

VILNIAUS KOLEGIJA  
ELEKTRONIKOS IR INFORMATIKOS FAKULTETAS



**LEISTA GINTI**

**Elektronikos ir informatikos fakulteto  
prodekanė**

\_\_\_\_\_ **dr. Loreta Savulionienė**

**2018 m. sausio mėn. 11 d.**

**INTERNETINĖ SKAITYKLA**

**BAIGIAMASIS DARBAS**

**BD 653I10001 PI14B**

**DIPLOMANTAS**

2018-01-11

**ROBERTAS CIŪNYS**

**VADOVAS**

2018-01-11

**DOC. DR. GENADIJUS  
KULVIETIS**

**RECENZENTAS (-Ė)**

2018-01-\_\_

VILNIAUS KOLEGIJA / UNIVERSITY OF APPLIED SCIENCES

FACULTY OF ELECTRONICS AND INFORMATICS



**AUTHORIZED BY**

**Vice Dean of Electronics and Informatics  
Faculty**

\_\_\_\_\_ **dr. Loreta Savulionienė**

**11 January, 2018**

## **ONLINE LIBRARY**

**FINAL PROJECT**

**BD 653I10001 PI14B**

**UNDERGRADUATE**

2018-01-11

**ROBERTAS CIŪNYS**

**SUPERVISOR**

2018-01-11

**DOC. DR. GENADIJUS  
KULVIETIS**

**REVIEWER**

2018-01-\_\_

Vilniaus kolegija

Elektronikos ir informatikos fakultetas

Programinės įrangos katedra

Valstybinis kodas: 653I10001

Data: 2018 – 01 – 11

Programų sistemų studijų programos baigiamojo darbo anotacija

Baigiamojo darbo tema: Internetinė skaitykla

Diplomantas: Robertas Ciūnys

Vadovas: Genadijus Kulvietis

Darbo apimtis - 61 p. teksto be priedų, 43 iliustr., 5 lent., 5 bibliografiniai šaltiniai, 1 priedai

### **Anotacija**

Baigiamajame bakauriniame darbe iškeliamą šiuometinė knygų skaitymo problema. Šiais laikais didžiąją laiko dalį daugelis naudotojų praleidžia internete ir dėl to lieka vis mažiau laisvo laiko arba noro skaityti knygas. Ir dėl to, nors popierinis variantas daugeliui vis dėlto yra patogesnis ir praktiškesnis, nuspręsta kurti internetinę skaityklą, kur prisijungę naudotojai galės skaityti, vertinti, komentuoti ar net įkelti savo knygas su galimybe išgarsėti. Taip pat naudotojui pateikiama patogi sąsaja, kur galima knygų sąrašę peržiūrėti knygos atributus, atlikti knygų paiešką ir filtravimą pagal nustatytus kriterijus. Neprisijungusiems naudotojams yra pateikiamas susipažinimas su sistema, o tai yra knygų sąrašas su visais parametrais, trumpas knygos aprašymas ir įvertinimas, kad būtų galimybė susidomėti ir užsiregistruoti sistemoje. Be to yra atskira administratoriaus sąsaja, kuri sudaro galimybę peržiūrėti visų naudotojų įkeltas knygas ir komentarus, ir jeigu kažkas neatitinka standartus, yra įžeidžiami kiti naudotojai ar tiesiog pateikiamas nekorektiškas turinys, administratorius savo nuožiūra gali redaguoti arba visiškai ištrinti tai iš sistemos. O, jeigu kažkuris naudotojas ir toliau kelia nekorektišką turinį arba yra neaktyvus, administratorius turi teisę tiesiog ištrinti jo sąsają, kad sistema būtų apsaugota nuo tolesnių negerovių.

Taigi pagrindinis tikslas yra leisti naudotojams laisvai skaityti ir įkelti savo knygas su galimybe išgarsėti, ir, kad visą tai galėtų daryti būnant įvairiose vietose ir bet koku laiku nevaržant savęs. Norimi rezultatai pasiekti ir naudotojai gali pilnavertiškai naudotis sistema.

Vilniaus kolegija / University of Applied Sciences  
Faculty of Electronics and Informatics  
Department of Software Development

State Code: 653E10005  
Date: 11 January, 2018

## Summary of the Final Project of Software Engineering Study Programme

Theme of the Final Project: Online Library

Undergraduate: Robertas Ciūnys

Supervisor: Genadijus Kulvietis

Volume of the work – 61 p. text without annexes, 43 pictures, 5 tables, 5 bibliographical entries, 1 annexes.

### Summary

The final thesis presents the problem of book reading in today's society. Most of the leisure time is usually spent on the internet and for that reason both the time and the will to read books is decreasing. Although for most people a paper book is more convenient and practical, I decided to make online library, where logged in users could read, rate, comment and even upload own books, with an opportunity to become famous. A user is also given a user-friendly GUI, where in a list users can review all the attributes of books, search for a certain book and filter them by given criteria. Users who are not logged in can take an introductory tour through the system, see a list of all books with their attributes, short book descriptions and book ratings to give users a chance to take an interest in the book library and register if they want. Furthermore, there is an admin interface, where an admin can review uploaded books and comments by all users and if something does not meet the standards, other users are offended or just given inappropriate content, an admin can according to his own discretion edit or simply delete it from the system. And if a user does not stop uploading content that is inappropriate or is not active admin has a right to delete his account, so that the system would be protected from further mischief.

All in all the main purpose of my system is to let users freely read and upload their own books with an opportunity for recognition and so it all could be done from various places and at any time without hassle. All pursued results are achieved and users can completely use this system.

# TURINYS

## 1. Turinys

2.	Įvadas .....	6
2.1.	Darbo tikslas .....	6
2.2.	Keliami uždaviniai .....	7
2.3.	Realizavimo priemonės .....	8
3.	Užduoties formuluotė .....	11
3.1.	Funkciniai reikalavimai .....	11
3.1.	Nefunkciniai reikalavimai .....	14
4.	Užduoties analizė .....	15
4.1.	Duomenų bazės lentelės .....	15
4.1.1	Naudotojas .....	15
4.1.2	Autorius .....	16
4.1.3	Skaitytojas .....	16
4.1.4	Knyga .....	17
4.1.5	Knygos būseną .....	17
4.1.6	Knygos atributai .....	18
4.2.	Klasių diagrama .....	18
4.3.	Esybių ryšių diagrama .....	20
4.4.	Panaudos atvejų diagramos .....	23
4.5.	Pagrindinio ir alternatyvaus scenarijaus lentelės bei veiklos diagramos .....	27
5.	Programinė realizacija .....	32
5.1.	Symfony karkaso įdiegimas ir konfigūravimas .....	33
5.1.1	Projekto sukūrimas .....	33
5.1.2	Konfigūracija .....	35
5.1.3	Verslo logikos organizavimas .....	35
5.1.4	Valdikliai .....	37
5.1.5	Šablonai .....	39
5.1.6	Formos .....	42
5.1.7	Apsauga .....	44
5.1.8	Interneto aktyvai .....	44
5.2.	Naudotojas .....	46
5.3.	Knyga .....	52
5.4.	Media .....	54
5.5.	Notes .....	55
6.	Naudotojo instrukcija .....	56

## **2. Įvadas**

Šiais laikais didžiąją laiko dalį daugelis naudotojų praleidžia internete ir dėlto lieka vis mažiau laisvo laiko arba noro skaityti knygas. Internetas pasiūlo patrauklesnius pasiūlymus naudotojams, kur beveik nereikalaujama ar visai nereikalaujama naudotojo pastangų ir viskas pateikiama „ant lėkštutės“. Tai gali būti filmai, serialai, visokios apžvalgos, naujienos, kur naudotojams yra tiesiog kišama informaciją ir viskas yra priimama už gryną pinigą. Su knygomis yra kitaip – žmogus skaito, gauna kažkokios informacijos ir jau po to pats kiekvienas individualiai apgalvoja, ką jis perskaitė, kokias gali padaryti iš to išvadas ir pasiimti sau tik tai ką pats nori. Taigi esant kuo daugiau galimybių šiuolaikiniams žmonėms skaityti knygas – šiuo atveju sukuriant internetinę skaityklą, galima bus nepaliekant savo komforto zonos atnaujinti, atrasti iš naujo savo seną skaitymo pomėgį arba atrasti naują, iki šiol nepatirtą jausmą, žmonės praturtės žiniomis, jutimais, kurių negautų iš šiuolaikinės žiniasklaidos, kuri didžiąją laiko dalį tiesiog skleidžia propagandą. Taip pat yra padidinama prieiga prie knygų, yra praplečiamos galimybės tiems, kurie galbūt negali gauti popierinio varianto, neturi arti gyvenamosios vietos bibliotekų ir net jeigu biblioteka yra arti, joje gali nebūt norimos knygos arba ją tuo metu gali skaityti kitas žmogus, o internetinėje skaitykloje tą pačią knygą gali skaityti neribojamas kiekis žmonių. Ir taip pat kadangi dabartiniu metu visi turi su savimi išmaniuosius įrenginius nereikėtų galvoti, apie knygų nešiojimą su savimi, nereikėtų nešioti papildomo svorio ir nebūtų taip, kad yra pamirštama kokia nors reikiama knyga, kaip pavyzdį galima pateikti mokyklų vadovėlius ir panašiai. Ypač tai matoma pradinėse klasėse, kur mokiniai ant nugaros tempiasi tokį svorį nuo vadovėlių ir kitų reikmenų, kad patys vos paeina ir jiems tai tikrai ne į naudą iš sveikatos pusės. Taigi internetinis knygų variantas galėtų sutvarkyti ir šią, šiuo metu gan aktualią, problemą.

### **2.1. Darbo tikslas**

Darbo tikslas yra sukurti sistemą leidžiančią naudotojams skaityti kitų autorių publikuotus kūrinius – internetinę skaityklą. O kadangi galima bus ne tik skaityti, bet ir įkelti savo kūrinius, tai sukurtoji sistema sudarys sąlygas visuotinai nežinomiems autoriams platesnes galimybes greičiau išgarsėti.

## **2.2. Keliami uždaviniai**

Darbo uždaviniai:

- suteikti naudotojams galimybę skaityti bet kokią sistemoje esamą knygą;
- knygų paieška pagal nustatytą kriterijų;
- įkelti savo knygą PDF formatu;
- komentuoti ir vertinti norimas knygas;

### **2.3. Realizavimo priemonės**

Internetinė skaitykla bus kuriama su PHP programavimo kalba naudojant „Symfony“ karkasą. Kodui rašyti bus naudojama PhpStorm programa, bus naudojama MySQL duomenų bazė, bus palaikomas lokalus serveris naudojant XAMPP programą.

Visą tai buvo pasirinkta, dėl to, kad šiuo metu dirbu įmonėje, kuri naudoja aukščiau išvardintas priemones ir jau turiu patirties su jomis. Taip pat man pačiam patinka su jomis dirbti.



## PROGRAMŲ SISTEMŲ (653I10001) STUDIJŲ PROGRAMOS STUDIJŲ REZULTATAI

Bendrosios kompetencijos		Studijų programos rezultatai (Studentas turėtų gebėti)	
1.	Bendravimo: gebėjimas bendrauti žodžiu ir raštu gimtąja (lietuvių) ir užsienio kalbomis, su žmonėmis, kurie nėra profesinės srities ekspertai.	1.1	Gebėti bendrauti ir bendradarbiauti valstybine ir užsienio kalbomis.
		1.2	Gebėti aiškiai perteikti profesines žinias.
2.	Tarpkultūrinė: gebėjimas dirbti daugiakultūroje aplinkoje, bendrauti ir bendradarbiauti su skirtingų kultūrų atstovais.	2.1	Gebėti dirbti daugiakultūroje aplinkoje.
		2.2	Gebėti elgtis etiškai ir profesionaliai.
3.	Iniciatyvumo ir verslumo: gebėti priimti sprendimus, laikytis lygių galimybių ir tolerancijos principo, vertinti ir užtikrinti darbo kokybę, prisitaikyti prie naujovių versle ir darbe.	3.1	Gebėti įvertinti rinkos pokyčius ir priimti sprendimus.
		3.2	Gebėti taikyti kokybės valdymo būdus.
		3.3	Gebėti savarankiškai studijuoti, tobulinti savo žinias, kelti kvalifikaciją
Dalykinės kompetencijos		Studijų programos rezultatai (Studentas turėtų gebėti)	
4.	Programinių sprendimų kūrimas	4.1	Gebėti analizuoti ir projektuoti duomenų struktūras.
		4.2	Gebėti projektuoti ir įgyvendinti algoritmus pasirinktomis programinėmis priemonėmis.
		4.3	Gebėti parinkti ir taikyti programinių sprendimų testavimo metodus.
		4.4	Gebėti parengti programinių sprendimų naudotojo ir techninę dokumentaciją.
		4.5	Gebėti projektuoti, kurti ir tobulinti duomenų bazes.
		4.6	Gebėti projektuoti, kurti ir tobulinti naudotojo sąsajas.
		4.7	Gebėti projektuoti, kurti naujus ir tobulinti esamus programinius sprendimus.
		4.8	Gebėti analizuoti, parinkti ir pritaikyti duomenų saugos sprendimus.
5.		5.1	Gebėti kurti ir tobulinti duomenų apdorojimo sprendimus.

	Duomenų bazių kūrimas ir valdymas (specializacija - Duomenų bazių sistemos).	<b>5.2</b>	Gebėti administruoti duomenų bazių valdymo sistemas.
<b>6.</b>	Internetinių paslaugų kūrimas ir valdymas (specializacija - Internetinės technologijos).	<b>6.1</b>	Gebėti kurti ir tobulinti internetinių paslaugų sprendimus.
		<b>6.2</b>	Gebėti administruoti internetinių paslaugų sistemas.

### **3. Užduoties formuluotė**

#### **3.1. Funkciniai reikalavimai**

Sistemos naudotojas galės būti trijų lygių: neregistruotas naudotojas galės skaityti trumpas bet kurių kūrinių ištraukas, registruotas naudotojas galės skaityti ir įkelti savo turinį ir sistemos administratorius saugos sistemą nuo nelegalaus ir nekorektiško turinio.

##### **1) Neregistruotas naudotojo funkcijos:**

- Naudotojas gali peržiūrėti visų įkeltų knygų sąrašą
- Jeigu naudotojas bando pridėti savo knygą, skaityti knygą, vertinti knygą, komentuoti knygą jam tai yra neleidžiama, nes jis neturi tam teisių
- Taip pat jam neleidžiama redaguoti jokių knygų
- Naudotojas gali peržiūrėti trumpą knygos aprašymą, sąrašė paspaudęs atitinkamą vietą
- Naudotojas gali filtruoti knygas, paspaudus ant norimo stulpelio, mažėjimo ir didėjimo tvarka
- Naudotojas gali ieškoti norimos knygos įvedus jos pavadinimą į paieškos lauką
- Naudotojas gali skaityti komentarus prie norimos knygos, paspaudus atitinkamą paveiksluką
- Naudotojas gali matyti knygos įvertinimą
- Naudotojas gali rūšiuoti knygas didėjimo arba mažėjimo tvarka

##### **2) Registruotas naudotojas galės:**

- Naudotojas gali užsiregistruoti užpildydamas reikiamus laukus: prisijungimo vardą, elektroninį paštą, vardą, pavardę, gimimo datą ir slaptažodį
- Jeigu kažkas yra įvesta neteisingai arba yra praleidžiamas koks nors laukas, sistema praneša apie tai naudotojui ir jis gali užpildyti neteisingai įvestus laukus
- Užsiregistravęs naudotojas gali prisijungti į sistemą
- Jeigu įvestas neteisingas prisijungimo vardas arba slaptažodis, sistema praneša apie tai naudotojui ir jis gali jungtis iš naujo
- Prisijungus naudotojas gali peržiūrėti knygų sąrašą
- Gali pridėti naują knygą
- Jeigu kažkas yra įvesta neteisingai arba yra praleidžiamas koks nors laukas, sistema praneša apie tai naudotojui ir jis gali užpildyti neteisingai įvestus laukus
- Jeigu įkeliamą knygą viršija nustatytą duomenų dydžio limitą, sistema neleidžia jos įkelti ir naudotojo yra prašoma bandyti įkelti iš naujo

- Taip pat jeigu įkelta knyga arba jos aprašas yra nekorektiški administratorius gali ištrinti tą knygą
- Naudotojas gali komentuoti knygas
- Jeigu komentarai yra korektiški, neižeidžiantys tie komentarai lieka, o jeigu pažeistos kažkurios taisyklės administratorius gali juos ištrinti
- Naudotojas gali vertinti knygas paspaudus vertinimo lauką
- Naudotojas gali redaguoti savo įkeltas knygas
- Tačiau jeigu bando redaguoti kitų įkeltas, sistema praneša, kad jis neturi tam teisių
- Naudotojas gali skaityti pasirinktą knygą
- Naudotojas gali peržiūrėti trumpą knygos aprašymą, sąrašą paspaudęs atitinkamą vietą
- Naudotojas gali filtruoti knygas, paspaudus ant norimo stulpelio, mažėjimo ir didėjimo tvarka
- Naudotojas gali ieškoti norimos knygos įvedus jos pavadinimą į paieškos lauką
- Naudotojas gali skaityti komentarus prie norimos knygos, paspaudus atitinkamą paveiksluką
- Naudotojas gali matyti knygos įvertinimą
- Naudotojas gali vertinti knygas
- Naudotojas gali tęsti skaityti norimą knygą, ten kur sustojo praeitą kartą sąrašą paspausdamas ant puslapio numerio esančio prie tos knygos
- Naudotojas gali rūšiuoti knygas didėjimo arba mažėjimo tvarka

### 3) Administratorius galės:

- Matyti kiek naudotojų, skaito atitinkamą knygą
- Administratorius gali prisijungti į sistemą
- Jeigu įvestas neteisingas prisijungimo vardas arba slaptažodis, sistema praneša apie tai administratoriui ir jis gali jungtis iš naujo
- Gali peržiūrėti knygų sąrašą
- Gali pridėti naują knygą
- Gali komentuoti knygas
- Gali vertinti knygas
- Jis gali ir skaityti ir redaguoti visas knygas
- Taip pat gali trinti knygas
- Gali trinti naudotojus
- Gali atšaukti naudotojų skaitomas knygas
- Gali filtruoti knygas, paspaudus ant norimo stulpelio, mažėjimo ir didėjimo tvarka

- Gali rūšiuoti knygas didėjimo arba mažėjimo tvarka
- Gali tęsti skaityti norimą knygą, ten kur sustojo praeitą kartą sąrašė paspausdamas ant puslapio numerio esančio prie tos knygos
- Vienintelis dalykas, kurio negali administratorius, tai ištrinti save

### **3.1. Nefunkciniai reikalavimai**

Taigi nefunkciniai reikalavimai:

- Sistema programuojama PHP kalba
- Naudojama programa PhpStorm
- Naudojamas Symfony karkasas
- Naudojamas lokalus XAMPP serveris
- Sistemos duomenų bazė MySQL
- Naudotojo kompiuteris turi turėti interneto ryšį
- Sistema prieinama per interneto naršyklę
- Palaikomos naršyklės: Mozilla Firefox, Chrome, Microsoft Edge

## **4. Užduoties analizė**

### **4.1. Duomenų bazės lentelės**

#### **4.1.1 Naudotojas**

Naudotojo lentelės laukai:

- Id
- Vardas
- Pavardė
- Prisijungimo vardas
- Elektroninis paštas
- Slaptažodis
- Gimimo data
- Rolė

Vardas – naudotojo vardas, kuris bus rodomas jo paskyroje

Pavardė – naudotojo pavardė, kuri bus rodomas jo paskyroje

Elektroninis paštas – šis laukas skirtas tam, kad kilus neaiškumams galima būtų susisiekti su naudotoju arba pranešti kažkokias žinias, naujienas

Prisijungimo vardas – kažkoks sugalvotas vardas, kurį naudotojas turės įvesti prisijungiant

Slaptažodis – kažkoks sugalvotas slaptažodis, kurį naudotojas turės įvesti prisijungiant

Gimimo data – laukas, kuris parodys koks naudotojo amžius ir pagal tai gali būti kažkokie apribojimai, pavyzdžiui administratorius matydamas knygą, skirtą suaugusiems, kurią skaito nepilnametis galės pašalinti tą knygą iš naudotojo skaitomų knygų sąrašo

Rolė – bus nurodoma ar jis yra neregistruotas, prisijungęs naudotojas ar administratorius. Šiuo atveju administratorius bus tik vienas, tai keisis tik prisijungusio ir neprisijungusio naudotojo teisės.

#### **4.1.2 Autorius**

Autoriaus lentelės laukai:

- Id
- Vardas
- Pavardė
- Gimimo data
- Patirtis
- Įkeltų knygų skaičius

Patirtis – bus matoma kiekvieno autoriaus patirtis, tam, kad, jeigu įkelta knyga yra parašyta jo paties, naudotojai galėtų suprast, ko gali tikėtis, pagal tokią patirtį. Registruojantis bus pasirinkimas iš tokių: pradedantysis, patyręs ir profesionalas. Nors žmonės galės pasirinkti bet ką, nepriklausant nuo to, kaip yra iš tikrųjų, vis tiek tai padės naudotojams labiau orientuotis.

Įkeltų knygų skaičius – bus matoma kiekvieno autoriaus įkeltų knygų skaičius kaip dar vienas jo patirties rodiklis ir taip pat bus matoma ar autorius yra aktyvus.

#### **4.1.3 Skaitytojas**

Skaitytojo lentelės laukai:

- Id
- Vardas
- Pavardė
- Gimimo data
- Skaitomų knygų skaičius

Skaitomų knygų skaičius – bus matoma kiekvieno skaitytojo skaitomų knygų skaičius, kad galima būtų pamatyti aktyviausius skaitytojus ir tuos, kurie užsiregistravę, bet nesinaudoja sistema.



#### 4.1.4 Knyga

Knygos lentelės laukai:

- Id
- Pavadinimas
- Autorius
- Puslapių skaičius
- Išleidimo metai
- Kalba
- Žanras
- Leidykla – naujas laukas
- ISBN

Pavadinimas – knygos pavadinimas

Autorius – knygos autorius

Puslapių skaičius – knygos puslapių skaičius. Gali būti, kad naudotojas nenorės skaityti daug puslapių turinčios knygos, tai galės pagal tai iškart atsirinkti.

Išleidimo metai – data, kada knyga buvo išleista

Kalba – kalba, kuria yra parašyta įkelta knyga. Tai padės suprast naudotojams ar knyga parašyta kalba, kuria jie supranta arba tiesiog yra patogiau skaityti

Žanras – bus matomas knygos žanras, kad naudotojai galėtų pagal tai atsirinkti savo mėgstamo žanro knygas ir nereikėtų veltui peržiūrėti visų knygų aprašymo

Leidykla – pridėtas neprivalomas laukas, kur, jeigu įkeliamą knygą yra išleista leidykla, užpildomas jos pavadinimas, o, jeigu išleidžia pats autorius – laukas paliekamas tuščias.

ISBN - tarptautinio standarto knygos numeris - unikalus knygos, skirtos naudoti komerciniais tikslais, numeris

#### 4.1.5 Knygos būseną

Knygos būsenos lentelės laukai:

- Id

- Įvertinimas
- Paskutinis skaitytas puslapis

Įvertinimas – kiekvienas žmogus galės įvertinti knygą nuo 1 iki 10 ir po to pagal visus įvertinimus bus išvedamas vidurkis.

Paskutinis skaitytas puslapis - prie tų knygų, kurias skaitė naudotojas, bus nurodomas paskutinis skaitytas puslapio numeris

#### **4.1.6 Knygos atributai**

Knygos atributų lentelės laukai:

- Id
- Aprašymas
- Komentarai
- Skaitytojų skaičius
- Knygos tipas

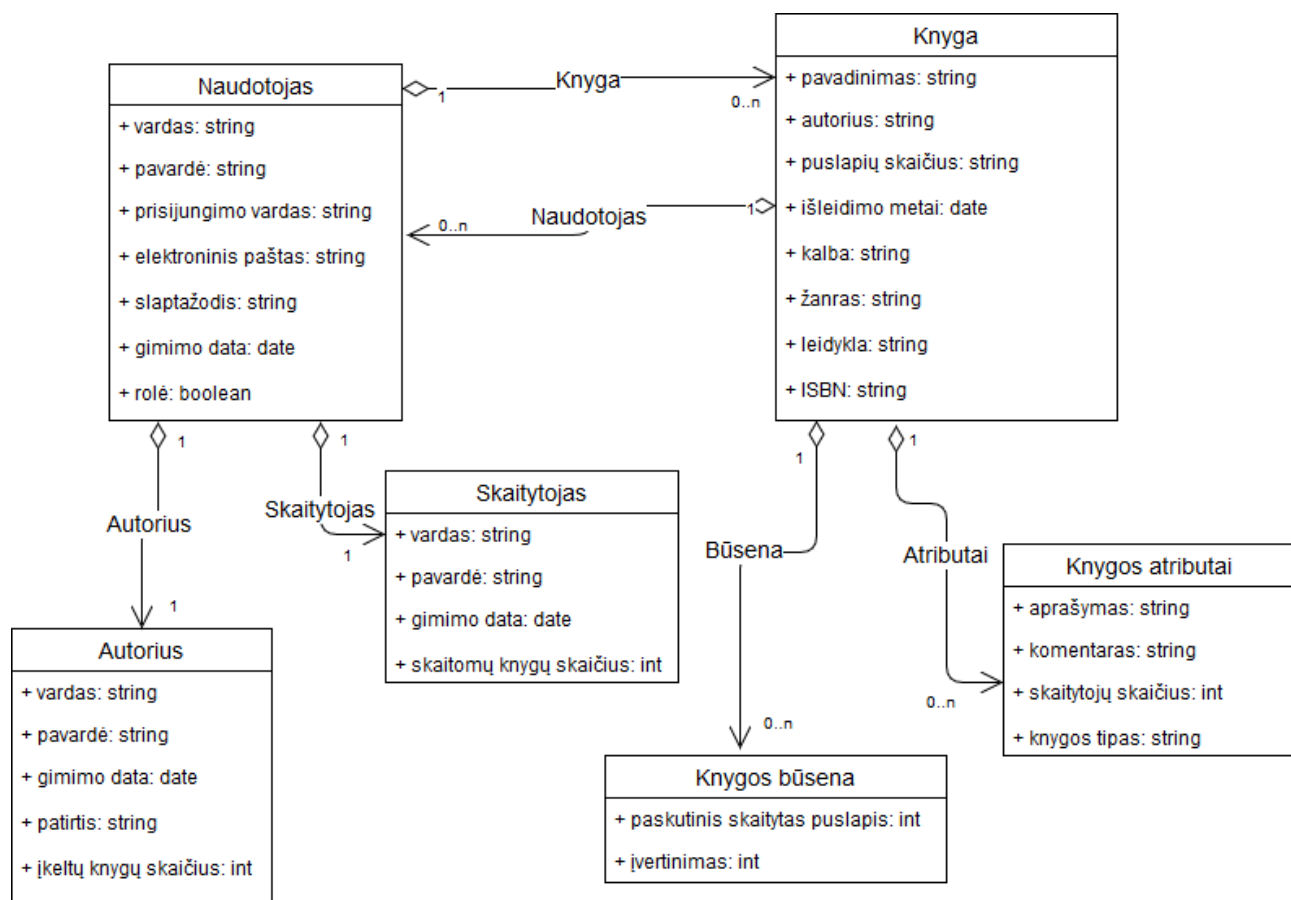
Aprašymas – bus pridėtas trumpas knygos aprašymas, kuris bendrai supažindins naudotojus apie ką ta knyga yra. Tai yra laukas, kuris bus matomas ir neprisijungusiems naudotojams, taigi pagal aprašymus žmogus gali suprasti ar jam verta prisijungti į sistemą ir skaityti esamas knygas.

Komentaras – galima bus prie kiekvienos knygos sąrašė pridėti komentarą, kurių kiekis nebus ribojamas. Tai galės būti ir knygos įvertinimas ir jos apžvalga arba šiaip pasiūlymai žmonėms ar verta skaityti šią knygą pateikus panašių knygų pavyzdžius. Tai kitas laukas, kuris padės apsispręsti neprisijungusiems naudotojams. Komentarų skaičius – bus skaičiuojama kiek komentarų prie kiekvienos knygos buvo pridėta, taip parodant labiausias aptarinėjamas knygas, o gal ir jų kokybę, gerumą.

Skaitytojų skaičius - prie kiekvienos knygos bus rodoma, kiek žmonių šiuo metu skaito tą knygą.

## **4.2. Duomenų struktūros diagrama**

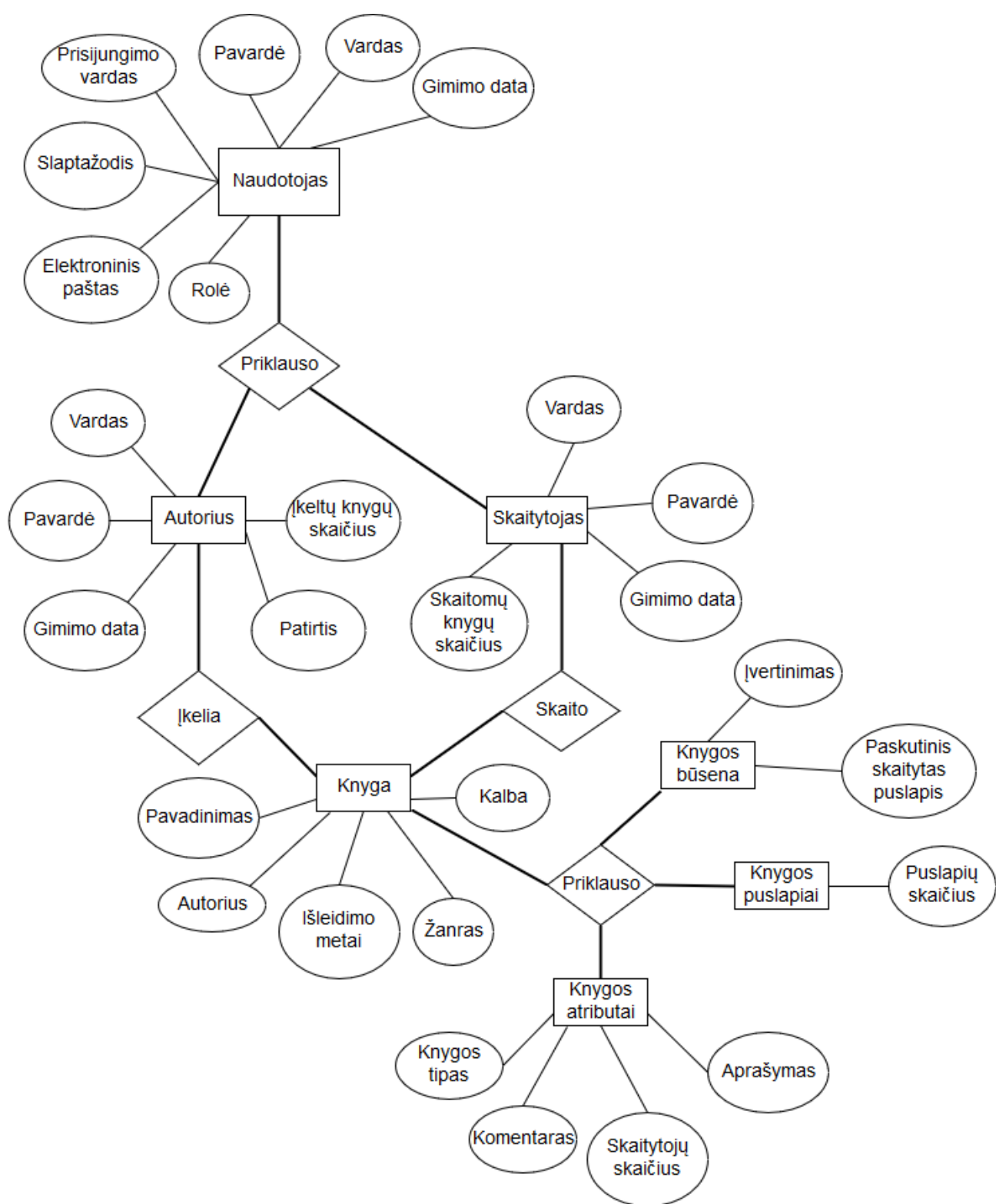
Nurodau duomenų struktūros klasės diagramą, pagal kurią buvo pridėti laukai į duomenų bazę (1pav.):



1pav. Duomenų struktūros diagrama

Kaip matome čia pavaizduoti ryšiai, kaip šios duomenų bazės lentelės tarpusavyje jungsis. Taigi Naudotojas su Knyga jungiasi ryšiu ‘One-to-many’, kas reiškia, kad tas pats naudotojas galės turėti daug knygų, o Knyga su Naudotoju taip pat jungiasi ryšiu ‘One-to-many’, nes tą pati knygą taip pat gali turėti daug naudotojų. Knyga jungiasi ryšiu ‘One-to-many’ su knygos puslapių, knygos atributų ir knygos būsenos lentelėmis, nes ta pati knyga turi daug puslapių, gali turėti daug skirtingų įvertinimų ir turėti daug komentarų, skaitytojų. Naudotojas jungiasi ryšiu ‘One-to-one’ su autoriumi ir skaitytoju, nes tas pats naudotojas gali būti tik vienas autorius ir vienas skaitytojas.

### 4.3. Esybių ryšių diagrama



2 pav. Esybių ryšių diagrama

Čia įkelta esybių ryšių diagrama (2pav.), kur matomi naudotojo ir knygos atributai. Išskiriu naudotoją į autorių ir skaitytoją ir toliau pagal tai skirstau į įkeliamą ir skaitomą knygą. Nors ir autorius ir skaitytojas gali skaityti knygas, aš pateikiu atskirai, kad būtų aiškiau, bet kadangi knyga yra susijusi bendrai su visais naudotojais, tai iš to aišku, kad ir autorius gali skaityti knygą. Taigi iš pradžių pateikiu bendrai naudotoją su atributais, kurie turi būti pateikiami registruojantis arba prisijungiant:

Naudotojas:

- Vardas
- Pavardė
- Gimimo data
- Prisijungimo vardas
- Slaptažodis
- Elektroninis paštas
- Rolė

Toliau naudotoją skirstau į autorių ir skaitytoją:

Autorius:

- Vardas
- Pavardė
- Gimimo data
- Patirtis
- Įkeltų knygų skaičius

Skaitytojas:

- Vardas
- Pavardė
- Gimimo data
- Skaitomų knygų skaičius

Tada atskirai pateikiu, kad autorius įkelia, o skaitytojas skaito knygas, tačiau viskas sueina į tą patį knygos lauką, kuris toliau plečiasi į kitus. Taigi knygos pagrindiniai atributai:

Knyga:

- Pavadinimas
- Autorius
- Išleidimo metai
- Kalba
- Žanras

Toliau eina smulkesni knygos atributai:

Knygos būseną:

- Įvertinimas
- Paskutinis skaitytas puslapis

Knygos puslapiai:

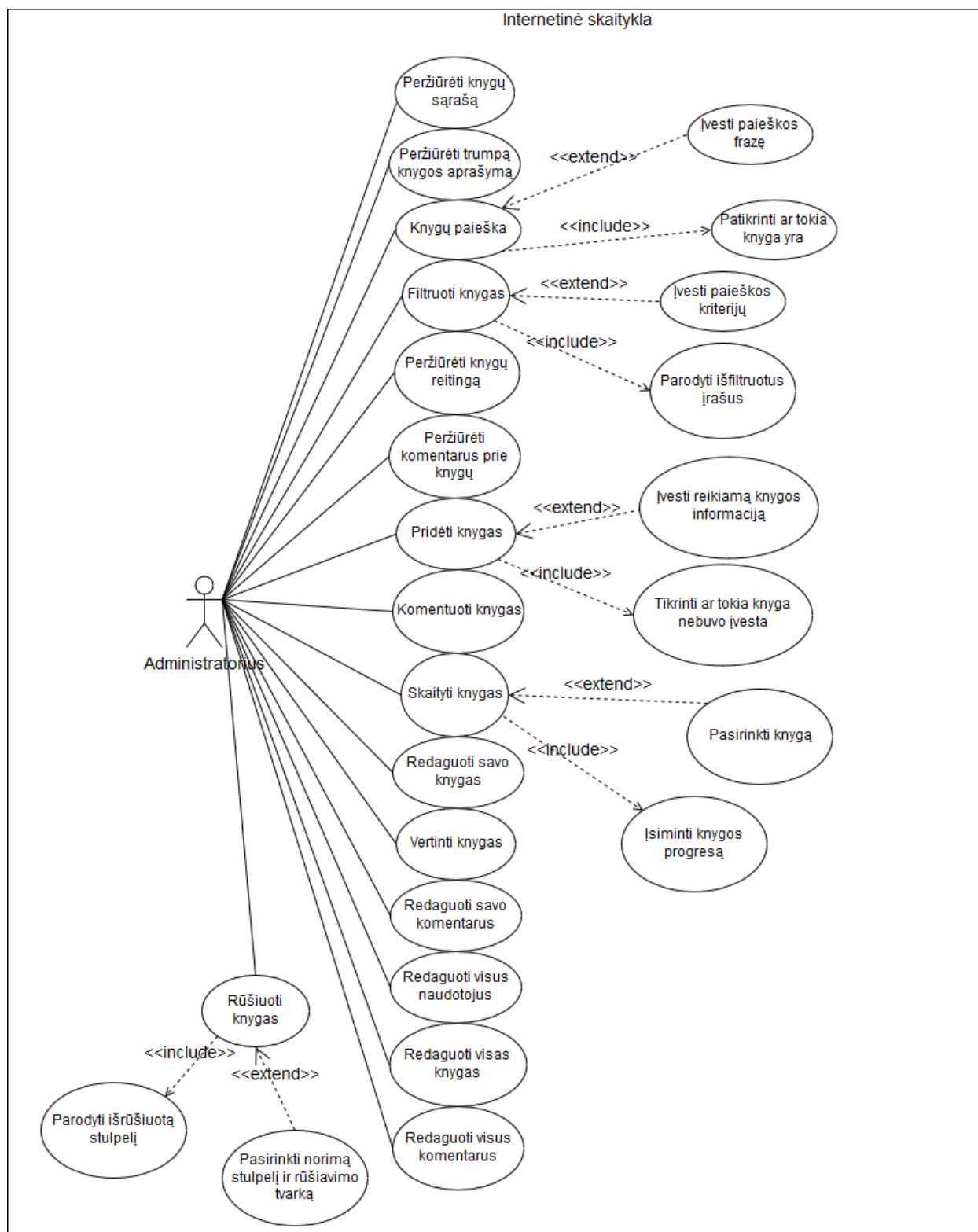
- Puslapių skaičius

Knygos atributai:

- Aprašymas
- Komentaras
- Skaitytojų skaičius
- Knygos tipas

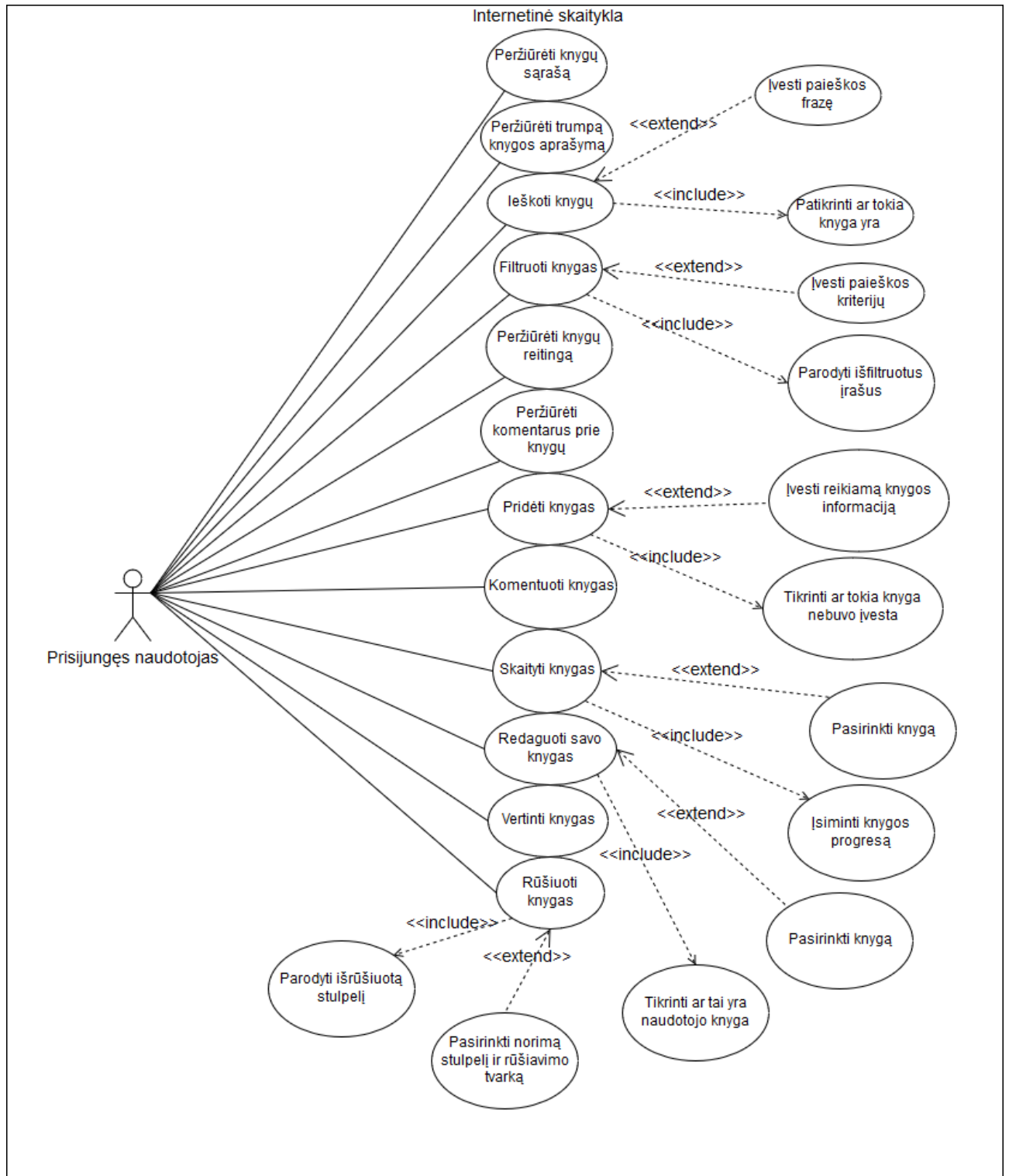
#### 4.4. Panaudos atvejų diagramos

Kadangi sistemoje bus trys rolės, tai nusprendžiau kiekvienai rolei padaryti atskira diagramą. Taigi administratoriaus rolės panaudos atvejų diagrama (3pav.):



3pav. Administratoriaus panaudos atvejų diagrama

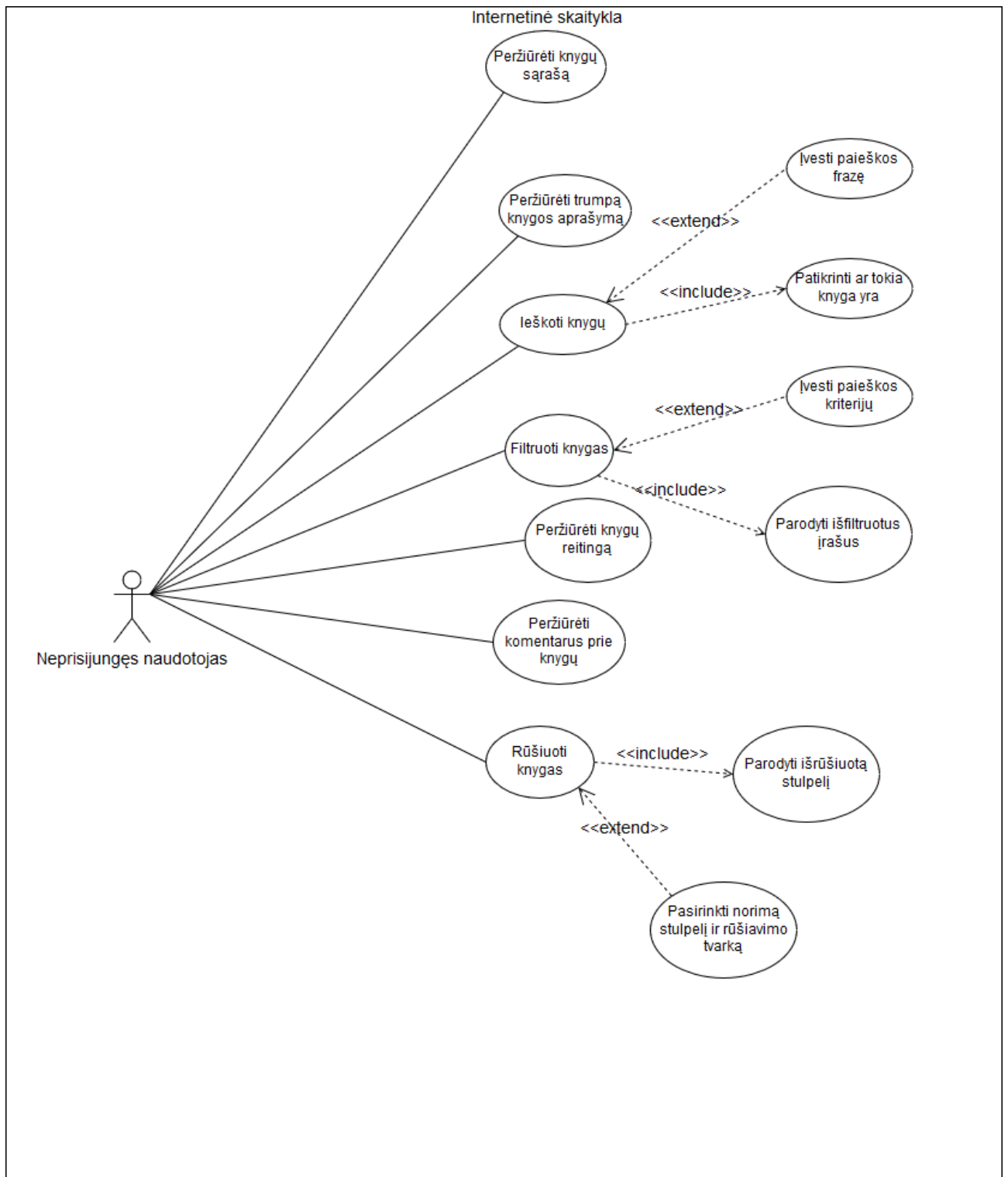
Toliau eina prisijungusio naudotojo rolės panaudos atvejų diagrama (4pav.):



4pav. Prisijungusio naudotojo panaudos atvejų diagrama

Ir galų gale neprisijungusio naudotojo rolės panaudos atvejų diagrama (5pav.):





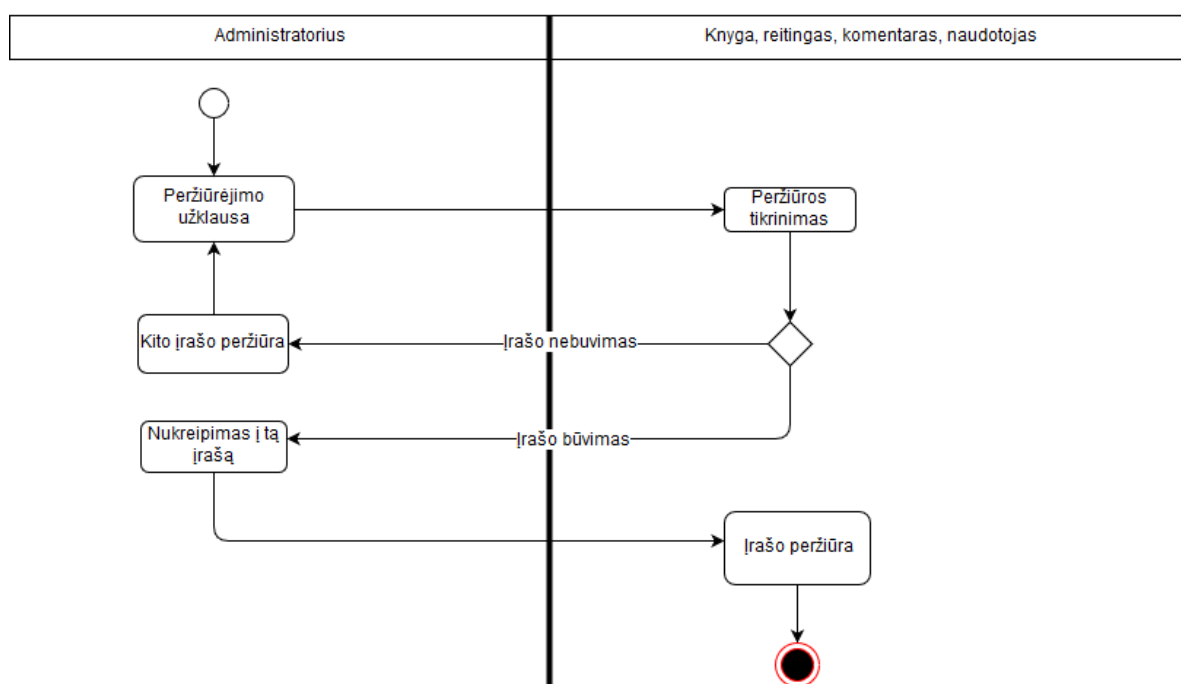
5pav. Nerisijungusio naudotojo panaudos atvejų diagrama



#### 4.5. Pagrindinio ir alternatyvaus scenarijaus lentelės bei veiklos diagramos

1 lentelė Peržiūros pagrindinis ir alternatyvus scenarijus

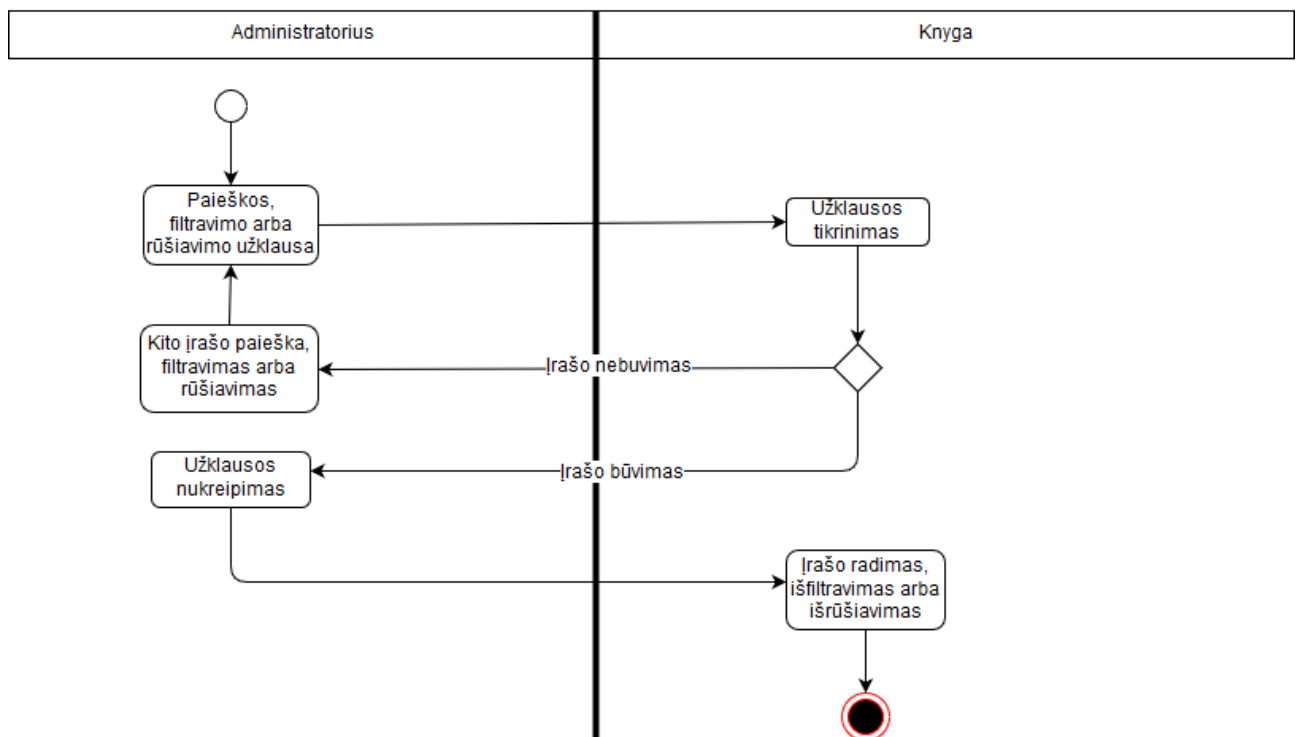
Pavadinimas	Peržiūrėti
Trumpas aprašymas	Administratorius peržiūri knygų sąrašą, reitingą, komentarus, naudotojus
Naudotojai (-ai)	Administratorius, naudotojai
Prieš sąlygos	Turi būti pateiktas knygų sąrašas, reitingas, komentarai, naudotojai
Pagrindinis scenarijus	Administratorius peržiūri norimus įrašus
Alternatyvūs scenarijai	Peržiūrėti įrašo neįmanoma, jeigu jis buvo ištrintas
Po sąlygos	Norimi įrašai yra peržiūrėti



6 pav. Peržiūros veiklos diagrama

2 lentelė Paieškos pagrindinis ir alternatyvus scenarijus

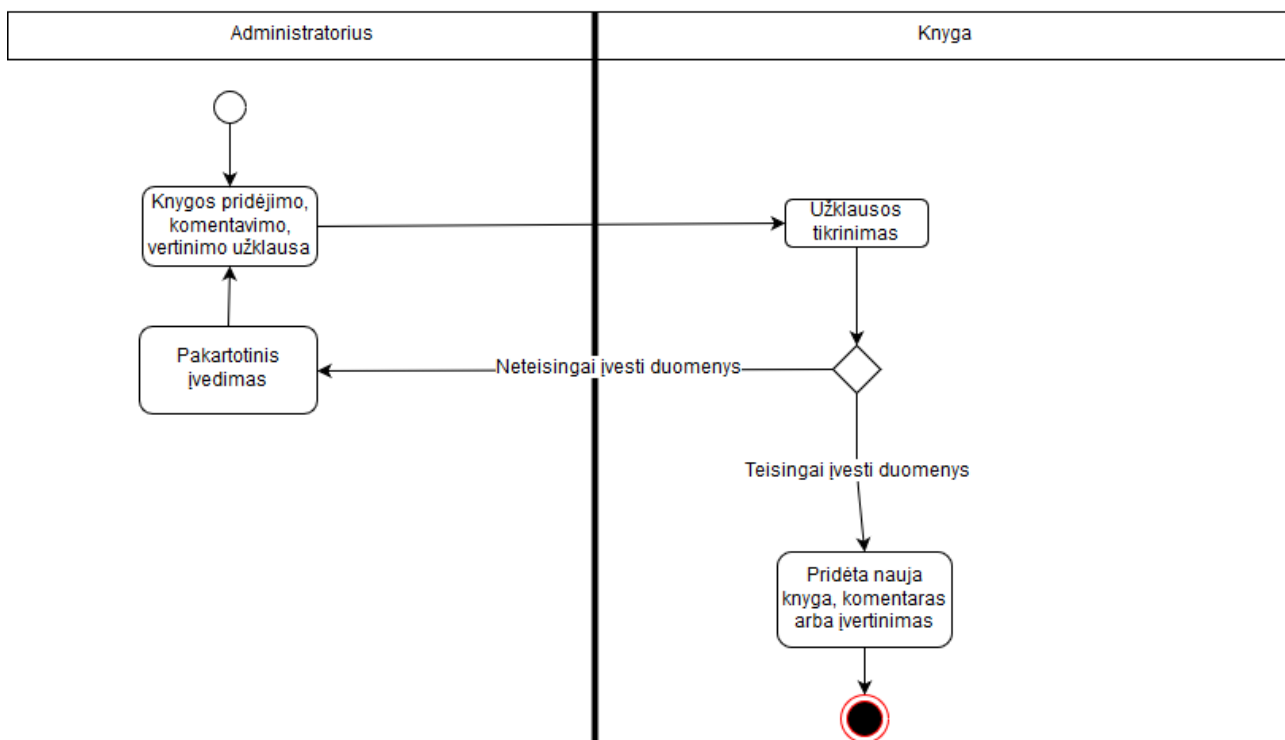
Pavadinimas	Ieškoti, filtruoti, rūšiuoti
Trumpas aprašymas	Administratorius ieško, filtruoja arba rūšiuoja knygas
Naudotojai (-ai)	Administratorius
Prieš sąlygos	Turi būti pateiktos paieškos, filtravimo ir rūšiavimo galimybės
Pagrindinis scenarijus	Administratorius ieško, filtruoja arba rūšiuoja norimus įrašus
Alternatyvūs scenarijai	Ieškoti, filtruoti arba rūšiuoti įrašų neįmanoma, jeigu tokių įrašų nėra
Po sąlygos	Norimi įrašai randami, išfiltruojami arba išrūšiuojami



7pav. Paieškos veiklos diagrama

3 lentelė Pridėjimo pagrindinis ir alternatyvus scenarijus

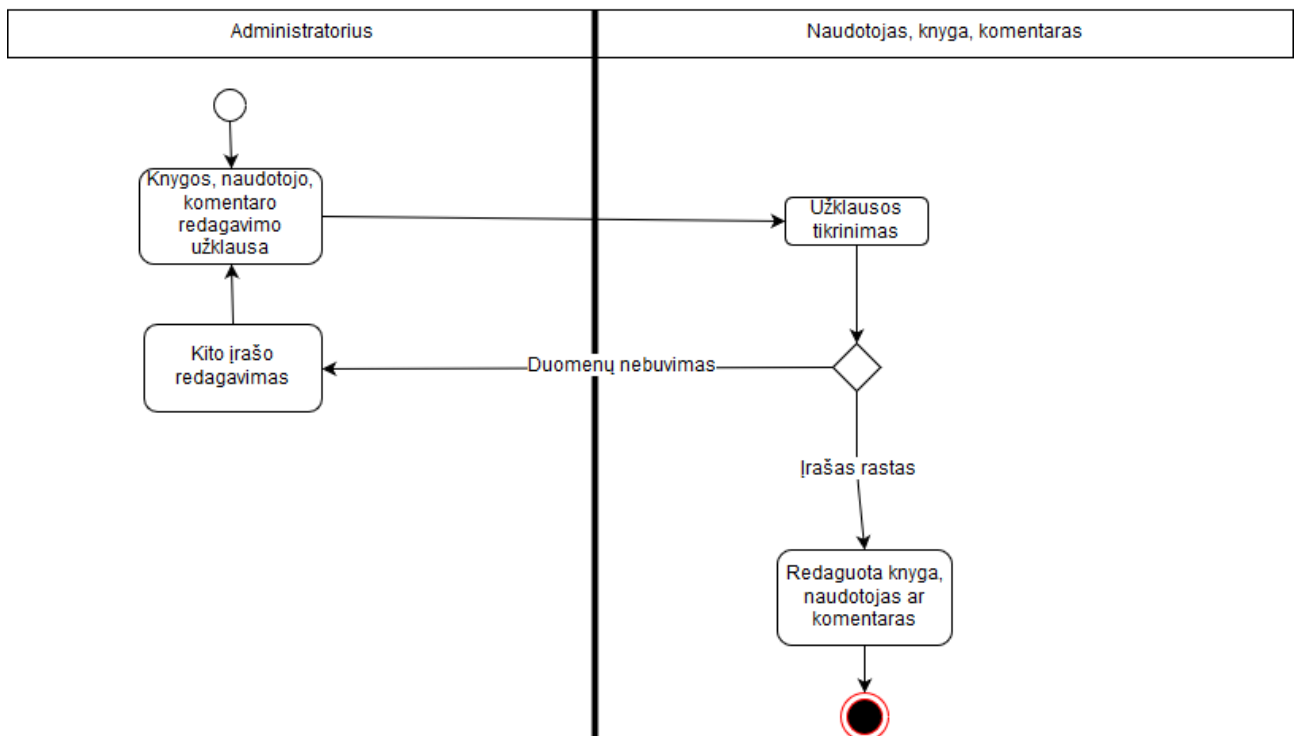
Pavadinimas	Pridėti, komentuoti, vertinti
Trumpas aprašymas	Administratorius prideda, komentuoja arba vertina knygas
Naudotojai (-ai)	Administratorius
Prieš sąlygos	Turi būti pateikta galimybė pridėti, komentuoti arba vertinti knygas
Pagrindinis scenarijus	Administratorius prideda, komentuoja arba vertina knygas
Alternatyvūs scenarijai	Pridedant knygą yra užpildomi ne visi laukai arba yra užpildomi, bet neteisingai, įvedamas per ilgas komentaras
Po sąlygos	Sukuriama nauja knyga, parašomas komentaras ir įvertinama knyga



8pav. Pridėjimo veiklos diagrama

4 lentelė Pridėjimo pagrindinis ir alternatyvus scenarijus

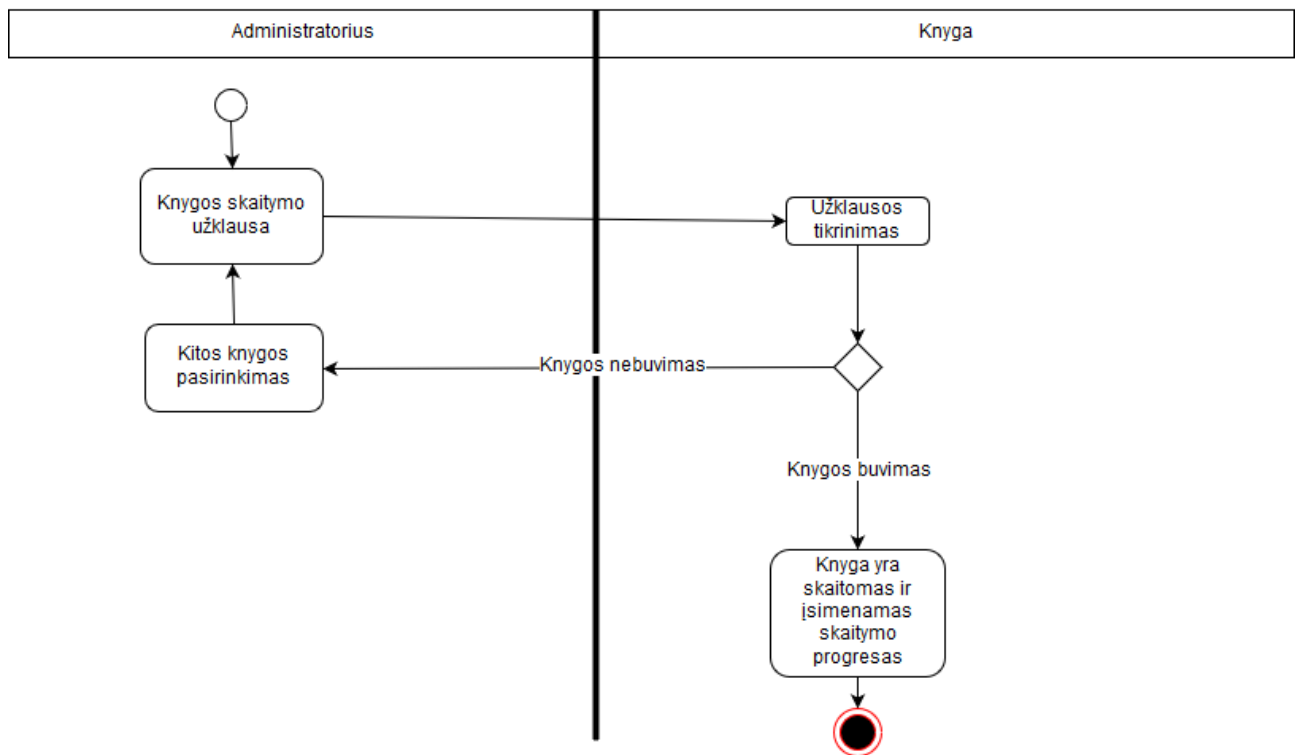
Pavadinimas	Redaguoti
Trumpas aprašymas	Administratorius redaguoja (pakeičia arba ištrina) norimas knygas, komentarus, aprašymus, naudotojus
Naudotojai (-ai)	Administratorius, naudotojas
Prieš sąlygos	Turi būti pridėta knyga, komentaras, aprašymas arba naudotojas
Pagrindinis scenarijus	Administratorius redaguoja (pakeičia arba ištrina) norimą knygą, komentarą, aprašymą, naudotoją
Alternatyvūs scenarijai	Redaguoti arba ištrinti įrašo neįmanoma, jeigu jo nėra
Po sąlygos	Redaguojamos knygos, komentarai, aprašymai arba naudotojai



9pav. Redagavimo veiklos diagrama

5lentelė Skaitymo pagrindinis ir alternatyvus scenarijus

Pavadinimas	Skaityti
Trumpas aprašymas	Administratorius skaito norimas knygą
Naudotojai (-ai)	Administratorius
Prieš sąlygos	Turi būti pridėta knyga
Pagrindinis scenarijus	Administratorius skaito norimą knygą
Alternatyvūs scenarijai	Skaityti knygos neįmanoma, jeigu jos nėra
Po sąlygos	Yra skaitoma knyga ir įsimenamas skaitymo progresas



10pav. Skaitymo veiklos diagrama

## **5. Programinė realizacija**

Pradedant kurti „Internetinę skaityklą“ iš pradžių reikėjo pasirinkti kokią programavimo kalbą naudosi. Kadangi aš jau turiu kažkiek realios patirties darbe su PHP programavimo kalba, tai nedaug mąstydamas ją ir pasirinkau. Taip pat kadangi naudojam Symfony karkasą, tai nusprendžiau naudoti ir jį. Tačiau jį naudoti, kaip pasirodė, yra ne taip ir paprasta, nes reikia atlikti nemažai konfigūracijų po jo įdiegimo. Taigi iš pradžių aprašau Symfony karkaso įdiegimo ir konfigūravimo procesą.



## 5.1. Symfony karkaso įdiegimas ir konfigūravimas

### 5.1.1 Projekto sukūrimas

Čia bus aprašomas projekto sukūrimas nuo pat pradžių. Taigi iš pradžių reikia atsisiųsti ir įsidiegti Symfony Installer ir jį naudoti savo projektų sukūrimui. Taip pat reikia turėti įsirašius kokį nors lokalų serverį (pavyzdžiui XAMPP). Toliau yra padaryta nuoroda į tai, kaip instaliuoti ir paruošti naudojimui Symfony. Pagrindiniai konfigūracijos punktai atsidarius komandinę eilutę (11pav.):

```
# Linux and macOS systems
$ sudo mkdir -p /usr/local/bin
$ sudo curl -LS https://symfony.com/installer -o /usr/local/bin/symfony
$ sudo chmod a+x /usr/local/bin/symfony

# Windows systems
c:\> php -r "readfile('https://symfony.com/installer');" > symfony
```

11pav. Pagrindiniai konfigūracijos punktai

Linux ir macOS operacinėse sistemose yra sukuriami globali symfony komanda, o Windows reikia perkelti symfony failą į lokalaus serverio direktoriją, tam, kad būtų sukurta globali komanda arba perkelti į bet kokią Jums tinkančią direktoriją (12pav.):

```
# for example, if WAMP is used ...
c:\> move symfony c:\wamp\bin\php

# ... then, execute the command as:
c:\> symfony

# moving it to your projects folder ...
c:\> move symfony c:\projects

# ... then, execute the command as
c:\> cd projects
c:\projects> php symfony
```

12pav. Perkėlimas į direktoriją

Po įdiegimo galima sukurti savo Symfony programą naudojant komandą „new“ (13pav.):

```

$ cd projects/
$ symfony new blog

# Windows
c:\> cd projects/
c:\projects\> php symfony new blog

```

**13pav. Programos sukūrimas**

Tiesa, tai yra ne visai pirmas dalykas, kurį reikėjo daryti atsisiunčiant Symfony Installer. Iš pradžių reikėjo instaliuoti Composer į komandinę eilutę įrašius „composer install“. Instaliavus jį, galima į Symfonį karkasą įtraukti norimas bibliotekas.

Sukūrus projektą ir įvedus „blog/“ direktoriją bus matomi automatiškai sukurti failai ir direktorijos (14pav.):

```

1  blog/
2  └─ app/
3     └─ config/
4     └─ Resources/
5  └─ bin
6     └─ console
7  └─ src/
8     └─ AppBundle/
9  └─ var/
10  └─ cache/
11  └─ logs/
12  └─ sessions/
13 └─ tests/
14  └─ AppBundle/
15 └─ vendor/
16 └─ web/

```

**14pav. Automatiškai sukurti failai ir direktorijos**

Taip pat oficialiame tinklalapyje buvo aprašoma kiekvienos direktorijos rekomenduojamas pritaikymas:

- app/config/ - visa nurodyta konfigūracija bet kokiai aplinkai
- app/Resources/ - visi šablonai ir vertimo failai
- src/AppBundle/ - Symfony valdikliai ir keliai, domeno kodas ir visa verslo logika
- var/cache/ – visi „cache“ failai
- var/logs/ - visi logų failai
- var/sessions/ - visi sesijos failai

- tests/AppBundle/ - automatiniai testai
- vendor/ - šioje direktorijoje „Composer“ instaliuoja programos priklausomybes ir vartotojas čia nieko neturėtų redaguoti
- web/ - stiliaus, JavaScript failai ir paveikslukai

Taip pat reikėjo sukonfigūruoti lokalų serverį, kurį paleisdavau per programą „XAMPP“. O būtent reikėjo nustatyti portus, per kuriuos pasileistų lokalus serveris, tai iš esamo 8000 keičiau į 8080.

### 5.1.2 Konfigūracija

Šioje skiltyje aprašau pagrindinę projekto konfigūraciją ir kaip geriau ką naudoti. Bendrai konfigūracija apima skirtingas projekto dalis ir aplinkas ir dėl to Symfony rekomenduoja padalinti projekto konfigūraciją į tris dalis:

- app/config/parameters.yml – apibrėžiami su infrastruktūra susiję nustatymai (15pav.)

```
# app/config/parameters.yml
parameters:
    database_driver:   pdo_mysql
    database_host:     127.0.0.1
    database_port:     ~
    database_name:     symfony
    database_user:     root
    database_password: ~

    mailer_transport:  smtp
    mailer_host:       127.0.0.1
    mailer_user:       ~
    mailer_password:   ~

# ...
```

15pav. Su infrastruktūra susiję nustatymai

- app/config/parameters.yml.dist – apibrėžiami visi projekto parametrai
- app/config/config.yml – apibrėžiamas projekto funkcionalumas

Taip pat parametrų konfigūracijos pavadinimai turėtų būti kuo trumpesni ir sudaromi su bendru priešdėliu visai programai.

### 5.1.3 Verslo logikos organizavimas

Programinėje įrangoje verslo logika arba domeno logika yra aprašoma kaip programos dalis, kuri nurodo verslo taisykles kaip turi būti sukurti, rodomi, laikomi ir redaguojami duomenys. Symfony programose verslo logika yra visas kodas, kurį rašo vartotojas, kuris nėra priskirtas prie jokio karkaso, pavyzdžiui keliai ir valdikliai. Domenų klasės, Doktrinų esybės ir paprastos PHP klasės, kurios naudojamos kaip servaisai, yra geri verslo logikos pavyzdžiai.

Programos servisų pavadinimai turėtų būti kuo trumpesni, bet pakankamai unikalūs, kad prireikus juos galima būtų surasti projekte (17pav.):

```
1  # app/config/services.yml
2  services:
3      # keep your service names short
4      app.slugger:
5          class: AppBundle\Utils\Slugger
```

16pav. Programos servisų pavadinimai

Savo sukurtų servisų apibrėžimui reiktų naudoti YAML formatą, nes jis yra draugiškas pradedantiesiems ir glaustas. Taip pat nėra rekomenduojama turėti servisų klasių pavadinimus (18pav.):

```
1  # app/config/services.yml
2
3  # service definition with class namespace as parameter
4  parameters:
5      slugger.class: AppBundle\Utils\Slugger
6
7  services:
8      app.slugger:
9          class: '%slugger.class%'
```

17pav. Servisų klasių pavadinimai

Doktrinos esybėms ir „mapping“ informacijai reikia naudoti anotacijas, nes jos yra labiausiai patikimas ir greitas būdas tam dalykui (19pav.):

```

1 namespace AppBundle\Entity;
2
3 use Doctrine\ORM\Mapping as ORM;
4 use Doctrine\Common\Collections\ArrayCollection;
5
6 /**
7  * @ORM\Entity
8  */
9 class Post
10 {
11     const NUM_ITEMS = 10;
12
13     /**
14      * @ORM\Id
15      * @ORM\GeneratedValue
16      * @ORM\Column(type="integer")
17      */
18     private $id;
19
20     /**
21      * @ORM\Column(type="string")
22      */
23     private $title;

```

18pav. Anotacijų naudojimas doktrinių esybėms

#### 5.1.4 Valdikliai

Symfony naudoja filosofiją „ploni valdikliai ir stori modeliai“, tai reiškia, kad valdikliai turėtų savyje turėti tik ploną sluoksnį kodo, reikalingo koordinuoti skirtingas programos dalis. Toliau aprašoma 5-10-20 taisyklė – valdikliai turi apibrėžti 5 arba mažiau kintamuosius, turėti 10 arba mažiau metodų, kuriuose turi būti 20 arba mažiau kodo eilučių.

Gera praktika yra sukurtam valdikliui paveldėti FrameworkBundle bazinį valdiklį ir kelių konfigūracijai, „cachinimui“ ir apsaugai kada tik įmanoma naudoti anotacijas (20pav.):

```

1 # app/config/routing.yml
2 app:
3     resource: '@AppBundle/Controller/'
4     type:     annotation

```

19pav. Anotacijų naudojimas kelių konfigūracijai

Nepatariama naudoti @Template anotacijos valdiklio naudojamo šablono konfigūracijai. Taip turi atrodyti pagrindinio puslapio valdiklis (21pav.):

```

1 namespace AppBundle\Controller;
2
3 use Symfony\Bundle\FrameworkBundle\Controller\Controller;
4 use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
5
6 class DefaultController extends Controller
7 {
8     /**
9      * @Route("/", name="homepage")
10     */
11     public function indexAction()
12     {
13         $posts = $this->getDoctrine()
14             ->getRepository('AppBundle:Post')
15             ->findLatest();
16
17         return $this->render('default/index.html.twig', array(
18             'posts' => $posts
19         ));
20     }
21 }

```

20pav. Pagrindinio puslapio valdiklis

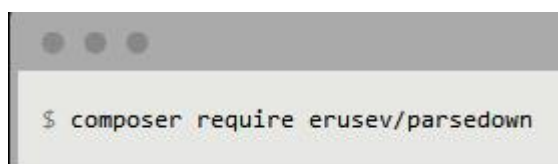
### 5.1.5 Šablonai

Kai php buvo sukurtas 20 metų atgal, programuotojams patiko jo paprastumas ir kaip jame gerai susimaišė HTML ir dinaminis kodas. Bet laikas bėga ir atsirado šablonų kalbos, kurios padarė juos dar geresnius. Vienas iš tokių šablonų yra Twig, kuri ir siūlo pagal geriausią praktiką naudoti Symfony. Dėl to Twig yra nustatytas kaip Symfony šablonas pagal nutylėjimą. Taip pat Twig yra vienintelis šablonas, kuris garantuotai veiks su Symfony 3.0.

Symfony siūlo laikyti visus programos šablonus `app/Resources/views/` direktorijoje. Prieš tai jie buvo laikomi tiesiog `Resources/views/` direktorijoje, tačiau taip kartu prie projekto dirbantiems dizaineriams buvo nepatogu ieškoti tų šablonų išsimėčiusių po visą projektą skirtinguose aplankuose ir naudojant `app/` aiškiai supaprastėja jų pavadinimai, pavyzdys (kairės lentelės pusėje kaip buvo anksčiau, o dešinėje, kaip yra dabar):

Templates Stored inside Bundles	Templates Stored in <code>app/</code>
<code>AcmeDemoBundle:Default:index.html.twig</code>	<code>default/index.html.twig</code>
<code>::layout.html.twig</code>	<code>layout.html.twig</code>
<code>AcmeDemoBundle::index.html.twig</code>	<code>index.html.twig</code>
<code>AcmeDemoBundle:Default:subdir/index.html.twig</code>	<code>default/subdir/index.html.twig</code>
<code>AcmeDemoBundle:Default/subdir:index.html.twig</code>	<code>default/subdir/index.html.twig</code>

Mano projektas reikalauja specialaus `md2html` Twig filtro, kad galima būtų paversti kiekvieno posto Markdown turinį į HTML. Tam, iš pradžių reikia instaliuoti `Parsedown` Markdown analizatorių kaip naują projekto priklausomybę (įrašyti komandą į komandinę eilutę) (22pav.):



21pav. Analizatoriaus instaliavimas

Toliau reikia sukurti naują Markdown servisą, kuris bus toliau naudojamas Twig plėtiniui. Serviso apibrėžimas reikalauja tik kelio iki klasės (23pav.):

```

1  # app/config/services.yml
2  services:
3      # ...
4      app.markdown:
5          class: AppBundle\Utils\Markdown

```

22pav. Serviso apibrėžimo kelias

Kaip ir Markdown klasei reikia apibrėžti tik vieną metodą, kad galima būtų paversti Markdown turinį į HTML (24pav.):

```

1  namespace AppBundle\Utils;
2
3  class Markdown
4  {
5      private $parser;
6
7      public function __construct()
8      {
9          $this->parser = new \Parsedown();
10     }
11
12     public function toHtml($text)
13     {
14         $html = $this->parser->text($text);
15
16         return $html;
17     }
18 }

```

23pav. Markdown turinio pavertimas į HTML

Toliau, reikia sukurti naują Twig plėtinį ir apibrėžti naują filtrą „md2html“ naudojant „Twig\_SimpleFilter“ klasę ir įterpti naują apibrėžtą „markdown“ servisą Twig plėtinio konstruktoriuje (25pav.):



```

1 namespace AppBundle\Twig;
2
3 use AppBundle\Utils\Markdown;
4
5 class AppExtension extends \Twig_Extension
6 {
7     private $parser;
8
9     public function __construct(Markdown $parser)
10    {
11        $this->parser = $parser;
12    }
13
14    public function getFilters()
15    {
16        return array(
17            new \Twig_SimpleFilter(
18                'md2html',
19                array($this, 'markdownToHtml'),
20                array('is_safe' => array('html'))
21            ),
22        );
23    }
24
25    public function markdownToHtml($content)
26    {
27        return $this->parser->toHtml($content);
28    }
29
30    public function getName()
31    {
32        return 'app_extension';
33    }
34 }

```

24pav. Naujo twig plėtinio sukūrimas

Galų gale reikėjo apibrėžti naują servisą, kad galima būtų naudoti Twig plėtinį aplikacijoje (26pav.):

```

1 # app/config/services.yml
2 services:
3     app.twig.app_extension:
4         class: AppBundle\Twig\AppExtension
5         arguments: ['@app.markdown']
6         public: false
7         tags:
8             - { name: twig.extension }

```

25pav. Naujo serviso apibrėžimas

### 5.1.6 Formos

Formos yra vienas iš daugiausiai netinkamai naudojamų Symfony komponentų dėl jų plačios naudojimo srities ir begalinių privalumų. Čia bus aiškinamos geriausios formų naudojimo praktikos norint pagreitinti darbo atlikimą.

Taigi yra siūloma kiekvieną formą aprašinėti PHP klasėse, o tas klases talpinti AppBundle/Form vardo vietoje (namespace) (27pav.):

```
1 namespace AppBundle\Form;
2
3 use AppBundle\Entity\Post;
4 use Symfony\Component\Form\AbstractType;
5 use Symfony\Component\Form\FormBuilderInterface;
6 use Symfony\Component\OptionsResolver\OptionsResolver;
7 use Symfony\Component\Form\Extension\Core\Type\TextareaType;
8 use Symfony\Component\Form\Extension\Core\Type\EmailType;
9 use Symfony\Component\Form\Extension\Core\Type\DateTimeType;
10
11 class PostType extends AbstractType
12 {
13     public function buildForm(FormBuilderInterface $builder, array $options)
14     {
15         $builder
16             ->add('title')
17             ->add('summary', TextareaType::class)
18             ->add('content', TextareaType::class)
19             ->add('authorEmail', EmailType::class)
20             ->add('publishedAt', DateTimeType::class)
21         ;
22     }
23
24     public function configureOptions(OptionsResolver $resolver)
25     {
26         $resolver->setDefaults(array(
27             'data_class' => Post::class,
28         ));
29     }
30 }
```

26pav. Formos aprašymas

Norint naudoti sukurtą klasę, reikia panaudoti „createForm()“ ir perduoti tos klasės pavadinimą (28pav.):

```

1  // ...
2  use AppBundle\Form\PostType;
3
4  // ...
5  public function newAction(Request $request)
6  {
7      $post = new Post();
8      $form = $this->createForm(PostType::class, $post);
9
10     // ...
11 }

```

27pav. Klasės pavadinimo perdavimas

Reikia atidžiai stebėti kur bus naudojamos formų klasės, tai padeda ateityje paprasčiau jas naudoti. Patariama aprašyti mygtukus šablonuose, o ne formų klasėse arba valdikliuose. Taip yra todėl, nes talpinant juos pastaruosiuose bus sumažinta tos formos apimtis ir būtų problematiška naudoti mygtukus postų redagavimui. Taip pat tuo atveju būtų sumaišomi aprašų žymėjimai su grynu PHP kodu. Taigi atskyrimas yra geriausia praktika ir dėl to siūloma talpinti visus su atvaizdavimu susijusius dalykus į vaizdų sluoksnį.

Geriausia paleisti formą naudojant formos valdiklių funkciją (29pav.):

```

1  {{ form_start(form, {'attr': {'class': 'my-form-class'}}) }}
2      {{ form_widget(form) }}
3  {{ form_end(form) }}

```

28pav. Formos valdiklių funkcija

Formos patvirtinimui naudojamas panašus šablonas (30pav.):

```

1 public function newAction(Request $request)
2 {
3     // build the form ...
4
5     $form->handleRequest($request);
6
7     if ($form->isSubmitted() && $form->isValid()) {
8         $em = $this->getDoctrine()->getManager();
9         $em->persist($post);
10        $em->flush();
11
12        return $this->redirect($this->generateUrl(
13            'admin_post_show',
14            array('id' => $post->getId())
15        ));
16    }
17
18    // render the template
19 }

```

29pav. formos patvirtinimas

### 5.1.7 Apsauga

Kalbant apie apsaugą Symfony siūlo nepriklausant nuo poreikių autentifikavimą konfigūruoti security.yml faile, naudojant „firewalls“ raktą. Ir yra rekomenduojama naudoti tik vieną „firewall“ įrašą su „anonymous“ aktyvuotu raktu. Slaptažodžių užkodavimui reikėtų naudoti „bcrypt“ kodatorių, o ne tradiciškai naudojamą „SHA-512“. Pagrindiniai „bcrypt“ privalumai yra „salt“ naudojimas apsisaugojimui nuo „rainbow table“ ir „brute-force“ atakų.

Autorizacijos geriausia praktika:

- security.yml faile naudoti „access\_control“
- kur tik įmanoma naudoti „@Security“ anotaciją
- tikrinti saugumą tiesiogiai „security.authorization\_checker“ servise, jeigu užklumpa sudėtingesnė situacija
- norint uždrausti prieigą prie bet kokio objekto kokiam nors naudotojui per administratoriaus sąsają naudoti Symfony ACL

Projekte norint sukurti naują įrašą reikia rolės „ROLE\_USER“.

### 5.1.8 Interneto aktyvai

Interneto aktyvai yra tokie dalykai kaip CSS, JavaScript ir paveikslukai, kurie padaro tinklalapį gražesnį. Dažniausiai jie būna talpinami Resources/public/ arba web/ direktorijoje. Tačiau

reikia būti atsargiems talpinant failus web/, nes tuo atveju failai tampa viešai prieinami. Interneto aktyvų sujungimui ir sumažinimui reikia naudoti Assetic.

Assetic yra aktyvų valdytojas, kuris gali kompiliuoti aktyvus padarytus su įvairiomis technologijomis, tokiomis kaip LESS, Sass ir CoffeScript. Visus šituos aktyvus apgaubia vienas Twig tagas.

## 5.2. Naudotojas

Kuriant naudotoją iš pradžių reikėjo sukurti user klasę, su reikiama atributais: paskyros vardas, elektroninis paštas, slaptažodis.

Toliau kūrėm naudotojo prisijungimą, tam reikėjo sukurti SecurityController, kuriame buvo tikrinami ir validuojami įvesti laukai, ar gerai įvestas elektroninis paštas, ar įvestas reikiamas slaptažodis ir panašiai. Ir taip pat čia buvo nurodoma, kad įvesti laukai būtų perduodami į atvaizdavimo twig failą, kur ir buvo prisijungimo formos vaizdas. O formos laukai yra UserType faile, kur yra nurodoma kokio tipo turi būti kiekvienas laukas. Dar buvo sukurta atsijungimo galimybė, kuri atjungia prisijungusį naudotoją iš sistemos. Prisijungęs naudotojas taip pat gauna user rolę, pagal kurią gali redaguoti, trinti ir kurti savo knygą.

Aišku logino neužteko, reikėjo ir registracijos, tai po to ją ir kūrėm. UserController faile reikėjo aprašyti, kad visa informacija įvedama į formos laukus būtų talpinama į duomenų bazę, kad po to galima būtų pagal tai prisijungti. Registracijoje taip pat yra nemažai validacijos: tikrinama ar gerai įvestas elektroninis paštas, ar pakartotinai įvestas slaptažodis atitinka originalą ir panašiai. Užsiregistravus naudotojas gali iš karto prisijungti, o prisijungęs yra nukreipiamas į pagrindinį puslapį, iš kurio pagal navigaciją jau gali keliauti į knygų sąrašą.

Taigi iš pradžių pateikiu naudotojo prisijungimo klases ir kartu jų metodus ir už ką jie atsakingi:

### 1. User.php

Tai yra pagrindinė naudotojo klasė, kurioje iš pradžių įvedami kintamieji atsakingi už prisijungimą, tokie kaip: \$username, \$password, \$roles ir kiti, kuriuos ne visus naudojam, bet jie turėjo būti įvesti, nes taip prašo naudojama klasė: UserInterface. Vėliau buvo aprašomi taip vadinami “seteriai” ir “geteriai”, kurie iš pradžių priskiria, o po to grąžina reikšmes, kaip pavyzdys galėtų būti: setUsername, setPassword, setRoles ir getUsername, getPassword, getRoles. Taigi aprašomi kintamieji atsakingi už prisijungimo vardą arba šiuo atveju elektroninį pašta, slaptažodį, kuri naudotojas turės įvesti, norėdamas prisijungti į sistemą, rolę, pagal kurias bus aišku ar naudotojas yra administratorius ar ne ir koks naudotojas įkėlė kokią knygą.

### 2. SecurityController.php

- loginAction

Tai yra naudotojo prisijungimo metodas, kuris mano sistemoje yra pasiekiamas naudojant „/login“ kelią ir turintis „user\_login“ pavadinimą. Toliau pasinaudojau oficialiausio Symfony puslapio dokumentacija, o būtent „How to Build a Traditional Login Form“, kur buvo pakankamai gerai paaiškinta, kaip yra gaunami autentifikavimo

nustatymai ir klaidos. Taip pat parametrų skiltyje nurodžiau prisijungimo vardą, kad jį naudotų toliau sukurtoje formoje. Po to reikėjo nurodyti kelią iki prisijungimo atvaizdavimo failo „login.html.twig“ naudojant „\$this->render()“ funkciją, kur taip pat buvo nurodyti formos sukūrimo ir klaidų parametrai.

- `logoutAction`

Tai yra naudotojo atsijungimo metodas, kuris mano sistemoje yra pasiekiamas naudojant „/logout“ kelią ir turintis „user\_logout“ pavadinimą. Čia jau yra aprašomas naudotojo atsijungimas ir nurodomas kelias, kur naudotojas bus nukreipiamas atsijungus, o būtent į pagrindinį „home“ puslapį.

### 3. Login.html.twig

Tai yra naudotojo prisijungimo formos atvaizdavimo failas, turintis įvairių validavimų ir tikrinimų ar įvestas egzistuojantis naudotojas. Ten yra aprašoma ir klaida, kuri yra parodoma, jeigu kažkas ne taip įvesta. Toliau yra aprašomi prisijungimui reikiami laukai: prisijungimo vardas (elektroninis paštas) ir slaptažodis. Formos pabaigoje yra aprašomas prisijungimo mygtukas, kuris arba prijungia vartotoją arba iškviečia klaidos pranešimus ir prašo naudotojo įvesti prisijungimo duomenys iš naujo. Be to formos pabaigoje dar pridėjau registracijos nuorodą, kad, jeigu naudotojas dar nėra užsiregistravęs, galėtų tai padaryt.

### 4. LoginFormAuthenticator.php

Tai yra klasė atsakinga už naudotojo autentifikavimą, kuri gauna visus reikiamus prisijungimui duomenis iš naudotojo, kaip ir pati naudotoją. Taip pat tikrina naudotojo įvestus duomenys ar jie sutampa su tais, kurie buvo išsaugoti registracijos metu. Be to yra aprašoma kur yra nukreipiamas naudotojas po sėkmingo prisijungimo ir gaunamas prisijungimo Url.

- `__construct`

Šis metodas yra visos klasės griaučiai. Čia aprašomi visi reikalingi kintamieji kartu su jų tipais ir susijusiomis klasėmis ir perduodami tolesniam naudojimui.

- `getCredentials`

Kadangi šis metodas yra kviečiamas kiekviena kartą kai būna pateikiama užklausa iš pradžių reikėjo patikrinti ar ta užklausa yra prisijungimo formos patvirtinimas ir ar yra naudojamas POST metodas. Buvo nurodyta, kad forma nukreiptų iškart į kelią „/login“, kad autentifikatorius iš kart imtųsi darbo. Taigi, jeigu prisijungimas yra patvirtintas bus tikrinami visi naudotojo įvesti duomenys, o jeigu ne yra nieko

nedaroma ir tiesiog grąžinamas “null”. Toliau yra sukuriamą pati prisijungimo forma ir padaroma taip, kad antrą kartą užėjus į tą pačią formą, pavyzdžiui įvedus neteisingą slaptažodį, prisijungimo vardas išliktų.

- `getUser`

Šis metodas aprašo kokių naudotojo duomenų reikia ieškant jo duomenų bazėje, o tai yra jo prisijungimo vardo, kuris šiuo atveju yra elektroninis paštas. Jeigu smulkiau, tai yra kreipiamasi į “User” duomenų bazės lentelę ir naudojama funkcija “findOneBy”, kuri pagal toliau nurodyta kriterijų “email” ir randa naudotoją.

- `checkCredentials`

Šis metodas tiesiog patikrina ar naudotojo įvestas slaptažodis yra teisingas ir jeigu taip, praleidžia naudotoją toliau.

- `onAuthenticationSuccess`

Čia nurodoma, kur naudotojas patenka, kai prisijungia.

- `getLoginUrl`

Čia yra generuojamas naudotojo prisijungimo kelias, o būtent “user\_login”.

## 5. `services.yml`

Čia yra užregistruojama `LoginFormAuthenticator` klasė įvedant serviso ir tos klasės pavadinimą:

```
app.security.login_form_authenticator:  
  class: AppBundle\Security\LoginFormAuthenticator  
  autowire: true
```

Taip pat čia yra aprašomas naudotojo slaptažodžio užkodavimas:

```
app.doctrine.hash_password_listener:  
  class: AppBundle\Doctrine\HashPasswordListener  
  autowire: true  
  tags:  
    - { name: doctrine.event_subscriber }
```

## 6. `security.yml`

Tai yra apsaugo failas ir jame prie `guard` skilties reikėjo nurodyti autentifikatoriaus pavadinimą:

```
guard:  
  authenticators:  
    - app.security.login_form_authenticator
```



Taip pat čia reikėjo pridėti atsijungimo kelią:

firewalls:

main:

logout:

path: /logout

Be to yra aprašoma taisyklė, kad norint užėti į admino teisių reikalaujančius kelius, naudotojo rolė turi būti „ROLE\_ADMIN“:

access\_control:

- { path: ^/admin, roles: ROLE\_ADMIN }

## 7. config\_dev.yml

Čia reikėjo pakeisti vieną parametą, kad nerodytų klaidos bandant po prisijungimo perkelti naudotoją į nurodytą puslapį:

web\_profiler:

intercept\_redirects: true

## 8. base.html.twig

Šis failas atsako už bendrą vaizdą, kuris yra perduodamas į visus puslapius. Taip pat jame aprašau naudotojo pasveikinimą užėjus į savo paskyrą:

```
<li>Welcome {{ app.user ? app.user.username : 'Guest' }}!</li>
```

Čia, kaip matome, jeigu naudotojas būna prisijungęs, yra parodomas jo prisijungimo vardas, o jeigu ne – tiesiog pasveikinamas svečias.

Toliau padaryta, kad, jeigu naudotojas prisijungia su role „ROLE\_USER“ arba „ROLE\_ADMIN“, jam parodomas atsijungimo mygtukas, o jeigu naudotojas būna neprisijungęs, jam pateikiama registracijos arba prisijungimo galimybė:

```
{% if is_granted('ROLE_USER') or is_granted('ROLE_ADMIN') %}
```

```
<li><a href="{{ path('user_logout') }}">Logout</a></li>
```

```
{% else %}
```

```
<li><a href="{{ path('user_register') }}">Register</a></li>
```

```
<li><a href="{{ path('user_login') }}">Login</a></li>
```

```
{% endif %}
```

## 9. HashPasswordListener.php (patikrinti ar tikrai prie logino)

- `__construct`

Šiame metode sukuriamas slaptažodžio užšifravimo kintamasis

- `getSubscribedEvents`

Čia yra naudojami du įvykių pavadinimai, kuriuos „Doctrine“ padaro prieinamais:

`prePersist`, kuris iškviečiamas iškart prieš įkeliant esybę ir `preUpdate`, kuris iškviečiamas prieš atnaujinant esybę.

- `prePersist`

Taigi kaip jau buvo prieš tai minėta, šis metodas kviečiamas prieš esybės įkėlimą ir parašant:

```
$entity = $args->getEntity();
```

nurodoma, kuri esybė turi būti išsaugota. Ir esybė yra grąžinama tik tada, jeigu yra susijusi su User klase.

- `encodePassword`

Šiame metode yra užkoduojamas slaptažodis. Užšifruotas slaptažodis perduodamas naudotojui, kaip paprasto teksto slaptažodis.

- `preUpdate`

Čia yra pasakoma, kad slaptažodžio laukas yra atnaujinamas, kitaip jis neišsisaugos.

## 10. all\_users\_list.html.twig

Čia yra pateikiamas visų naudotojų sąrašas, kad prireikus, administratorius galėtų juos redaguoti, trinti, o taip pat prisijungti su kito naudotojo paskyra, taip galėdamas patikrinti kaip viskas atrodo jam. Tai yra vadinama impersonifikacija ir persijungia iš sąrašo pasirinkus norimą naudotoją, pagal jo elektroninį paštą:

```
'?_switch_user=' ~ user.email
```

Taip pat, po to galima atsijungti iš to naudotojo vaizdo atgal į administratoriaus:

```
'?_switch_user=_exit'
```

Pateikiu naudotojo registracijos klases ir kartu jų metodus ir už ką jie atsakingi:

### 1. UserController.php

- registerAction

Iš pradžių yra sukuriamą registracijos forma pagal RegistrationType klasę. Po to tikrinama ar forma yra galima ir patvirtinama ir jeigu taip, į duomenų bazę perduodama registracijos metu pateikta informacija ir tuo pačiu perkeliama į pagrindinį puslapį. Galų gale yra paduodamas registracijos formos atvaizdavimo failas.

Be to užsiregistravus naudotojas yra iškart nukreipiamas į savo paskyrą:

```
return $this->get('security.authentication.guard_handler')
->authenticateUserAndHandleSuccess(
    $user,
    $request,
    $this->get('app.security.login_form_authenticator'),
    'main'
);
```

- userList

Pateikiamas visų užsiregistravusių naudotojų sąrašas.

- editAction

Suteikiama galimybė redaguoti naudotoją.

- deleteAction

Suteikiama galimybė trinti naudotoją.

### 2. RegistrationType.php

Čia yra registracijos formos failas, kur yra nurodomi visi reikiami laukai norint užsiregistruoti. Taip pat tų laukų tipai.

### 3. Register.html.twig

Čia yra aprašoma naudotojo registracijos forma su reikiamais naudotojo laukais ir patvirtinimo mygtuku.

### 5.3. Knyga

#### 1. BookController.php:

- showAction

Šis metodas leidžia naudotojui atsidaryti kiekvieną knygą atskirai. Iš pradžių iš duomenų bazės lentelės „Book“ ieškomas jos id, o po to jau pagal tą id yra randama knyga. Ir metodo pabaigoje visą tai yra perduodama atvaizdavimo „Books\_list.html.twig“ failui, kartu su knygos ir jos id atributais.

- newAction

Šiame metodo iš pradžių yra sukuriamas knygos objektas, po to panaudojant „BookType“ failą yra sukurama naujos knygos sukūrimo forma. Toliau tikrinama, kad, jeigu ši forma yra galima ir patvirtinta, knyga yra išsaugoma duomenų bazėje ir naudotojas yra nukreipiamas į knygų sąrašą. Be to, kuriant knygą į duomenų bazę pagal elektroninį pašta yra išsaugomas ir tuo metu prisijungęs naudotojas, kuris sukūrė tą knygą. Ir pabaigoje yra perduodamas naujos knygos pridėjimo atvaizdavimo failas „new.html.twig“ ir sukuriamas tos formos vaizdas.

- editAction

Čia yra knygos redagavimo metodas. Iš pradžių duomenų bazėje yra ieškoma knyga pagal jos id, po to sukurama redagavimo forma. Ir jeigu forma yra galima ir patvirtinta knygos laukų pakeitimai yra išsaugomi duomenų bazėje. Redagavus knygą naudotojas nukreipiamas į knygų sąrašą.

- deleteAction

Šiame metode duomenų bazėje ieškoma knyga pagal jos id ir kai randama ištrinama iš duomenų bazės. Po trynimo naudotojas patenka į knygų sąrašą.

#### 2. DefaultController.php:

- indexAction – tas pats kaip show action, patikrinti pagal kuri daro
- bookList

Čia knygų duomenų bazėje ieškomos visos knygos ir perteikiamos į „all\_books\_list.html.twig“ atvaizdavimo failą.

- getNotesAction – reikės keisti arba šalinti, nes čia kol kas pagal pateiktus duomenis rankiniu būdu parodo komentarus atsidarius bet kokią knygą.

- homeAction

Šis metodas tiesiog nukreipia naudotoją į pagrindinį puslapį.

3. BookRepository.php:

- getValuesList

Čia aprašoma užklausa į duomenų bazę, kuri randa knygas pagal id ir atvaizduoja didėjimo tvarka.

4. All\_books.html.twig

Tai yra knygo sąrašo atvaizdavimo failas, kur yra aprašomas pagrindinis puslapio pavadinimas, lentelė su parametrais, kurioje yra talpinami reikiami knygos duomenys, taip pat naujos knygos pridėjimo, redagavimo ir trynimo mygtukai pagal atitinkamas roles. Prie trynimo mygtuko dar pridėtas pranešimas, kuris klausia naudotojo ar šis tikrai nori ištrinti pasirinktą knygą.

5. Books\_list.html.twig

Tai yra norimos knygos atvaizdavimo failas, kur pateikiamas knygos paveikslukas, norimi parametrai ir atvaizduojami komentarai JavaScript pagalba.

## 5.4. Media

### 1. DefaultController.php:

- `FileUploadHandler`

Tai yra knygų įkėlimo į atitinkamą direktoriją metodas. Taigi iš pradžių sukuriamas masyvas su „false“ reikšme ir kintamasis, kuris gauna įkeltą failą. Tada užkoduojamas failo pavadinimas ir nurodoma direktorija, kur įkelti failai bus talpinami ir jeigu tokio failo įkelta dar nebuvo, leidžiama įkelti. Taip pat įkėlus failą jo užkoduotas pavadinimas yra įkeliamas į duomenų bazę į „Media“ objektą. Ir tada įkėlus failą tas pradžioj paminėtas masyvas tampa „true“ ir gauna failo pavadinimą. Pabaigoje tas kintamasis grąžinamas „JsonResponse“ būdu.

## 5.5. Notes

### 1. NoteController.php

- newAction

Komentaro sukūrimas nekviečiant formos failo, visi laukai aprašomi čia.

Pabaigoje komentarai išsaugojami duomenų bazėje ir perduodami į atvaizdavimo failą.

### 2. DefaultController.php

- getNotesAction

## 6. Naudotojo instrukcija

Tam, kad galima būtų paleisti mano programą reikės:

- Kokio nors lokalaus internetinio serverio (pvz XAMPP, WAMP)
- Internetinės naršyklės
- MySql duomenų bazės palaikymą
- PHP 5.\* versija arba daugiau
- Symfony 3.\* versija
- Turėti instaliuotą „Composer“
- Turėti visus reikiamus failus, kurie bus pateikti laikmenoje su programine dalimi

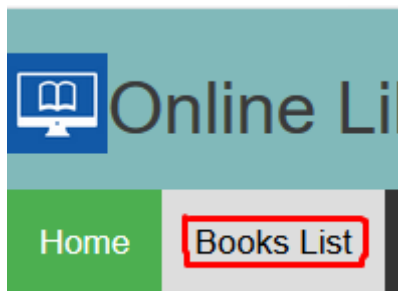
Realizavimo priemonės:

- Sistema programuojama PHP kalba
- Naudojama programa PhpStorm

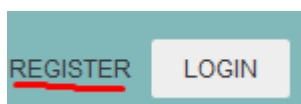
Taigi dabar pateiksiu naudotojo instrukciją skirtingų lygių naudotojams.

Pradėsiu nuo neregistruoto naudotojo, kuriam reikės mažiausiai instrukcijų, nes jis negalės pasinaudoti daugeliu funkcionalumų:

- Užėjęs į sistemą naudotojas iškart matys namų puslapį, kur bus trumpas Internetinės skaityklos aprašymas: (įkelti pav)
- Viršutiniame meniu galės nueiti į knygų sąrašą: (pav)



- Jame galės peržvelgti knygas, jų aprašymą, įvertinimą, komentarų skaičių. Taip pat galės ir peržiūrėti visus komentarus, bet aišku nei pridėti, nei redaguoti jų negalės
- Taip pat jis negalės nei pridėti nei skaityti knygų, tik susipažinti su jomis
- Aišku, jeigu norės išbandyti daugiau, turi galimybę užsiregistruoti, o tai galės padaryti spaudžiant register mygtuką pagrindiniame meniu dešinėje: (pav)





Ten naudotojui reikės įvesti atitinkamus laukus ir paspausti register mygtuką, norint užsiregistruoti:

## Register!

**Username**

**Email**

**Name**

**Password**

**Repeat Password**

Register

Toliau bus prisijungusio naudotojo instrukcija:

- Užėjęs į sistemą naudotojas iškart matys namų puslapį, kur bus trumpas Internetinės skaityklos aprašymas: (ikelti pav)
- Galės nueiti į knygų sąrašą su mažomis galimybėmis, o jeigu norės daugiau galės prisijungti spaudžiant login mygtuką(pav) :

## Login!

**Username**

**Password**

Login  Register

- Prisijungęs naudotojas bus iškart nukreipiamas į knygų sąrašą (pav):

### Books List

Id	Title	Author	Release year	Publisher	Genre	Language	Pages number	Description	Comments count	Rating	Readers count	Last read page
1	<a href="#">Vejo vardas</a>	Patrick Rothfuss	11/07/2012			English	450	Very good book	3	10	4	342
2	<a href="#">Ziedu valdovas</a>	J.R.R. Tolkien	02/02/2012			English	420	Fantasy book about hobits				

- Ten jis galės pridėti naujos knygos aprašymą su visais parametrais (pav):

### Books List

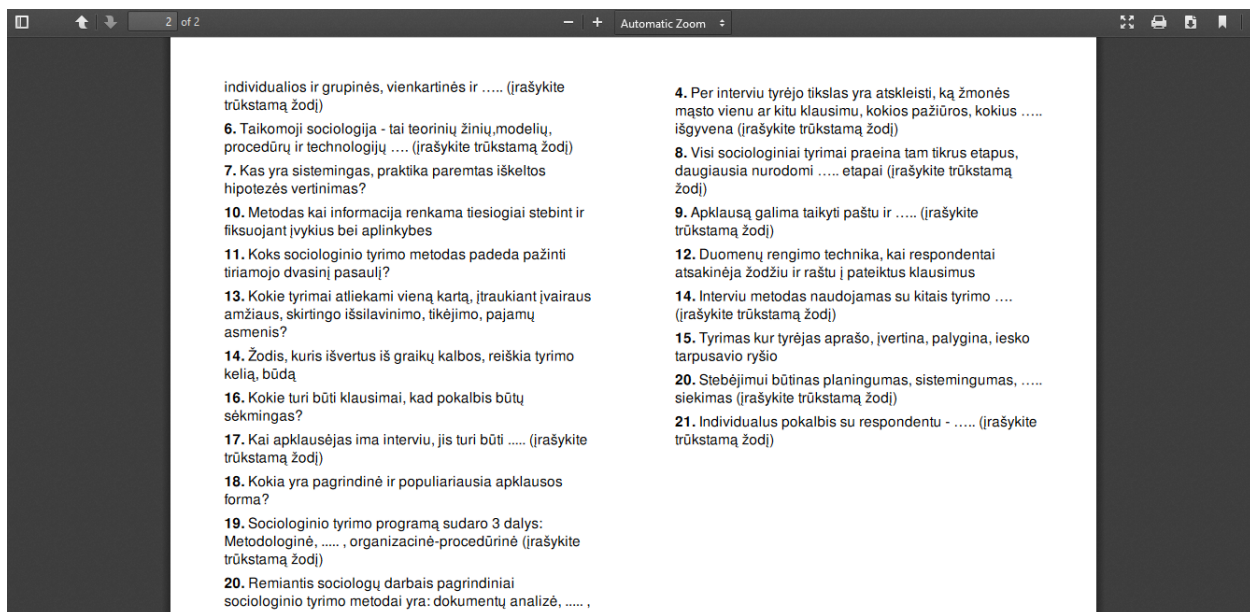
Id	Title	Author	Release year	Publisher	Genre	Language	Pages number	Description	Comments count	Rating	Readers count	Last read page	
													<a href="#">+</a>

- Įkeliant knygą bus perkeliamas jo vardas ir pavardė, šiuo atveju naudotojas negalės įkelti kitų autorių knygų
- Įkelti knygą (pav):

Drop files here to upload


Save

- Vertinti knygas (pav):
- Komentuoti knygas (pav):
- Skaityti knygas (pav):

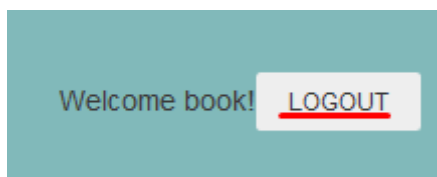


- Redaguoti ir trinti savo įkeltų knygų aprašymus (pav):

### Books List

Id	Title	Author	Release year	Publisher	Genre	Language	Pages number	Description	Comments count	Rating	Readers count	Last read page	+
1	<a href="#">Vėjo vardas</a>	Patrick Rothfuss	11/07/2012			English	450	Very good book	3	10	4	342	 
2	<a href="#">Ziedu valdovas</a>	J.R.R. Tolkien	02/02/2012			English	420	Fantasy book about hobits					 

- Norint pasikeisti savo prisijungimo vardą arba slaptažodį, naudotojas galės tai padaryti pagrindiniame meniu užėjus į savo paskyrą (pav):
- Padaręs viską ką norėjo naudotojas galės atsijungti paspaudus atitinkamą mygtuką (pav):



Galų gale liko administratoriaus instrukcija (čia aprašau tik tuos punktus, kurių neturi kitų lygių naudotojai, visą tai ką gali kiti administratorius aišku taip pat gali):













- Prisijungimas bus toks pat kaip eilinių naudotojų, o naujo administratoriaus registracija nebus galima, nes bus tik vienas pagrindinis administratorius, kuris bus atsakingas už visą sistemą
- Administratorius gali užėjus į knygų sąrašą redaguoti ir trinti ne tik savo knygas, o ir bendrai visų naudotojų įkeltas (pav):

### Books List

Id	Title	Author	Release year	Publisher	Genre	Language	Pages number	Description	Comments count	Rating	Readers count	Last read page	+
1	<a href="#">Vėjo vardas</a>	Patrick Rothfuss	11/07/2012			English	450	Very good book	3	10	4	342	 
2	<a href="#">Ziedu valdovas</a>	J.R.R. Tolkien	02/02/2012			English	420	Fantasy book about hobits					 



- Taip pat galės trinti nekorektiškus komentarus (pav):
- Pagrindiniame meniu paspaudus Users lauką atsidarys visų užsiregistravusių naudotojų sąrašas, kur administratorius galės trinti ir redaguoti ką panorėjęs (pav):

### Users List

Id	Email	Username	Name	Surname	
1	user@gmail.com	user			  
2	admin@gmail.com	admin			  
3	hnioba@gmail.com	hnioba			  
4	labas@gmail.com	labas			  

- Taip administratorius iš šio sąrašo galės užėti į kitą naudotojo paskyrą:

## Users List

Id	Email	Username	Name	Surname	
1	user@gmail.com	user			  

- Kaip ir visi kiti naudotojai darbo pabaigoje administratorius gali atsijungti iš sistemos

## **7. Išvados**

Šio baigiamojo darbo metu man sėkmingai pavyko įgyvendinti išsikeltus tikslus ir užsibrėžtas užduotis. Pavyko:

- Leisti neprisijungusiems naudotojams peržvelgti pagrindinę knygų informaciją
- Suteikti galimybę užsiregistruoti naujiems naudotojams
- Prisijungimo galimybė
- Leisti naudotojams laisvai skaityti knygas
- Leisti naudotojams įkelti savo knygas
- Leisti naudotojams vertinti knygas
- Leisti naudotojams komentuoti knygas
- Leisti naudotojams redaguoti ir trinti savo knygas
- Leisti administratoriui visas teises redaguoti ir trinti visą turinį – ir knygas, ir komentarus ir pačius naudotojus