Španělsko: Internetové sítě AB [cit. 2022-04-01]. Dostupné z: https://www.linuxadictos.com/cs/phpstorm-un-excelente-ide-para-php-multiplataforma.html

[21] MACHÁČ, Marek. Jak efektivně pracovat v PhpStorm?. In: Interval [online]. Brno: ZONER software, 2014 [cit. 2022-03-28]. Dostupné z: https://www.interval.cz/clanky/jak-efektivne-pracovat-v-phpstorm/

[22] WELLING, Luke a Laura THOMSON. Mistrovství PHP a MySQL. Brno: Computer Press, 2017. ISBN 978-80-251-4892-1.

[23] History of PHP. In: Php [online]. The PHP Group, c2001-2022 [cit. 2022-05-19]. Dostupné z: https://www.php.net/manual/en/history.php.php#history.php

[24] LOCKHAR, Josh. Modern PHP: New Features and Good Practices. First ed. USA: O'Reilly Media, Inc., 2015. ISBN 978-1-491-90501-2.

[25] What's New in PHP 8: (Features, Improvements, and the JIT Compiler). In: Kinsta [online]. Kinsta Inc., 2022 [cit. 2022-04-15]. Dostupné z: https://kinsta.com/blog/php-8/#php-8-improvements-and-new-features

[26] POTENCIER, Fabien. Symfony 5: The Fast Track. 1.0.6. France: Symfony SAS, 2020. ISBN 978-2-918390-37-4.

[27] GOLOFIT, Piotr. What is PHP Framework Symfony? Explained for executives. In: Accesto [online]. Poland: accesto, 2021 [cit. 2022-04-19]. Dostupné z: https://accesto.com/blog/what-is-php-framework-symfony-explained-for-executives/

[28] POTENCIER, Fabian. Open-Source cross-pollination. In: Symfony [online]. 2008 [cit. 2022-04-19]. Dostupné z: https://symfony.com/blog/open-source-cross-pollination

[29] VAUGHAN, Jack. MariaDB. In: TechTarget [online]. Boston: TechTarget, 2018 [cit. 2022-03-29]. Dostupné z: https://www.techtarget.com/searchdatamanagement/definition/MariaDB

[30] ŠTRAUCH, Adam. Databáze MariaDB válcuje MySQL. In: Root [online]. Praha: Internet Info, 2008 [cit. 2022-05-19]. Dostupné z: https://www.root.cz/clanky/databaze-mariadb-valcuje-mysql/

[31] DYER, Russell J. T. Learning MySQL and MariaDB: Heading in the Right Direction with MySQL and MariaDB. USA: O'Reilly Media, Inc., 2015. ISBN 1449362907.

[32] About DBeaver. In: DBeaver [online]. 2021 [cit. 2022-05-19]. Dostupné z: https://github.com/dbeaver/dbeaver/wiki

[33] Slovníček vysokoškolských pojmů. In: Gaudeamus [online]. Brno: MP-Soft, 2022 [cit. 2022-05-01]. Dostupné z: https://gaudeamus.cz/brno/studenti/slovnicek-vs-pojmu

[34] SQL Injection. In: W3school [online]. Refsnes Data, c1999-2022 [cit. 2022-05-10]. Dostupné z: https://www.w3schools.com/sql/sql_injection.asp

[35] HRANICKÝ, Jan. Lekce 2 - Technika útoku SQL injection. In: ITnetwork: Učíme národ IT [online]. Praha: itnetwork, 2022 [cit. 2022-05-10]. Dostupné z: https://www.itnetwork.cz/php/bezpecnost/technika-utoku-sql-injection

[36] KIRSTEN, S. Cross Site Scripting (XSS). In: Owasp [online]. 2022 [cit. 2022-05-11]. Dostupné z: https://owasp.org/www-community/attacks/xss/

[37] Cross-site scripting: What is cross-site scripting (XSS)?. In: Portswigger [online]. PostSwigger, 2022 [cit. 2022-05-11]. Dostupné z: https://portswigger.net/web-security/cross-site-scripting

[38] ŠŤASTNÝ, Mgr. Bc. Jakub. Trestní postih DoS/DDoS útoků. In: Epravo [online]. Praha: EPRAVO, 2020, s. 1 [cit. 2022-05-11]. ISSN 1213-189X. Dostupné z: https://www.epravo.cz/top/clanky/trestni-postih-dosddos-utoku-110941.html

[39] Co to je DDoS útok a jak se dělá?. In: Diit [online]. Praha: CDR server, 2012, s. 1 [cit. 2022-05-11]. ISSN 1804-5405. Dostupné z: https://diit.cz/clanek/co-to-je-ddos-utok-a-jak-se-dela

[40] SHIROKOVA, Anna. Útoky hrubou silou stále fungují, z redakčních systémů dělají botnety. In: Root [online]. Praha: Internet Info, 2017 [cit. 2022-05-12]. Dostupné z: https://www.root.cz/clanky/utoky-hrubou-silou-stale-funguji-z-redakcnich-systemu-delaji-botnety/

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

AD	Active Directory
CSRF	Cross-Site Request Forgery
CSS	Cascading Style Sheets
DoS	Denial of Services
DDoS	Distributed Denial of Service
ER	Entity-realitonship
HTML	HyperText Markup Language
HR	Human Resources
IDE	Integrated Development Environment
IS	Informační systém
LDAP	Lightweight Directory Access Protocol
MVC	Mocel-View-Controller
TUAP	Tvorba úvazků akademických pracovníků
UML	Unified Modeling Language
URL	Uniform Resource Locator
PHP	Hypertext Preprocessor

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: OVLÁDÁNÍ APLIKACE

Akademické roky -> tlačítko na hlavní stránce -> Akademické roky -> Vytvořit nový akademický rok

Následně jde akademický rok smazat

Studijní plány -> tlačítko na hlavní stránce -> Studijní plány -> Vytvořit studijní plán

Následně jde studijní plán zobrazit, upravit nebo smazat

Při zobrazení studijního plánu jsou možnosti -> Přidat ročníky do studijního plánu a přidat do akademického roku studijní plán

Tlačítko Přidat ročníky -> vytvoříme nový ročník -> následně si ročník můžeme zobrazit, přidat předměty nebo smazat

Tlačítko Přidat do akademického roku -> přidáme daný studijní plán do akademického roku

Předměty -> můžeme vytvořit, editovat nebo smazat

Učebny -> můžeme vytvořit nebo smazat

Tvorba akcí -> Vybereme rok -> zobrazí se dané předměty pro přidělení studentů a následné generování akcí

Přehled akcí -> Přehled akcí -> zde vybereme rok a pokud tu jsou vygenerované akce, tak k jednotlivým akcím přidělíme akademického pracovníka

Akademické pracovníci -> Správa uživatelé -> Vytvořit uživatele -> zadat potřebné informace

Následně uživatele můžeme zobrazit a zjistit jaké má úvazky a akce

PŘÍLOHA P II: CD

Součástí fyzické verze diplomové práce je přiložené CD, jehož obsahem je DP v PDF/A formátu a zdrojový kód aplikace na tvorbu úvazků akademických pracovníků.