Vytvoření webové aplikace knihovny

Klára Vychodilová

Bakalářská práce
2019

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2018/2019

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Klára Vychodilová**
Osobní číslo:	**A16076**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Softwarové inženýrství**
Forma studia:	**prezenční**
Téma práce:	**Vytvoření webové aplikace knihovny**
Téma anglicky:	**The Creation of a Library Web Application**

Zásady pro vypracování:

1. **Analyzujte problematiku a vypracujte literární rešerši na dané téma.**
2. **Navrhněte strukturu databáze a webových stránek.**
3. **Proveďte výběr vhodného hostingu.**
4. **Realizujte aplikaci prostřednictvím vybraných vhodných nástrojů.**
5. **Zajistěte správu a zabezpečení aplikace.**

Rozsah bakalářské práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. **CONOLLY, Thomas, Carolyn E. BEGG a Richard HOLOWCZAK. Mistrovství - databáze: profesionální průvodce tvorbou efektivních databází. Brno: Computer Press, 2009. ISBN 978-80-251-2328-7.**
2. **ELMASRI, Ramez a Sham NAVATHE. Fundamentals of database systems. Seventh edition. Hoboken, NJ: Pearson, [2016]. ISBN 978-0133970777.**
3. **KOFLER, Michael a Bernd ÖGGL. PHP 5 a MySQL 5: průvodce webového programátora. Brno: Computer Press, 2007. ISBN 978-80-251-1813-9.**
4. **SUMMERFIELD, Mark. Python 3: výukový kurz. Brno: Computer Press, 2010. ISBN 978-802-5127-377.**
5. **HLAVENKA, Jiří. [ET AL.]. Vytváříme WWW stránky. 7., aktualiz. vyd. Brno: CP Books, 2005. ISBN 8025108015.**

Vedoucí bakalářské práce: **doc. Ing. Zdenka Prokopová, CSc.**
Ústav počítačových a komunikačních systémů

Datum zadání bakalářské práce: **3. prosince 2018**

Termín odevzdání bakalářské práce: **15. května 2019**

Ve Zlíně dne 7. prosince 2018

doc. Mgr. Milan Adámek, Ph.D.
děkan

prof. Mgr. Roman Jašek, Ph.D.
garant oboru

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské 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 bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou 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 – bakalářskou 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 bakalářské 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 bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské 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 bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně dne 15.5.2019

Klára Vychodilová, v.r.

podpis autora

ABSTRAKT

Tato bakalářská práce je zaměřena na návrh a realizaci webové aplikace knihovny. Aplikace slouží jak pro běžné uživatele tak i správce této webové aplikace. Umožňuje zobrazení novinek a informací o dané knihovně, jež tuto aplikaci využívá. Dále zahrnuje evidenci a editaci knih, autorů, žánrů, vydavatelství a uživatelů dané knihovny.

Klíčová slova: webová aplikace, knihovna, evidence, Django, Python

ABSTRACT

This bachelor work is focused on draft and realization of library web application. The application serves both ordinary users and administrators of this web application. It allows view news and information about the library that uses the app. It also includes registration and editing books, authors, genres, publishers and users of the library.

Keywords: web application, library, evidence, Django, Python

Ráda bych tímto poděkovala vedoucí mé práce paní doc. Ing. Zdence Prokopové, CSc. za cenné rady a čas, který mi věnovala. Taktéž bych ráda poděkovala mé rodině a příteli za jejich obrovskou podporu během mého studia.

OBSAH

ÚVOD

Při výběru zaměření mé práce mi byla inspirací obecní knihovna v místě mého bydliště. Vzhledem k probíhajícímu 21.století byl i k nám samozřejmě zaveden internet, ale místní knihovny stále lpí na papírové administraci.

Mým cílem je vytvořit jednoduchou intuitivní webovou aplikaci knihovny, jež by třeba jednou mohla být využívána v místní obecní knihovně. Jsem toho názoru, že by to osobám, které mají odpovědnost za vedení místní knihovny, usnadnilo práci. Taktéž obyvatelům obce by tento systém mohl přinést užitek. Seznam knih, kterými knihovna disponuje by byl dostupný i z domova, což by mohlo přilákat možné zájemce o vypůjčení knih, přestože danou knihovnu nikdy předtím nenavštívili. Velmi často tato situace nastává v případě, kdy mají děti ve škole zadanou povinnou četbu a jejich rodinná knihovna jí bohužel nedisponuje, případně rodiče nemají prostředky k jejímu zakoupení.

V teoretické části této práce je popsán současný stav trhu s již existujícími knihovními systémy, použité technologie v rámci celé práce a možnosti výběru hostingu.

V praktické části je popsán návrh a samotný vývoj webové aplikace od počátku až po výslednou aplikaci. Součástí práce jsou i ukázky částí kódu, jejich popis a náhled výsledné aplikace. Taktéž je zde shrnuta správa a zabezpečení aplikace.

I. TEORETICKÁ ČÁST

1 Knihovní systémy

V historii, kdy zde ještě nebyla možnost využít elektronického ukládání dat, byla data zpracovávána v listinné podobě. Po zavedení počítačů do knihoven bylo využíváno i různých kancelářských softwarů, nic z toho však plně nedostačovalo všem potřebným funkčním požadavkům. Za tímto účelem byly vytvořeny knihovní systémy, které měly sloužit ke sjednocení dat knihovny do elektronické podoby. V dnešní době nám knihovní systémy umožňují více, než jen pouhou evidenci knih. Každý knihovní systém disponuje různými funkcionalitami – moduly, ze kterých si potom zákazník, jež by rád využíval tento knihovní systém, vybere. Mezi hlavní moduly patří výpůjčka, akvizice, katalogizace a modul sloužící ke správě systému.

Modul výpůjční nám slouží ke správě výpůjček, rezervací a čtenářů. Dále může být součástí modulu i výpočet poplatků za případné pozdní vrácení knihy, či její poškození.

Modul akvizice eviduje objednávky a faktury k dokumentům či rozpočty.

Modul katalogizace eviduje informace o knihách, dokumentech v knihovně.

Modul správy systému eviduje data o jednotlivých účtech využívajících systém, jejich přístupová práva apod.

Každý knihovní systém má však vždy odlišné jednotlivé modely. I tím je způsobeno to, že systémy můžeme dělit na komerční a open source systémy. Firmy využívají poptávky po funkcích, které jiný systém nemá a tím činí jimi vytvořený systém žádaným.

1.1 Komerční systémy

Komerční knihovní systémy se vyznačují především tím, že za své služby, jako je využívání jejich systému vyžadují poplatek za licenci. Případné další služby, jako je zaškolení, rozšíření systému či technická podpora jsou taktéž zpoplatněny. Cena licencí jednotlivých systémů se dále může odvíjet od možného počtu svazků, které daná knihovna má v plánu spravovat.

1.1.1 Clavius

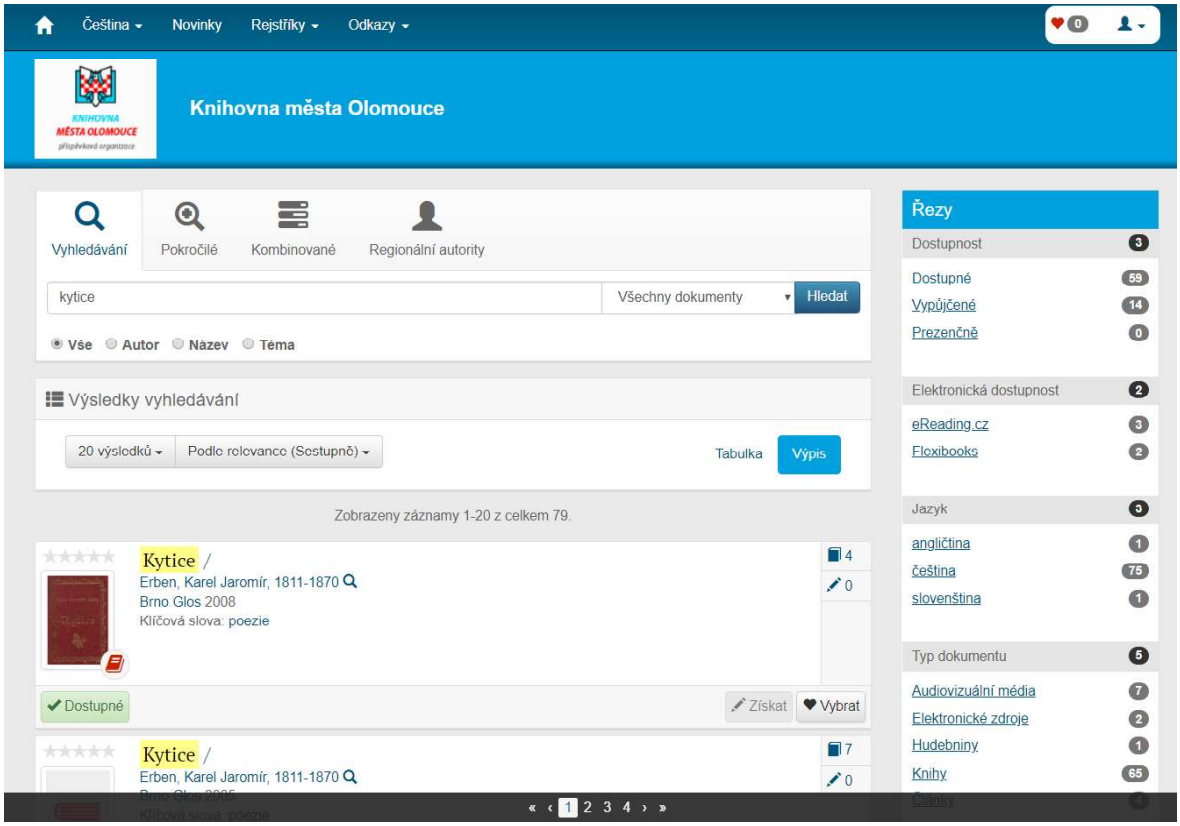
Je komerční systém vytvářený od roku 1998 firmou LANius s.r.o. [1]. V České republice se tento knihovní systém dostal do prodeje v roce 1999 a od roku 2003 je vytvořena i slovenská verze Clavia. Ještě před vytvořením systému Tritius byl systém Clavius nejrozšířenějším knihovním softwarem v České a Slovenské republice. V oblibě byl jak u větších či menších veřejných knihoven, tak i u knihoven, které byly zapojeny do regionálního knihovního systému. Taktéž byla nabízena zvýhodněná verze pro školní knihovny, které disponují méně než 20 tisíci svazky. Dále je vhodné poznamenat, že funkce výpůjček je u tohoto systému realizována pomocí čteček a čárových kódů.

Clavius nabízí neomezenou velikost vstupních oblastí, www katalog či podporu hlavních knihovnických standardů. Není však omezen jen pro knihovny, může být využíván státními institucemi, vysokými školami, muzei či archivy. Přestože je tento systém nejvíce rozšířený, mezi jeho nevýhody spadá končící podpora v roce 2020. Nástupcem toho systému je níže zmíněný systém Tritius.

1.1.2 Tritius

Tritius je obecně katalogizační systém pro evidenci různých dokumentů, který není určen jen pro knihovny. Jeho vlastnosti a funkce se však osvědčily tak, že je Tritius schopným nástupcem knihovního systému Clavius. Hlavní výhodou tohoto systému je především to, že je webový, tudíž snadno dostupný. Není třeba složitých instalací či braní velkých ohledů na hardware, operační systém. K provozu stačí pouze běžný internetový prohlížeč. Firma však dává k dispozici seznam podporovaných webový prohlížečů a doporučení.

Stejně jako Clavius tento systém podporuje nejpoužívanější moderní i historické standardy. Například umožňuje podporu všech existujících národních abeced, využití online plateb, strukturovatelnost systému a vylepšené uživatelské rozhraní. I zde jsou nabízeny moduly s různými funkcionalitami, jako je webový katalog, katalogizace, meziknihovní výpůjční služba, akvizice a mnoho dalších [2]. Tritius se také pyšní plánovanými funkcemi a vizí jako je například vybudování analýzy čtenářských dat, která poté mohou napomoci zvýšit efektivitu se zaměřením na určité věkové skupiny.



Obr. 1.1 Vyhledávání v knihovně města Olomouce - systém Tritius [3]

Základní rozložení prvků stránky bývá často zachováno, je však umožněno grafické přizpůsobení jednotlivým knihovnám a to především z pohledu barev, tak aby k sobě vše včetně loga knihovny ladilo. Z náhledu je patrné vyhledávání, kde je umožněno nastavit parametry vyhledávání dle autorů, názvů či témat. Dále je v pravé části panel sloužící k širší specifikaci vyhledávání dle určitých kritérií, jako je jazyk nebo typ dokumentu.

1.2 Open source systémy

Open source knihovní systémy jsou volně dostupné knihovní systémy s otevřeným zdrojovým kódem. Tyto systémy tedy nabízí licenci k jejich používání zcela zdarma. Je však nutné podotknout, že každý tento systém má nějaké určité hardwarové i softwarové nároky, tudíž se i tak nevyhneme počátečním a dlouhodobým investicím. K nejčastějším z nich patří například instalace softwaru, pořízení serveru, jeho provoz apod. Mezi nejčastěji využívané open source systémy patří Evergreen a Koha.

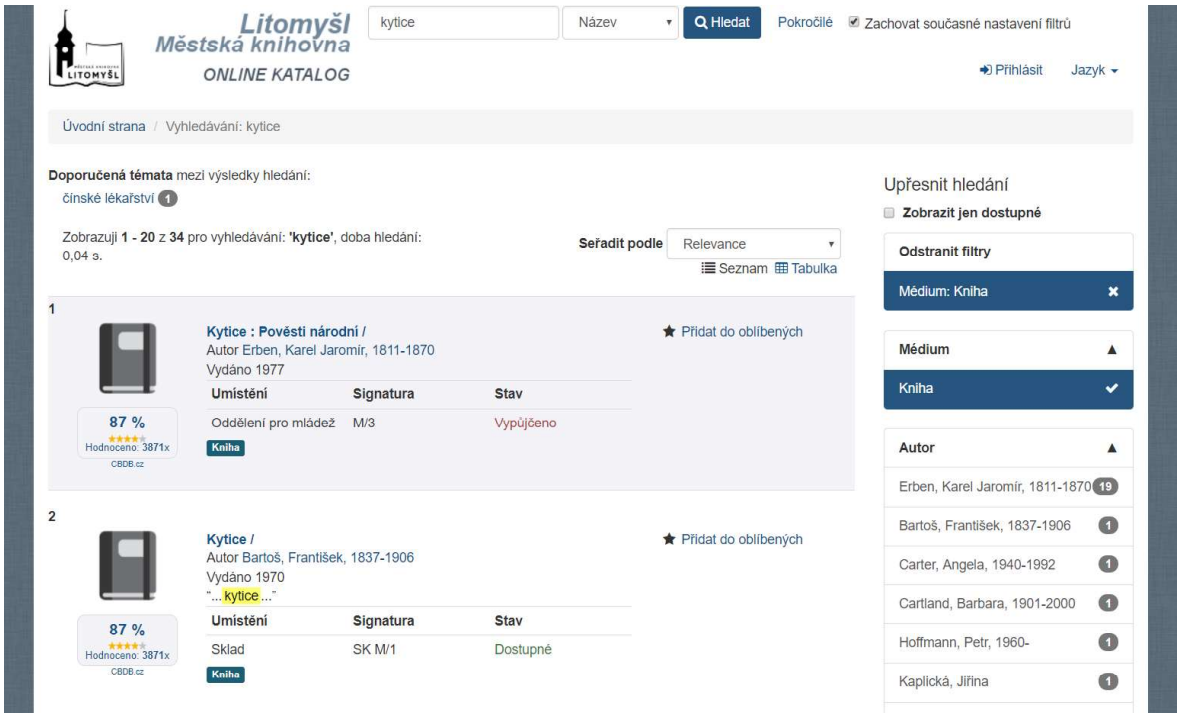
1.2.1 Evergreen

Knihovní systém Evergreen je vyvíjen od roku 2004, avšak ke spuštění došlo až v roce 2006. Tento systém je hojně využíván ve Spojených státech amerických, kde taktéž

vznikl. Ve světě je však využíván tisíci knihovnami ve 140 státech světa [4]. I tento systém má svou komunitu, která je zastoupená i v České republice. Tudíž i k tomuto systému nalezneme přeloženou dokumentaci. Systém je využíván veřejnými a univerzitními knihovnami. Evergreen nabízí modul katalogizace, statistiky, cirkulace, akviziční či modul ke správě katalogu knih. Systém umožňuje využití jedné instalace pro několik knihoven zároveň.Tuto funkcionalitu ocení především větší knihovny s více pobočkami, není však vyloučeno využití systému i menšími knihovnami.

1.2.2 Koha

Jedním z prvních open source knihovních systémů je právě Koha, který je od roku 1999 vyvíjen novozélandskou firmou Katipo Communications Ltd. K největším výhodám tohoto systémů patří komunita. Vzhledem k tomu, že je Koha systém s přístupným zdrojovým kódem, je to právě komunita, která se zasluhuje o rozvoj systému a zlepšování jeho funkcí. Tato komunita je v hojném počtu zastoupená i zde v České republice, což přineslo výhody českým uživatelům, a to například možnost pracovat se systémem v českém jazyce či uživatelské manuály v českém jazyce. Tak jako předchozí systémy, nabízí i Koha modul akvizice, katalogizace, správy čtenářů či statistiky. Taktéž zde nalezneme modul cirkulace, který má na starosti poplatky, rezervace, upozornění či tisk přihlášek nebo průkazů. Pro rok 2016 je známo 54 knihoven v České republice, jež využívají tento systém [5].



Obr. 1.2 Vyhledávání v knihovně města Litomyšl - systém Koha [6]

Vzhled vyhledávání systému Koha je podobný jako u výše popsaného systému Tritius. Při srovnání webových aplikací různých knihoven využívajících systém Koha, se vzhledově liší jen velmi lehce. Zpočátku se vzhled může zdát lehce strohý, cílem však není zaujmout čtenáře perfektním vzhledem webu, ale především funkcionalitou a snadným nalezením potřebných funkcionalit.

V hlavičce stránky nalezneme pole pro vyhledávání, kde taktéž můžeme definovat dle jakého parametru si přejeme vyhledávat, případně zvolit pokročilé vyhledávání. I u tohoto systému nalezneme v pravé části panel, sloužící k širší specifikaci vyhledávání dle žánru, období či média vydání. Taktéž můžeme zvolit v jakém jazyce si přejeme web ovládat, standardní výbavou webů bývá anglický a německý jazyk, v nabídce je však navíc kromě jazyka českého i polský jazyk.

2 Použité technologie

2.1 LaTeX

Systém LaTeX je založen na sázecím programu TeX, který vyvinul Donald Ervin Knuth. LaTeX je ve skutečnosti balík maker programu TeX a byl vyvinut v roce 1985 Lesliem Lamportem. Nyní je udržován a vyvíjen projektem LaTeX3.

LaTeX je vysoce kvalitní typografický systém, který zahrnuje funkce určené pro tvorbu technické či vědecké dokumentace. Taktéž může být tento typografický systém využit pro jakoukoliv formu publikování jako jsou technické zprávy, knihy. Systém je založen na myšlence usnadnění práce uživateli. Cílem je, aby se uživatelé mohli co nejvíce a plně věnovat obsahu dokumentu, nikoli ztrácení času nad návrhy se špatným stylováním nebo formátováním. Při práci s tímto systémem využíváme specifické a již definované značky, podobně jako v jazyce HTML.

Mezi výhody můžeme zařadit profesionálnější výstup jak po estetické tak i typografické stránce. Další z výhod je dostupná dokumentace, ať už oficiální či různé podpůrné návody a články jak se s tímto systémem naučit. Nutno zmínit, že LaTeX je dostupný jako svobodný software a nabízí nespočet funkcí pro vkládání obrázků, vytváření seznamu literatury apod.

K nevýhodám patří učení se s tímto systémem. Je to však velice sporné. Pro technicky založeného člověka bude jistě seznámení s tímto systémem jednodušší než pro člověka, který nikdy nevyužil žádný značkovací jazyk či systém podobný tomuto. Dále je nutno zmínit, že při potřebě vytvořit pouze krátkou zprávu, se může zdát zbytečné vytvářet nový soubor se spoustou příkazů.

Poslední spornou nevýhodou je instalace, v dnešní době však existují online editory. Při vytváření této práce byl využit právě online editor OverLeaf, který umožňuje kompilaci v reálném čase [7]. Uživatel může okamžitě vidět jak zdrojový text, tak i překompilovaný hotový dokument.

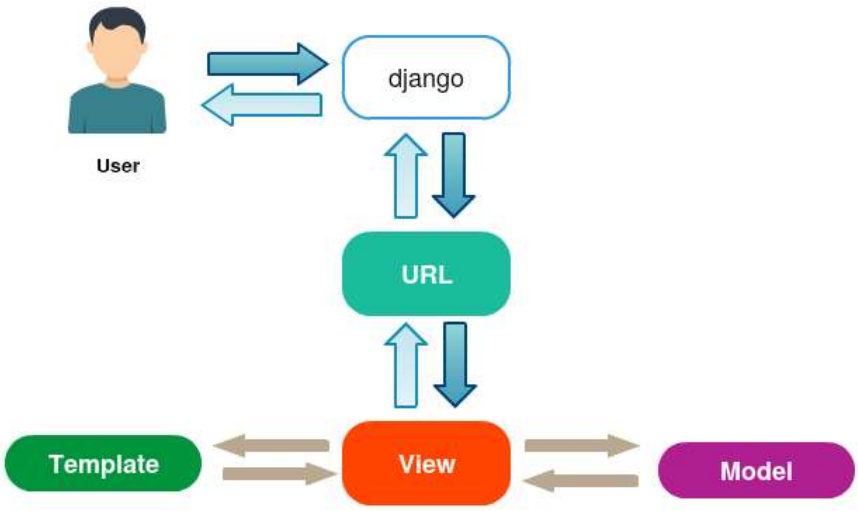
2.2 Django framework

Django je open source webový framework založený na jazyku Python. Vznik tohoto frameworku nelze přímo určit, je však známo, že k jeho vývoji došlo již mezi roky 2003 a 2004 v kansaském deníku Lawrence Journal-World. V době vzniku bylo Django bráno jako interní projekt tohoto deníku.O vznik Django frameworku se zasloužili programátoři Adrian Holovaty a Simon Willison, kteří v deníku zastávali pozici programátorů a správců webových portálů. Vzhledem k tomu, že pod sebou měli nespočet projektů, které bylo nutno spravovat, hledali nástroj, který by jim tuto práci usnadnil, urychlil a byl udržovatelný. Přestože oba ovládali jazyk PHP a redakční systémy založené na tomto jazyku, udržování webů na bázi PHP bylo velmi časově náročné, a proto bylo velkým bodem zlomu, když se tito dva programátoři shodli na jazyku Python. Dalším krokem tudíž bylo rozhodnutí přejít na vývoj webových aplikací v Pythonu. Protože oba měli jistou představu o tom, jak by měl daný webový vývoj vypadat, k samotnému vytvoření Djanga došlo až po zvážení všech dostupných nástrojů. Již od začátku však nebylo v plánu vytvořit hotový framework a Django bylo bráno jako pouhé CMS, kde každá jejich aplikace byla vyvíjena pro určitý web. Později se tyto aplikace začaly zobecňovat, spojovat a tím se začínalo vyvíjet samotné Django. Po skončení stáže Simona Willisona byl na jeho místo jako náhrada přijat Jacob Kaplan-Moss, který se dále podílel na vývoji [8]. Následně byl framework Django posouzen jako použitelný a vypuštěn do světa pro uživatele jako open source pod liberální BSD licencí. Název Django je odvozen od jména jazzového kytaristy Django Reinhardta, kterého měl v oblibě Adrian Holovaty.

Obr. 2.1 Django logo [9]

Mezi přední výhody frameworku patří vývoj založený na principu DRY (Don't Repeat Yourself). Úmyslem tedy je rozvrhnout práci tak, aby docházelo k co nejmenšímu opakování kódu. Struktura projektu je tedy z části pevně daná, což činí Django přehledným. Tato vlastnost je nám v případě potíží velmi nápomocná a usnadňuje řešení problémů při komunikaci s komunitou a dalšími vývojáři. Další z předních vlastností

je založení frameworku na návrhovém vzoru MVC (Model-View-Controller). Django si však tento přístup převzalo po svém a princip přístupu označuje jako MTV (Model-Template-View). Tento přístup bychom mohli jednoduše popsat tak, že v databázovém modelu nalezneme data, která dále promítneme do View a HTML Template.



Obr. 2.2 Django MVT

Django taktéž podporuje ORM (Object-Relation Mapping). Objektový přístup k databázi pracuje tak pouze s objekty a funkcemi Pythonu [10]. Objekty jsou transformovány na dotazy a díky implementovanému ORM je poskytnuta vyšší optimalizace dotazů. Možnost psaní vlastních SQL dotazů je zachována, a to především pro definování složitějších dotazů.

Django toho však nabízí mnohem více:

- automatické generování administračního rozhraní
- generování formulářů z databázového modelu
- šablonovací systém
- podpora kešování
- automatická validace
- nespočet malých jednoúčelových aplikací
- podpora „hezkých“ odkazů na jednotlivé stránky
- provoz více webů z jedné instalace

Pro upřesnění, mezi jednoúčelovými aplikacemi nalezneme třeba komentářový systém, autentizaci uživatelů či generování RSS. Podporou „hezkých“ odkazů je zde myšlena logická stavba odkazu, která je pro uživatele nejen lehce zapamatovatelná, ale také je jasným vodítkem k tomu, jaký obsah na stránce máme očekávat.

Příklad „hezkého“ odkazu:

http://www.webovastranka.cz/produkty/kolo/chalapecke-kole-vel-25

Klasický odkaz:

http://www.webovastranka.cz/index.php?id=25-hash=jiklsmnjsd

2.3 Python

V jádru je Python snadno osvojitelným objektově orientovaným interpretovaným jazykem a řadíme jej mezi hybridní jazyky [11]. Podle potřeby programátora a dané úlohy je možno využít jak objektově orientované, procedurální, tak funkcionální paradigma. Navrhovatelem tohoto jazyka je nizozemský programátor Guido van Rossum. V komunitě Pythonu je stále brán jako benevolentní doživotní vůdce, což znamená, že stále dohlíží na vývoj jazyka a jeho rozhodnutí v tomto směru jsou velmi významná.

Snahou jazyka Python je, aby byl intuitivní, snadný jak pro zápis tak čtení a je brán jako jazyk expresivní. Expresivní jazyky se vyznačují tím, že často stačí napsat pouhých pár řádků a kód je funkční. Tuto vlastnost je možno zpozorovat při totožné funkcionalitě kódu v odlišných programovacích jazycích jako je například C++ nebo Java. Jazyk je multiplatformní, tudíž je instalace umožněna na většině běžně dostupných platformách jako je Windows či unixových systémech, kde je ve většině distribucích již součástí základní instalace. I když se Python vzhledem k výkonu řadí k pomalejším jazykům, výkon běžných aplikací je oproti aplikacím v jiném jazyku jen nepatrně pomalejší. Tento nepatrný rozdíl je způsoben využití jazyka C, v němž jsou napsány všechny kritické knihovny, a který s jazykem Python spolupracuje. I přesto se však Python nedoporučuje na výkonově náročné programy. Jazyk Python můžeme také najít v různých aplikacích, kde je vkládán jako součást. Mezi nejznámější patří například modelovací a vykreslovací software Blender, kde Python slouží jako skriptovací jazyk a pomocí těchto skriptů následně můžeme obohatit obraz o generování různých objektů.

Obr. 2.3 Python logo [12]

2.4 HTML

HTML, přesným názvem Hypertext Markup Language je jeden z nejzákladnějších nástrojů pro tvorbu webu. Jedná se o značkovací jazyk, který využívá značky, někdy označované jako tagy k popisu obsahu webu. Každá stránka webu je zvlášť popsána značkami či elementy jazyka HTML v jednotlivých souborech s příponou html.

Základním principem jazyka je skutečnost, že jde vždy o textový formát. Obrázky, animace a jiná binární data nejsou přímo umístěna v souboru s textovým popisem webové stránky, ale je na ně pouze odkázáno [13] . Použití značek tohoto jazyka je jasně dáno pravidly jeho používání. Elementy s jejich parametry jsou uzavírány do špičatých závorek, přesněji řečeno značek pro menší než a větší než. Dále rozlišujeme elementy párové a nepárové. Párové značky se vyznačují tím, že při jejich použití je nutno aplikovat jak počáteční tak i koncovou značku. Mezi tyto značky můžeme například zařadit značky pro tvorbu nadpisů, které jsou dány písmenem a číslovkou pro odlišení úrovně nadpisu.

Ukázka použití počáteční a koncové značky:

```
<h1> Ukazka pouziti HTML tagu </h1>
<b> Tag pro vyrazny text </b>
```

Existují však i značky, jejichž použití je podmíněno užitím pouze počátečního tagu. V tomto případě je následné použití koncového tagu volitelné, jeho uzavření se odvodí z kontextu a ukončí se automaticky [14] . Při odvozování, zda-li se má daný příkaz ukončit, se vychází z jednoduchých již určených pravidel. Příkladem těchto značek je značka pro označení odstavců, seznamů nebo buněk či řádků v tabulce.

Dalším typem jsou značky nepárové, u nichž je použití koncové značky zakázané. Jazyk HTML je však velmi tolerantní jazyk, a při použití koncové značky u nepárových tagů je prohlížeče zpravidla ignorují. Tento jev se děje především z toho důvodu, že prohlížeč není schopen tento příkaz rozkódovat, tudíž jeho zobrazení vynechá [13] .

Posledním typem jsou značky, kde je volitelná jak počáteční tak koncová značka. Do této kategorie spadají takové tagy, které jsou při vytváření webové stránky povinné. V případě, zapomene-li uživatel tyto značky použít, vytvoří se automaticky samy.

Příklady těchto značek:

<html> <head> <body>

Hlavním stavebním kamenem jazyka HTML jsou tedy elementy, které mohou obsahovat text, další elementy nebo mohou zůstat prázdné. Dále využívá jazyk HTML atributů, které nám udávají informace o obsahu dokumentu a tvoří tak parametry elementů. Dle zavedeného pravidla by měly být jak elementy, tak atributy psány malými

písmeny. Od verze HTML5 již není nutno používat uvozovky pro definování hodnot atributů, jedná se však o zavedené pravidlo z dřívějších verzí, které je stále ze zvyku dodržováno [15] .

Atributy mohou přijímat nespočet typů parametrů, najdou se však i atributy s omezenějším příjmem a lze využít pouze specifické hodnoty. Atributy mohou přijímat hodnotu výčtového typu, číselnou, předdefinovanou i hodnotu ve formě URL adresy.

Ukázka elementu, kde *for* tvoří atribut elementu *label* a *jmeno* vstupní hodnotu.

```
<label for = "jmeno"> Jmeno </label>
```

U jazyka HTML se můžeme taktéž setkat s takzvanými vnořenými značkami. Příkladem jsou například položky v seznamu. V tomto případě je vnější element označující seznam nazýván jako rodič a vnitřní element označující položku seznamu jako potomek.

Ukázka vnořených značek, kde *<ol>* představuje číslovaný seznam a *li* položku tohoto seznamu.

```
<ol >
    <li>Bod seznamu</li>
    <li>Bod seznamu</li>
</ol>
```

1. Bod seznamu
2. Bod seznamu

2.5 CSS

CSS neboli kaskádové styly slouží k definování zobrazení HTML dokumentů a tvoří tak jejich nadstavbu. Definovat styly elementů pomocí CSS lze přímo v HTML souboru. Aby byl HTML dokument přehlednější, je možné CSS styly definovat ve zvláštním souboru s příponou css. Tímto je docíleno oddělené struktury dokumentu a definice formy jeho stylování. V CSS souborech tak můžeme definovat použití barev, písma, umístění a další vlastnosti elementů použitých v HTML souborech. Díky CSS souborům můžeme sdílet stejné stylování pro různé HTML soubory.

Existují 3 druhy zápisu CSS stylů. První z nich se vytváří pomocí zápisu párového tagu *<style>* do hlavičky HTML dokumentu.

Ukázka tohoto použití:

```
<style type="text/css">
    h2 {color: blue}
    a  {
        font-family: Lucida Console;
        color: green}
</style>
```

Tímto zápisem definujeme, že každý nadpis úrovně *<h2>* v tomto HTML dokumentu bude vždy modré barvy a každý odkaz bude zelené barvy písmem Lucida Console.

Další možností definování stylu je zahrnout styl jako atribut přímo určitému elementu v HTML souboru.

```
<h2 style="color:blue">Nadpis</h2>
```

Poslední možnost definice CSS stylů spočívá ve vytvoření externího souboru s css příponou. Budeme-li využívat styly definované v tomto souboru, je pak vhodné vložit do hlavičky HTML souboru tzv. link na tento soubor.

```
<link rel="stylesheet" type="text/css" href="styl.css">
```

V případě, že jsme takto učinili, jsou nám nyní k dispozici styly definované CSS souboru. Nastane-li situace, kdy chceme jeden typ elementu definovat více styly, je nezbytné použít třídy, identifikátory a selektory. Identifikátor se vyznačuje především tím, že v HTML dokumentu by se měl element s určitým identifikátorem objevit právě jen jednou. V ukázce je nejprve zaznamenán zápis stylu za použití identifikátoru v CSS dokumentu, který se vyznačuje použitím znaku # . Následuje ukázka použití identifikátoru v HTML souboru.

```
CSS soubor:
        #zelenabarva {color:green;}
HTML soubor:
    <p id="zelenabarva">Zeleny odstavec </p>
```

Potřebujeme-li jeden element nastylovat vícekrát, doporučuje se využít třídu. Definice třídy v CSS souboru se vyznačuje použitím tečky a následně názvu dané třídy. Název by neměl obsahovat znaky typu otazník, vykřičník a počátečním znakem by neměla být číslice. V případě nevhodně nazvané třídy se nám následně v HTML dokumentu nemusí naše styly uplatnit. Ukázka klasické a vícenásobné třídy.

```
CSS soubor:
        .zelenabarva {color:green;}
        .definovanytext {font-family: Lucida Console;}
HTML soubor:
<p class="zelenabarva">Zeleny odstavec </p>
<p class="zelenabarva definovanytext">Vicenasobna trida</p>
```

Taktéž lze definovat styl několika elementům najednou.

```
h1, h2, h3 { color:green; }
```

Dále můžeme definovat styl zanořených elementů. V případě ukázky definujeme všechny odkazy uvnitř odstavce a specifikujeme, že budou zelené.

```
CSS soubor:
    p a { color:green; }
HTML soubor:
   <p>Odstavec <a>zeleny odkaz</a></p>
```

Posledním bodem, který je nutno zmínit jsou selektory, kterých existuje nespočet. Například selektor * vybere všechny elementy, které bude umožněno vybrat a aplikuje na ně definované pravidlo. Tento selektor lze také přidat před název elementu v CSS souboru a je tak dáno, že na každý typ tohoto elementu bude aplikována pravidla.
V případě, je-li selektor umístěn za elementem, bude na vše, co tento element obsahuje aplikováno pravidlo. V jednom CSS souboru můžeme využít jak identifikátorů, tak tříd i selektorů.

2.6 Bootstrap

Bootstrap je jedním z nejpopulárnějších frontend frameworků a open source projektů na světě. Původně byl vytvořen designéry a vývojáři na Twitteru v polovině roku 2010 [16]. K vytvoření univerzálního CSS frameworku je vedla především nekonzistence vzhledu různých aplikací ve společnosti Twitter. I přesto, že se na vývoji podílela společnost Twitter, byl framework uvolněn a je dostupný jako open source od roku 2011 [17].

Bootstrap nabízí nespočet nástrojů sloužících k vývoji webu a webových aplikací, ale taktéž základní šablony nebo již hotové šablony. Šablony jsou k dispozici volně i za poplatek. V tomto případě se však jedná o šablony na profesionální úrovni.

Hlavní důvody, proč používat Bootstrap při vytváření webů:

- snadné použití
- dokumentace s příklady
- responzivita
- flat design
- snadnější pozicování pomocí grid systému
- dostupný zdarma
- kompatibilita s prohlížeči

Bootstrap lze využívat dvěma způsoby. V prvním případě si můžeme na oficiálních stránkách Bootstrapu stáhnout zdrojové soubory CSS a JavaScript pluginů, zahrnout je do složky našeho projektu a úvést cesty k těmto souborům v HTML dokumentu.

Další možností je implementace odkazů na CSS styly do *<head>* hlavičky našeho HTML dokumentu nad vlastní použité CSS soubory. Následuje přidání JavaScriptu před ukončovací značku *</body>*. Tento způsob je z hlediska rychlosti implementace rychlejší, avšak v případě výpadku nebo velkého přetížení CDN (Content Delivery Network) je výhodou mít soubory implementované přímo složce našeho projektu.
V případě nefunkčnosti CDN tak můžeme zamezit špatné funkcionalitě a stylování naší stránky, tudíž to neohrozí případnou návštěvnost.

3 Webhosting

Webhosting je služba, která umožňuje pronájem prostoru pro naše webové stránky. Na trhu nalezneme hostingy placené i neplacení. Ty jsou však většinou velmi omezeny prostorem a službami. V některých případech mohou na naše webové stránky vkládat nevhodné reklamy a jsou nespolehlivé [18].

Při výběru hostingu tudíž hraje velkou roli částka, kterou jsme ochotni investovat do provozu webu. Prioritou při výběru hostingu je najít střed mezi cenou, spolehlivostí a nabízeným prostorem.

Máme-li zájem o pouhé vyzkoušení nasazení projektu na web a jeho testování, nabízí se služba PythonAnywhere. Služba poskytuje zkušební účet pro nasazení jedné aplikace a úložiště 512 MB [19]. Skončí-li zkušební lhůta služba nabídne placený hosting, jehož cena se pohybuje od 5$ do 500$ za měsíc.

Další možností je Python hosting služby Roští.cz. Služba nabízí možnost nasazení nekonečně mnoha projektů, úložiště v rozsahu od 256 MB až do 50 GB či zálohování dat každé tři hodiny a databáze 1x denně. Nalezneme zde i přehlednou dokumentaci s popisem nasazení projektu v českém jazyce. Nejmenší startovací balíček tohoto hostingu je možné získat za 99 Kč měsíčně a nabízí nám prostor 2 GB pro naše data [20].

Máme-li zájem o levnější hostingové služby, je zde možnost zahraničních webhostingů, jejichž cena pohybuje v rozmezí od 30 Kč až do stovek Kč za měsíc. Jak bylo zmíněno dříve, tyto levnější hostingové služby mohou disponovat slabou technickou podporou.

Pokud bychom ani u zahraničních poskytovatelů nenašli pro nás cenově a funkčně dostupný hosting, nabízí se možnost využít VPS (Virtual Private Server) servery založené na linuxové podpoře. Mezi českými poskytovateli se jedná například o FORPSI Cloud. Ten nabízí 20 GB úložiště již od 70 Kč měsíčně. Další českou službou je WEDOS.
V případě volby WEDOS VPS si zvolíme námi potřebné serverové prostředky, dle kterých je poté vyhodnocena celková cena [20].

Zhodnotíme-li všechny nabízené možnosti, nejlépe cenově se jeví využití VPN serveru. Při výběru je však nutné brát ohledy na náročnost nasazení projektu a taktéž na možnosti zálohování dat. Pro samotné testování projektu mimo lokální server se nejlépe hodí služba PythonAnywhere, pro kterou je k nalezení i dokumentace s postupem.

II. PRAKTICKÁ ČÁST

4 Úvod do problematiky

Prvotním záměrem bylo vytvořit aplikaci a navrhnout její využívání místní obecní knihovně v oblasti mého bydliště. S přihlédnutím k tomu, že se obec skládá ze tří místních částí a hned dvě tyto části disponují vlastní malou knihovnou, vznikl nápad vytvořit jednu aplikaci, kterou budou využívat obě místní knihovny.

Obě tyto knihovny jsem navštívila, abych měla přesný přehled o tom, jak to zde funguje. Paní knihovnice mne seznámila s informacemi a úkony, jakými se při výpůjčkách a vrácení knih zabývají. Také mi sdělila, které informace je nutné uchovávat, aby bylo možné v případně nevrácení knihy čtenáře kontaktovat.

Již od založení místních knihoven je zde veden pouze jediný způsob uchovávání dat o výpůjčkách, čtenářích a knihách, jimiž daná knihovna disponuje. Knihovník má k dispozici tištěnou, abecedně seřazenou kartotéku všech čtenářů. Na každého čtenáře tedy existuje listina, ve které jsou k dispozici kontaktní údaje čtenáře, věk, identifikační číslo a prostor pro zapisování výpůjček a jejich vrácení. Výpůjčky jsou zapisovány názvem knihy, identifikačním číslem knihy a taktéž datem vypůjčení. Při vrácení se připíše datum vrácení.

Kartičku, kde jsou zapsané výpůjčky, dostane po registraci čtenářského členství a zaplacení vstupního poplatku knihovník i čtenář. Cílem je, aby i čtenář měl případně přehled o tom, které knihy měl či aktuálně má vypůjčené. Cena poplatku za členství se v místních knihovnách dělí do dvou skupin, děti do 15 let a dospělé. Členství se zakládá pro obě skupiny na rok a registrační poplatek činí 40 Kč pro děti a 60 Kč pro dospělé.

Jednou za 4 měsíce je v každé z těchto dvou knihoven doplněna literatura díly z výměnného fondu, který knihy dopraví do knihoven, předá jejich seznam a vyzvedne knihy zapůjčené z předchozího období.

5 Návrh a vývoj aplikace knihovního systému

5.1 Požadavky

5.1.1 Funkční požadavky

Systém musí umožnit:

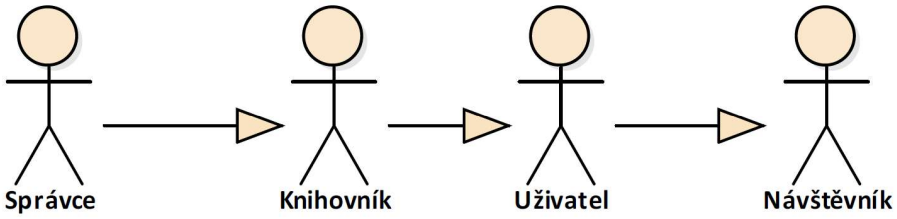
- zobrazit katalog knih
- vyhledávání v katalogu knih
- zobrazit detail knihy
- vytvoření, editaci, smazání knihy
- zobrazit výpis autorů
- vyhledávání v katalog autorů
- zobrazit detail autora
- vytvoření, editaci a smazání autora
- zobrazit výpis žánrů
- vyhledávání v katalogu žánrů
- vytvoření, editaci, smazání žánru
- zobrazit výpis vydavatelství
- vyhledávání v katalogu vydavatelství
- vytvoření, editaci, smazání vydavatelství
- zobrazit aktuality
- vytvoření, editaci, smazání aktuality
- zobrazit katalog uživatelů
- zobrazit profil uživatele
- vyhledávání v katalogu uživatelů
- vytvoření, editaci, smazání uživatelů
- prodloužení členství uživatele
- zobrazit výpůjčky
- vyhledávání v katalogu výpůjček
- vytvoření, editaci, smazání výpůjček
- vrácení knihy v rámci výpůjčky
- editaci informací o knihovně
- zobrazit výpis knihovníků
- vytvoření, editaci, smazání knihovníků
- přihlášení uživatele
- odhlášení uživatele
- vytvářet uživatelské účty s unikátním emailem
- změnu a reset hesla

5.1.2 Nefunkční požadavky

- dostupná technická IT podpora
- jednoduchost použití
- spolehlivost hostingu
- rychlost odezvy

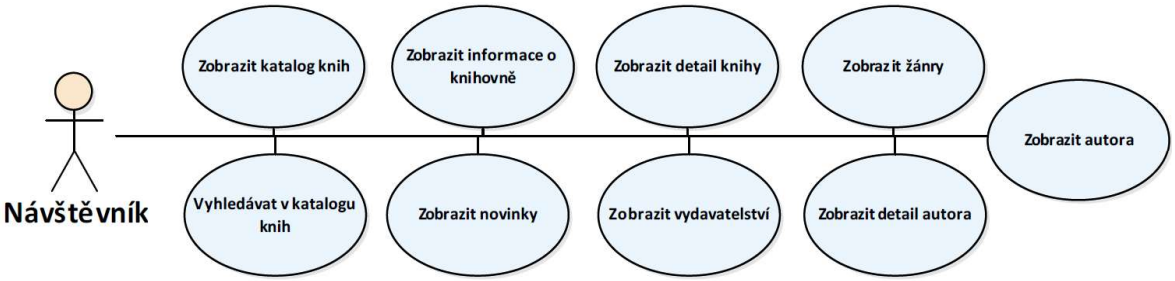
5.2 Use Case model

Use Case model je definován aktéry a případy užití. Model se při návrhu systému vytváří za účelem náhledu, co by daný systém měl umět a kdo jej bude obsluhovat.
V návrhu mé knihovní aplikace jsou celkem 4 aktéři. Mezi jednotlivými aktéry je definována dědičnost. Aktéři od sebe dědí vše od jednoduchých případů užití až po stěžejní administrační případy užití.



Obr. 5.1 Use Case model - Aktéři

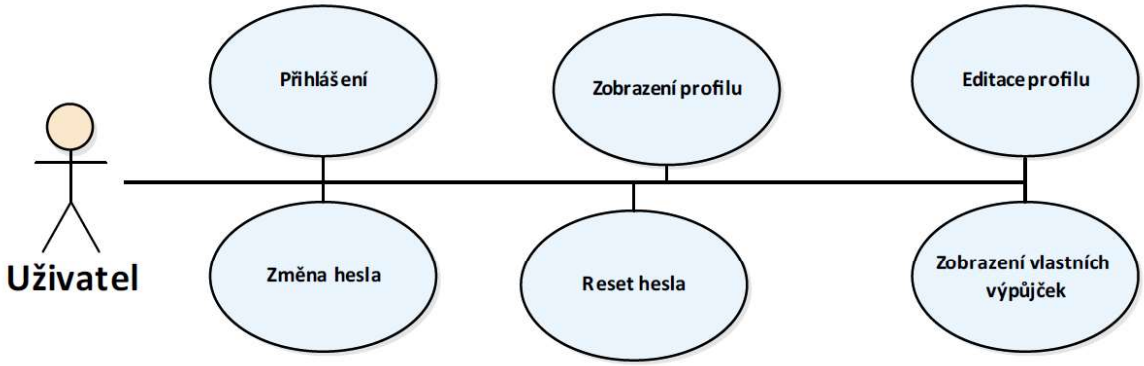
5.2.1 Návštěvník



Obr. 5.2 Use Case model - Návštěvník

Návštěvníkovi je umožněno zobrazení katalogu knih, vyhledávání v katalogu či detailní informace o knize. Systém je takto postaven, aby i nepřihlášení uživatelé mohli mít přehled o titulech v knihovně. Všechny tyto funkce může uživatel využít i v sekci týkající se autorů knih, vydavatelství a žánrů. Nezbytnou součástí je i zobrazení informací o knihovně a přehled novinek.

5.2.2 Uživatel

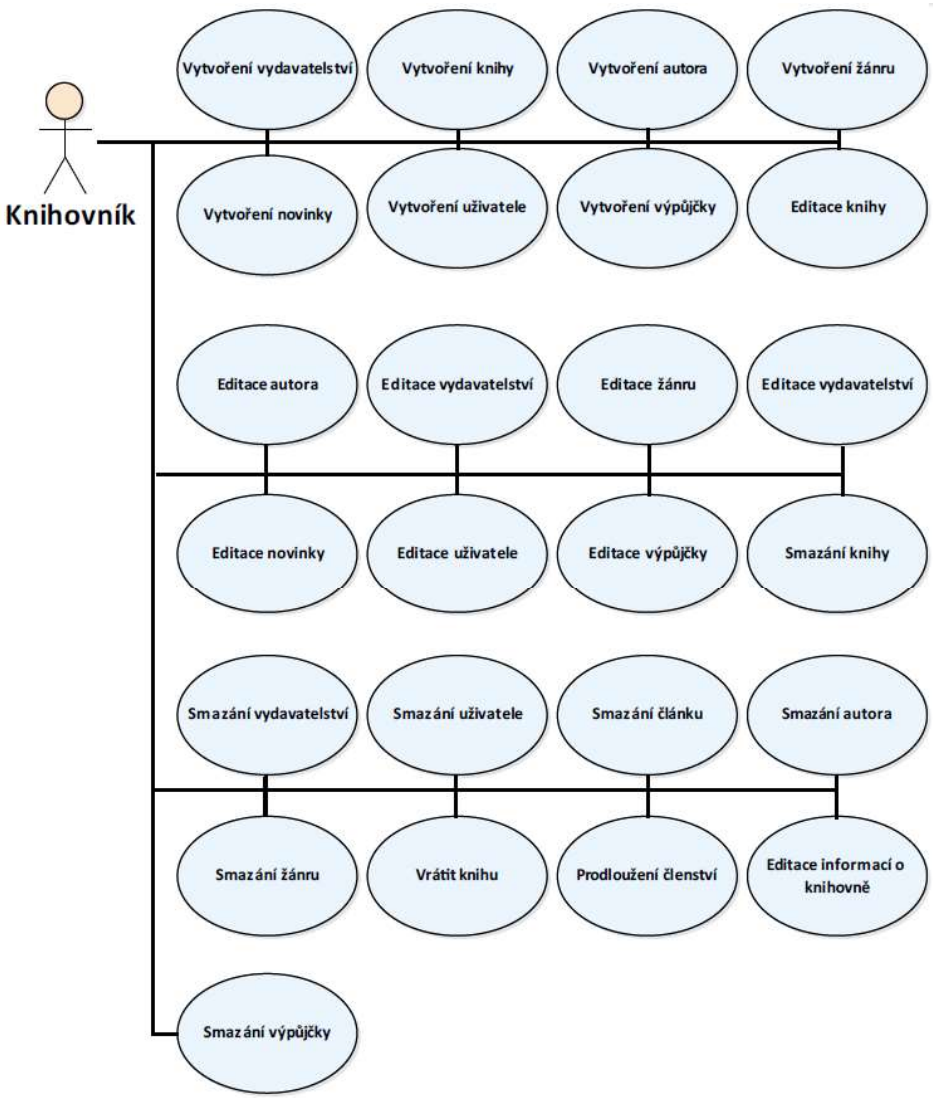


Obr. 5.3 Use Case model - Uživatel

Vzhledem k dědičnosti mezi aktéry má uživatel stejné možnosti jako návštěvník. Uživatelský účet je však rozšířen o možnost přihlášení k vlastnímu účtu. Každý uživatel následně vlastní profil. S profilem jsou spojeny další případy užití jako je editace účtu, změna hesla či možnost změnit heslo v případě, kdy uživatel heslo zapomene.

Důležitým případem užití je zobrazení vlastních výpůjček. Uživatel má přehled o vlastních aktuálních i minulých výpůjčkách knih.

5.2.3 Knihovník

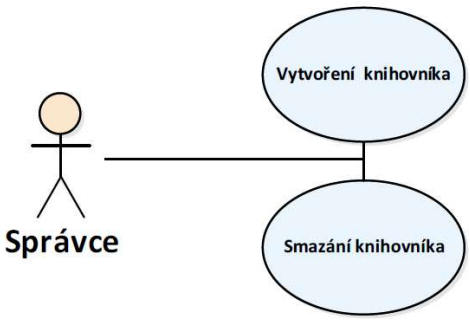


Obr. 5.4 Use Case model - Knihovník

Významným aktérem tohoto systému je právě knihovník, který díky dědičnosti taktéž nabývá práv uživatele a návštěvníka. Stěžejní funkce tohoto aktéra jsou především správa knih, uživatelů, výpůjček a novinek týkajících se knihovny. Všechny tyto akce samozřejmě zahrnují jak vytvoření, editaci, tak i mazání záznamů.

U výpůjček je knihovník oprávněn k jejich vytváření, editaci, mazání a především k vrácení knihy vypůjčené uživatelem. V případě končícího členství u uživatele má knihovník možnost členství za určitých podmínek prodloužit.

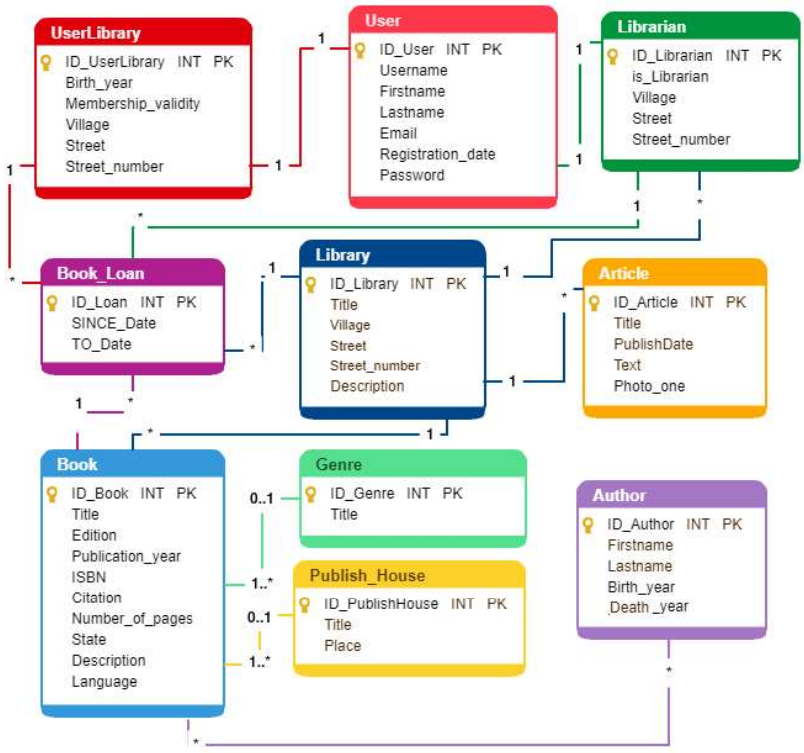
5.2.4 Správce



Obr. 5.5 Use Case model - Správce

Správce dědí od všech předchozích aktérů. Navíc je mu umožněno vytvářet a mazat knihovníky. Také se stará o správu celé aplikace. Přistupuje k ní přes djangem automaticky generovanou administraci.

5.3 ER diagram



Obr. 5.6 ER diagram

ER diagram (Entity-relationship model) znázorňuje entity a vztahy mezi nimi.

Aplikaci knihovny tvoří celkem 10 entit.

Vazební tabulka u vztahů M:N (* : *) je k nahlédnutí v návrhu databáze.

5.3.1 Entity

Každá z entit je definována primárním klíčem a dalšími atributy.

Library - knihovna je definována názvem, adresou a popisem sloužícím zejména pro popis knihovny a otevírací dobu.

User - obecně uživatel je zastřešující entita pro entitu UserLibrary a Librarian. Je definována uživatelským jménem, jménem, příjmením, emailem, heslem a taktéž datem registrace.

Librarian - knihovník je definován adresou a především proměnnou, která určuje, že jde právě o knihovníka.

UserLibrary - uživatel knihovny je rozšiřující entita obecného uživatele a je definována atributy, jež je nutné evidovat v rámci knihovny. Spadá sem rok narození, platnost členství a adresa.

BookLoan - výpůjčka je definována především datem *od, do*

Book - kniha je definována atributy, jež je nutné znát při katalogizaci knih a taktéž jsou i zajímavostí. Patří sem název, vydání, rok vydání, ISBN, citace, počet stran, stav, jazyk a popis děje knihy.

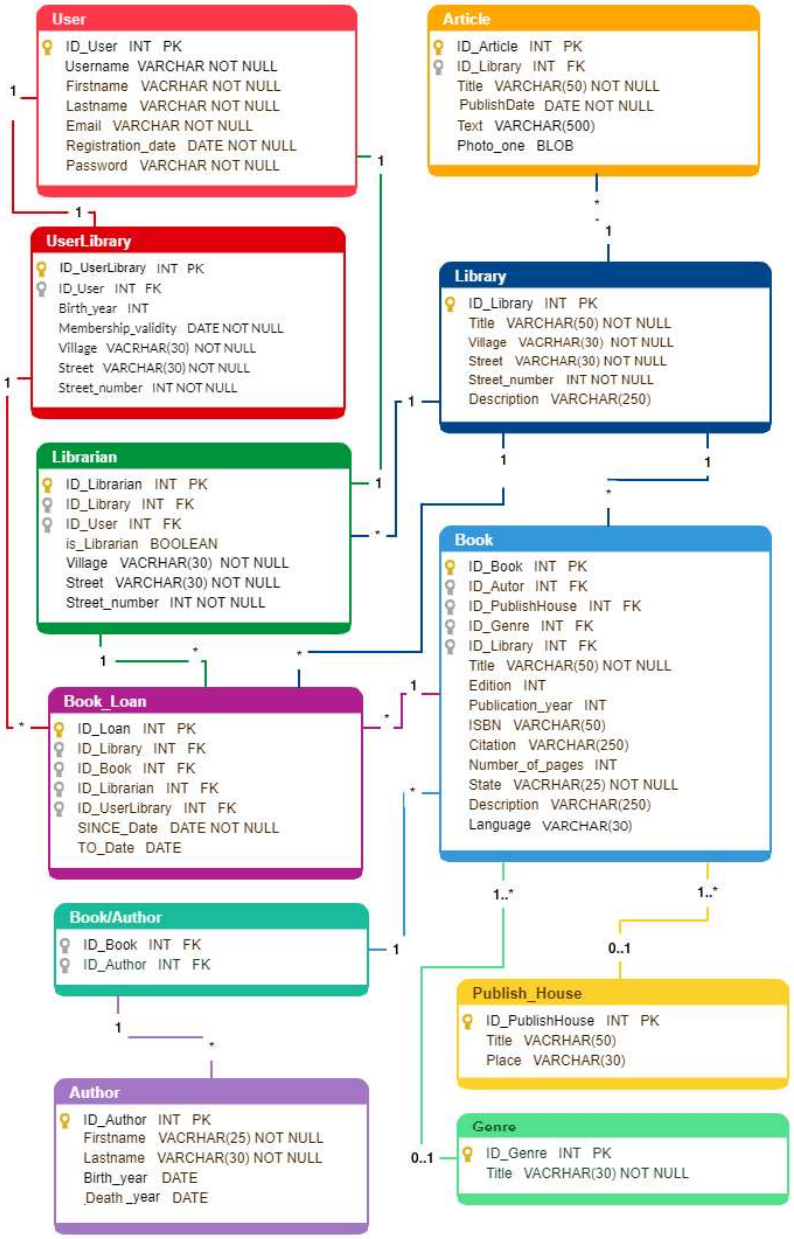
Genre - žánr je definován názvem.

PublisHouse - vydavateltsví je definováno především názvem a místem.

Author - autor je zpravidla definován jménem, příjmením, případně rokem narození či úmrtí.

Article - aktualita je definována nadpisem, datem vydání, autorem a především textem a fotografií.

5.4 Návrh databáze



Obr. 5.7 Databázový model

Databázový model slouží k přehledu a definici, z jakých tabulek se bude skládat naše databáze. Tabulky jsou definovány atributy s konkrétním datovým typem. Nalezneme zde primární i cizí klíče, které nám definují relace neboli logické vazby mezi jednotlivými tabulkami. Nachází-li se v návrhu databáze vazba many-to-many (M:N), je třeba v návrhu zahrnout tzv.mezi-tabulku. Příkladem je vztah mezi tabulkou Book a Author. Každá kniha může být napsána několika autory, a každý autor může být autorem několika knih.

Django framework využívaný při realizaci webové aplikace se neváže na používanou databázi. Z toho důvodu byly v rámci návrhu databáze použity univerzální datové typy. Níže je možno vidět tabulku vyjadřující, které typy použité databáze SQLite3 zastřešují typy použité v návrhu.

Univerzální typ	**SQLite3**
INT	INTEGER
VARCHAR(255) TEXT	TEXT
BLOB no datatype specified	BLOB
BOOLEAN DATE	NUMERIC

Obr. 5.8 Převodní tabulka

6 Realizace

6.1 Instalace a založení projektu

Tato kapitola se věnuje vývoji webové aplikace a popisuje postup vývoje od založení projektu až po dokončení aplikace. Při vývoji této aplikace byl využit framework Django. Před instalací samotného Django frameworku je nutná instalace Pythonu.
V rámci této práce byla využita verze Pythonu 3.5.2. Dále je nutné zvolit vhodnou verzi daného frameworku. S výběrem vhodné verze nám před instalací pomáhá především oficiální dokumentace. Instalace je možná přímo v terminálu pomocí jednoduchého příkazu přímo ve složce, kde plánujeme daný projekt založit. V rámci práce byla využita verze djanga 2.0.4.

```
pip install django==2.0.4
```

Ukázka kódu 1 Instalace django

Po úspěšné instalaci následuje vytvoření projektu *mysite* v naší složce.

```
py -m django startproject mysite
```

Ukázka kódu 2 Založení projektu

Tímto příkazem se vygeneroval nový adresář obsahující tyto soubory.

```
BP/
    mysite/
        __init__.py
        settings.py
        urls.py
        wsgi.py
    manage.py
```

Ukázka kódu 3 Obsah adresáře

Soubor *_init_.py* - prázdný soubor, udává že se jedná o Python balíček.

Soubor *urls.py* - slouží k definici URL adres a návaznost na části aplikace (views).

Soubor *wsgi.py* - soubor sloužící ke konfiguraci při nasazení webové aplikace.

Soubor *settings.py* - konfigurační soubor společný pro veškeré projekty v adresáři. Slouží například ke konfiguraci přesměrování po přihlášení/odhlášení uživatele nebo konfiguraci databáze, časového pásma.

Soubor *manage.py* - soubor umožňující správu projektu přes příkazovou řádku.
S jeho použitím se setkáváme především při vytváření databáze, spouštění aplikace na testovacím lokálním serveru.

Před vytvořením projektu je důležité zvolit, jakou databázi budeme využívat. Django využívá jako výchozí nastavenou databázi sqlite3. V oficiální dokumentaci je vypsán výčet dalších možných alternativ, jako je například PostgreSQL, MySQL, Oracle a jejich podpora.

Při vývoji této aplikace bylo využito výchozího nastavení databáze - sqlite3. K založení projektu je opět využito příkazové řádky, kde pomocí jednoduchého příkazu definujeme i název aplikace *kis*.

```
python manage.py startapp kis
```

Ukázka kódu 4 Vytvoření projektu

Django vytvoří adresář dle názvu, který jsme zadali s následujícími soubory.

```
__init__.py
admin.py
apps.py
models.py
tests.py
```

Ukázka kódu 5 Obsah adresáře kis

Soubor *admin.py* - administrace aplikace kis.

Soubor *apps.py* - konfigurace aplikace kis.

Soubor *models.py* - správa datového modelu, definice struktury databáze.

Soubor *tests.py* - jednotkové testy.

Aby bylo možné spustit aplikaci, po vytvoření projektu je nutné v souboru *settings.py* nově vytvořenou aplikaci zaznamenat do *INSTALLED_APPS*.

6.2 Vytvoření modelů

Při práci s daty django využívá ORM přístupu. V rámci projektu si v souboru *models.py* definujeme třídy, které pak framework automaticky převede na databázové příkazy dané databáze. Definice tříd zahrnuje i definici fieldu (atribut tabulky), příkladem je typ atributu. V rámci projektu bylo využito i definice atributu *null*, který definuje, zda daný sloupec v tabulce akceptuje hodnotu *null*. Obdobou je atribut *blank*, který se využívá při validaci v rámci formulářů.

Chceme-li při vytváření atributu využít cizího klíče, musíme definovat o jakou vazbu se jedná. Framework nabízí použití všech obvyklých databázový relací jako je One-to-One, Many-to-Many i Many-to-One. S tímto souvisí taktéž nastavení proměnné *on_delete*, která udává, co se bude dít se záznamem v případě, že dojde k vymazání objektu cizího klíče.

```
1  class Book_Loan(models.Model):
2      Book = models.ForeignKey('Book',
3              on_delete=models.CASCADE,
4              verbose_name='Kniha')
5
6      Librarian = models.ForeignKey('Librarian',
7                  on_delete=models.SET_NULL,
8                  verbose_name='Knihovnik',
9                  null=True, blank=True, )
10
11     User = models.ForeignKey('UserLibrary',
12              on_delete=models.CASCADE,
13              verbose_name='Uzivatel')
14
15     Library = models.ForeignKey('Library',
16               on_delete=models.CASCADE,
17               verbose_name='Knihovna')
18
19     SINCE_date = models.DateField('Od')
20     To_date = models.DateField('Do', null=True, blank=True)
21
22     def __str__(self):
23         return self.Book.Title
24
25     class Meta:
26         verbose_name = 'Vypujcka'
27         verbose_name_plural = 'Vypujcky'
```

Ukázka kódu 6 Model výpůjčky

Využití cizího klíče v rámci modelu můžeme vidět na řádcích *3, 7, 12, 16*. Field *Book* má definovaný jako cizí klíč třídu *Book*, tak *on_delete = CASCADE*. Nastane-li situace, kdy je kniha s daným cizím klíčem vymazána, dojde i ke smazání všech jejích výpůjček. Stejná situace nastává taktéž u fieldu *User* a *Library*. U fieldu *Librarian* můžeme vidět jiné nastavení proměnné *on_delete = SET_NULL*. To udává, že pokud dojde k smazání knihovníka s daným primárním klíčem, v tabulce dojde pouze k nastavení hodnoty proměnné na *NULL* a daná výpůjčka nebude vymazána.

U uživatele je to takto nastaveno zejména proto, že při smazání uživatele nás dále již nezajímá, jaké výpůjčky tento uživatel měl a soustředíme se především na aktivní

uživatele. V případě vytváření statistik podle výpůjček je tohle sice nevhodná volba, avšak statistiky lze dosáhnout i jinými způsoby.

Nutno podotknout, že v rámci tříd není nutné definovat sloupec s primárním klíčem, framework si jej automaticky při vytváření tabulky přidá.

V modelu *Book* můžeme vidět využití vazby Many-to-Many a tzv. *help_text*, který se automaticky zobrazí v rámci formuláře. V třídě *Meta* definuje tzv.lidsky čitelný název objektu jak singulární tak plurální tvar.

```
1  class Book(models.Model):
2      Author = models.ManyToManyField('Author',
3              help_text="Pro vyber vice nez jednoho
4              autora stiskni klavesu CTRL
5              a klikem mysi vyber.")
6      ...
7      def __str__(self):
8          return self.Title
9
10     class Meta:
11         verbose_name = 'Kniha'
12         verbose_name_plural = 'Knihy'
```

Ukázka kódu 7 Část modelu knihy

Jsou-li všechny modely nadefinovány, je nutné provést tzv.migrace. Nejprve je potřeba upozornit na to, že došlo ke změně v souboru *models.py* pomocí tohoto příkazu.

```
python manage.py makemigrations kis
```

Ukázka kódu 8 Django - makemigrations

Následuje příkaz pro vytvoření tabulek dle definice v souboru *models.py*.

```
python manage.py migrate
```

Ukázka kódu 9 Django - migrate

Migrace je nutné provádět vždy, když provedeme změnu v souboru *models.py*. Framework poté upraví tabulky dle nových migrací. Výpis všech migrací - změn v souboru nalezneme ve složce migrations, kde se po každé migraci vytvoří nový soubor. Tato funkcionalita je velkou výhodou zejména v případě, kdy si nejsme úplně jisti aktuálním návrhem databáze a máme v plánu jej třeba rozšířit.

6.3 Forms - formuláře

Formuláře jsou v projektu využívány za účelem vytváření, editace či přihlášení. Definovány jsou pomocí tříd v námi vytvořeném souboru *forms.py* ve složce projektu *kis*. Před vytvářením formulářů je nezbytným krokem import modelů, pro které formuláře definujeme.

```
1  from django.contrib.auth.forms import UserCreationForm
2  from django import forms
3  from django.contrib.auth.models import User
4
5  class ExtendedUserCreationForm(UserCreationForm):
6      email = forms.EmailField(required=True)
7      jmeno = forms.CharField(max_length=30,
8                              label='Jmeno')
9      prijmeni = forms.CharField(max_length=50,
10                                label='Prijmeni')
11
12     class Meta:
13         model = User
14         fields = ('username', 'email', 'jmeno',
15                   'prijmeni', 'password1', 'password2')
16
17     def save(self, commit=True):
18         user = super().save(commit=False)
19         user.email = self.cleaned_data['email']
20         user.first_name = self.cleaned_data['jmeno']
21         user.last_name = self.cleaned_data['prijmeni']
22
23         if commit:
24             user.save()
25
26         return user
```

Ukázka kódu 10 Registrační formulář

Ve formuláři můžeme přímo určit, o jakou formu fieldu formuláře se jedná, popisek *label* či pomocný text. V ukázce registračního formuláře pracujeme s výchozím *User* modelem a určujeme *fieldy*, které má daný formulář obsahovat. Pomocí *user_save()* uložíme data z fomuláře do databáze.

V rámci knihovní aplikace jsou fieldy výchozího *User* modelu nedostatečné, z toho důvodu vytváříme následující formulář, který nám zajistí rozšíření o všechna potřebná data uživatele knihovny, které jsme si definovali v modelu *UserLibrary*. Spadá sem platnost členství, rok narození pro vyhodnocení výše poplatku za členství a adresa uživatele pro případný kontakt. U tohoto formuláře můžeme pomocí řádku *č.4* dopředu definovat platnost členství o rok později od data registrace.

```
1  class UserProfileForm(forms.ModelForm):
2      today = datetime.date.today()
3      Membership_validity = forms.DateField(
4                  initial=today.replace(
5                          year=today.year + 1))
6
7      class Meta:
8          model = UserLibrary
9          fields = ('Birth_year', 'Membership_validity',
10         'Village', 'Street', 'Street_number')
```

Ukázka kódu 11 Registrační formulář - rozšíření

V ukázce vytvoření výpůjčky je nezbytností umožnit výpůjčku pouze těch knih, které jsou v knihovně k dispozici. Tohoto docílíme vyfiltrováním objektů dle stavu knihy, ukázka níže - řádek *č.4*.

```
1  class AddLoan(forms.ModelForm):
2      Book = forms.ModelChoiceField(
3                label='Kniha',
4              queryset=Book.objects.filter(State=IN))
5
6      SINCE_date = forms.DateField(
7                  initial=datetime.date.today(),
8                  label='Od')
9      class Meta:
10         model = Book_Loan
11         fields = ('Book', 'User', 'Library',
12                   'Librarian', 'SINCE_date',
13                   'To_date',)
```

Ukázka kódu 12 Vytvoření výpůjčky

V rámci práce byly vytvořeny formuláře pro vytvoření či editaci.

```
1  class AddEditGenreForm(forms.ModelForm):
2      class Meta:
3          model = Genre
4          fields = ('Title',)
```

Ukázka kódu 13 Formulář pro přidání a editaci žánru

6.4 Views - pohledy

Views neboli pohledy jsou taktéž důležitým stavebním kamenem každé django aplikace. Django views obstarávají vzájemnou kooperaci mezi jednotlivými částmi frameworku jako jsou formuláře, URLs atd. Views definujeme jako pythonovské třídy nebo funkce v souboru *views.py*. Každý z pohledů musí mít minimálně jeden parametr, který se odvíjí od očekávané funkcionality pohledu. Následné využití těchto tříd, funkcí nadefinujeme zavoláním funkce v souboru *urls.py*. Při vývoji je umožněno vytvářet vlastní pohledy nebo využít již předpřipravené obecné Generic views, které framework nabízí. V rámci práce bylo využito *ListView*, které slouží k výpisu listu objektů a *DetailView*, k zobrazení detailu jednotlivých objektů.

V ukázce č.14 můžeme vidět využití předpřipraveného *ListView*. Třídě je přidělen model, se kterým spolupracuje, název šablony a taktéž *context_object_name*, pomocí kterého je umožněno k jednotlivým objektům přistupovat a vypisovat další vlastnosti objektu. V tomto případě *Title*, *Author* a zbylé fieldy definované v modelu.

Proměnná *paginate_by* slouží k regulaci výpisu počtu novinek na stránku. Pomocí proměnné queryset definujeme objekty výpisu, případně řazení pomocí *order_by*.

```
1  class ArticleIndex(generic.ListView):
2      model = Article
3      template_name = 'kis/article_index.html'
4      context_object_name = 'articles'
5
6      paginate_by = 3
7      queryset = Article.objects.all().order_by('-id')
8
9  class BookView(generic.DetailView):
10     model = Book
11     template_name = 'kis/book_detail.html'
```

Ukázka kódu 14 ListView - Výpis novinek

Využití předpřipraveného *DetailView*.

```
1  class BookView(generic.DetailView):
2      model = Book
3      template_name = 'kis/book_detail.html'
```

Ukázka kódu 15 DetailView - Detail knihy

Při vytváření nového objektu se až na vytváření uživatele funkce liší zcela minimálně. V první řadě je zásadní použít správný formulář, řádek *č.3*, ověřit validitu, uložit data a následně přesměrovat uživatele.

```
1  def addBook(request):
2      if request.method == 'POST':
3          form = AddBook(request.POST)
4          if form.is_valid():
5              form.save()
6              return redirect('book_index')
7      else:
8          form = AddBook()
9      context = {'form': form}
```

Ukázka kódu 16 Přidání knihy

Při editaci již vytvořeného objektu je nutné uživateli zobrazit data, která byla definována při vytváření objektu. Tohoto docílíme přiřazením editovaného objektu proměnné *instance*. Díky parametru *pk* v definici funkce přesně víme, o který záznam se jedná a můžeme s ním pracovat viz.řádek *č.5* . Nezbytným krokem při editaci je následná validace a uložení.

```
1  def edit_book(request, pk):
2      book = Book.objects.get(pk=pk)
3      if request.method == 'POST':
4          form = EditBookForm(request.POST,
5                  instance=Book.objects.get(pk=pk))
6          if form.is_valid():
7              form.save()
8              return redirect('book_index')
9
10     form = EditBookForm(instance=book)
11     context = {'form': form}
12     return render(request, 'edit_book.html', context)
```

Ukázka kódu 17 Editace knihy

Při mazání objektu taktéž využíváme parametru v definici funkce. Pomocí primárního klíče je nám umožněno smazat určitý záznam viz.řádek *č.4,5*. Dále je vhodné uživatele informovat o provedení, k tomu slouží *messages.error*.

```
1  def delete_book(request, pk):
2      if request.method == 'POST':
3          book = Book.objects.get(pk=pk)
4          book.delete()
5          messages.error(request, 'Kniha vymazana')
6      return redirect('book_index')
```

Ukázka kódu 18 Smazání knihy

V ukázce č.19 můžeme vidět zpracování dvou formulářů v rámci jednoho view. Dochází ke zpracování dat výchozího *User* a *UserLibrary* modelu pomocí formulářů definovaných v souboru *forms.py*. Při zpracování údajů je důležitou součástí ověření, zda jsou data validní viz. řádek *č.6* a následně data zpracovat, uložit a přesměrovat uživatele pomocí *return redirect(url název)*.

```
1  def register(request):
2      if request.method == 'POST':
3          form = ExtendedUserCreationForm(request.POST)
4          profile_form = UserProfileForm(request.POST)
5          if form.is_valid() and profile_form.is_valid():
6              user = form.save()
7              profile = profile_form.save(commit=False)
8              profile.user = user
9              profile.save()
10             username = form.cleaned_data.get('username')
11             password = form.cleaned_data.get('password1')
12             user = authenticate(
13                     username=username,
14                     password=password)
15
16             return redirect('users_index')
17     else:
18         form = ExtendedUserCreationForm()
19         profile_form = UserProfileForm
20     context = {'form': form, 'profile_form': profile_form}
21     return render(request, 'register.html', context)
```

Ukázka kódu 19 View registrace uživatele

6.5 Templates - šablony

Šablonovací systém frameworku lze využít k definici jednotlivých šablon, které následně mohou být dále rozšiřovány jinými šablonami. Zpravidla se této funkce využívá při vytváření *base.html* šablony, kde bývá definován základní vzhled webu a pomocí speciálních značek je definován prostor, který je dále rozšiřován.

```
<div class="content" style="min-height:400px;">
    {% block content %}
     ---- > prostor pro obsah ostatnich sablon < ----
    {% endblock %}
</div>
```

Ukázka kódu 20 Base šablona

Chceme-li využít *base.html* šablonu, je nutné v nové šabloně definovat její použití pomocí *extends*.

```
{% extends "base.html" %}
{% load crispy_forms_tags %}
{% block content %}
...
{% endblock %}
```

Ukázka kódu 21 Použití base šablony

Užitečnou vlastností šablonovacího systému je především definice stylů v základní šabloně. V závěru se tedy vyvarujeme redundantní definici stylů, menu, patičky webu v každém z *html* souborů naší aplikace.

6.6 URLs

Soubor *urls.py* slouží k propojení jednotlivých views - pohledů přímo s určitou URL adresou. Každé URL adrese je přiřazeno view a taktéž hodnota *name*. Při vytvoření projektu pomocí příkazové řádky automaticky vzniká soubor *urls.py*, který je společný pro všechny vytvořené aplikace v tomto adresáři. V souboru je však možné definovat přesměrování na námi vytvořenou aplikaci a taktéž na administraci.

```
1 from django.contrib import admin
2 from django.urls import path, include
3 urlpatterns = [
4       path('admin/', admin.site.urls, name='admin'),
5       path('',include('kis.urls')),]
```

Ukázka kódu 22 Obsah urls.py

Vytvořením nového souboru *urls.py* v adresáři naší aplikace *kis* docílíme lepšího přehledu o url adresách přímo naší konkrétní aplikace. V ukázce č.23 můžeme na řádcích *č.13, 17* vidět použití primárního klíče přímo v url adrese.

```
1  urlpatterns = [
2   path('', views.index, name='index'),
3   url(r'^logout/$', views.logout_view, name='logout'),
4
5   url('vypujcky/',
6         views.LoanIndex.as_view(),
7         name='loan_index'),
8
9   path('addvypujcka/',
10          views.addLoan,
11          name='add_loan'),
12
13  path('back_loan/<int:pk>/$',
14           views.back_loan,
15           name='back_loan'),
16
17  path(r'loanedit/<int:pk>/$',
18          views.edit_loan,
19          name='edit_loan'),
20  ...
21  path("novyuzivatel/",
22           views.register,
23           name="register"),
24
25
26 ]+ static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Ukázka kódu 23 Obsah kis/urls.py

Na jednotlivé URL adresy se následně můžeme přímo odkazovat v šablonách přes proměnnou *name*. Nastavení této proměnné má velkou výhodu zejména v okamžiku změny URL adresy, kdy proměnná *name* zůstává neměnná.

```
1  ...
2  <li class="nav-item">
3          <a class="nav-link" href="{% url 'loan_index' %}">
4          Vypujcky</a>
5  </li>
6  <li class="nav-item">
7          <a class="nav-link" href="{% url 'register' %}">
8          Pridat uzivatele</a>
9  </li>
10 <li class="nav-item">
11         <a class="nav-link" href="{% url 'administration' %}">
12         Sprava knihovny</a>
13 </li>
14 ...
```

Ukázka kódu 24 Odkazování

6.7 Spuštění aplikace

Aplikaci je možno spustit na lokálním serveru pomocí příkazové řádky.

```
python manage.py runserver
```

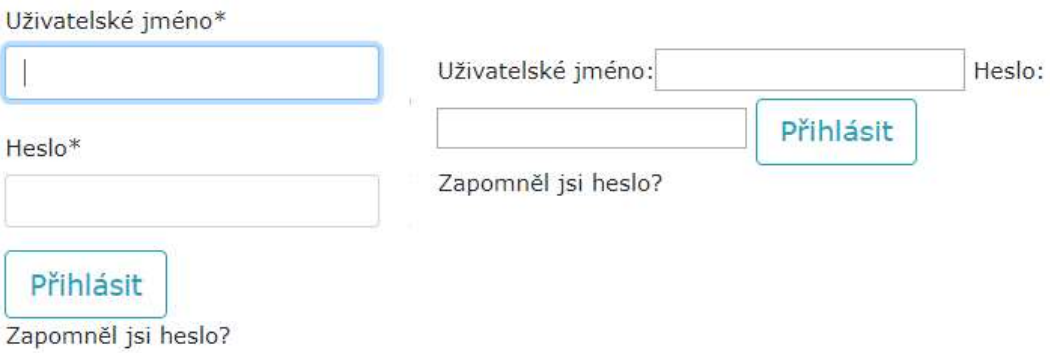
Ukázka kódu 25 Spuštění aplikace

7 Technologie

7.1 Bootstrap

Na vzhledu výsledné aplikace se velmi podílí využití sady nástrojů *Bootstrap 4.0*. Především využití grid systému výrazně zjednodušuje pozicování prvků. Taktéž je využito možnosti definování menu a prvků typu *button, alert* atd. Jednou z hlavních výhod je vytvoření responzivního webu.

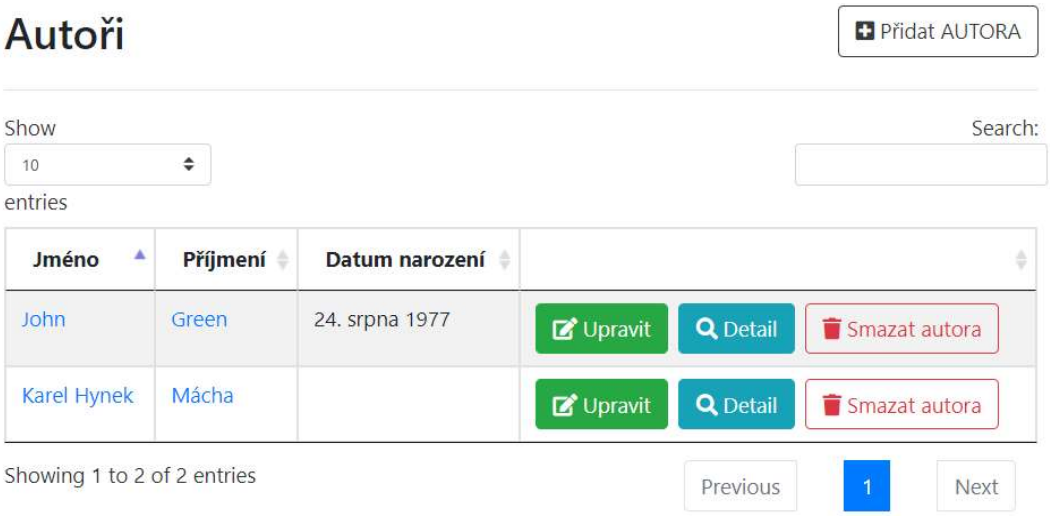
Dalším důležitým prvkem je využití *django-crispy-forms*, které umožňuje upravit vykreslení formulářů dle našich vlastních stylů.



Obr. 7.1 Formulář s crispy-forms - formulář bez crispy-forms

7.2 DataTables

Použitím *DataTables* pluginu bylo zajištěno vyhledávání a taktéž řazení položek v tabulkách. Jednou z výhod tohoto pluginu je možnost stránkování záznamů.



Obr. 7.2 Výpis autorů - použití DataTables

8 Správa a zabezpečení aplikace

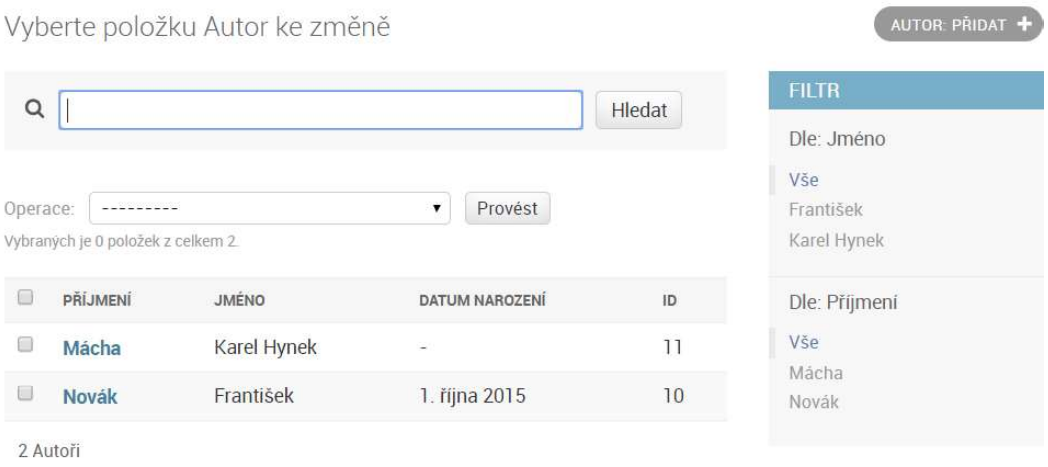
8.1 Správa aplikace - Administrace

Automaticky generovaná administrace je jedna z předních výhod Djanga. Pro přístup k administraci je nezbytné vytvořit tzv. superuživatele neboli admina. Při vytváření budeme požádáni o zadání povinných údajů jako je uživatelské jméno, email a heslo.

```
python manage.py createsuperuser
```

Ukázka kódu 26 Vytvoření admina

Abychom mohli pracovat s objekty, je nutné je zahrnout v souboru *admin.py*. Před registrací modely naimportujeme. Taktéž můžeme definovat, které fieldy chceme v administraci u jednotlivých modelů zobrazit, podle kterých si přejeme filtrovat nebo vyhledávat.

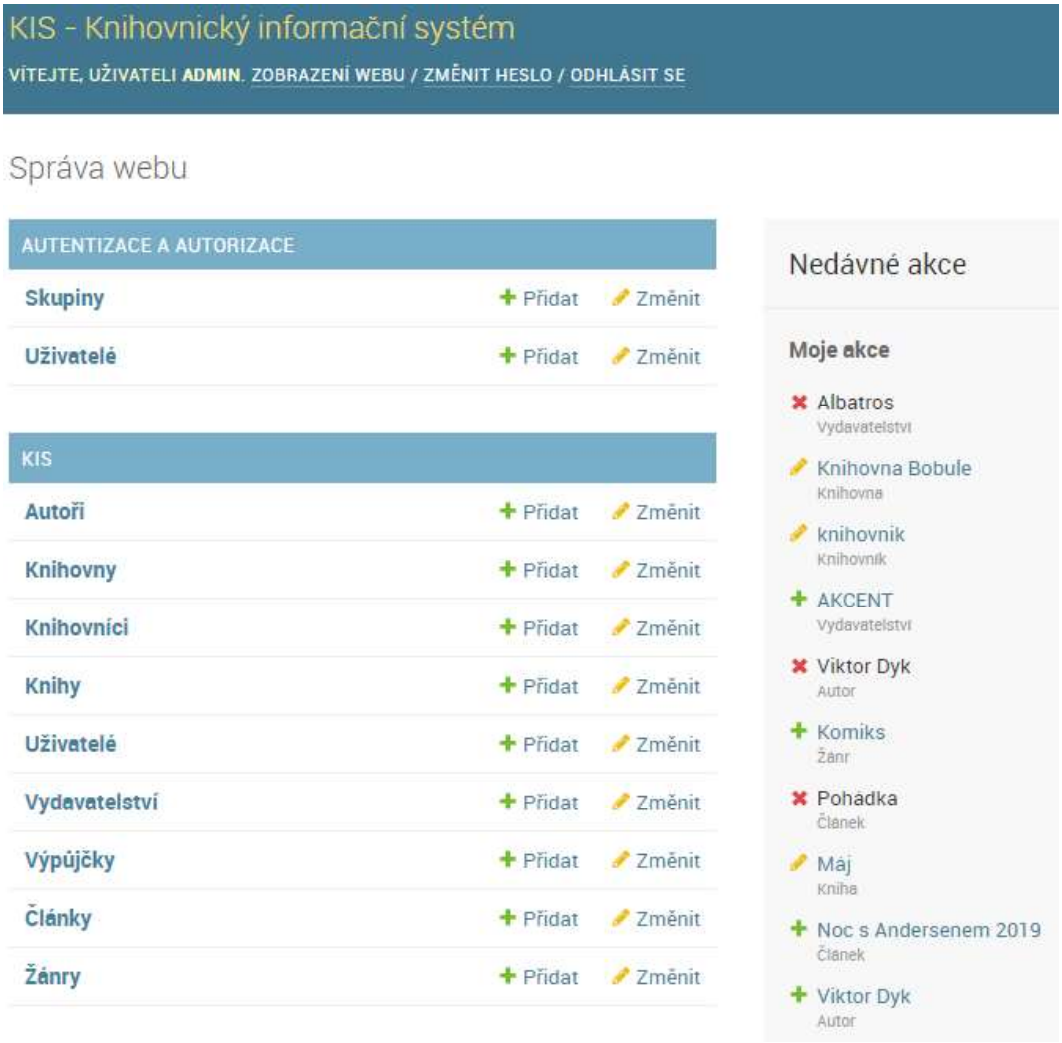


Obr. 8.1 Náhled administrace - Autor

```
1  from .models import Book, Book_Loan, Author,UserLibrary,Publish_House, ..
2  class AuthorAdmin(admin.ModelAdmin):
3      list_display = ('Lastname', 'Firstname',
4                      'Birth_Date', 'id')
5      list_filter = ('Firstname', 'Lastname')
6      search_fields = ('Firstname', 'Lastname')
7      ...
8  admin.site.register(Book, BookAdmin)
9  admin.site.register(Book_Loan, Book_LoanAdmin)
10 admin.site.register(Article)
```

Ukázka kódu 27 Část admin.py souboru

V náhledu administrace můžeme přímo vidět všechny spravované modely. V pravé části jsou vidět nedávné akce - přidání, odebrání a editace včetně názvu.



Obr. 8.2 Náhled administrace

8.2 Zabezpečení

Zabezpečení aplikace je v případě uchovávání uživatelských dat velmi důležitou částí.

8.2.1 Autentizace

Využitím django uživatelského ověřovacího systému je ošetřeno například zabezpečení přihlašování, registrace nebo ověřování. Uživatelský ověřovací systém zajišťuje kontrolu správně zadaných přihlašovacích údajů, správnost zadaného hesla. Při registrace nového uživatele ověřuje rovnost hesla a potvrzovacího hesla a zda je zadaný email či uživatelské jméno unikátní. Díky vlastnostem frameworku je zabezpečeno i ukládání hesla, které by nikdy nemělo být surově ukládáno do databáze, aniž by byl použit hash. Django taktéž disponuje ochranou proti CSRF (Cross-site request forgery). Této vlastnosti je využito zejména ve všech formulářích aplikace.

8.2.2 Reset hesla

Nastane-li situace, kdy uživatel zapomene heslo je uživateli umožněno heslo změnit. Po zadání emailu je pomocí uživatelského ověřovacího systému ověřena existence uživatele v systému. Následuje zaslání emailu s instrukcemi, jak dále postupovat. Uživatel v tomto emailu nalezne odkaz, který umožní nastavení nového hesla. Pokud by uživatel chtěl tento odkaz využít vícekrát, je jeho použití omezeno a uživatel je vyzván k opětovnému zadání emailu.

```
1  From: webmaster@localhost
2  To: k_vychodilova@utb.cz
3  Date: Mon, 01 May 2019 20:22:42 -0000
4  Message-ID: <155717416281.14720.
5  14646556046719922137@DESKTOP-UVLGE4O>
6
7  Zdravime ctenare!
8  V nedavne dobe bylo zazadano o reset hesla uzivatelskeho uctu
9  timto emailem k_vychodilova@utb.cz.
10 Pro reset hesla kliknete na odkaz nize:
11
12 http://localhost:8000/accounts/reset/OQ/563-4fb28958c35541ed261a/
13
14 Pekny den preje tym knihovniku!
```

Ukázka kódu 28 Reset hesla - email

8.2.3 Omezení přístupu

Aplikace nabízí nejen prohlížení záznamů, ale také jejich vytváření, editaci nebo mazání. Podstatou této aplikace je, že ne každý uživatel má stejná práva. Zejména uživatelé knihovny mají striktně omezená práva a to pouze na prohlížení. Možnost editace je uživatelům knihovny zpřístupněna pouze při změně vlastního uživatelského profilu. Omezení přístupu můžeme definovat přímo v šablonách nebo views.

```
1  {% if user.is_authenticated %}
2  {% if user.librarian.is_knihovnik %}
3  ...
4  {% endif %}
5  {% endif %}
```

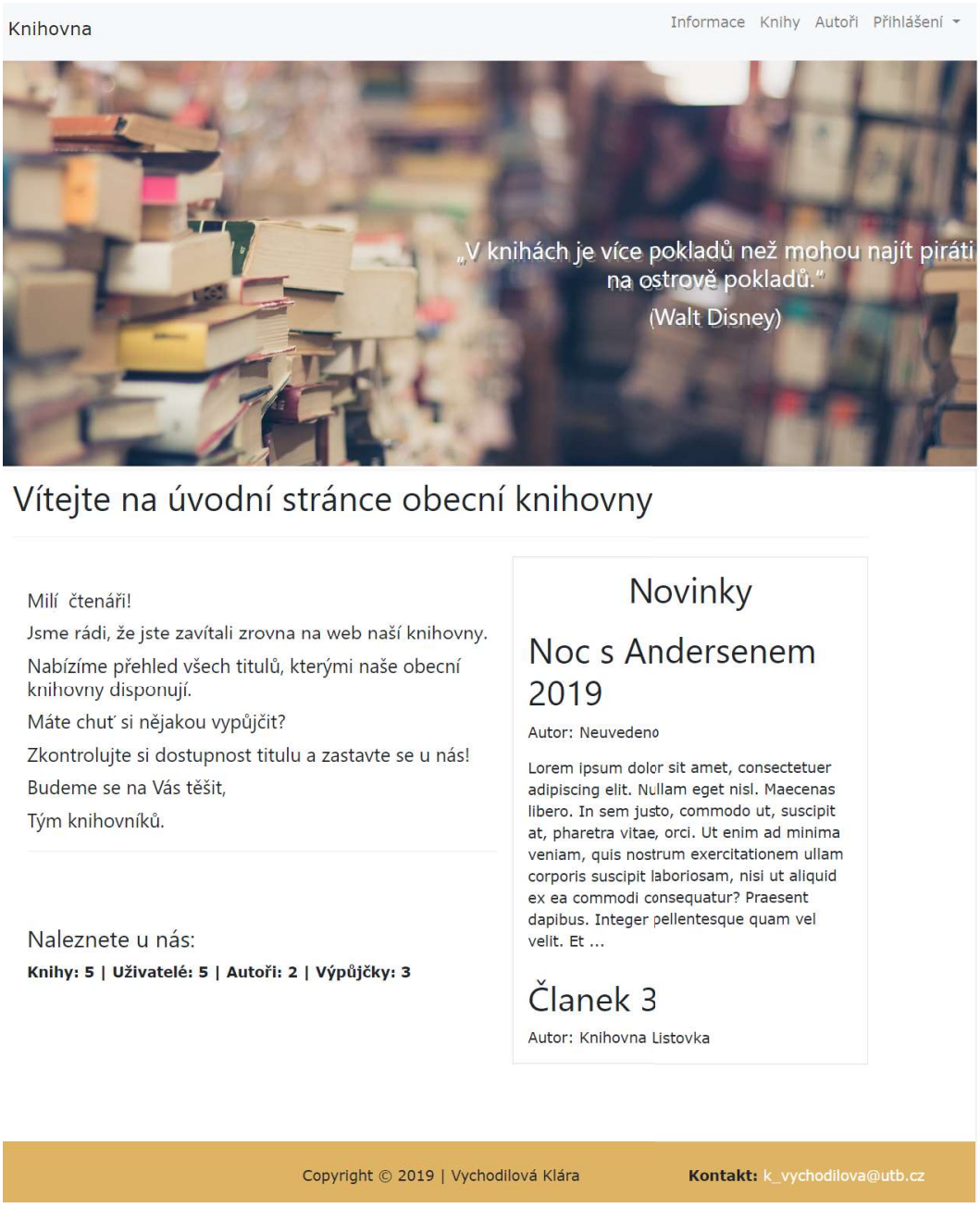
Ukázka kódu 29 Omezení přístupu 1

V ukázce č. 28 je možno vidět ověření, zda je uživatel přihlášen a zda se jedná o knihovníka. Chceme-li definovat zobrazení v opačném případě,pych použijeme negaci `not`.

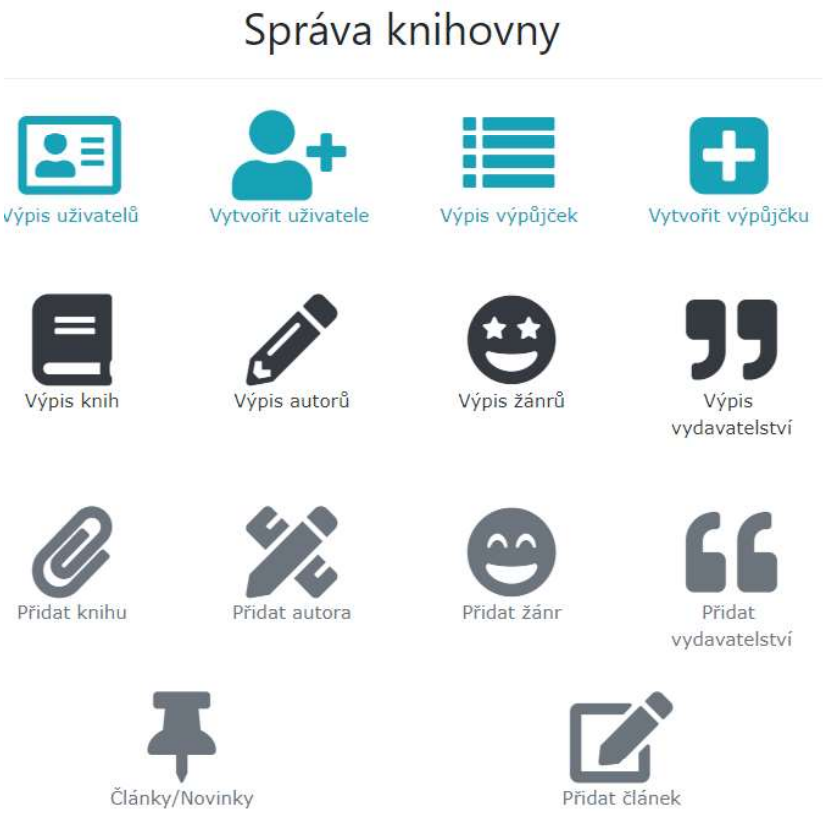
```
1  {% if not user.librarian.is_knihovnik %}
2  {% if not user.is_authenticated %}
3      ...
4  {% endif %}
5  {% endif %}
```

Ukázka kódu 30 Omezení přístupu 2

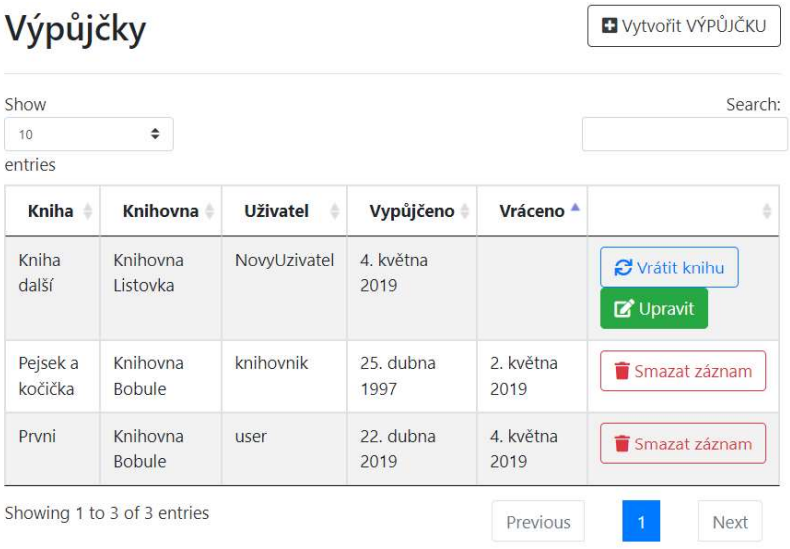
9 Výsledná aplikace



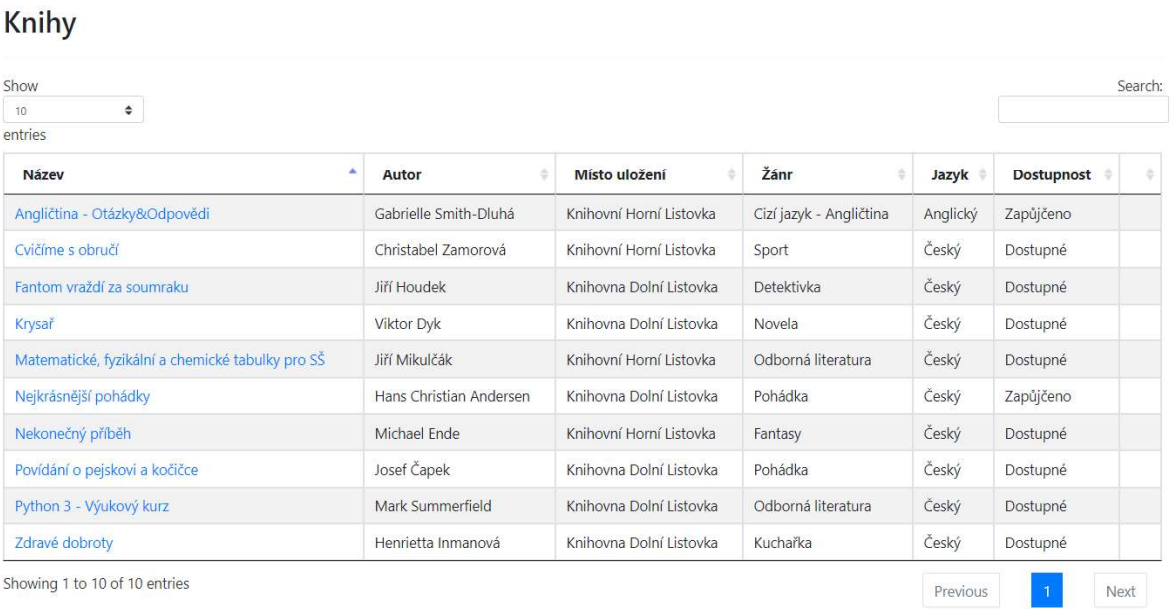
Obr. 9.1 Úvodní strana webu



Obr. 9.2 Správa knihovny - knihovník



Obr. 9.3 Výpůjčky



Knihy

Show 10 entries

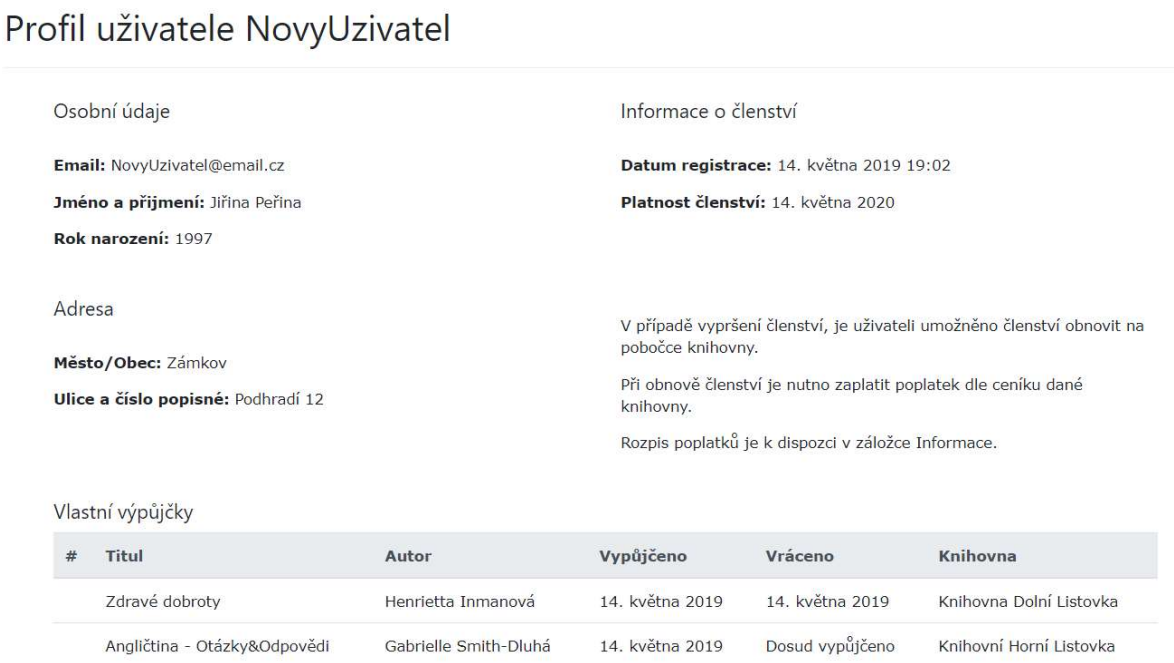
Search:

Název	Autor	Místo uložení	Žánr	Jazyk	Dostupnost
Angličtina - Otázky&Odpovědi	Gabrielle Smith-Dluhá	Knihovní Horní Listovka	Cizí jazyk - Angličtina	Anglický	Zapůjčeno
Cvičíme s obručí	Christabel Zamorová	Knihovní Horní Listovka	Sport	Český	Dostupné
Fantom vraždí za soumraku	Jiří Houdek	Knihovna Dolní Listovka	Detektivka	Český	Dostupné
Krysař	Viktor Dyk	Knihovna Dolní Listovka	Novela	Český	Dostupné
Matematické, fyzikální a chemické tabulky pro SŠ	Jiří Mikulčák	Knihovní Horní Listovka	Odborná literatura	Český	Dostupné
Nejkrásnější pohádky	Hans Christian Andersen	Knihovna Dolní Listovka	Pohádka	Český	Zapůjčeno
Nekonečný příběh	Michael Ende	Knihovní Horní Listovka	Fantasy	Český	Dostupné
Povídání o pejskovi a kočičce	Josef Čapek	Knihovna Dolní Listovka	Pohádka	Český	Dostupné
Python 3 - Výukový kurz	Mark Summerfield	Knihovna Dolní Listovka	Odborná literatura	Český	Dostupné
Zdravé dobroty	Henrietta Inmanová	Knihovna Dolní Listovka	Kuchařka	Český	Dostupné

Showing 1 to 10 of 10 entries

Previous 1 Next

Obr. 9.4 Výpis knih - z pohledu návštěvníka, uživatele



Profil uživatele NovyUzivatel

Osobní údaje

Email: NovyUzivatel@email.cz

Jméno a příjmení: Jiřina Peřina

Rok narození: 1997

Adresa

Město/Obec: Zámkov

Ulice a číslo popisné: Podhradí 12

Informace o členství

Datum registrace: 14. května 2019 19:02

Platnost členství: 14. května 2020

V případě vypršení členství, je uživateli umožněno členství obnovit na pobočce knihovny.

Při obnově členství je nutno zaplatit poplatek dle ceníku dané knihovny.

Rozpis poplatků je k dispozici v záložce Informace.

Vlastní výpůjčky

#	Titul	Autor	Vypůjčeno	Vráceno	Knihovna
	Zdravé dobroty	Henrietta Inmanová	14. května 2019	14. května 2019	Knihovna Dolní Listovka
	Angličtina - Otázky&Odpovědi	Gabrielle Smith-Dluhá	14. května 2019	Dosud vypůjčeno	Knihovní Horní Listovka

Obr. 9.5 Profil uživatele

10 Možná vylepšení

Aktuální verze aplikace splňuje minimální podmínky pro správu knihovny. Pořádá-li knihovna akce, jedním z vylepšení by mohlo být založení galerie. Dalším bodem vylepšení by mohla být vyšší nadstavba statistiky, jako je například výpis deseti nejžádanějších knih, nebo počty výpůjček v jednotlivých měsících. Další možností je rozšířit aplikaci o možnost rezervace knih. Vzhledem k tomu, že byl systém navrhnutý pro velmi malou knihovnu, nebylo zde nutné zahrnout rezervační systém knih.

ZÁVĚR

Cílem této bakalářské práce bylo navrhnout a vytvořit webovou aplikaci knihovny pomocí vhodných nástrojů. Zacíleno bylo především na jednoduchost a přehlednost, tak aby bylo jednoduché ovládání pro různé věkové kategorie.

Návrh struktury vycházel z existujících knihoven obcí v oblasti mého bydliště a byl směrován k jejich potřebám. Ty byly víceméně všechny úspěšně splněny.

Vývoj aplikace probíhal za použití frameworku Django. Nástroj velmi usnadnil vývoj některých částí aplikace a to zejména autentizační část.

V aplikaci uchováváme data o knihách, autorech, vydavatelstvích, žánrech, také o uživatelích a knihovnících. Webová aplikace nabízí přístup nepřihlášeným uživatelům k prostému přehledu aktualit, informací o dané knihovně a knižní vybavenosti. Pro uživatele, jež uhradili členský poplatek je k dipozici uživatelský účet, kde po přihlášení mají možnost nahlédnout do vlastních výpůjček. Jako knihovník má uživatel možnost vytvářet výpůjčky, prodloužit platnost členství, editovat a mazat záznamy.

Bude-li u místní obecní knihovny zájem využívat tuto aplikaci, jistě by mohlo dojít k naplnění některých plánů k vylepšení aplikace, jako je například rozšíření statistiky či založení galerie.

SEZNAM POUŽITÉ LITERATURY

[1] LANius s.r.o. Clavius - základní informace, [Online]. Dostupné z: http://www.lanius.cz/clavius/info.htm. [cit. 2019-03-10].

[2] TRITIUS. Webová aplikace, [Online]. Dostupné z: https://www.tritius.cz/vlastnosti. [cit. 2019-03-10].

[3] Knihovna města Olomouce. Výsledky vyhledávání, [Online]. Dostupné z: https://tritius.kmol.cz/search?q=kytice&area=-1&field=0, 2017. [cit. 2019-03-11].

[4] EVERGREEN. Knihovní systém - WikiKnihovna, [Online]. Dostupné z: http://wiki.knihovna.cz/index.php?title=EVERGREEN,_knihovn%C3%AD_syst%C3%A9m, 2010. [cit. 2019-03-10].

[5] R-BIT TECHNOLOGY s.r.o. Otevřený integrovaný knihovní systém, [Online]. Dostupné z: https://www.koha-v-knihovne.cz/sites/default/files/ke-stazeni/koha.pdf. [cit. 2019-03-13].

[6] Knihovna města Litomyšl. Výsledky vyhledávání, [Online]. Dostupné z: https://katalog.knihovna-litomysl.cz/Search/Results?lookfor=kytice&type=AllFields, 2019. [cit. 2019-03-11].

[7] DAVIDEK Kryštof. LaTeX Co, proč a jak, [Online]. Dostupné z: https://medium.com/edtech-kisk/latex-co-pro%C4%8D-a-jak-a4c1f4641616. [cit. 2019-03-24].

[8] WILLISON Simon. What is the history of the Django web framework..., [Online]. Dostupné z: https://simonwillison.net/2010/Aug/24/what-is-the-history/, 2010. [cit. 2019-03-10].

[9] Django Software Foundation. Official Django Logos, [Online]. Dostupné z: https://www.djangoproject.com/community/logos/, 2019. [cit. 2019-05-11].

[10] DVOŘÁK Pavel. Django: Úvod a instalace, [Online]. Dostupné z: https://www.zdrojak.cz/clanky/django-uvod-a-instalace/, 2009. [cit. 2019-03-13].

[11] Mark SUMMERFIELD. *Python 3 Výukový kurz.* Computer Press, 1st edition, 2010. ISBN 978-80-251-2737-7.

[12] Python Software Foundation. The Python Logo, [Online]. Dostupné z: https://www.python.org/community/logos/, 2019. [cit. 2019-05-11].

[13] HLAVENKA Jiří. *Vytváříme WWW stránky a spravujeme moderní web site.* Computer Press, 6th edition, 2002. ISBN 80-7226-748-5.

[14] JAHODA Bohumil. Různé druhy HTML značek, [Online]. Dostupné z: `http://jecas.cz/html-znacky`. [cit. 2019-03-24].

[15] CASTRO Elizabeth a HYSLOP Bruce. *HTML5 a CSS3 - Názorový průvodce tvorbou WWW stránek.* Computer Press, 1st edition, 2012. ISBN 978-80-251-3733-8.

[16] BOOTSTRAP. About, [Online]. Dostupné z: `https://getbootstrap.com/docs/4.3/about/overview/`. [cit. 2019-03-11].

[17] ČÁPKA David. Lekce 1 - Úvod do CSS frameworku Bootstrap, [Online]. Dostupné z: `https://www.itnetwork.cz/html-css/bootstrap/kurz/uvod-do-css-frameworku-bootstrap`. [cit. 2019-03-11].

[18] JANOVSKÝ Dušan. Hosting a co od něj požadovat, [Online]. Dostupné z: `https://www.jakpsatweb.cz/hosting.html`. [cit. 2019-05-1].

[19] PYTHONANYWHERE. Plans and pricing, [Online]. Dostupné z: `https://www.pythonanywhere.com/pricing/`, 2019. [cit. 2019-05-1].

[20] ROŠTÍ.CZ. Ceník, [Online]. Dostupné z: `https://rosti.cz/cenik/`, 2019. [cit. 2019-05-1].

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

BSD	Berkeley Software Distribution
CDN	Content Delivery Network
CMS	Content management system
CSS	Cascading Style Sheets
CSRF	Cross-site request forgery
DRY	Don't Re-peat Yourself
ER	Entity relationship
HTML	Hypertext Markup Language
GB	Giga Byte
HTML	Hypertext Markup Language
MB	Mega Byte
MTV	Model-Template-View
MVC	Model-View-Controller
ORM	Object-Relation Mapping
PK	Primary key
PHP	Personal Home Page
SQL	Structured Query Language
URL	Uniform Resource Locator
VPN	Virtual Private Network
VPS	Virtual Private Server

SEZNAM OBRÁZKŮ

Seznam ukázek kódu

SEZNAM PŘÍLOH

K bakalářské práci je přiloženo CD, na kterém je k dispozici výsledná aplikace, ukázkové video a bakalářská práce v elektronické verzi.