**UNIVERZITA KONŠTANTÍNA FILOZOFA V NITRE**

FAKULTA PRÍRODNÝCH VIED

Systém pre on-line publikovanie študijných materiálov a kníh

bakalárska práca

Nitra 2016 Peter Baláž

**UNIVERZITA KONŠTANTÍNA FILOZOFA V NITRE**

FAKULTA PRÍRODNÝCH VIED

Systém pre on-line publikovanie študijných materiálov a kníh

bakalárska práca

Študijný odbor: 9.2.9 Aplikovaná informatika

Študijný program: Aplikovaná informatika

Školiace pracovisko: Katedra informatiky

Školiteľ: Mgr. Martin Drlík, PhD.

Nitra 2016 Peter Baláž

# Abstrakt

BALÁŽ, Peter: Systém pre on-line publikovanie študijných materiálov a kníh. [Bakalárska práca]. Univerzita Konštantína Filozofa v Nitre. Fakulta prírodných vied. Školiteľ: Mgr. Martin Drlík, PhD. Stupeň odbornej kvalifikácie: Bakalár odboru Aplikovaná informatika. Nitra: FPV, 2015. 46 s.

Bakalárska práca sa zaoberá vývojom softvéru slúžiaceho na elektronickú publikáciu študijných materiálov a kníh. Je členená na teoretickú a praktickú časť. Teoretická časť je zameraná na problematiku životného cyklu vývoja softvéru, vodopádový a špirálový model, inkrementálny a prototypový prístup, a agilné metodiky vývoja softvéru, presnejšie extrémne programovanie, lean development, testami riadený vývoj a SCRUM. Praktická časť bakalárskej práce sa zameriava na návrh a implementáciu aplikácie slúžiacej na on-line publikovanie študijných materiálov a kníh čo pozostáva z určenia požiadaviek, vytvorenia diagramu prípadov použitia, diagramu tried a entitno-relačného modelu, zvolenia vhodného postupu a technológií, návrhu používateľského rozhrania, samotnej implementácie a testovania.

Kľúčové slová: Informatika. Prírodné vedy. Vývoj softvéru. Agilné metodiky. Extrémne programovanie. SCRUM.

# ABSTRACT

BALÁŽ, Peter: System for On-line Publishing. [Bachelor Thesis]. Constantine the Philosopher University in Nitra. Faculty of Natural Sciences. Supervisor: Mgr. Martin Drlík, PhD. Degree of Qualification: Bachelor of Applied Informatics. Nitra: FNS, 2015. 46 p.

Thesis deals with software development serving the electronic publication of books and study materials. It is divided into theoretical and practical part. The theoretical part is focused on the problems of software development life cycle, waterfall and spiral model, prototyping and incremental approach, and agile software development methodologies like Extreme Programming, Lean Development, the test-driven development and SCRUM. The practical part of the thesis focuses on the design and implementation of an application for online publishing study materials and books, which consists of the identification of requirements, creating use case diagram, class diagram, entity relationship model, selecting the appropriate processes and technologies, design the user interface and itself implementation and testing.

Keywords: Informatics. Natural Science. Software development. Agile methodology. Extreme Programming. SCRUM.

# Obsah

[Obsah 3](#_Toc448858291)

[Úvod 5](#_Toc448858292)

[1 Analýza súčasného stavu skúmanej problematiky 6](#_Toc448858293)

[1.1 Vodopádový model 6](#_Toc448858294)

[1.2 Inkrementálny prístup 7](#_Toc448858295)

[1.3 Prototypový prístup 8](#_Toc448858296)

[1.4 Špirálový model 8](#_Toc448858297)

[1.5 Agilné metodiky 9](#_Toc448858298)

[1.5.1 Extrémne programovanie 9](#_Toc448858299)

[1.5.2 Lean development 11](#_Toc448858300)

[1.5.3 Vývoj riadený testami 12](#_Toc448858301)

[1.5.4 SCRUM Development process 13](#_Toc448858302)

[2 Ciele bakalárskej práce 15](#_Toc448858303)

[Čiastkové ciele: 15](#_Toc448858304)

[3 Návrh a riešenie 16](#_Toc448858305)

[3.1 Diagram prípadov použitia 16](#_Toc448858306)

[3.2 DIagram tried 17](#_Toc448858307)

[3.3 Návrh dátovej vrstvy 18](#_Toc448858308)

[3.3 Spôsob vývoja 19](#_Toc448858309)

[3.4 Výber programového prostredia 19](#_Toc448858310)

[3.5 Implementačná Architektúra 19](#_Toc448858311)

[3.6 Návrh grafického prostredia 22](#_Toc448858312)

[3.6.1 Dostupné publikácie 23](#_Toc448858313)

[3.6.2 Prihlásenie 23](#_Toc448858314)

[3.6.3 Registrácia 23](#_Toc448858315)

[3.6.4 Zabudnuté heslo 24](#_Toc448858316)

[3.6.5 Vytvorenie a editácia publikácie 24](#_Toc448858317)

[3.6.6 Prehliadanie publikácie 24](#_Toc448858318)

[3.6.7 Vytvorenie a editácia kapitoly 25](#_Toc448858319)

[3.6.8 Prehliadanie kapitoly 25](#_Toc448858320)

[3.6.9 Používateľská knižnica 26](#_Toc448858321)

[3.6.10 Žiadosti na potvrdenie autorstva 26](#_Toc448858322)

[3.6.11 Žiadosti na povolenie prístupu k publikáciám 26](#_Toc448858323)

[3.7 Výsledky a kontrola kvality 27](#_Toc448858324)

[3.7.1 Funkčné testovanie 27](#_Toc448858325)

[3.7.2 Výkonnostné a záťažové testovanie 27](#_Toc448858326)

[3.7.3 Výsledky testov 28](#_Toc448858327)

[Záver 29](#_Toc448858328)

[Zoznam bibliografických odkazov 30](#_Toc448858329)

[Zoznam príloh 32](#_Toc448858330)

# Úvod

Predstavme si, že sme vysokoškolský pedagóg, ktorý má pripravené študijné materiály pre svojich študentov v elektronickej podobe na svojom lokálnom disku na celý akademický rok a chceli by sme, aby sa tieto materiály k študentom dostávali postupne počas celého akademického roka tak, aby reprezentovali vždy len prebratú časť učiva. V prípade, že škola nemá k dispozícii systém, ktorý by umožňoval efektívne zdieľať materiály týmto spôsobom, zostáva nám použiť papierovú formu alebo niektorý zo systémov tretích strán ako napríklad cloudové úložiská ktorými sa opäť znižuje efektivita a komfortnosť. V súčasnosti aj napriek rozširujúcej sa informatizácii spoločnosti je riešení ktoré podobný priebeh umožňujú stále nedostatok, najmä pokiaľ sa jedná o centralizované riešenia.

Našim cieľom je navrhnúť a implementovať webovú aplikáciu, ktorá by centralizovane umožňovala práve takýmto spôsobom postupne zverejňovať kapitoly publikácie podľa uváženia jej autora a zároveň získavať spätnú väzbu od jej čitateľov.

Práca je členená do troch kapitol. Prvá kapitola pojednáva o softvérovom inžinierstve. Prináša stručný pohľad na príčiny jeho vzniku, opisuje prvotné problémy vývoja softvéru a predstavuje inžinierske postupy, ktoré vznikli ako reakcia na tieto problémy. Zároveň je prehľadom niektorých dostupných metodík vývoja softvéru.

Druhá kapitola pojednáva o cieli bakalárskej práce a jej čiastkových cieľoch.

V tretej kapitole je obsiahnutá analýza a návrh samotného riešenia. Zároveň pojednáva o návrhu dátovej vrstvy, spôsobe vývoja, výbere programovacieho jazyka a nástrojoch, implementačnej architektúre a návrhoch grafického prostredia. Vo svojom závere táto kapitola pojednáva o funkčnom, výkonnostnom a záťažovom testovaní a ich výsledkoch.

# 1 Analýza súčasného stavu skúmanej problematiky

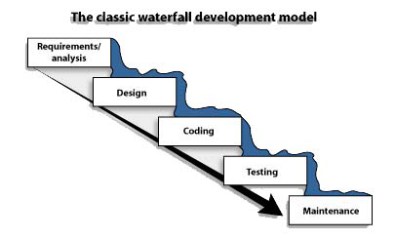
S rozmachom informatizácie neustále stúpa potreba kvalitnejších, rozšíriteľnejších a nezávislejších informačných systémov dodaných v čo najkratšom čase. Pre uspokojenie týchto požiadaviek je však potrebné zefektívniť procesy počas vývoja informačného systému, a preto vzniklo odvetvie informatiky nazývané softvérové inžinierstvo.

Softvérové inžinierstvo je štúdium a aplikácia inžinierstva na návrh, vývoj, implementáciu a údržbu softvéru systematickým spôsobom (Laplante, 2007). Sommerville (2007) softvérové inžinierstvo definuje ako inžiniersku disciplínu, ktorá sa zaoberá všetkými aspektami vývoja softvéru od raných fáz špecifikácie systému až po údržbu po zavedení softvéru do prevádzky. Softvérové inžinierstvo prešlo dlhým vývojom, čo malo za následok vznik a vývoj rôznych metodík tvorby softvéru a jej plánovania. Medzi najznámejšie prístupy plánovania vývoja softvéru patria:

* vodopádový prístup,
* špirálový prístup,
* inkrementálny prístup,
* prototypový prístup,
* agílne prístupy.

## 1.1 Vodopádový model

Vodopádový model je pomenovaný podľa podobnosti grafického znázornenia priebehu s vodopádom. Jeho etapy na seba kaskádovito nadväzujú v presne určenom poradí. Každá fáza je ukončená kompletnou dokumentáciou a ďalšia môže začať až po ukončení predchádzajúcej.



Obrázok Grafické znázornenie vodopádového modelu[[1]](#footnote-1)

Zmeny sa vykonávajú len veľmi ťažko a po určení zmien zákazníkom je potrebné celý proces zopakovať. Ďalšou nevýhodou je, že zákazník nie je s vývojom softvéru takmer vôbec spätý, nemôže takmer vôbec ovplyvniť ani zadanie ani funkcionalitu. Častokrát sa teda stáva, že zákazník nedostane softvér vypracovaný podľa svojich predstáv, čo vedie k pripomienkam a predlžovaniu času potrebného na vývoj takéhoto softvéru. Kvôli týmto nevýhodám je aj napriek svojej intuitívnosti a jednoduchosti vodopádový vhodný len na jednoduché projekty s jasným cieľom s dobre definovanými požiadavkami, takže ho môžeme pokladať len za teoretický spôsob vývoja softvéru (Rajib, 2009). Napriek tomu je však často dodržiavaný pri písaní softvérovej dokumentácie. Vývoj softvéru využitím vodopádového modelu sa skladá z týchto etáp:

* štúdia uskutočniteľnosti,
* analýza požiadaviek a špecifikácia,
* dizajn,
* programovanie a testovanie,
* integrácia a systémové testovanie,
* údržba.

(Rajib, 2009).

## 1.2 Inkrementálny prístup

Inkrementálny prístup je vhodný na kombináciu sekvenčných a iteračných metodík softvérového vývoja. Cieľom je obmedziť projektové riziká rozdelením projektu na menšie časti a zjednodušiť zavedenie zmien počas vývoja. Základné princípy inkrementálneho vývoja sú:

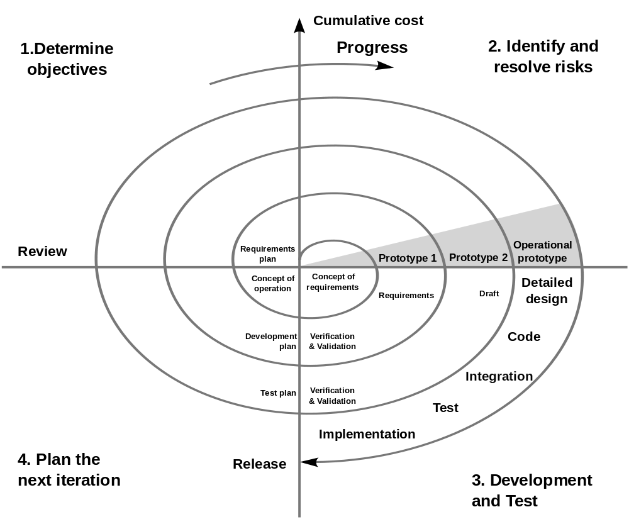
* sú robené série malých vodopádov, kde každý vodopád je určený pre určitú časť softvéru a je dokončený pred pokračovaním na ďalší prírastok, alebo
* všeobecné požiadavky sú definované skôr než sa pristúpi k evolučnému vývoju pomocou malých vodopádov, alebo
* prvotný koncept je definovaný pomocou vodopádu, nasleduje iteratívny prototypový prístup, ktorý vrcholí inštaláciou finálneho prototypu ako funkčného systému (CMS,2008).

## 1.3 Prototypový prístup

V prípade prototypového prístupu sa počas vývoja vytvára množstvo neúplných softvérových riešení, tzv. prototypov, ktoré postupne nadobúdajú finálnu podobu. Prototypový prístup nie je kompletnou metodikou vývoja, ale skôr prístupom k jednotlivým častiam väčších tradičných metodík. Používateľ je zapojený do celého procesu vývoja, čo znižuje riziko, že softvér bude treba prerábať a taktiež rozdelením projektu na menšie časti znižuje závažnosť projektových rizík. Na druhú stranu sa však súčasne vyvíja niekoľko prototypov, z ktorých veľká časť neprejde do samotného vývoja (CMS,2008).

## 1.4 Špirálový model

Špirálový model bol navrhnutý tak, aby zahŕňal najlepšie vlastnosti modelov prototypovania a vodopádu doplnené o nový komponent hodnotenia rizík. Termín špirála sa používa na opis procesu, ktorý prebieha pri vývoji softvéru.



Obrázok Špirálový model[[2]](#footnote-2)

V tomto modeli, podobne ako pri modeli prototypovania, je vyvinutá počiatočná verzia systému, ktorá je následne modifikovaná na základe výsledkov hodnotenia zákazníkom. Na rozdiel od modelu prototypovania, je však vývoj každej verzie navrhnutý pomocou krokov zapojených do vodopádového modelu. Následne sa s každou iteráciou systém nabaľuje až pokým nie je kompletný a pripravený na prebratie zákazníkom (State University of New York, 2016).

## 1.5 Agilné metodiky

Vo februári roku 2001 sa stretli softvéroví vývojári aby prediskutovali ľahké metodiky vývoja. Následne vydali publikáciu Manifesto for Agile Software Development v ktorej hovoria o uprednostnení individualít a interakcií pred procesmi a nástrojmi, funkčný softvér pred obsiahlou dokumentáciou, zákazníkovu spoluprácu pred vyjednávaním o kontrakte a reakciu na zmenu pred dodržiavaním plánu(Beck, 2001). Agilný manifest zakladá na dvanástich princípoch:

* Zákazníkova spokojnosť vďaka skorej a priebežnej dodávke softvéru.
* Vítanie zmien požiadaviek, a to aj v neskorých fázach vývoja.
* Funkčný softvér je dodávaný často.
* Blízka denná spolupráca medzi predajcami a vývojármi.
* Projekty sú postavené na motivovaných jedincoch.
* Konverzácia tvárou v tvár je najlepšia forma komunikácie.
* Funkčný softvér je hlavným meradlom pokroku.
* Udržateľný rozvoj, ktorý je schopný udržať konštantnú rýchlosť.
* Trvalá pozornosť k dobrému dizajnu a technickej dokonalosti.
* Jednoduchosť, ako umenie maximalizovať prácu ktorá ešte nebola spravená, je nevyhnutná.
* Najlepšie architektúry, požiadavky a návrhy pochádzajú zo seba organizovaných tímov.
* Tím pravidelne uvažuje o tom, ako byť efektívnejší, a podľa toho sa upravuje (Kent, 2001).

Podľa Larmana (2004) sú agilné metodiky v porovnaní s tradičným softvérovým inžinierstvom cielené na komplexné systémy a projekty s dynamickou, nedeterministickou a nelineárnou charakteristikou, kde presné odhady, stabilné plány a predpovede je v raných fázach ťažké presne určiť, a zároveň veľké dopredu navrhnuté dizajny a usporiadania by mohli spôsobiť veľké množstvo odpadu. To znamená, že nie sú ekonomicky stabilné. Tieto základné argumenty a skúsenosti v odvetví pomohli tvarovať agilné metodiky do ich súčasného tvaru.

## 1.5.1 Extrémne programovanie

Extrémne programovanie je jednou z najrozšírenejších agilných techník, nakoľko predstavuje predvídateľný, účinný a flexibilný spôsob vývoja softvéru, no je vhodná najmä pre malé tímy skladajúce sa z maximálne desiatich členov, ktoré sa musia rýchlo vysporiadať s nejasným a často sa meniacim zadaním. Základom extrémneho programovania je presvedčenie, že jediným zmerateľným, jednoznačným a overiteľným zdrojom informácií je zdrojový kód (Kadlec, 2004). Podľa Becka (1999) je extrémne programovanie o sociálnych zmenách. Ide v ňom o opustenie návykov a vzorov, ktoré boli adaptívne v minulosti, ale dnes bránia vo vykonávaní práce čo najlepšie. Ide o to vzdať sa ochrany, ktorá interferuje s  produktivitou.

Extrémne programovanie vychádza z princípov a postupov bežných pri vývoji softvéru, no privádza ich do extrémov. Pri extrémnom programovaní sa kód neustále reviduje vďaka využitiu párového programovania, kedy sa programátori navzájom kontrolujú, čo pomáha predísť chybám priebežne počas celej doby vývoja. Následne sa kód necháva testovať vývojármi v podobe jednotkových automatizovaných testov, a tiež používateľmi v podobe funkčných testov. Ďalším typickým znakom XP je refaktorizácia, počas ktorej všetci každodenne upravujú a vylepšujú návrh zdrojového kódu. XP sa zároveň vyznačuje krátkymi iteráciami, ktoré nie sú týždne, mesiace či roky, ale sekundy, minúty, hodiny. Tento znak je označovaný kontinuálna integrácia.

Extrémne programovanie upravuje tri tradičné premenné vývojového cyklu a pridáva štvrtú:

* kvalita,
* čas,
* náklady,
* šírka zadania (Shore, 2007).

Dôležitou myšlienkou XP je, že zákazníci, zadávatelia a manažéri zadávajú len ľubovoľné tri z týchto premenných a štvrtú, poslednú, premennú určuje vývojový tím. Ak zadávatelia s výslednou premennou nie sú spokojní, majú možnosť vstupné premenné predefinovávať do tej doby, kým nebudú spokojní výslednou hodnotou (Kadlec, 1999).

Ideovým základom XP je komunikácia, jednoduchosť, spätná väzba, odvaha a rešpekt čo však nestačí na vytvorenie funkčnej metodiky, kvôli čomu tiež obsahuje dvanásť základných postupov, ktoré vedú k vytvoreniu kvalitného softvéru:

* Plánovacia hra, ktorej cieľom je zapojiť celý tým do plánovania.
* Malá verzia, čo znamená, že nová verzia sa vydáva často.
* Metafora, ktorá slúži na popis funkcionality softvéru pomocou príbehu, čo zabezpečuje, že jej rozumejú všetky zainteresované strany.
* Jednoduchý návrh, teda najjednoduchší návrh, ako je danom momente možné.
* Testovanie, čiže spôsob ako udržať kvalitu aj napriek absencii explicitne definovaného návrhu.
* Refaktorizácia, ktorá pomáha zvyšovať kvalitu kódu jeho zjednodušovaním bez akejkoľvek zmeny chovania softvéru.
* Párové programovanie, pri ktorom zdrojový kód píše navzájom sa kontrolujúca dvojica programátorov.
* Spoločné vlastníctvo, hovoriace o myšlienke, že každý môže meniť ktorúkoľvek časť kódu v akomkoľvek čase.
* Nepretržitá integrácia, čo znamená, že systém je integrovaný, zostavený a testovaný minimálne raz denne.
* Štyridsať hodinový pracovný týždeň, nakoľko dlhodobé nadčasy vedú k znižovaniu efektivity.
* Skutočný zákazník alebo používateľ sa počas vývoja stáva súčasťou tímu.
* Dodržiavanie štandardov písania zdrojového kódu, čím je dosiahnutá vyššia čitateľnosť a kvalita zdrojového kódu.

## 1.5.2 Lean development

Shore (2007) definuje Lean Development ako systematický prístup k identifikácii a odstráneniu možného plytvania v priebehu celého vývoja, čím sa pokúša dodať zákazníkovi bezchybný produkt a uspokojiť tým jeho požiadavky čo znamená, že jeho hlavnou myšlienkou je odstránenie všetkého nadbytočného, čo v priebehu vývoja vzniká a môže negatívne ovplyvniť náklady, efektivitu a kvalitu výsledného produktu.

Lean Development sa dá zosumarizovať pomocou siedmych princípov, ktoré sú podobné konceptu štíhlej výroby:

* Odstránenie všetkého, čo zákazník nechce, nepoužije alebo pre neho nemá hodnotu.
* Posilnenie učenia pomocou krátkych iteračných cyklov a získavanie spätnej väzby od zákazníka.
* Odloženie závažných rozhodnutí na čo najpokročilejšiu dobu, kvôli možným rozhodnutiam zákazníka zmeniť požiadavku po dlhšom zvážení.
* Potreba dodať produkt v čo najkratšom čase, nakoľko je tým umožnené dostatočne skoro získavať spätnú väzbu a zmeny získane na jej základe začleniť do procesu vývoja.
* Dodanie potrebnej pozitívnej motivácie a prostriedkov tímu pre jeho posilnenie.
* Budovanie integrity systému, či už architektonickej, vnímanej alebo koncepčnej je ďalším dôležitým princípom, obsiahnutým v koncepte štíhlej výroby.
* Posledný z princípov hovorí o nutnosti vnímať produkt aj ako celok, nakoľko veľké systémy sú zložené z viacerých častí, ktoré nemusia byť vyvíjané tým istým tímom, a preto je potrebné zabezpečiť vzájomnú kompatibilitu súčastí (Poppendieck, M.; Poppendieck, T. 2003).

## 1.5.3 Vývoj riadený testami

Beck (2012), ktorý je považovaný za znovuobjaviteľa tejto metódy, napísal, že bol inšpirovaný starou knihou v ktorej sa nachádzal postup, podľa ktorého bolo potrebné vziať pásku so vstupnými údajmi, manuálne napísať pásku s výstupnými údajmi podľa očakávaní a programovať dovtedy, pokým výstupná páska neobsahovala dáta podľa očakávaní.

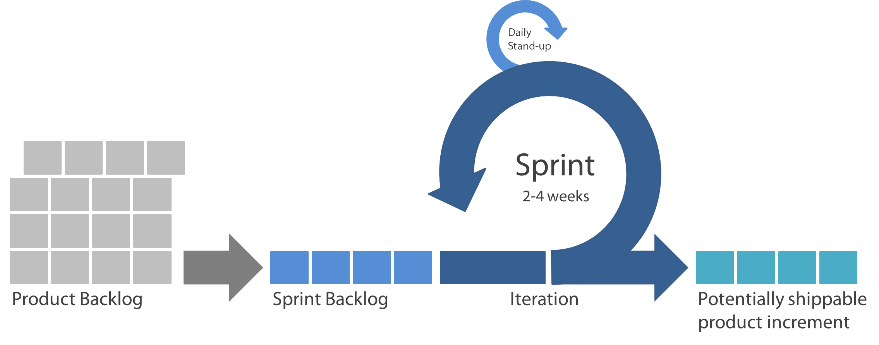
Test Driven Development je metóda, ktorej základom je rýchly cyklus testovania, programovania a refaktoringu. Pri pridávaní funkcionality môžu prejsť desiatky týchto cyklov programovania, zavádzania a zdokonaľovania softvéru po malých krokoch, pokým nezostane nič, čo by bolo potrebné pridať alebo odobrať. Pokiaľ je táto metóda použitá správne, pomáha v zdokonaľovaní dizajnu, dokumentovaní verejných rozhraní a predchádzaní budúcim omylom (Shore, 2007).

Podľa Becka (2003) pozostáva cyklus vývoja použitím tejto metódy z piatich opakujúcich sa krokov:

* Vytvorenie testu, ktorým v TDD začína každá nová funkcionalita.
* Spustenie všetkých testov pre zistenie, či novo vytvorený test neprejde testovaním, čo je v tomto prípade žiaduce.
* Napísanie kódu, ktorý novo vytvoreným testom prejde bez ohľadu na jeho kvalitu.
* Spustenie všetkých testov za účelom zistenia, či napísaná časť kódu testami prejde, a zároveň negatívne neovplyvní ostatné funkcionality.
* Refaktorovanie kódu pre zvýšenie jeho kvality odstránením duplicít, správnym pomenovaním metód a premenných prípadne úprava na tvar vďaka ktorému môže profitovať z rozpoznateľných návrhových vzorov.

## 1.5.4 SCRUM Development process

Pôvod slova scrum je možné nájsť v športe nazývanom ragby, kde sa ním označuje metóda pri reštarte hry keď sú hráči natlačení tesne pri sebe s hlavami dole s cieľom zmocniť sa lopty. Táto metodika sa radí medzi agilné a je definovaná ako flexibilná stratégia pre vývoj produktu, kde celý tím pracuje ako jednotlivec pre dosiahnutie spoločného cieľa (Takeuchi;Nonaka, 1986).



Obrázok Grafické znázornenie SCRUM procesu[[3]](#footnote-3)

Vývoj podľa tejto metodiky prebieha v presne daných časových úsekoch nazývaných šprinty. Každý šprint trvá mesiac alebo menej a po jeho skončení je tím schopný dodať funkčný a otestovaný produkt, následne začína nové plánovanie a cyklus sa opakuje až pokým nie je produkt dokončený.

Hlavným dokumentom SCRUM-u je Backlog, ktorý tvorí utriedený list všetkého, čo môže byť potrebné v produkte a majiteľ produktu zodpovedá za jeho obsah, dostupnosť a utriedenie. Tým ako je produkt používaný, naberá na hodnote a trh ponúka spätnú väzbu, Backlog rastie a je vyťažujúcejší (Schwaber; Sutherland, 2014). Metodika uvádza tri základné role ktoré sa na vývoji podieľajú. Prvou je majiteľ produktu, ktorého hlavnou úlohou je komunikácia, vďaka ktorej je schopný zžiť sa s členmi tímu a ostatnými zúčastnenými stranami, čo vedie k riadeniu projektu správnym smerom (Pichler, 2010). Nasledujúcou rolou je vývojársky tím, ktorý je zodpovedný za dodávku potenciálne publikovateľného produktu na konci každého šprintu. Vývojárske tímy musia byť zostavené tak, aby ich členovia mali všetky potrebné schopnosti na dokončenie celého prírastku. Poslednou rolou je vedúci scrum-u. Jeho úlohou je odstraňovať prekážky, ktoré bránia tímu v dokončení cieľov, a zároveň dohliada na to, aby scrum prebiehal tak, ako bolo zamýšľané. Zároveň je jeho náplňou práce aj poskytovať služby ako majiteľovi produktu, tak vývojárskemu tímu, ako napríklad:

* nachádza spôsoby efektívnej tvorby produktového Backlogu,
* chápe produktové plánovanie v empirickom prostredí,
* zabezpečuje, aby majiteľ produktu vedel zostaviť Backlog tak, aby bola maximalizovaná jeho hodnota,
* upravuje udalosti scrumu podľa požiadaviek alebo potreby,
* organizuje vývojársky tím,
* pomáha tímu vo vytváraní produktov s vysokou kvalitou (Schwaber; Sutherland, 2014).

# 2 Ciele bakalárskej práce

Cieľom bakalárskej práce je navrhnúť a vytvoriť webovú aplikáciu pre elektronické publikovanie publikácií, ktorá umožní autorom postupne zverejňovať jednotlivé kapitoly publikácií, sprístupňovať ich registrovaným čitateľom a získavať spätnú väzbu.

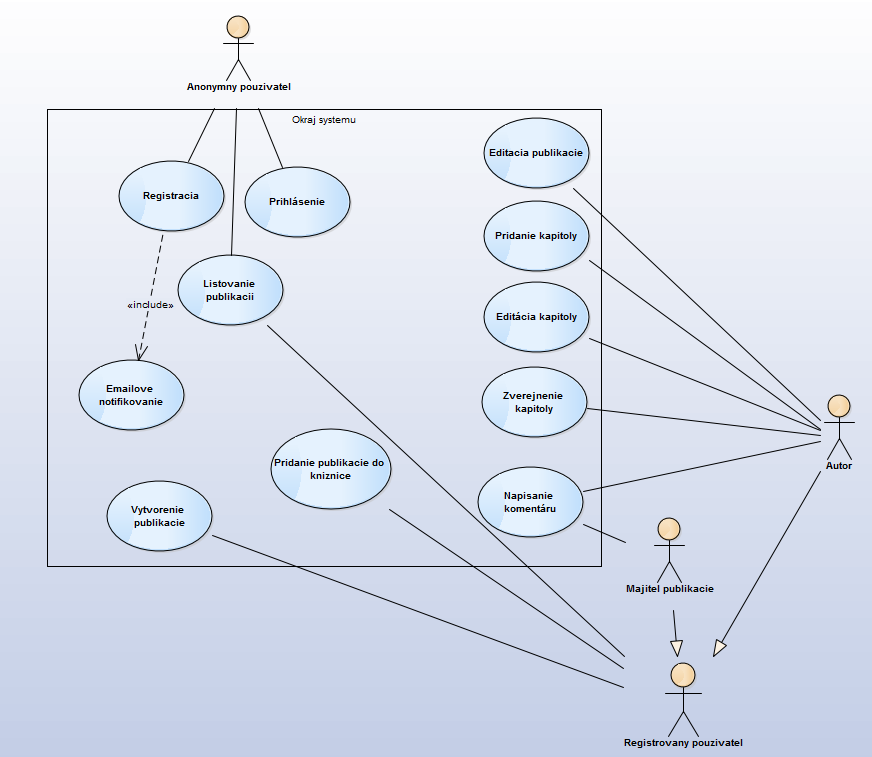
## Čiastkové ciele:

* navrhnúť diagramy podľa UML štandardu,
* navrhnúť databázovú štruktúru,
* zvoliť spôsob vývoja,
* navrhnúť implementačnú architektúru,
* navrhnúť používateľské prostredie,
* implementovať navrhnuté riešenie.

# 3 Návrh a riešenie

## 3.1 Diagram prípadov použitia

V našom prípade, podľa požiadaviek vyplývajúcich zo zadania práce vychádzame z predpokladu, že systém by mal obsahovať rolu registrovaného a anonymného používateľa a autora. My sme sa v tomto prípade rozhodli priamo zadefinovať len role registrovaného a anonymného používateľa. Rola registrovaného používateľa sa ďalej v systéme aplikačne rozdeľuje na autora a majiteľa danej publikácie.

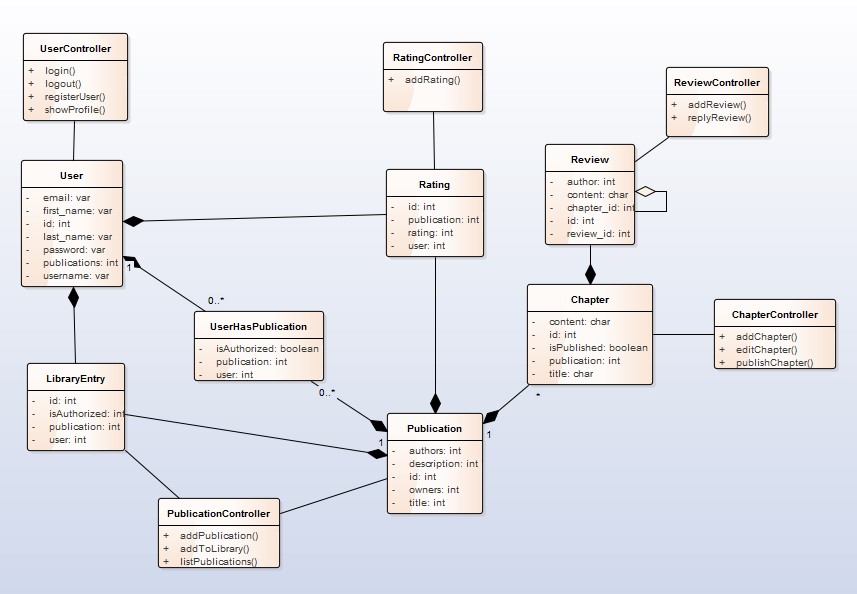


Obrázok Diagram prípadov použitia

Anonymný používateľ má k dispozícii len základné mechaniky ako je prihlásenie, registrácia a listovanie v publikáciách, zatiaľ čo registrovaný používateľ má k dispozícii možnosť vytvoriť publikáciu a tým sa v rámci danej publikácie dostať do role autora, prípadne zvoliť si existujúcu publikáciu a požiadať jej autora o povolenie prístupu. Pokiaľ autor toto povolenie používateľovi pridelí, ten sa automaticky stáva majiteľom publikácie a je mu umožnené pristupovať k jednotlivým kapitolám bez obmedzení, prípadne poskytovať spätnú väzbu pre autora. V prípade, že sa používateľ nachádza v roli autora, je mu umožnené editovať danú publikáciu, pridávať ,editovať a zverejňovať jej kapitoly, odpovedať na prichádzajúci spätnú väzbu a udeľovať povolenia pre prístup k publikácii.

## 3.2 DIagram tried

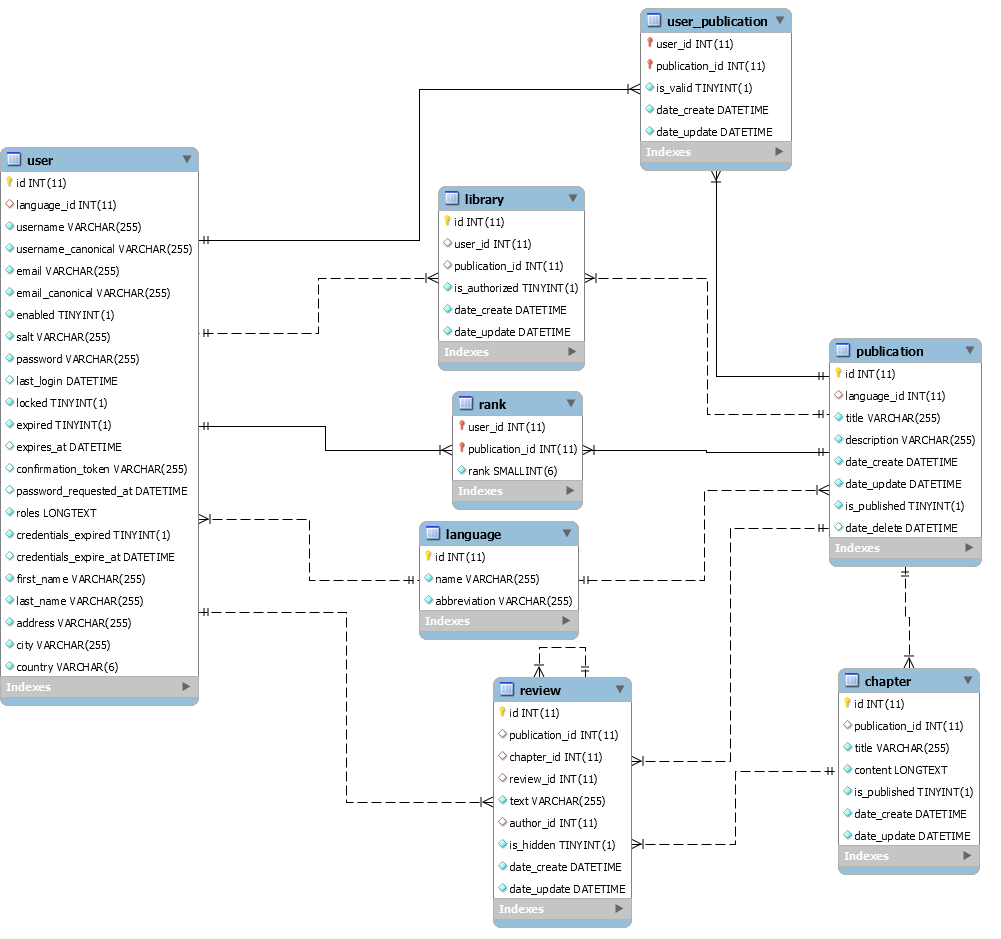
Diagram tried (Obrázok 6) vznikol ako podklad k počiatočným stretnutiam a ujasneniu rozsahu systému. Vzhľadom k agilnému spôsobu vývoja výsledné riešenie nie je totožné s pôvodným konceptom vo všetkých častiach, nakoľko bolo nutné niektoré časti zmeniť z praktického hľadiska, prípadne prispôsobiť kvôli kompatibilite so zvoleným frameworkom. Nakoľko sa vychádza z predpokladu, že zvolený framework bude podporovať objektovo relačné mapovanie(ORM), diagram tried neobsahuje metódy a triedy spojené s perzistenciou, tvorbou, ani vyhľadávaním dát.



Obrázok Diagram tried

## 3.3 Návrh dátovej vrstvy

Na základe diagramu tried na sme pristúpili k zostaveniu návrhu dátovej vrstvy (Obrázok 7). Tento návrh zohľadňuje všetky zadefinované entity v diagrame tried, s uvedenými atribútmi. Okrem zadefinovaných atribútov z diagramu tiež obsahuje atribúty nápomocné pri správe a údržbe systému. Sú v ňom obsiahnuté všetky potrebné indexy, obmedzenia a vzťahy medzi entitami spolu s dátovými typmi jednotlivých stĺpcov. Zároveň je z veľkej časti navrhnutý tak, aby každý atribút bol buď kľúčovým alebo priamo závislým na celom kľúči, zároveň aby bol každý nekľúčový atribút plne funkčne závislý na primárnom kľúči a aby domény všetkých atribútov obsahovali len atomické hodnoty.



Obrázok Entitno-relačný diagram

## 3.3 Spôsob vývoja

Po úvodnom stanovení približného rozsahu bol stanovený spôsob vývoja. Z dôvodu nepravidelných stretnutí nemôžme hovoriť o konkrétnej metodike, a však pokúšali sme sa dodržať pravidelné dvojtýždňové iterácie vzhľadom na vývoj zdrojového kódu. Pred každou iteráciou celý systém čiastočne prešiel používateľskými testami, z ktorých sme získali spätnú väzbu, na základe ktorej sme sa rozhodli, či do systému zapracujeme navrhované zmeny.

Vývoj je verzionovaný pom ocou nástroja GIT. Každá iterácia bola odklonená do vlastnej vývojovej vetvy a po skončení iterácie bola zlúčená s vetvou produkčnou.

## 3.4 Výber programového prostredia

Pri výbere programovacieho jazyka sme sa rozhodovali medzi PHP a Java. Brali sme do úvahy viacero aspektov, no najviac zavážila predpokladaná náročnosť k používateľskému prostrediu a zložitosť učenia daného jazyka. Nakoniec sme sa na základe požiadavky na čo najvyššiu funkčnú efektivitu čiže možnosti veľkého prírastku funkcionality v krátkom čase rozhodli pre jazyk PHP.

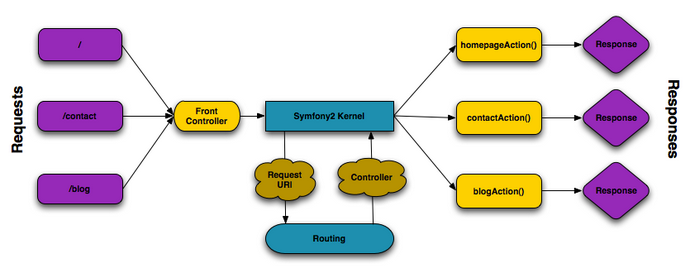
Pre vysokú potrebu funkčnej efektivity sme sa rozhodli pre požitie frameworku čo je softvérová štruktúra, ktorá slúži ako podpora pri programovaní. Obsahuje rôzne už implementované návrhové vzory a nástroje, ktoré programátor môže znovu použiť a nemusí ich znova programovať. Na základe odporúčaní, osobných skúseností, veľkosti komunity a kvality dokumentácie sme sa rozhodli pre Symfony Framework vo verzii 3.0.

Vývoj používateľského prostredia prebiehal pomocou moderných technológií HTML5, CSS3 a javascriptového frameworku jQuery.

Výhodou takto zvoleného riešenia je jeho multiplatformovosť a veľká bezpečnosť, nakoľko používané vzory z frameworku neobsahujú bezpečnostné diery. Použitím javascriptového frameworku bolo možné použiť veľké množstvo funkcionality, ktorá bola otestovaná naprieč všetkými používanými prehliadačmi a tak sa nebolo potrebné zdržiavať zložitým testovaním.

## 3.5 Implementačná Architektúra

Webová služba vyvíjaná monoliticky je dnes už prakticky odsúdená k rýchlemu zániku, nakoľko rozširovanie a údržba takéhoto systému sú po krátkej dobe prakticky nemožné. Väčšina frameworkov z tohto dôvodu siaha po architektonickom vzore Model-View-Controller(MVC). Pre dlhšiu životnosť webovej aplikácie je totiž nutné čo najviac oddeliť dátovú, aplikačnú a prezentačnú vrstvu. Symfony oproti týmto frameworkom poskytuje nástroje riadiacej časti, vizuálnej časti ale nie modelovej. Je len na vývojárovi, či si vytvorí model sám alebo použije nejaký nástroj, napríklad ORM. Symfony je Request/Response framework čiže jeho základné princípy sa točia okolo HTTP špecifikácie. Na obrázku 8 je znázornený aplikačný tok v rámci Symfony frameworku. Prichádzajúce požiadavky sú interpretované trasovaním a posunuté funkciám radiča ktoré vracajú objekt odpovede. Každá stránka aplikácie je definovaná v konfiguračnom súbore trasovania, ktorý mapuje rôzne adresy na rozdielne PHP funkcie. Úlohou každej PHP funkcie, nachádzajúcej sa v radiči, je použiť informáciu z požiadavky na vytvorenie a vrátenie objektu odpovede.

 Obrázok Request-Response aplikačný tok[[4]](#footnote-4)

Najlepšou a najsilnejšou črtou Symfony frameworku je systém balíčkov. Tieto balíčky sú niečo ako zásuvné moduly pri iných softvéroch, no na rozdiel od nich v Symfony frameworku je balíček každá jeho súčasť od jeho jadra až po kód ktorý napíše programátor pri jeho používaní. Dalo by sa povedať, že balíček je štruktúrovaná množina súborov, ktorá implementuje jednu funkcionalitu a ktorá je ľahko zdielateľná medzi vývojármi.

Najčastejším dôvodom pre problémové udržiavanie, rozšírenie a úpravu aplikácie je priveľká závislosť od iných komponentov. Jedným zo spôsobov ako túto závislosť zredukovať, je pretrhaním väzieb tým, že jednotlivé závislosti sa do objektov ktoré ich využívajú vkladajú buď pomocou konštruktora pri inštanciovaní triedy alebo pomocou príslušných setterov. Na popis práve takéhoto procesu sa používa dizajnový vzor Dependency Injection ktorého podpora je priamo implementovaná v jadre nami zvoleného frameworku. Tento dizajnový vzor slúži na vkladanie závislostí medzi jednotlivými komponentami programu tak, aby jeden komponent programu mohol používať druhý bez toho, aby mal v dobe zostavovania programu referenciu. Martin Fowler(2004) poukazuje na tri rôzne vzory, ako je možné objektu predať externú referenciu iného objektu.

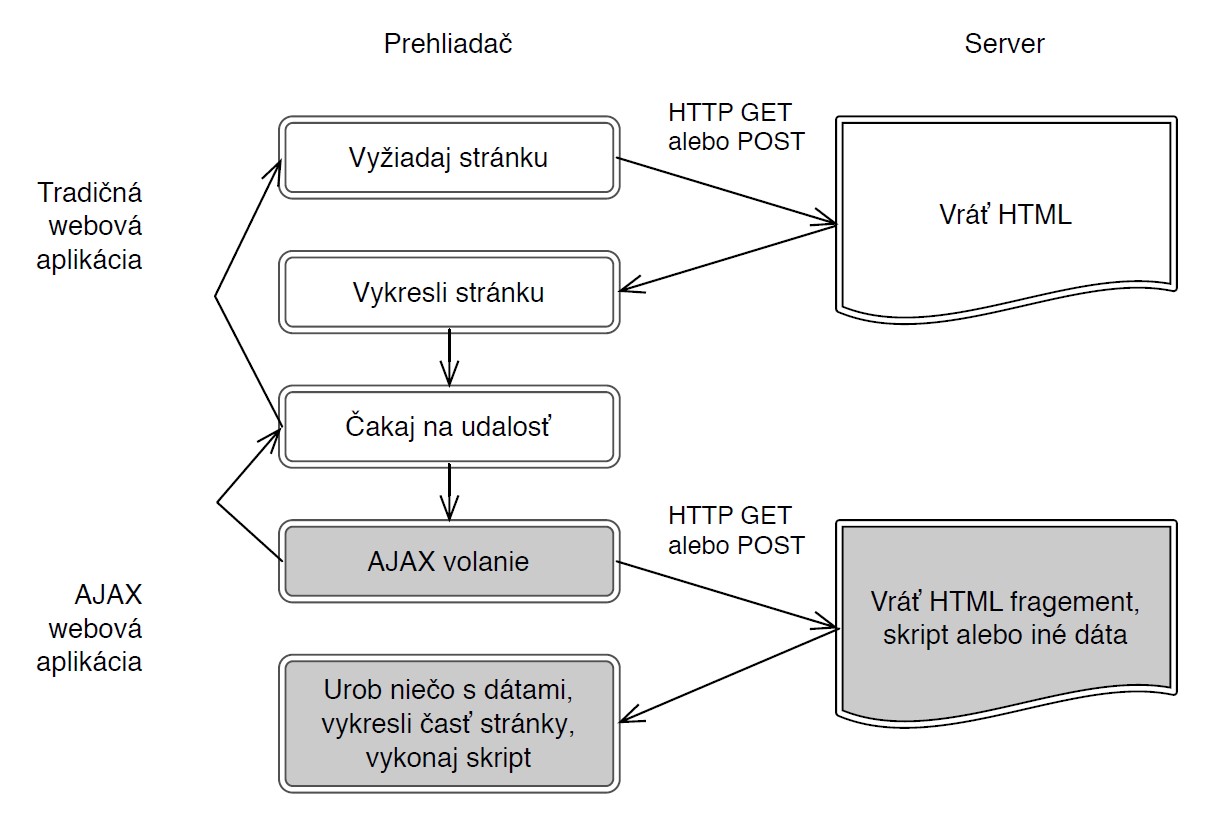
1. Vkladanie rozhraním, pri ktorom externý modul, ktorý je do objektu pridaný implementuje rozhranie, ktoré objekt očakáva v dobe zostavenia programu.
2. Vkladanie setter metódou, pomocou ktorej je možné objektu vložiť závislosť.
3. Vkladanie konštruktorom, pri ktorom je závislosť objektu vkladaná pri jeho inštanciovaní.

Pre potrebu práce s dátami ktorá nie je priamo podporovaná nami zvoleným frameworkom sme sa rozhodli pre použitie ORM frameworku Doctrine, ktorý je plne kompatibilný so Symfony frameworkom, veľmi dobre spĺňa naše potreby a zároveň je dostupný ako hotový balíček pre Symfony. Doctrine sa skladá z dvoch projektov. Object Relational Mapper a Database Abstraction Layer(DBAL) na ktorého základe je postavený(Doctrine Project, 2016). Zatiaľ, čo je entita v relačnom databázovom systéme reprezentovaná ako riadok, prípadne skupina riadkov v tabuľke, v objektovo-orientovanom jazyku je spravidla reprezentovaná ako inštancia triedy. Táto odlišná reprezentácia entít má za následok vznik ORM, ktorý sa stará o konverziu medzi relačnou databázou a objektami. Týmto sa snaží dať vývojárovi unifikovaný prístup k entite, s ktorou v aplikácii pracuje.

Oba tieto nami zvolené frameworky obsahujú čítač anotácií, ktorý umožňuje konfigurovanie objektov pomocou jednoduchých anotácií, ktoré získava z jednotlivých tried pomocou reflexie. V takto pripravených triedach framework dokáže daným triedam sám vložiť pri vytvorení objektu ich závislosti, nastavenia, obmedzenia prípadne ich na základe anotácií validovať, rozhodovať o použití kľúčov, dátových typov, vložiť dáta z iných tabuliek či dokonca generovať SQL dotazy pre vytvorenie alebo úpravu tabuliek. Nakoľko je získavanie reflexných tried v PHP pomerne pomalým procesom, oba frameworky používajú internú cache na to, aby si triedy v ktorých tieto anotácie využívajú uložili už predgenerované na opätovné použitie.

Pre tvorbu používateľského prostredia sme zvolili šablónovací systém Twig, ktorý je podobne ako Doctrine dostupný priamo ako balíček pre framework Symfony. Šablónovacie systémy obsahujú všetku potrebnú funkcionalitu pre vytváranie používateľského prostredia, v našom prípade za použitia značkovacieho jazyka HTML. Umožňujú jednoduchú prácu s objektami určenými na zobrazenie vo webovom prehliadači, jednoduché znovupoužitie opakovaných častí kódu a z veľkej časti pomáhajú zabraňovať cross site scripting útokom(Juras, 2014).

Z dôvodu, že počas používania systému dochádza k interakcii medzi používateľmi, je potrebné systém navrhnúť tak, aby bol schopný distribuovaných prístupov. Z tohto dôvodu musí časť komunikácie medzi klientom a serverom prebiehať asynchrónne(Obrázok 9).



Obrázok Požiadavky na server v AJAX webovej aplikácii.[[5]](#footnote-5)

Klasický spôsob komunikácie by bol síce zrozumiteľný pre osobu, ktorá danú akciu vyvolala, no v prípade, že táto akcia súvisí s iným používateľom, ktorý práve aplikáciu používa, tento by sa o vyvolaní akcie dozvedel až po obnovení stránky.

## 3.6 Návrh grafického prostredia

Najdôležitejšou časťou pri vývoji aplikácie je návrh grafického prostredia. Pri návrhu sme sa inšpirovali stránkami ktoré používajú podobné mechaniky ako náš systém. Ako pomôcku sme použili wireframeovací nástroj Balsamiq Mockups ktorý slúži na jednoduché navrhovanie používateľských protredí.

Používateľ sa pri používaní aplikácie stretne s nasledovnými nástrojmi:

* Dostupné publikácie
* Prihlásenie
* Registrácia
* Zabudnuté heslo
* Vytvorenie a editácia publikácie
* Prehliadnutie publikácie
* Vytvorenie a editácia kapitoly
* Prehliadnutie kapitoly
* Používateľská knižnica
* Žiadosti na potvrdenie autorstva
* Žiadosti na povolenie prístupu k publikácii

### 3.6.1 Dostupné publikácie

Stránka dostupných publikácií je zároveň úvodnou stránkou po príchode do systému. Sú na nej zobrazené všetky zverejnené publikácie aj pre anonymného používateľa v podobe interaktívneho rámu ktorý po kliknutí presmeruje používateľa na stránku prehliadania publikácie. Rám zodpovedajúci každej publikácii obsahuje jej názov, používateľské hodnotenie v podobe hviezd, popis publikácie a informáciu o autorstve. Po prihlásení do systému sa používateľovi sprístupní bočný panel, ktorý obsahuje možnosti pre vytvorenie novej publikácie. V prípade, že je používateľ zároveň autorom niektorej z publikácii, pri tejto sa v hornej časti rámu zobrazí panel obsahujúci tlačidlá pre zmazanie a editáciu.

### 3.6.2 Prihlásenie

Stránka s prihlasovacím formulárom je dostupná pre anonymného používateľa cez tlačidlo Prihlásenie v pravej časti horného panelu alebo je na ňu automaticky presmerovaný pri pokuse o navštívenie stránky na ktorú nemá ako anonymný používateľ oprávnenie. Prihlasovací formulár obsahuje textové polia pre zadávanie používateľského mena a hesla, tlačidlo pre potvrdenie a hypertextové odkazy na stránku pre registráciu a obnovu strateného hesla.

### 3.6.3 Registrácia

Stránka s registračným formulárom je dostupná cez hypertextový odkaz umiestnený na stránke s prihlasovacím formulárom. Registračný formulár obsahuje textové polia pre zadávanie používateľových údajov ako sú email, používateľské meno, heslo s overením, meno, priezvisko, adresu, mesto, krajinu, preferovaný jazyk, Turingov test pre odfiltrovanie registračných botov a potvrdzovacie tlačidlo.

### 3.6.4 Zabudnuté heslo

Stránka s formulárom pre obnovenie strateného hesla je podobne ako stránka registrácie dostupná pomocou hypertextového odkazu umiestneného v prihlasovacom formulári. Formulár na tejto stránke pozostáva z jediného textového poľa, do ktorého používateľ vpisuje používateľské meno alebo email a potvrdzovacie tlačidlo.

### 3.6.5 Vytvorenie a editácia publikácie

Stránka s formulárom pre vytvorenie a editáciu publikácie je dostupná v prípade vytvárania novej publikácie cez tlačidlo v bočnom paneli na stránke dostupných publikácií len v prípade, že je používateľ prihlásený do systému. V prípade editácie je dostupná pomocou tlačidla na editáciu v lište umiestnenej v ráme ukážky publikácie, prípadne pomocou tlačidla editácie v bočnom paneli na stránke prehliadania publikácie a to len v prípade, ak je používateľ zároveň jej autorom. Samotný formulár pozostáva z textových polí pre zadanie nadpisu a stručného popisu publikácie, poľa s výberom jazyka v ktorom bude daná publikácia písaná, poľa s výberom autorov a potvrdzovacieho tlačidla. V prípade, že sa k tejto stránke pokúsi pristúpiť anonymný používateľ, bude presmerovaný na stránku s prihlasovacím formulárom.

### 3.6.6 Prehliadanie publikácie

Stránka umožňujúca prehliadnutie publikácie je dostupná všetkým používateľom cez interaktívny rám, umiestnený na stránke dostupných publikácií, prípadne v používateľovej knižnici. Základné zobrazenie je horizontálne rozdelené do troch celkov.

Prvý celok obsahuje názov publikácie spolu s hodnotením v podobe hviezd ktorý je horizontálne rozdelený čiarou od informácie o autorstve a informácie o napísaní a poslednej editácie publikácie. Následne sa nachádza rám obsahujúci popis publikácie.

Druhý celok obsahuje zoznam dostupných kapitol. Každá kapitola sa nachádza v samostatnom interaktívnom ráme, ktorý obsahuje názov kapitoly horizontálnou čiarou oddelený od informácie o napísaní a poslednej editácii kapitoly. Pod touto informáciou sa nachádza ukážka kapitoly v podobe prvého odstavca.

V treťom, poslednom celku sa nachádza súhrn používateľských hodnotení zo všetkých zverejnených kapitol spolu s odpoveďami na jednotlivé hodnotenia.

Možnosti každého z uvedených celkov sa menia vzhľadom na používateľskú rolu vo vzťahu k danej publikácii. V prípade, že je používateľ prihlásený, ale nie je autorom ani potvrdeným majiteľom publikácie, sprístupní sa mu možnosť požiadať autora o pridelenie práva k prečítaniu publikácie v pomocou tlačidla umiestneného v bočnom paneli. V prípade, že je používateľ prihlásený a zároveň vedený ako autor publikácie, sú mu sprístupnené funkcie pre editáciu a zmazanie publikácie a možnosť pridať novú kapitolu pomocou tlačidiel v bočnom paneli. Zároveň sa rám obsahujúci ukážku jednotlivých kapitol stáva hypertextovým odkazom umožňujúci prečítanie danej kapitoly. V prípade že je používateľ prihlásený, nie je vedený ako autor publikácie no má pridelené povolenie k jej prečítaniu, je mu umožnené hodnotiť publikáciu kliknutím na diagram hviezd vedľa nadpisu v prvej časti. Zároveň sa rámy s ukážkami kapitol rovnako ako v prípade autora stávajú hypertextovými odkazmi. Každému používateľovi môže byť povolené prideliť hodnotenie len jeden krát.

### 3.6.7 Vytvorenie a editácia kapitoly

Stránka obsahujúca formulár pre vytvorenie a editáciu kapitoly je dostupná len pre autora publikácie ku ktorej má byť daná kapitola pridružená. Na túto stránku sa používateľ dostane v prípade tvorby novej kapitoly pomocou tlačidla umiestneného v bočnom paneli zo stránky prehliadania publikácie. V prípade editácie sa na túto stránku používateľ taktiež dostane použitím tlačidla v bočnom paneli, no v tomto prípade zo stránky prehliadania kapitoly.

Formulár umiestnený na stránke pozostáva z textového poľa zodpovedného za nadpis kapitoly ,WYSIWYG editora ktorý slúži na formátované písanie samotného obsahu kapitoly a potvrdzovacieho tlačidla pre uloženie zmien.

### 3.6.8 Prehliadanie kapitoly

Stránka, na ktorej je používateľovi umožnené prečítať si jednotlivé kapitoly je dostupná po kliknutí na interaktívny rám na stránke prehliadania publikácie v prípade, že je používateľ autorom prípadne majiteľom publikácie. V hornej časti zobrazenia sa nachádza nadpis kapitoly spoločne s celkovým hodnotením publikácie. Pod ktorým je čiarou oddelená informácia o autorstve , dátume a čase publikácie a poslednej úpravy. Nižšie sa nachádza panel obsahujúci samotný obsah kapitoly v zodpovedajúcom formáte. Na konci sa nachádza časť s hodnoteniami a formulár pre napísanie hodnotenia k danej kapitole. Každý používateľ má právo napísať jediné hodnotenie, na ktoré však môže napísať neobmedzený počet odpovedí. Na každé hodnotenie má právo odpovedať jedine jeho autor alebo autor publikácie.

Autor publikácie má navyše k dispozícii tlačidlá pre editáciu a publikovanie resp. stiahnutie publikácie, ktoré sa nachádzajú v bočnom paneli.

### 3.6.9 Používateľská knižnica

Stránka používateľskej knižnice je dostupná priamo z hlavnej ponuky. V prípade, že používateľ nie je prihlásený, je presmerovaný na prihlasovací formulár. Táto stránka je veľmi podobná stránke s dostupnými publikáciami, no sú v nej zobrazené len publikácie, na ktoré dostal používateľ povolenie od ich autora čiže je z pohľadu aplikácie vedený ako ich majiteľ.

### 3.6.10 Žiadosti na potvrdenie autorstva

V prípade, že sa niektorý s používateľov pri písaní publikácie rozhodne uviesť okrem svojej osoby aj iných autorov, títo sa v popise publikácie nezobrazia do doby, pokým na to nedajú súhlas. Ten môžu udeliť pomocou nástroja, ktorý sa dostupný z hlavného menu po prihlásení používateľa. V zobrazení tohto nástroja sa nachádza tabuľka, ktorej riadky reprezentujú jednotlivé žiadosti o autorstvo. V pravej časti každého riadku sa nachádza tlačidlo, ktoré umožňuje túto žiadosť potvrdiť. Každé zobrazenie žiadosti pozostáva z názvu kapitoly, zoznamu všetkých zvolených autorov vrátane tých, ktorí ešte autorstvo nepotvrdili, času a dátumu kedy bola publikácia vytvorená, a taktiež kedy bola naposledy upravená. Pri potvrdzovaní žiadosti sa používa AJAXové volanie, po ktorého dokončení je na pár sekúnd zobrazená správa indikujúca úspešnosť danej požiadavky.

### 3.6.11 Žiadosti na povolenie prístupu k publikáciám

Pre pridelenie prístupu niektorému z používateľov, ktorí oň požiadali, musí autor použiť nástroj, ktorý je dostupný rovnako ako v prípade žiadostí na potvrdenie autorstva pomocou voľby v hlavnej ponuke. Tento nástroj sa od nástroja na potvrdenie autorstva vizuálne líši len v dátach, ktoré sú používateľovi zobrazené v tabuľke. V tomto prípade sa tabuľka skladá zo stĺpca obsahujúceho používateľské meno žiadateľa, názvu publikácie, zoznamu autorov a dátumu a času reprezentujúceho čas poslania žiadosti.

## 3.7 Výsledky a kontrola kvality

Pre kontrolu výslednej aplikácie sme sa rozhodli použiť funkčné, výkonnostné a záťažové testy. Tieto testy overujú nie len kvalitu a použiteľnosť, ale taktiež pomáhajú lepšie odhadnúť náročnosť aplikácie a stým spojený návrh konfigurácie pre uvedenie do prevádzky.

### 3.7.1 Funkčné testovanie

Vývoj aj napriek nepoužitiu plnohodnotnej agilnej metodiky prebiehal v krátkych iteráciách na konci ktorých sa k hlavnej vetve vždy pripájal otestovaný a funkčný kód zároveň priebežne prebiehalo definovanie a úprava funkcionalít.

Verzionovanie kódu bolo zabezpečené pomocou nástroja GIT. Pričom otestovaná a funkčná aplikácia sa vždy nachádzala v produkčnej vetve, zatiaľ čo vývoj prebiehal v úplne inej vetve. Samotné funkčné testovanie teda prebiehalo priebežne počas celej doby vývoja. Spätná väzba bola zbieraná formou osobných správ prípadne emailov. Všetky pripomienky sme sa snažili zapracovať vždy do nasledujúcej iterácie.

### 3.7.2 Výkonnostné a záťažové testovanie

Testovanie prebiehalo pomocou konzolovej aplikácie ApacheBench, ktorá dokáže podať predstavu o výkonnosti danej aplikácie v prevádzke na danom serveri. Výstupom je počet obslúžených požiadaviek a ich latencia.

V rámci webovej aplikácie bola na testovanie použitá úvodná stránka, ktorá je zároveň výpočtovo najzložitejšia. Pomocou ApacheBench bolo z osobného počítača pripojeného pomocou optickej siete s šírkou pásma 250/100Mbit na server s aplikáciou odoslaných 1000 požiadaviek. Testované boli rôzne počty konkurentných požiadaviek.

Pre potreby testovania bol zriadený virtuálny server s konfiguráciou uvedenou v tabuľke 1.

Tabuľka Použitá hardvérová konfigurácia virtuálneho servera

|  |  |
| --- | --- |
| Procesor: | 2x2.4GHz |
| Pamäť RAM: | 1.5GB |
| Softvérová konfigurácia: | Ubuntu 14.04  Apache/2.4.7  MariaDB 5.5.47  PHP 5.6.20 |

### 3.7.3 Výsledky testov

V rámci výkonnostného testovania sme robili tri testy s celkovým počtom 1000 požiadaviek pri 40, 20 a 5 konkurentných spojeniach.

Tabuľka Výsledky testov pri 40, 20 a 5 konkurentných spojeniach

|  |  |  |  |
| --- | --- | --- | --- |
| Počet požiadaviek (v %) | 40 konkurentných spojení | 20 konkurentných spojení | 5 konkurentných spojení |
| 50% | 2234ms | 1089ms | 275ms |
| 66% | 2388ms | 1144ms | 291ms |
| 90% | 2776ms | 1261ms | 330ms |
| 95% | 3153ms | 1330ms | 346ms |
| 98% | 3424ms | 1778ms | 375ms |
| 99% | 3569ms | 1848ms | 402ms |
| 100% | 3844ms | 2158ms | 473ms |

Podľa výsledkov uvedených v tabuľke 2 môžeme vidieť, že 50% požiadaviek pri 40 konkurentných spojeniach bolo obslúžených v čase približne 2,2 sekundy, pričom 90% požiadaviek v čase menšom ako 2,8 sekundy a najdlhšia požiadavka trvala približne 3,8 sekundy. Ďalšie výsledky testu ukázali že server dokázal spracovať približne 17 požiadaviek za sekundu.

V prípade testu s 20 konkurentnými spojeniami bol server schopný obslúžiť 50% požiadaviek v čase pod 1,1 sekundy, zároveň 90% požiadaviek obslúžil za menej ako 1,3 sekundy a najdlhšia požiadavka trvala približne 2,2 sekundy. V tomto prípade server dokázal za sekundu spracovať približne 18 požiadaviek.

Posledný test s 5 konkurentnými spojeniami ukázal, že 50% spojení bolo dokončených v čase pod 0,3 sekundy, 98% spojení pod 0,4 sekundy zatiaľ čo najdlhšia požiadavka trvala približne 0,47 sekundy. Aj v tomto prípade server podobne ako v prípade prvého a druhého testu dokázal za sekundu obslúžiť približne 18 požiadaviek.

Z vykonaných testov môžeme usúdiť, že nami zvolený virtuálny server by bol pri malej až strednej záťaži postačujúci, no pri vysokej záťaži by bolo čakanie na obslúženie požiadavky používateľsky nekomfortné. Pre urýchlenie aplikácie by bolo možné použiť proxy pre distribuovanie požiadaviek medzi viacero serverov, prípadne alokovať výkonnejší hardware.

# Záver

Náplňou našej práce bolo navrhnúť a vytvoriť webovú aplikáciu pre elektronické publikovanie publikácií, ktorá by umožnila autorom postupne zverejňovať jednotlivé kapitoly publikácií, sprístupňovať ich registrovaným čitateľom a získavať spätnú väzbu.

Pri vývoji sme použili v súčasnosti jedny z najpoužívanejších technológií pre vývoj webových aplikácií. Zároveň sme kládli dôraz na rozšíriteľnosť, jednoduchosť a udržateľnosť aplikácie čo sme docielili použitím viacerých návrhových vzorov a technológiami, ktoré sa neustále vyvíjajú a rozširujú svoje možnosti použitia.

Myslím, že sa nám úspešne podarilo navrhnúť a vytvoriť programové riešenie, ktoré umožnilo autorom efektívne a nenáročne publikovať publikácie, postupne zverejňovať ich jednotlivé kapitoly a získavať spätnú väzbu. Zároveň sa nám podarilo rovnako efektívne čitateľom umožniť k týmto publikáciám pristupovať. Systém sme navrhli tak, aby sme používateľom sprístupnili všetky dostupné nástroje pre publikovanie a čítanie v čo najkomplexnejšej no zároveň jednoduchej podobe.

Nakoľko vývoj samotného praktického výstupu prebiehal v zmysle agilných metodík, výsledné riešenie sme neustále podrobovali funkčným, výkonnostným a záťažovým testom. Napriek skutočnosti, že výsledná aplikácia nikdy nebola nasadená do ostrej prevádzky, plánujeme naďalej pracovať na jej vývoji pre sprístupnenie širšej škály nástrojov a funkcií pre vyhľadávanie, filtrovanie či čítanie používateľom.

Som presvedčený, že zadanie sa nám podarilo splniť.

# Zoznam bibliografických odkazov

BECK, K. 1999. Extreme Programming Explained. Addison-Wesley, 1999. 224 s. ISBN 978-0-201-61641-5.

Beck, K. 2002. Extrémní programování. Grada Publishing, 2002. 160 s.   
ISBN 80-247-0300-9.

BECK, K. a kol. 2001. Manifesto for Agile Software Development. [online]. Agile Alliance, 2001. [cit.:2016-03-01]. Dostupné na internete: <http://www.agilemanifesto.org/iso/sk/>

BECK, K. a kol. 2001. Principles behind the Agile Manifesto. [online]. Agile Alliance, 2001. [cit.:2016-03-01]. Dostupné na internete: <http://www.agilemanifesto.org/principles.html>

BECK, K. 2003. Test Driven Development: By Example. Pearson Education, 2003. 220 s. ISBN 0-321-14653-0

BECK, K. 2012. Why does Kent Beck refer to the "rediscovery" of test-driven development?. [online]. 2012. [cit. 2016-03-03]. Dostupné na internete: <https://www.quora.com/Why-does-Kent-Beck-refer-to-the-rediscovery-of-test-driven-development>

Centers for Medicare & Medicaid Services. 2008. Selecting a development approach. [online]. 2005. [cit.: 2016-03-01]. Dostupné na internete: <https://www.cms.gov/research-statistics-data-and-systems/cms-information-technology/xlc/downloads/selectingdevelopmentapproach.pdf>.

DOCTRINE PROJECT. 2016. [online]. 2016. [cit. 2016-04-17].  
Dostupné na internete: < http://www.doctrine-project.org/about.html >

FOWLER, M. 2004. Inversion of Control Containers and the Dependency Injection pattern. [online]. 2004. [cit. 2016-04-17]. Dostupné na internete:   
< http://www.martinfowler.com/articles/injection.html >

JURAS, M. 2014. Výběr vhodného šablonovacího systému pro webintegrační projekt na platformě PHP. [online]. 2014. [cit. 2016-04-17]. Dostupné na internete: <http://www.web-integration.info/cs/blog/vyber-vhodneho-sablonovaciho-systemu-pro-webintegracni-projekt-na-platforme-php/>

KADLEC, V. 2004. Agilní programování: metodiky efektivního vývoje softwaru. Computer Press, 2004. 278 s. ISBN 80-251-0342-0.

LAPLANTE,P.A. 2007. What every engineer should know about Software Engineering. CRC Press, 2007. 328 s. ISBN 978-0-8493-7228-5.

LARMAN, C. 2004. Agile and Iterative Development: A Manager’s Guide. Addison-Wesley, 2004. 368 s. ISBN 978-0-13-111155-4.

MARTIN, J. 1991. Rapid Application Development. Macmillan, 1991. 816 s. ISBN 0‑02‑376775‑8.

PICHLER, R. 2010. Agile Product Management with Scrum: Creating Products that Customers Love. Addison-Wesley, 2010. 160 s. ISBN 978-0-321-60578-8.

POPPENDIECK, M.; POPPENDIECK, T. 2003. Lean Software Development: An Agile Toolkit. Addison-Wesley, 2003. 240 s. ISBN 978-0-321-15078-3.

RAJIB, M. 2009. Fundamentals of software engineering. PHI Learning Pvt. Ltd., 2009. 464 s. ISBN 978-81-203-3819-7.

Shore, J. 2007. The Art of Agile Development. O’Reilly Media, 2007. 440 s. ISBN 978-0-5965-2767-9.

SCHWABER, K.; SUTHERLAND, J. 2014. The Scrum Guide. [online]. Scrum Alliance, 2014. [cit. 2016-03-04]. Dostupné na internete: <http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-US.pdf>.

SOMMERVILLE, I. 2007. What is software engineering?. Pearson Education, 2007. 840 s. ISBN 0-321-31379-8.

STATE UNIVERSITY OF NEW YORK. 2016. A survey of system development process models. [online]. 2016. [cit.: 2016-02-29]. Dostupné na internete: <https://www.ctg.albany.edu/publications/reports/survey\_of\_sysdev?chapter=9>.

TAKEUCHI, H.;NONAKA, I. 1986. New New Product Development Game. [online]. Harvard Business Review, 1986. [cit. 2016-03-04]. Dostupné na internete: <http://www.enterprisescrum.com/publications/The%20New%20New%20Product%20Development%20Game%20-%20Nonaka%20and%20Takeuchi.pdf>

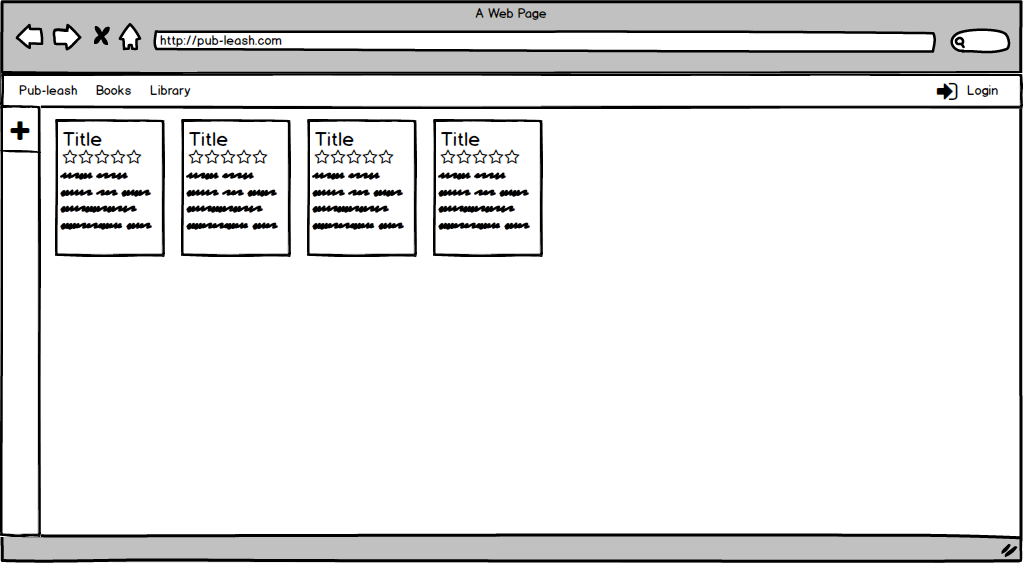
# Zoznam príloh

Príloha A – Návrhy obrazoviek aplikácie

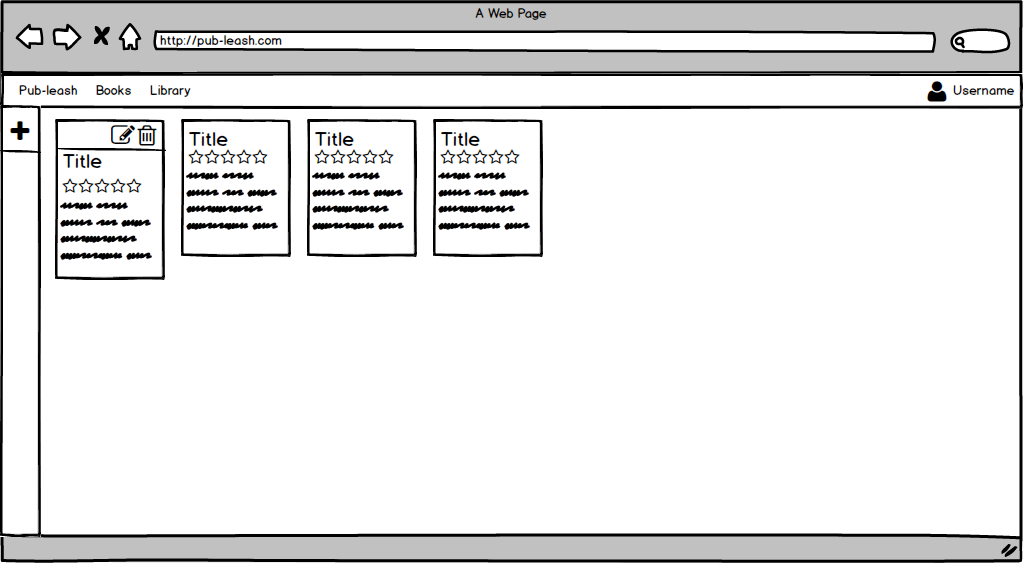
Príloha B – Ukážky aplikácie

Príloha C – CD obsahujúce ďalšie prílohy

Príloha A

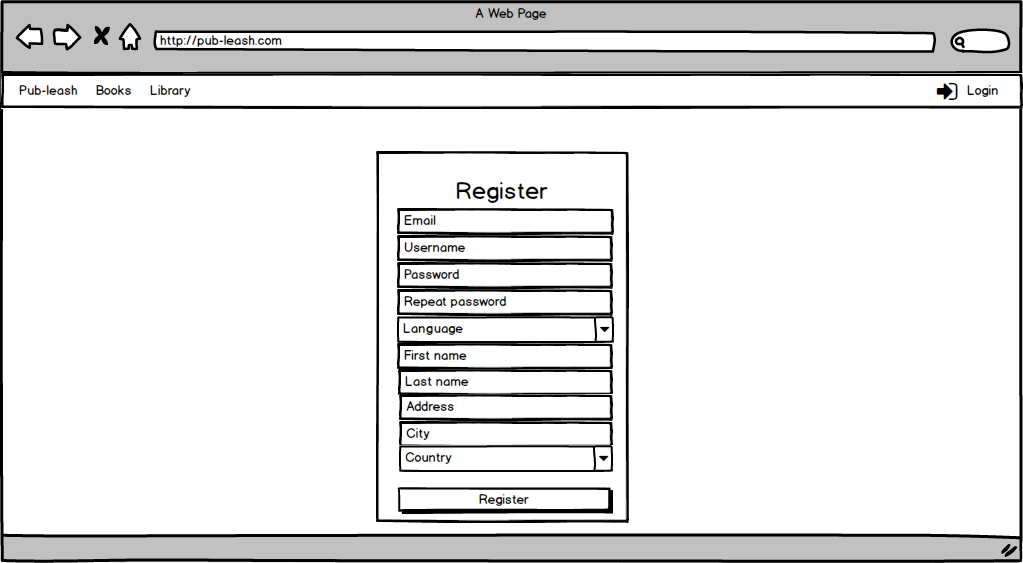
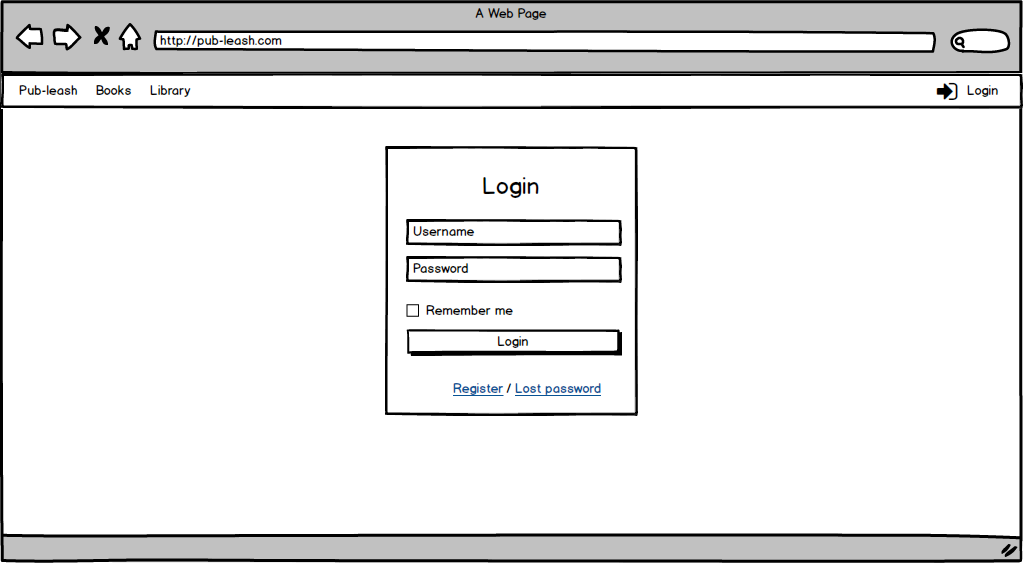


Príloha A1 Návrh hlavnej stránky pre anonymného používateľa



Príloha A2 Návrh hlavnej stránky pre prihláseného používateľa

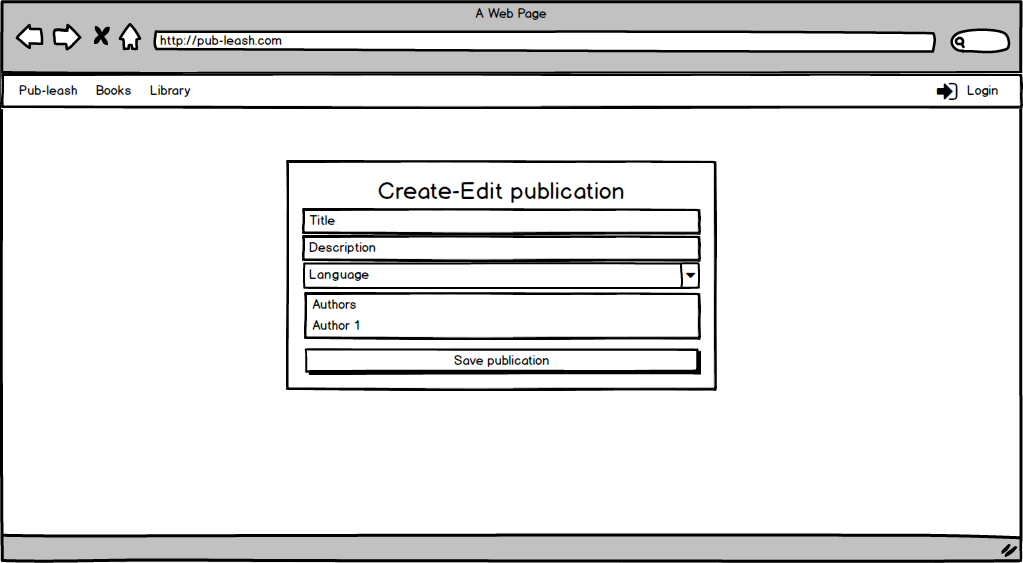
Príloha A3 Návrh prihlasovacieho formulára



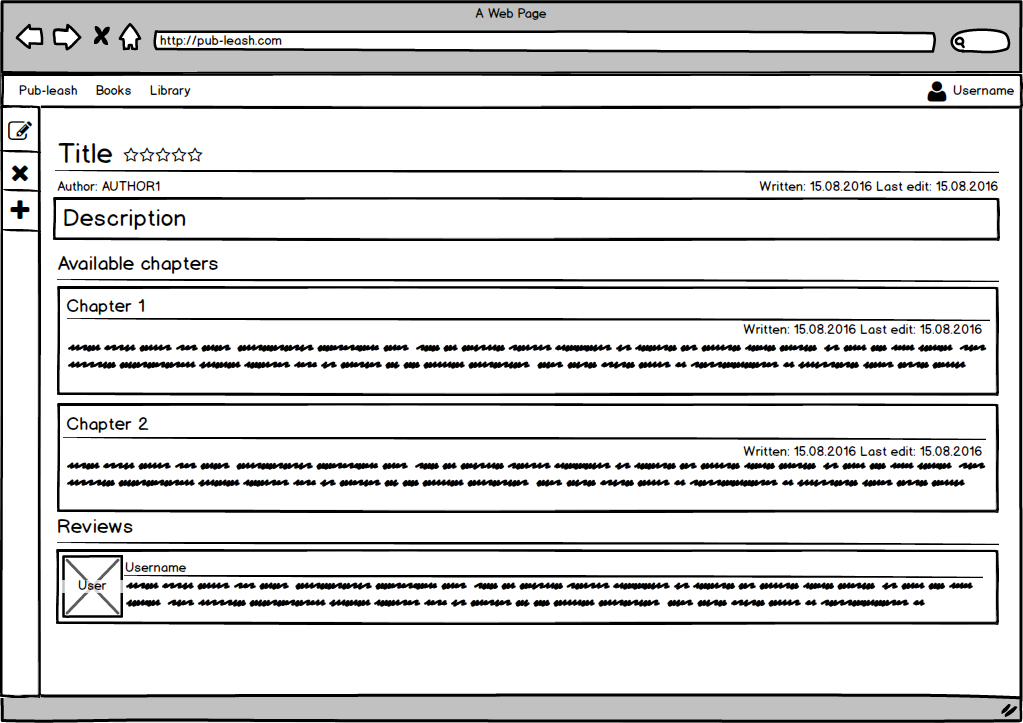
Príloha A4 Návrh registračného formulára



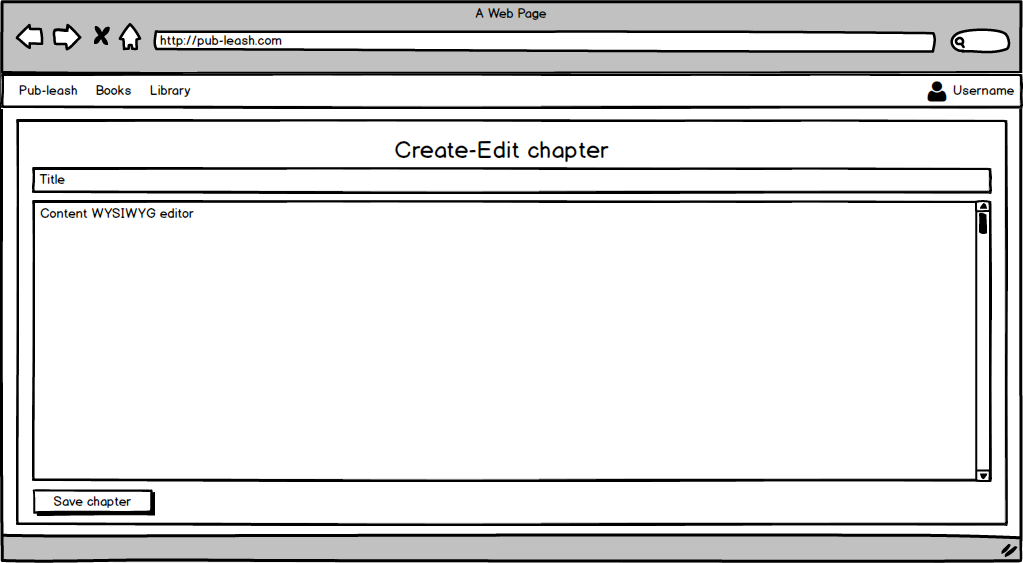
Príloha A5 Návrh formulára pre obnovenie hesla



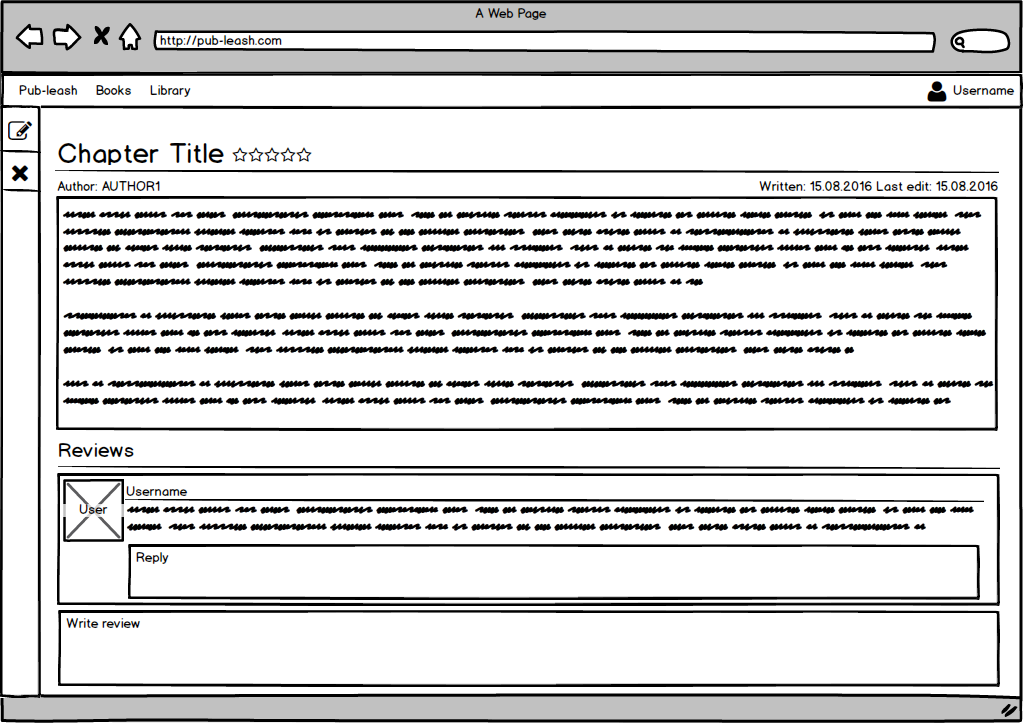
Príloha A6 Návrh formulára pre vytvorenie a editáciu kapitoly



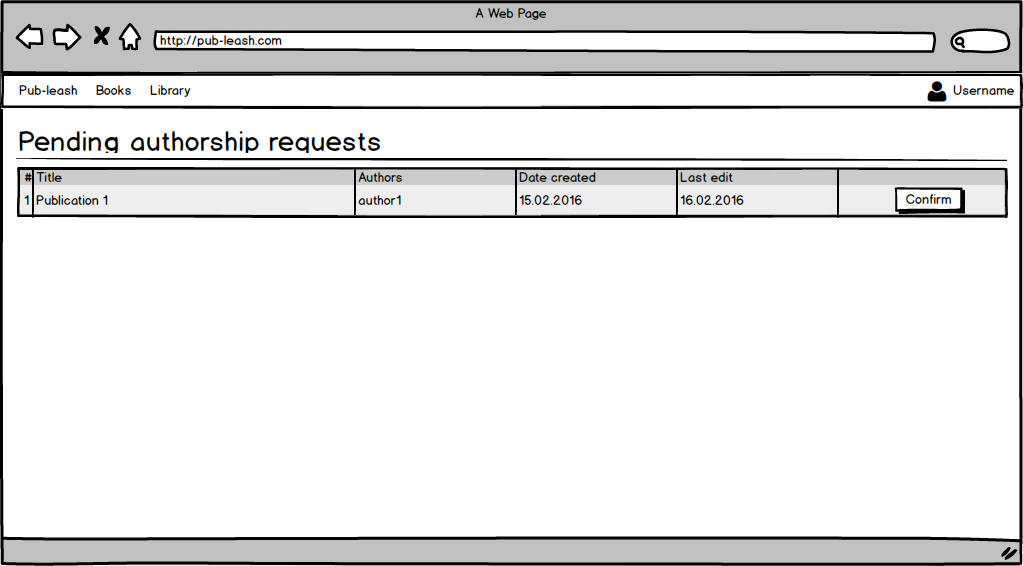
Príloha A7 Návrh obrazovky pre prehliadanie publikácie



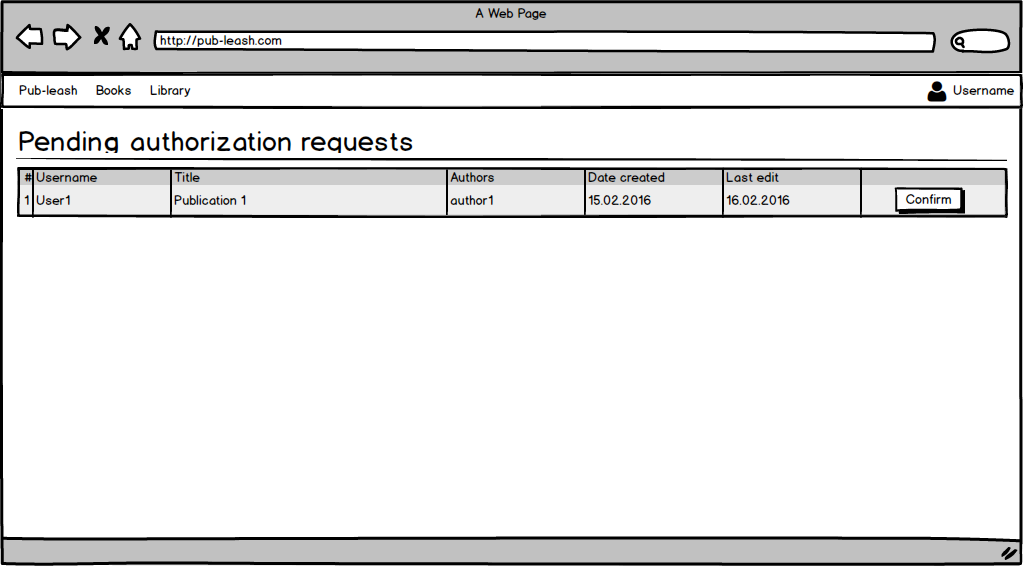
Príloha A8 Návrh formulára pre vytvorenie a editáciu kapitoly



Príloha A9 Návrh stránky pre čítanie kapitoly

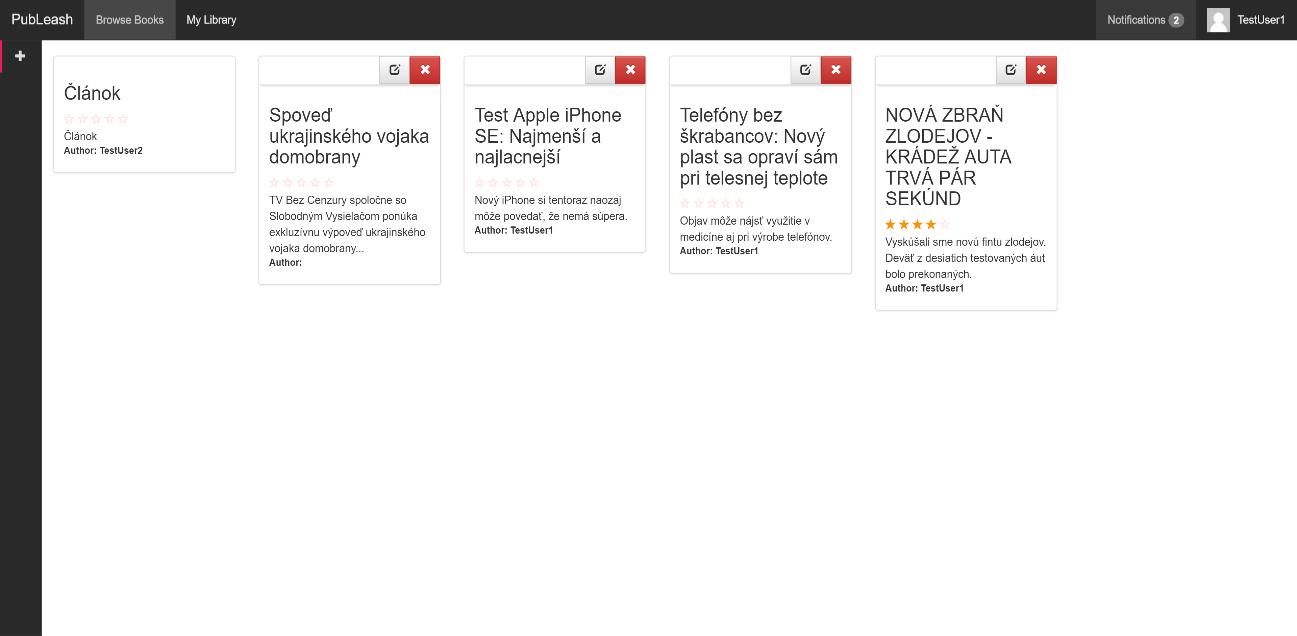


Príloha A10 Návrh stránky pre potvrdzovanie autorstva

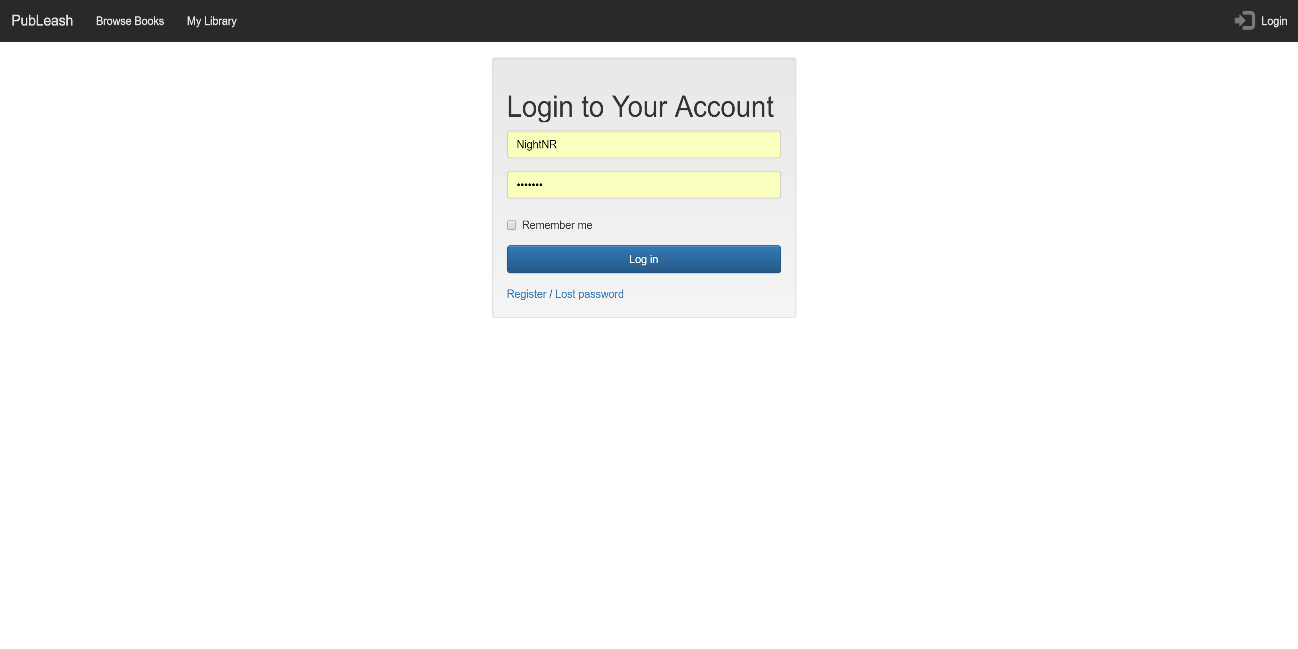


Príloha A11 Návrh obrazovky pre udeľovanie povolenia k čítaniu publikácie

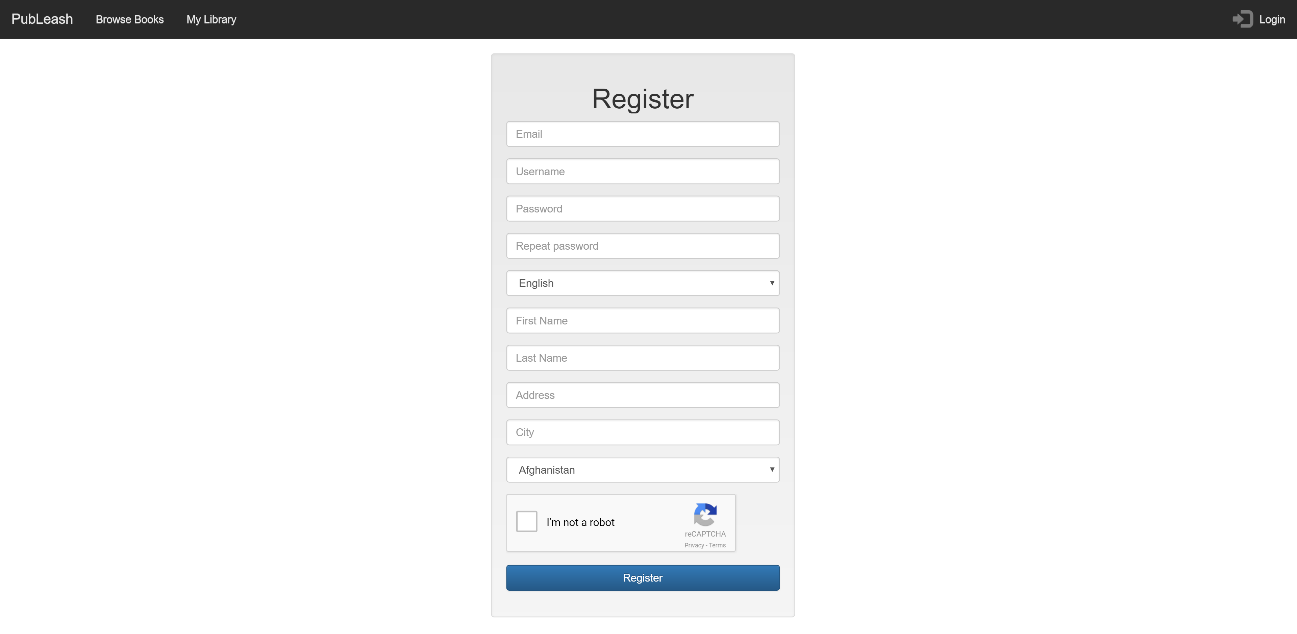
Príloha B



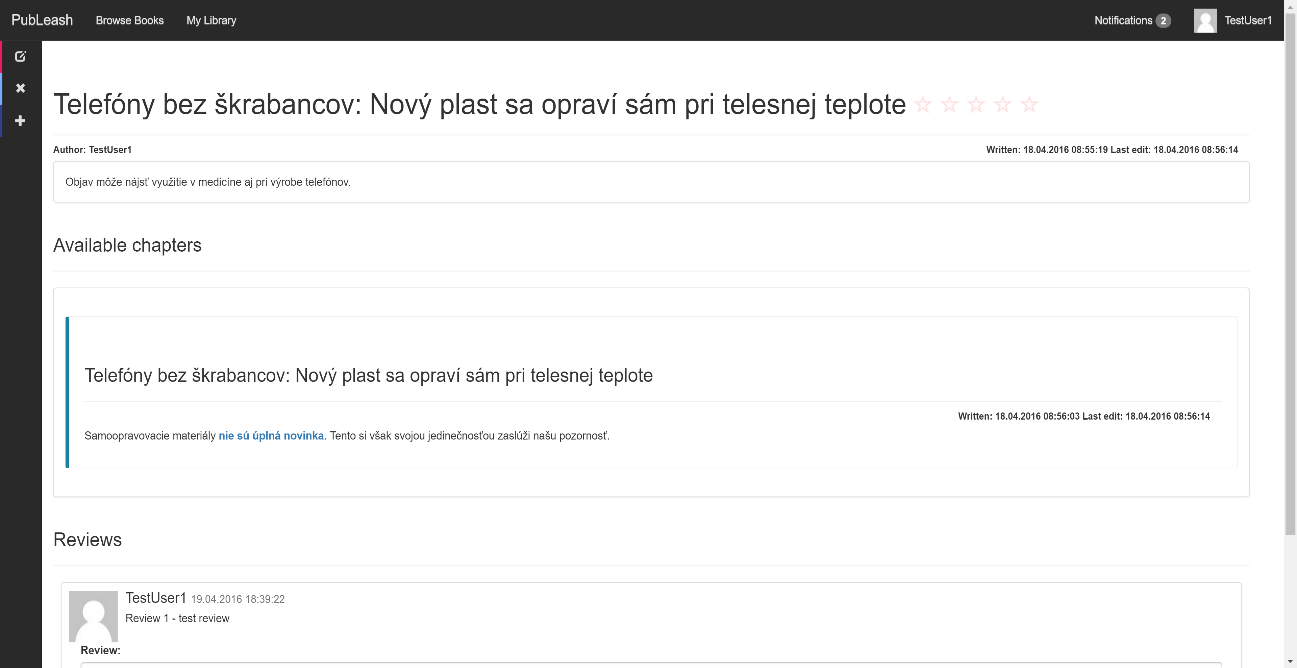
Príloha B1 Ukážka hlavnej obrazovky po prihlásení



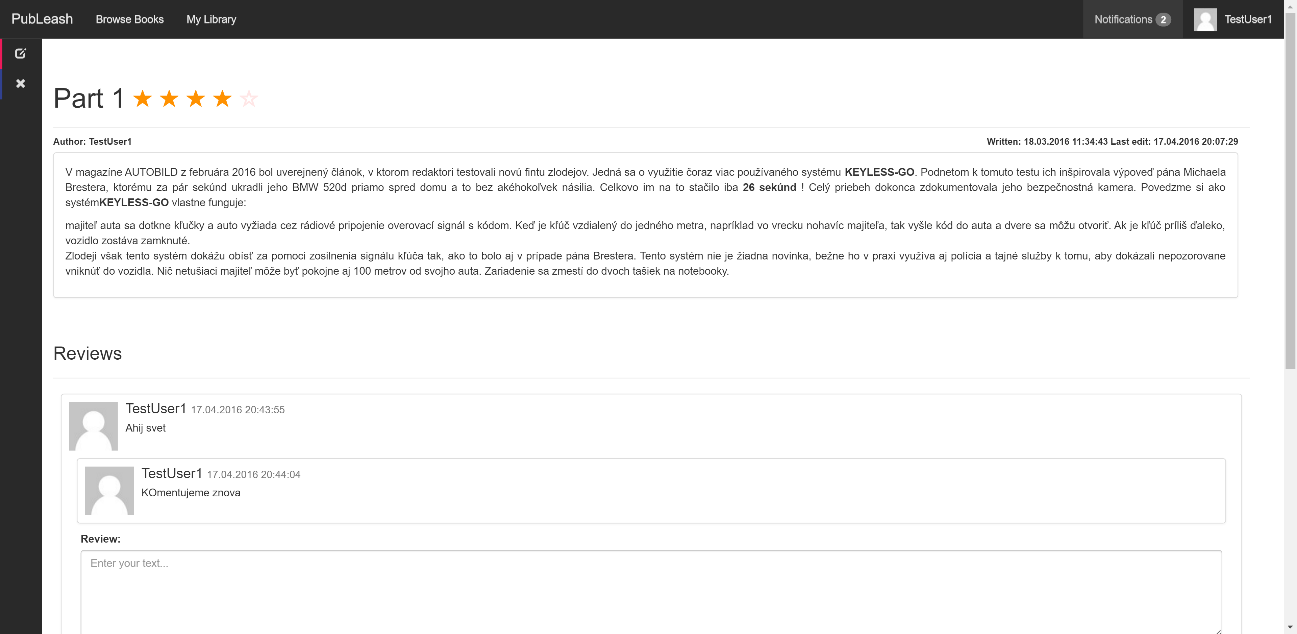
Príloha B2 Ukážka prihlasovacieho formulára



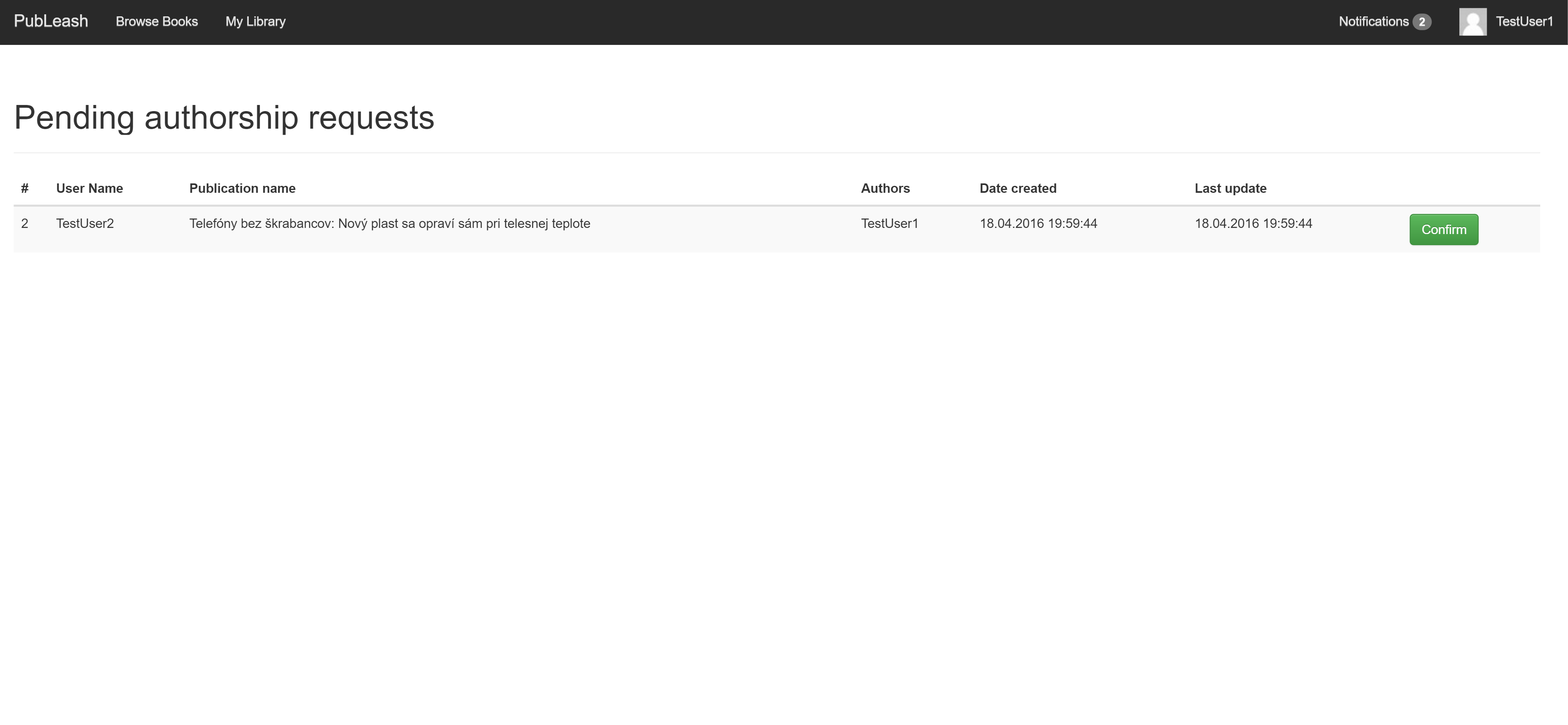
Príloha B3 Ukážka registračného formulára



Príloha B4 Ukážka obrazovky s publikáciou



Príloha B5 Ukážka obrazovky čítania kapitoly spolu s ukážkou spätnej väzby



Príloha B6 Ukážka obrazovky so žiadosťou o povolenie autorstva

Príloha C

CD Obsahujúce:

* zdrojový kód aplikácie,
* projekt návrhu obrazoviek v programe Balsamiq Mockups,
* projekt programu Enterprise Architect obsahujúci diagram prípadov použitia a diagram tried,
* projekt programu MySQL Workbench obsahujúci entitno-relačný diagram,
* túto prácu vo formáte PDF a docx.

1. Zdroj: https://embeddedsystemsvvce.files.wordpress.com/2013/02/waterfall-copy1.jpg?w=395 [↑](#footnote-ref-1)
2. Zdroj: https://upload.wikimedia.org/wikipedia/commons/thumb/e/ec/Spiral\_model\_(Boehm,\_1988).svg/1000px-Spiral\_model\_(Boehm,\_1988).svg.png [↑](#footnote-ref-2)
3. Zdroj: http://www.zenexmachina.com/assets/themes/zxm/images/page-artefacts/scrum-process-01.png [↑](#footnote-ref-3)
4. Zdroj: http://symfony.com/doc/current/book/http\_fundamentals.html [↑](#footnote-ref-4)
5. Zdroj: Agilné metodológie riadenia vývoja softvéru – Diplomová práca – Bc. Martin Kacvinský [↑](#footnote-ref-5)