|  |  |  |
| --- | --- | --- |
|  | УНИВЕРЗИТЕТ У НОВОМ САДУ  **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ** |  |

Вук Радмиловић

**Систем за препоруке при раду азила за животиње**

ЗАВРШНИ РАД

- Основне академске студије -

Нови Сад, 2024.

|  |  |
| --- | --- |
|  | УНИВЕРЗИТЕТ У НОВОМ САДУ ⚫ **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**  21000 НОВИ САД, Трг Доситеја Обрадовића 6 |
| **КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА** |

|  |  |  |  |
| --- | --- | --- | --- |
| Редни број, **РБР**: | |  | |
| Идентификациони број, **ИБР**: | |  | |
| Тип документације, **ТД**: | | Монографска публикација | |
| Тип записа, **ТЗ**: | | Текстуални штампани документ | |
| Врста рада, **ВР**: | | Дипломски рад | |
| Аутор, **АУ**: | | Вук Радмиловић | |
| Ментор, **МН**: | | доц. Др Синиша Николић, ФТН Нови Сад | |
| Наслов рада, **НР**: | | Систем за препоруке при раду азила за животиње | |
| Језик публикације, **ЈП**: | | Српски | |
| Језик извода, **ЈИ**: | | Српски/енглески | |
| Земља публиковања, **ЗП**: | | Србија | |
| Уже географско подручје, **УГП**: | | Војводина | |
| Година, **ГО**: | | 2024 | |
| Издавач, **ИЗ**: | | Ауторски репринт | |
| Место и адреса, **МА**: | | Факултет техничких наука, Трг Доситеја Обрадовића 6, Нови Сад | |
| Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога) | | бр. поглавља 7/страница 63/цитата 65/табела 0/слика 41 | |
| Научна област, **НО**: | | Електротехничко и рачунарско инжењерство | |
| Научна дисциплина, **НД**: | | Системи базирани на знању | |
| Предметна одредница/Кључне речи, **ПО**: | | системи базирани на правилима, препорука кућних љубимаца, азили за животиње, представа доменског знања | |
| **УДК** | |  | |
| Чува се, **ЧУ**: | | Библиотека ФТН, Трг Доситеја Обрадовића 6, Нови Сад | |
| Важна напомена, **ВН**: | |  | |
| Извод, **ИЗ**: | | У раду је описан систем који пружа препоруке при раду азила за животиње укључујући и препоруке кућних љубимаца потенцијалним будућим власницима. Коришћени су системи базирани на правилима. Детаљно је описана спецификација и имплементација система. Приказано је коришћење апликације од стране радника и потенцијалног власника кућног љубимаца. | |
| Датум прихватања теме, **ДП**: | |  | |
| Датум одбране, **ДО**: | |  | |
| Чланови комисије, **КО**: | Председник: | др Јелена Сливка, ван. проф., ФТН Нови Сад |
|  | Члан: | др Марко Марковић, доц., ФТН Нови Сад |
|  | Члан: |  | Потпис ментора |
|  | Члан, ментор: | др Синиша Николић, доц., ФТН Нови Сад |  |

|  |  |
| --- | --- |
|  | UNIVERSITY OF NOVI SAD ⚫ **FACULTY OF TECHNICAL SCIENCES**  21000 NOVI SAD, Trg Dositeja Obradovića 6 |
| **KEY WORDS DOCUMENTATION** |

|  |  |  |  |
| --- | --- | --- | --- |
| Accession number, **ANO**: | |  | |
| Identification number, **INO**: | |  | |
| Document type, **DT**: | | Monographic publication | |
| Type of record, **TR**: | | Textual material | |
| Contents code, **CC**: | | Bachelor thesis | |
| Author, **AU**: | | Vuk Radmilovic | |
| Mentor, **MN**: | | Sinisa Nikolic, PhD | |
| Title, **TI**: | | Recommendation system for animal shelters | |
| Language of text, **LT**: | | Serbian | |
| Language of abstract, **LA**: | | Serbian/English | |
| Country of publication, **CP**: | | Serbia | |
| Locality of publication, **LP**: | | Vojvodina | |
| Publication year, **PY**: | | 2024 | |
| Publisher, **PB**: | | Author’s reprint | |
| Publication place, **PP**: | | Faculty of Technical Sciences, Trg Dositeja Obradovića 6, Novi Sad | |
| Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes) | | no. of chapters 7/pages 63/quotes 65/tables 0/pictures 41 | |
| Scientific field, **SF**: | | Electrical and computer engineering | |
| Scientific discipline, **SD**: | | Knowledge based systems | |
| Subject/Key words, **S**/**KW**: | | Rule based systems, pet recommendation, pet shelters, representation of domain knowledge | |
| **UC** | |  | |
| Holding data, **HD**: | | Library of Faculty of Technical Sciences, Trg Dositeja Obradovića 6, Novi Sad | |
| Note, **N**: | |  | |
| Abstract, **AB**: | | This paper describes a recommendation system for running animal shelters which includes pet recommendation for potential pet owners. It utilizes rule-based systems. The system’s specification and implementation are described in detail. Use cases for both workers and potential pet owners are presented. | |
| Accepted by the Scientific Board on, **ASB**: | |  | |
| Defended on, **DE**: | |  | |
| Defended Board, **DB**: | President: | Jelena Slivka, PhD, аssoc. prof., FTN Novi Sad |
|  | Member: | Marko Marković, PhD, assist. prof., FTN Novi Sad |
|  | Member: |  | Menthor's sign |
|  | Member, Mentor: | Siniša Nikolić, PhD, assist. prof., FTN Novi Sad |  |

|  |  |  |
| --- | --- | --- |
|  | УНИВЕРЗИТЕТ У НОВОМ САДУ ⚫ **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**  21000 НОВИ САД, Трг Доситеја Обрадовића 6 | Број: |
|  |
| **ЗАДАТАК ЗА ЗАВРШНИ РАД** | Датум: |
|  |

*(Податке уноси предметни наставник - ментор)*

| Студијски програм: | **Софтверско инжењерство и информационе технологије** | | |
| --- | --- | --- | --- |
| Студент: | **Вук Радмиловић** | Број индекса: | **SV73/2020** |
| Степен и врста студија: | **Основне академске студије** | | |
| Област: | **Електротехничко и рачунарско инжењерство** | | |
| Ментор: | **доц. др Синиша Николић** | | |
| НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА  ИЗДАЈЕ СЕ ЗАДАТАК ЗА ЗАВРШНИ РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА:   * проблем – тема рада; * начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна; | | | |

**НАСЛОВ ЗАВРШНОГ РАДА:**

|  |
| --- |
| **Систем за препоруке при раду азила за животиње** |

**ТЕКСТ ЗАДАТКА:**

|  |
| --- |
| Описати систем за препоруке при раду азила за животиње који укључује и препоруке кућних љубимаца потенцијалним будућим власницима. Специфицирати и описати архитектуру система. Анализирати правила за анализу рада азила и доношење закључака. Документовати решење. |

|  |  |
| --- | --- |
| Руководилац студијског програма: | Ментор рада: |
|  |  |

|  |
| --- |
| Примерак за: - Студента; - Ментора |

Садржај

[1. Увод 1](#_Toc176525271)

[2. Стање у области 3](#_Toc176525272)

[2.1. Научна област 3](#_Toc176525273)

[2.2. Апликације 5](#_Toc176525274)

[*2.2.1.* *Buzz to the rescues* 5](#_Toc176525275)

[*2.2.2.* *Pawlytics* 8](#_Toc176525276)

[*2.2.3.* *Shelterluv* 8](#_Toc176525277)

[*2.2.4.* *PDSA Get PetWise* 9](#_Toc176525278)

[*2.2.5.* *Be chewy pet quiz* 10](#_Toc176525279)

[2.3. Пресек стања 10](#_Toc176525280)

[3. Коришћене технике и технологије 13](#_Toc176525281)

[*3.1.* *React* 13](#_Toc176525282)

[*3.2.* *Spring Boot* 13](#_Toc176525283)

[*3.3.* *Drools* 15](#_Toc176525284)

[4. Спецификација 19](#_Toc176525285)

[4.1. Дијаграм случајева коришћења 19](#_Toc176525286)

[4.2. Дијаграм класа 20](#_Toc176525287)

[4.3. Дијаграм секвенци 22](#_Toc176525288)

[5. Имплементација 23](#_Toc176525289)

[5.1. Клијентска апликација 23](#_Toc176525290)

[5.2. Серверска апликација 25](#_Toc176525291)

[5.3. Моделски део апликације 28](#_Toc176525292)

[5.4. Drools апликација 31](#_Toc176525293)

[6. Приказ имплементираног решења 37](#_Toc176525294)

[6.1. Радник у азилу за животиње 37](#_Toc176525295)

[6.2. Потенцијални власник кућног љубимца 41](#_Toc176525296)

[7. Закључак 45](#_Toc176525297)

[Литература 47](#_Toc176525298)

[Биографија 53](#_Toc176525299)

# Увод

Технолошке иновације савременог доба допринеле су порасту друштвене дистанце и отуђења у животима људи. Ово је довело до све већег осећаја усамљености и слабљења друштвених веза многих људи [[1]](#Reference1) [[2]](#Reference2). Кућни љубимци су добар начин да се човеку врати потреба за социјалним контактом и успостави емпатија према осталим живим бићима. Показало се и да поседовање кућних љубимаца знатно помаже деци у развоју друштвених вештина, поред повећавања самопоуздања и самопоштовања [[3]](#Reference3) [[4]](#Reference4).

Истовремено, азили за животиње постају све већи а њихови капацитети су често попуњени [[5]](#Reference5). Одржавање азила је изазовно због недовољног финансирања и недостатка радне снаге. Око 35% мачака и паса живе на улици или у азилима [[6]](#Reference6). На улицама су склони болести и нападима од стране других животиња или људи. Као домаће животиње не могу живети прикладне животе без људи.

Ови проблеми би се могли делимично решити системом који би олакшао рад азила за животиње. Систем за препоруке при раду азила за животиње би тежио да олакша радницима уобичајене радње и обавезе, дајући им више времена да се посвете животињама. Аутоматизацијом послова и препорукама учинио би рад азила ефикаснијим и ефектнијим. Тиме би се посао могао вршити и са мање радника, смањујући трошкове рада азила и омогућавајући да се та средства преусмере на ширење капацитета и маркетинг.

Доношење одлуке којег кућног љубимца удомити је вишефакторско и укључује потребно доменско знање о томе која животиња је прикладна за какве људе у каквим условима живота. Систем би могао да обухвати и модул за препоруку кућних љубимаца, чиме би се будућим власницима олакшао избор новог кућног љубимца.

Циљ овог рада је демонстрација употребе Система базираних на знању (СБНЗ) [[7]](#Reference7) у склопу веб апликације за решавање ових проблема. Апликацију могу користити радници у азилу за животиње и потенцијални власници кућних љубимаца који траже препоруке. Систем доноси одлуке и пружа одговарајуће препоруке на основу унетог доменског знања и дефинисаних правила. Доменско знање биће реализовано помоћу правила дефинисаних у *Drools* [[8]](#Reference8) алату.

У следећем поглављу је анализом сличних решења приказано стање у области. У трећем поглављу описане су технике и технологије коришћене при изради система. Четврто поглавље садржи спецификацију решења представљену кроз одговарајуће дијаграме. Детаљи конкретне имплементације су приказани у петом поглављу. Шесто поглавље садржи приказ имплементираног решења кроз очи корисника. У последњем поглављу се налазе донесени закључци и уочен простор за будући рад.

# Стање у области

Сходно проблемима наведеним у уводу, многи људи се данас одлучују за усвајање кућних љубимаца. Многи људи одустају од својих кућних љубимаца и напуштају их. Број кућних љубимаца на улицама и азилима се повећава у многим градовима широм света [[9]](#Reference9) [[10]](#Reference10) [[11]](#Reference11). Упркос томе, јавно доступне апликације за препоруке при управљању азила и даље не постоје. Апликације које се баве препоруком кућних љубимаца постоје, али су махом неозбиљне и ограничене.

У овом раду описан је систем који је реализован употребом експертских система базираних на правилима (*Rule-based systems*) [[12]](#Reference12). У традиционалним системима, цела логика програма је директно смештена у коду. Због тога је теже и компликованије променити је. У динамичним системима, где су честе измене пословне логике, овакав приступ представља значајан проблем. Системи базирани на правилима измештају знање из кода у екстерно написана правила. Тиме чак и људи који нису програмери могу да мењају правила и да се баве доменским знањем, док су програмери фокусирани на развој саме апликације.

## Научна област

Постоје многи примери употребе експертских система базираних на правилима.

Рад [[13]](#Reference13) представља дизајн експертског система који има за циљ да пацијенту пружи основу за дијагнозу неких очних болести. Пацијент је анкетиран а систем на основу одговора закључује дијагнозу. Систем чува чињенице и закључке, и обрађује базу података како би извукао правила која допуњују базу знања.

Рад [[14]](#Reference14) представља експертски систем који се може користити за успешну дијагнозу проблема са храњењем код деце. Систем је дизајниран и имплементиран да помаже родитељима у дијагностици ових проблема и да им пружа препоруке о томе како да се носе са децом.

Рад [[15]](#Reference15) јесте опис експертског система који помаже људима да детектују здравствене проблеме везане за зависност од видео игара. Систем пружа информације о проблему а нуди и решења.

Рад [[16]](#Reference16) описује систем базиран на правилима који помаже у дијагнози проблема са ушима и чулом слуха. Систем ради тако што се пацијенти анкетирају са циљем добијања дијагнозе и препорука за брзо лечење.

Рад [[17]](#Reference17) представља експертски систем који се може користити за успешну дијагнозу бола у леђима. Систем успешно дијагностификује проблеме и предлаже начине лечења.

Многи радови су написани о теми проблема везаних за азиле за животиње.

Рад [[18]](#Reference18) се бави проблемима везаним за азиле у Северној Америци. Говори о проблемима који се јављају за људе због повећане количине животиња на улицама, попут дивљих животиња које улазе у урбана подручја и приградска насеља због тога што тамо имају дивље мачке као плен. Дискутују се проблеми са здрављем и стањем животиња због пренатрпаних азила којима су попуњени капацитети. Радници немају довољно времена да социјализују животиње, дресирају их ни лече. Лекови притом често нису ни доступни азилу, па се ни не могу лечити све болести. Радника нема довољно а често су волонтери који нису обучени ни образовани. Многи ови проблеми узроковани су недостатком финансија. Због свих њих, многе животиње живе стресније животе у азилима и понашање им се погоршава. Због тога је теже наћи им дом а и када се пронађе тешко је задржати их тамо јер се често врате назад у азил након неког времена због лошег понашања. Многи ови проблеми могли би бити донекле надомештени добрим системом за препоруке при раду азила за животиње.

Рад [[19]](#Reference19) представља проблеме везане за збрињавање паса у азилима за животиње. Описује лоше стање у многим азилима због практичних ограничења. У многим азилима пси су изложени повећаним психолошким стресом. То се дешава због нагле промене рутине која ни у азилу није конзистентна и чињенице да су далеко од куће и власника. Такође, они су окружени странцима и често су у окружењу пуном непредвидиве и интензивне буке. Због тога многи пси који су остављени азилима због лошег понашања постају још гори. Рад представља препоруке и начине на које се може управљати азилом како би се стање побољшало.

Постоји више радова који се баве веб или мобилном апликацијом која би помогла азилима и потенцијалним усвајачима да сазнају више о азилу и усвоје животињу. Неки од њих укључују и могућности управљања азилом у смислу административног менаџмента самог азила, али се не баве препорукама нити користе системе базиране на знању.

Рад [[20]](#Reference20) представља мобилну апликацију која пружа могућност прегледа различитих азила и њихове понуде животиња за усвајање. Помаже потенцијалним власницима да ступе у контакт са азилима и закажу термин за преглед животиња и усвајање.

Рад [[21]](#Reference21) се бави апликацијом за информисање о азилима, животињама и бризи о њима са фокусом на добро корисничко искуство и леп дизајн. Укључује и анкету за проверу компатибилности са животињом, све у циљу повећања броја прикладно усвојених животиња.

Рад [[22]](#Reference22) пружа азилима могућност да се лакше представе на интернету. Омогућава им да прикажу своје животиње и подстакну лакше информисање о њима које би водило ка лакшем усвајању. Могуће је и вршити и промовисати донације азилу.

Рад [[23]](#Reference23) омогућава азилима да се региструју и пруже информације о животињама које збрињавају као и да админисративно прате стање азила и њихових радника. У питању је мобилна апликација која би требала да олакша рад азилима за животиње и омогући лакше информисање за потенцијалне власнике.

Радови који обрађују тему система за препоруке при раду азила за животиње у општем смислу попут овог нису пронађени. Ипак, постоји мноштво радова који се тичу различитих система који имају за циљ пружање помоћи радницима азила при одређеним радњама, за пројектовање азила и генерално о софтверу везаном за азиле за животиње и кућне љубимце.

Рад [[24]](#Reference24) омогућава радницима у азилу да лакше и ефикасније управљају азилом помоћу сензора животних услова. Услови за живот и физиолошко стање животиња се могу пратити преко веб сајта. Подаци се визуализују и графички приказују менаџерима и ветеринарима. Студија је рађена са псима али је циљ аутора проширити систем и за друге животиње.

Рад [[25]](#Reference25) се бави оптимизацијом дизајна и распореда кавеза у азилу за животиње. Циљеви су максимизација капацитета азила, минимизација тензије између паса и угодан приступ за особље и посетиоце азила. Оптимизација користи више критеријума и могуће је поштовати више различитих приоритета.

Рад [[26]](#Reference26) се бави застарелим технологијама коришћеним у многим азилима са циљем побољшања рада азила кроз ажурирање технологије. Аутори рада стреме ка смањењу броја успављених животиња кроз повећање ефикасности рада азила. Фокус је на бољем корисничком искуству и отклањању ограничења која су постојала при раду са животињама.

Рад [[27]](#Reference27) представља преглед техника за класификацију слика са циљем побољшања брзине усвајања животиња преко интернета. Аутоматизује препоруке азилима при постављању слика животиња како би се животиње брже усвојиле. Анализира које карактеристике мењају брзину усвајања и предвиђа брзину усвајања животиње.

Разматран је и један рад [[28]](#Reference28) који детаљно описује критеријуме по којима се може утврдити који кућни љубимци су прикладни за које власнике. Представљају се различите категорије по којима се услови, стање, искуство и жеље власника узимају у обзир при одабиру кућног љубимца. Иако није у питању апликација, рад је теоријски уско везан са препоруком кућних љубимаца, једним од модула система за препоруке при раду азила за животиње.

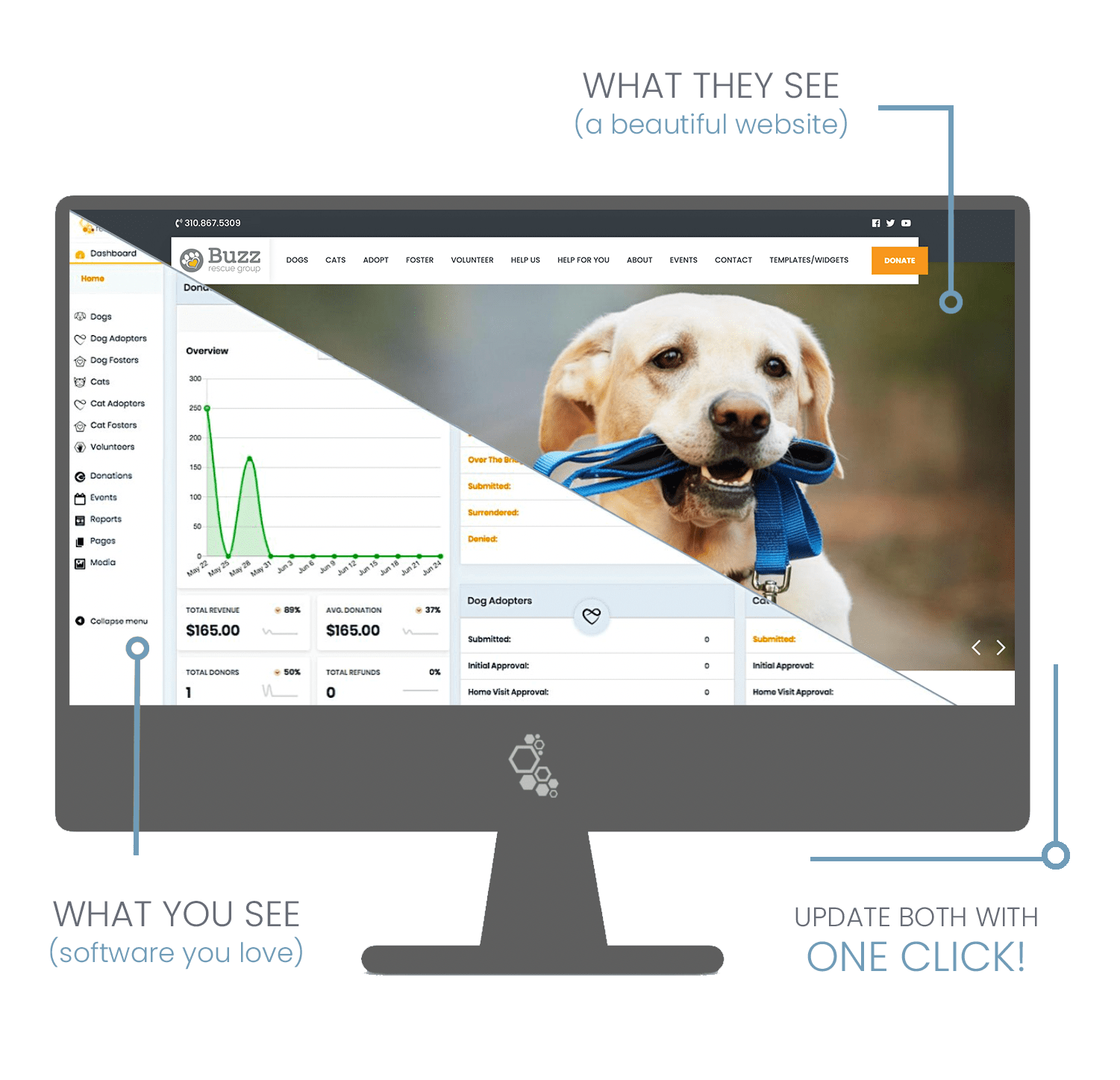
Пронађен је један рад [[29]](#Reference29) који се, између осталог, бави и препоруком кућних љубимаца. У питању је рад који се фокусира на олакшавање усвајања животиња из азила. Циљ рада је да смањи број успављених животиња у азилу. Део везан за препоруку кућних љубимаца функционише на основу анкете која је прилично једноставна и не дозвољава екстензивну проверу услова живота корисника нити његових жеља.

## Апликације

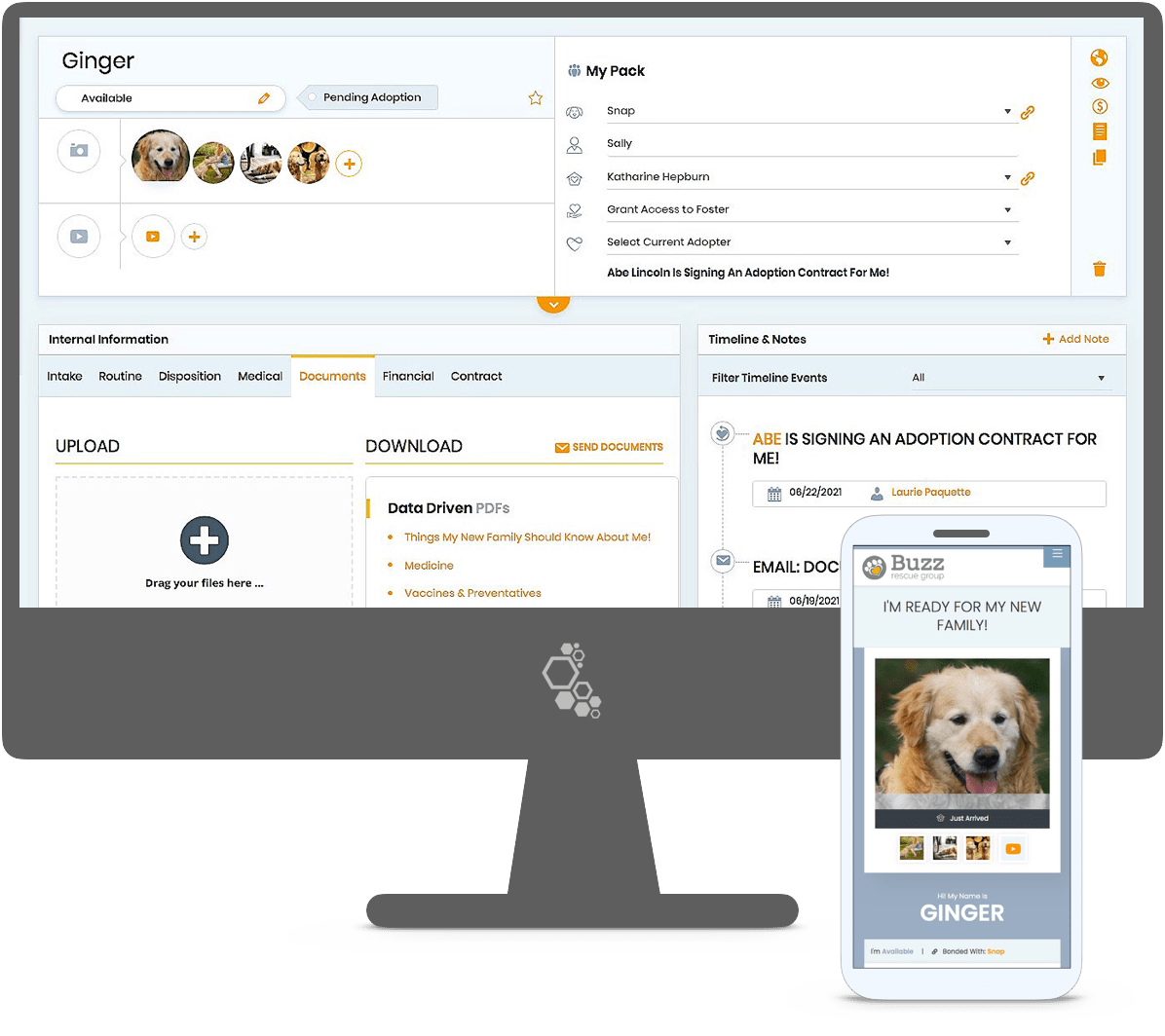
Апликација која укључује препоруке при управљању азила за животиње и препоруке кућних љубимаца за потенцијалне власнике није пронађена претрагом интернета. Није пронађена ниједна апликација која се бави препорукама при управљању азила за животиње. Пронађене су апликације које се фокусирају на управљање самим азилом из административне перспективе. Веб странице које се баве препоруком кућних љубимаца постоје, али су уже у обиму и/или неозбиљно урађене. Такође, нису интегрисане са системом за препоруке при управљању азила. У наредном делу разматрају се и описују неке од апликација за управљање азилима, и то: *Buzz to the rescues*, *Pawlytics* и *Shelterluv*. Након тога анализиране су и две апликације за препоруку кућних љубимаца: *PDSA Get PetWise* и *Be chewy* pet quiz.

### *Buzz to the rescues*

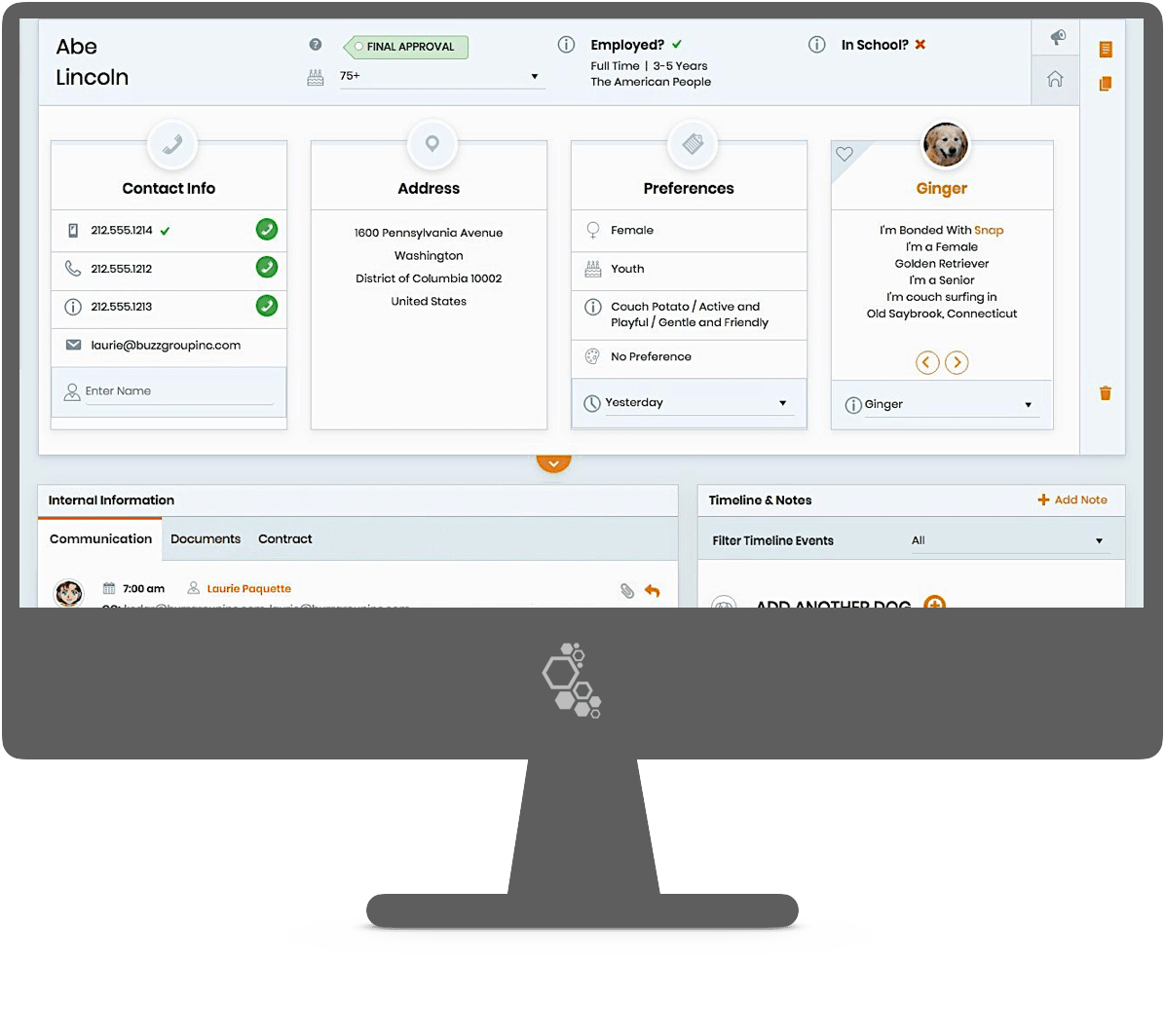
*Buzz to the rescues* [[30]](#Reference30) нуди систем за управљање азилима за животиње. Укључени су и сајт за управнике азила и сајт за кориснике који су заинтересовани за усвајање (Слика 2.1). Могуће је мењати и приказати информације о животињама, постављати слике, мењати њихове статусе, постављати и прегледати медицинску документацију као и различите специјалне потребе животиња (Слика 2.2). Доступна је и веб и мобилна апликација. Потенцијални усвојитељи могу регистровати своје налоге, навести своје преференције (Слика 2.3), прегледати животиње и пријављивати се за усвајање. Управитељи азила могу прегледати те профиле и пријаве и прихватити или одбијати их. Могуће је извршити потписивање уговора преко интернета. Систем нуди многе статистике и визуелизације везане за рад азила. Пружа преглед финансијских информација везаних за пословање азила (Слика 2.4) и нуди опцију донирања новца. Укључује и интеграцију са веб апликацијама *Petfinder* [[31]](#Reference31) и *Adopt a pet* [[32]](#Reference32), које се баве промоцијом усвајања животиња из различитих азила широм Северне Америке.



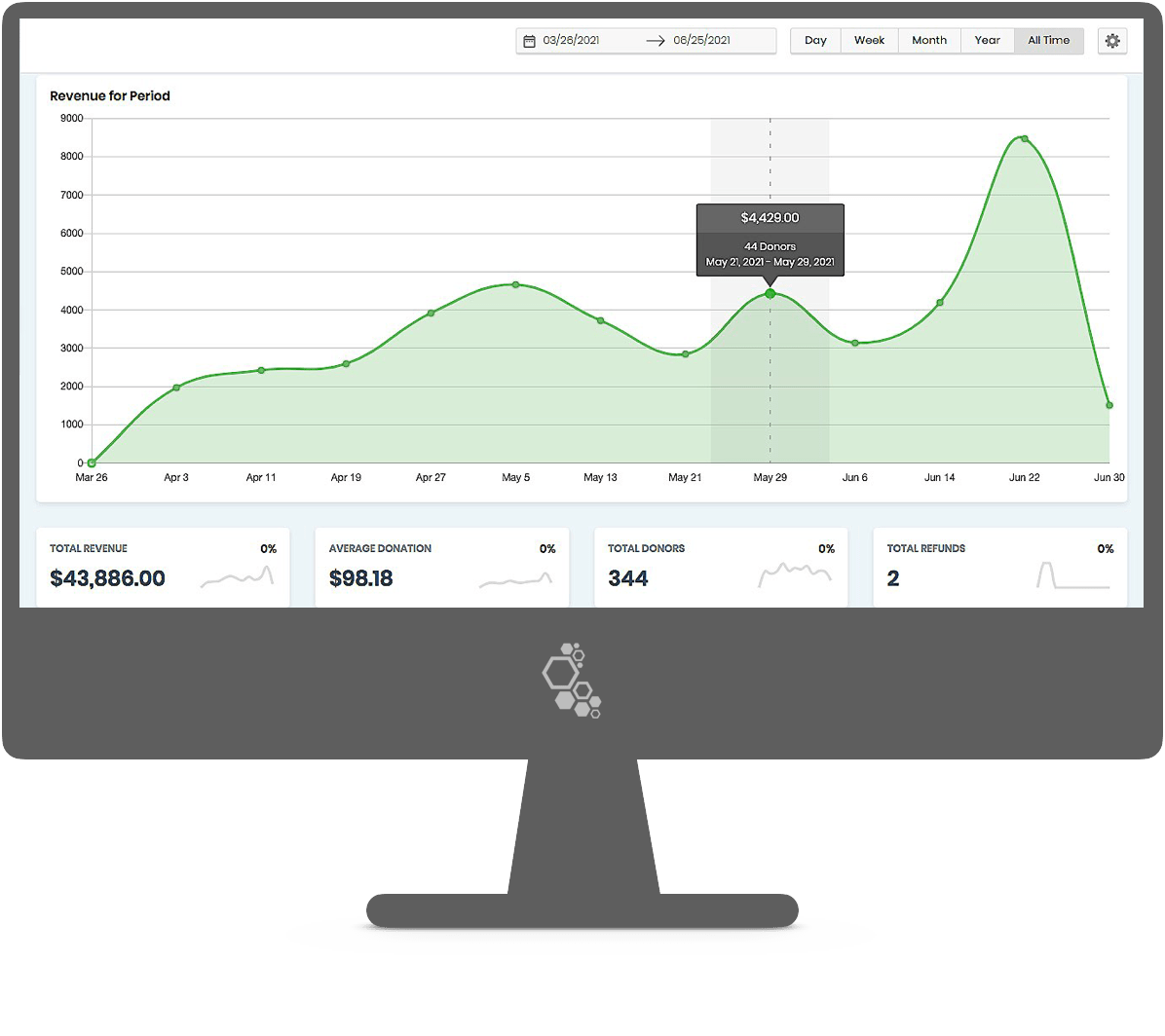
Слика 2.1 [[30]](#Reference30)



Слика 2.2 [[30]](#Reference30)



Слика 2.3 [[30]](#Reference30)



Слика 2.4 [[30]](#Reference30)

### *Pawlytics*

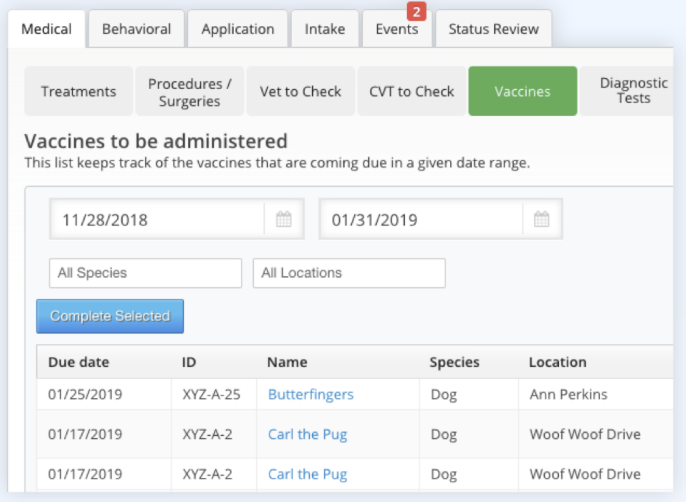
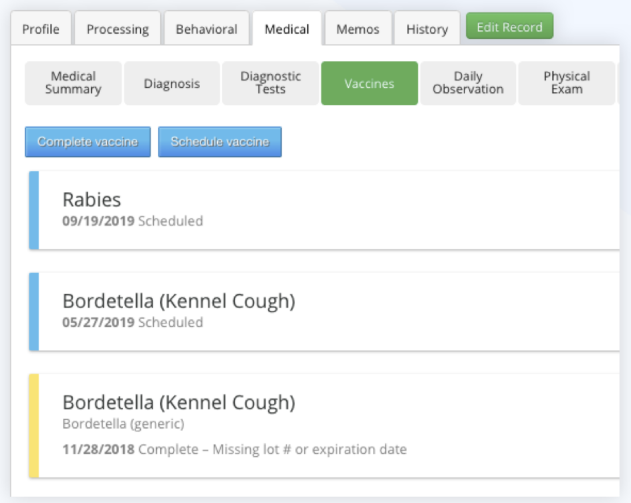
*Pawlytics* [[33]](#Reference33) пружа софтвер базиран на облаку за управљање азилима за животиње. Нуди једноставно управљање животињама и запосленима. Могуће је вршити пријаве за усвајање путем интернета. Омогућене су донације, животиње имају своје профиле (Слика 2.5), одржавају се архиви животиња и пружа се екстензивна корисничка подршка. Цена софтвера је веома компететивна. Могу је користити азили различитих величина и плаћа се само оно што се користи.



Слика 2.5 [[33]](#Reference33)

### *Shelterluv*

*Shelterluv* [[34]](#Reference34) представља решење за управљање азилима за животиње. Корисници и животиње имају своје профиле који се могу пунити са прилично обимном количином података. Имају свој *shelterpay* систем за лаке аутоматизоване пријаве за усвајање љубимаца. Корисници могу да врше донације, а сем пријаве за усвајање љубимаца могуће је и извршити целу трансакцију путем интернета, укључујући дигитално потписивање уговора и плаћање. Могуће је организовати задатке из различитих сфера посла за раднике унутар саме апликације како би организација посла била лакша (Слика 2.6). У понуди је и напредни систем за праћење медицинског стања животиња (Слика 2.7). Нуде брзо доступну корисничку подршку.

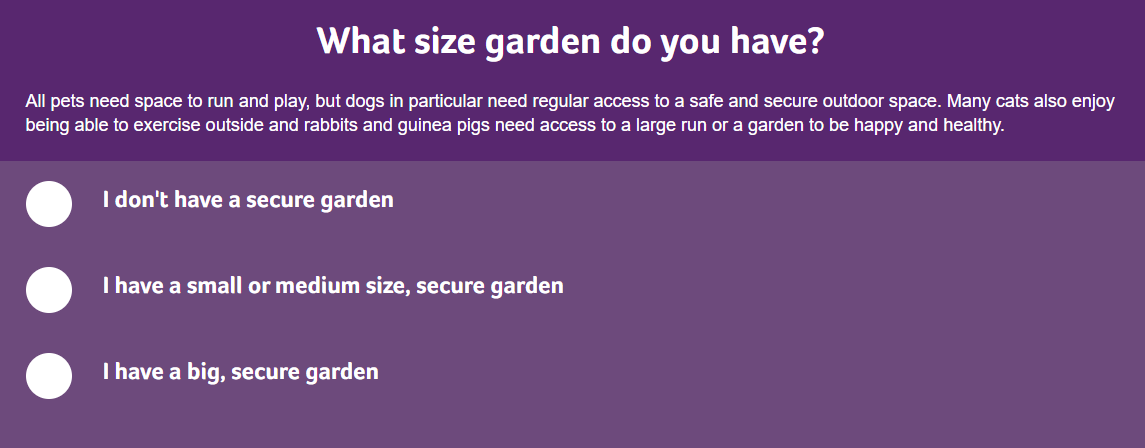
 

Слика 2.6 [[34]](#Reference34) Слика 2.7 [[34]](#Reference34)

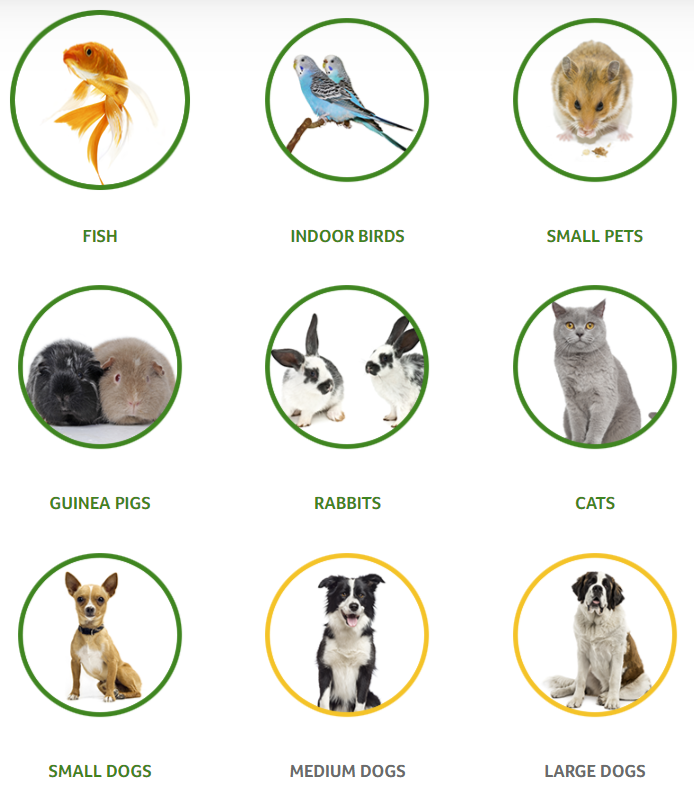
### *PDSA Get PetWise*

*PDSA* [[35]](#Reference35) је ветеринарска непрофитна организација која се труди да обезбеди бригу за болесне и повређене кућне љубимце чији власници не могу да приуште лечење. Баве се и промоцијом кућних љубимаца и њиховог здравља, као и образовањем деце и јавности у вези животиња.

*Get PetWise* [[36]](#Reference36) је део њиховог веб сајта који путем анкете сакупља информације о кориснику ради препоруке прикладног кућног љубимца. Анкета (Слика 2.8 ) се састоји од 5 питања у вези величине дворишта, нивоа активности, количине расположивог времена за кућног љубимца, новчаних средстава и нивоа знања о кућном љубимцу. Након тога излази препорука врста животиња које су пригодне за будућег власника. Постоји и листа животиња које би могле бити прикладне, када би потенцијални власник направио неке промене у свом окружењу и навикама. Препоручене животиње су уоквирене зеленом бојом, а оне које би биле препоручене са изменама жутом (Слика 2.9 ). Сајт је лепо дизајниран, корисничко искуство је одлично и сви одговори имају одговарајуће слике асоциране са њима. Број потенцијално препоручених врста животиња је прилично велик.



Слика 2.8 [[36]](#Reference36)



Слика 2.9 [[36]](#Reference36)

### *Be chewy pet quiz*

*Chewy* [[37]](#Reference37) је интернет продавница са намирницама и другим артиклима за кућне љубимце. *Be chewy* [[38]](#Reference38) је њихова подстраница која служи као блог који поставља различите чланке који могу бити занимљиви за људе који имају или желе кућне љубимце. Између осталог, та страница садржи и део са квизом [[39]](#Reference39) који има за циљ препоруку кућног љубимца.

Препорука се врши анкетом која садржи седам питања где сваки има пет потенцијалних одговора. Овде има и питања везаних за разлоге зашто корисник жели кућног љубимца као и о условима живота потенцијалног власника и спремности на трошење времена на дресирање и чишћење. На крају анкете, након потврде одговора, приказује се препорука једне врсте животиње која би требала бити најприкладнија за будућег власника.

## Пресек стања

Прве три описане апликације свој посао прилично добро обављају, али се ипак не баве препорукама при управљању азила за животиње, већ само пружају алате за администрацију самог азила. Притом треба имати у виду да и те апликације имају одређене недостатке у оквиру домена примене апликације. *Buzz to the rescues* ради само са псима и мачкама, док апликација представљена овим радом ради са низом различитих врста животиња. Њен фокус је на америчке азиле. Све три апликације залазе прилично детаљно у управљање азилима. Ипак, треба имати у виду да су ти системи незгодни за људе који нису вешти са руковањем компјутерима и сличним системима. Потребан је период учења и обуке за раднике који су често нестручни волонтери. *Shelterluv* се показује лошије од својих алтернатива по томе што им нису сви уписани подаци покривени извештајима и по томе што не рукују са информацијама о исхрани.

Последње две описане апликације се баве препоруком кућних љубимаца. Постојање таквих апликација је веома корисно, и те апликације нуде прилично добар дизајн и корисничко искуство. *PDSA Get PetWise* је темељније развијена апликација у односу на *Be Chewy* квиз, али ни једна ни друга апликација не пружа детаљан увид у профил потенцијалног власника кућног љубимца. Нуде слична питања која се налазе и у модулу описаном у овом раду, али не нуде многа питања која се такође налазе у модулу система описаног овим радом. Због недовољне количине питања за анализу *PDSA Get PetWise* има тенденцију да прикаже превелик број потенцијалних кућних љубимаца. Ово се дешава јер су питања преширока да би се избор сузио на мањи број животиња. Са друге стране, *Be Chewy* квиз даје само једну животињу као препоруку. Ипак, она је често скоро насумично одабрана јер систем нема довољно података да донесе правилну одлуку. Због тога се представља само једна од многих потенцијалних животиња које би одговарале власнику, и то често не она која би му највише одговарала.

Систем за препоруке при раду азила за животиње који је описан у раду тежи да интегрише све добре праксе примећене у датим апликацијама, али да пружи корисницима оно што фали на тржишту. Систем би требао не само да олакша администрацију азила већ и да пружи неке препоруке при њеном руковођењу. Тај систем би требао да има и интегрисан модул за препоруку кућних љубимаца који би био озбиљније и обимније урађен од алтернатива.

# Коришћене технике и технологије

Апликација се структурално састоји од серверске стране са интегрисаним системом базираним на знањем и клијентске стране. Клијентска страна развијена је помоћу *React*-а [[40]](#Reference40). Серверска страна је направљена користећи *Spring Boot* [*[41]*](#Reference41). Системи базирани на знању су имплементирани помоћу *Drools*-а. Комуникација између клијентске и серверске стране врши се користећи *REST (Representational State Transfer)* [[42]](#Reference42) архитектуру путем *HTTP (Hyper Text Transfer Protocol)* [[43]](#Reference43) протокола.

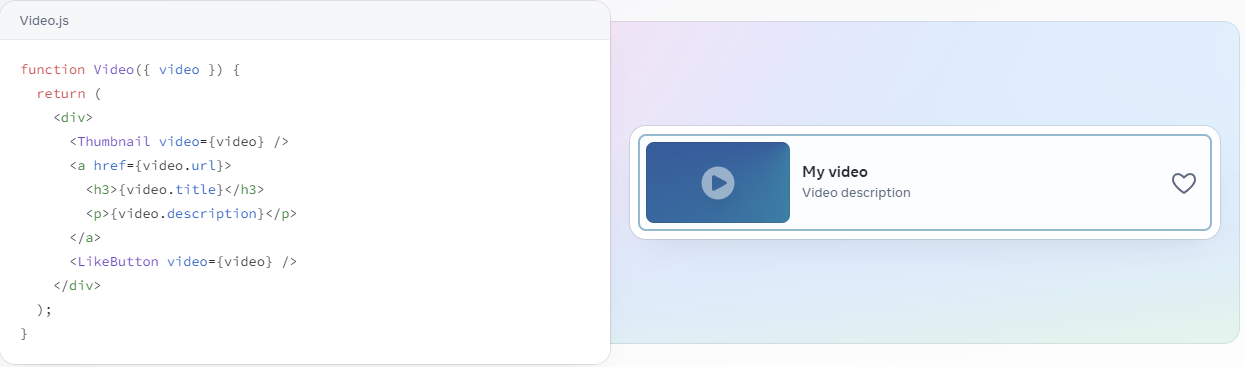
## *React*

*React* је *JavaScript* [[44]](#Reference44) радни оквир. Користи се за развијање веб апликација клијентске стране а створен је како би се лакше стварали комплексни кориснички интерфејси са скуповима података који се мењају кроз време [[40]](#Reference40). Циљеви развојног тима су били и скалабилност и одрживост.

*React* функционише на принципу креирања компонената који представљају делове корисничког интерфејса и који се могу поново употребљавати са истим или другим подацима [[40]](#Reference40). Користи *Virtual DOM* како би се ажурирање странице вршило на најефикаснији начин. Сваки пут када се страница ажурира, рачуна се најмањи број промена које је неопходно извршити [[40]](#Reference40). Служи се *JSX*-ом како би трансформисао *HTML* [*[45]*](#Reference45) код који је исписан у синтаксу која одговара *React*-у како би рендеровао елементе у *JavaScript*-у [[40]](#Reference40).

Све компоненте садрже нека своја својства и имају своје стање. Својства се не мењају, док је стање променљиво. Стање се иницијализује на почетку животног циклуса компоненте али се и мења током тог животног циклуса [[40]](#Reference40).

Слика 3.1 приказује пример кода написаног за *React* заједно са приказом који тај код генерише. Компоненте су представљене као функције које могу да примају аргументе и које враћају неки код који дефинише изглед и понашање те компоненте. Компоненте се могу угњеждавати, а постоје како предефинисане компоненте, тако и оне које корисници сами дефинишу или узимају из библиотека које су направили други корисници.



Слика 3.1 [[46]](#Reference46)

## *Spring Boot*

*Spring* је радни оквир за развој апликација написан у *Java* [[47]](#Reference47) језику. Подржава *Dependency Injection* [[48]](#Reference48) и Аспект-оријентисано програмирање [[49]](#Reference49). Садржи велики број модула који се могу користити, попут *Spring Data*, *Spring Security*, *Spring Cloud*, *Spring Web* [*[41]*](#Reference41)итд... У случају система за препоруке при раду азила за животиње, *Spring* је коришћен у контексту стварања веб сервиса, што и јесте њена главна улога данас. *Dependency Injection* је коришћен ради лакшег тестирања и веће флексибилности при раду.

*Spring* одликују одређене карактеристике. Има за циљ пружање избора на сваком нивоу дизајна, где је могуће кроз *Dependency Injection* променити провајдера перзистенције у било ком тренутку [[50]](#Reference50). Исти принцип важи и за друге промене попут инфраструктурних аспеката и интеграцијама са *API*-јима трећих страна [[50]](#Reference50). Приоритизује флексибилност, не намеће један начин да се ради било шта, подржава широк спектар потреба апликација из различитих перспектива [[50]](#Reference50). Нема превише значајне промене између верзија што значи да је лако одржавати апликације кроз промене верзија [[50]](#Reference50).

*Spring Boot* је *Spring* пројекат који олакшава стварање *Spring* апликација. Нуди конфигурације за различите често коришћене *Spring* апликације и укључује све библиотеке потребне за њих [[51]](#Reference51). Бира подразумевану конфигурацију, али опет нуди потпуну флексибилност јер се све библиотеке и сви алати могу заменити. Веб сервери у понуди су *Tomcat* [[52]](#Reference52) и *Jetty* [*[53]*](#Reference53) [[54]](#Reference54).

*Spring Boot* апликације се обично могу представити као спој четири слоја (Слика 3.2).



Слика 3.2 [[55]](#Reference55)

*Presentation Layer* (презентациони слој) обрађује HTTP захтеве, претвара JSON параметре у објекте, аутентификује захтеве и прослеђује их слоју пословне логике [[55]](#Reference55). *Business Layer* (слој пословне логике) управља свом пословном логиком [[55]](#Reference55). Садржи сервисне класе и користи услуге које пружа слој за приступ подацима [[55]](#Reference55). Врши и валидацију и ауторизацију [[55]](#Reference55). *Persistence Layer* (слој за приступ подацима) садржи сву логику за чување података и преводи објекте из сервисних класа у редове базе података [[55]](#Reference55). *Database* (слој базе података) изводи све операције везане за базу података попут уноса, читања, мењања и брисања података [[55]](#Reference55).

Архитектура *Spring Boot* апликације се може представити и кроз следећи дијаграм (Слика 3.3 )



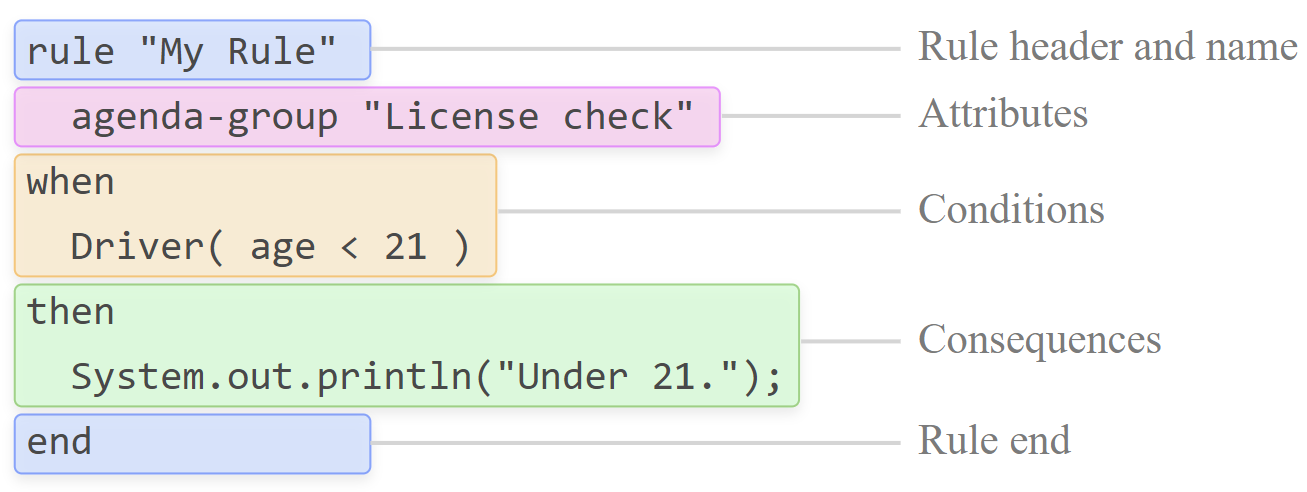
Слика 3.3 [[55]](#Reference55)

*Client* (клијент) шаље *HTTP* захтеве апликацији [[55]](#Reference55). Њих прима контролер, који пресликава објекте послате захтевом у оне прикладне за обрађивање од стране сервисног слоја [[55]](#Reference55). Сервисни слој садржи пословну логику, врши све потребне обраде и операције, и по потреби позива репозиторијуме који комуницирају са базом података. Затим се све то враћа до контролера који враћа одговор клијенту.

## *Drools*

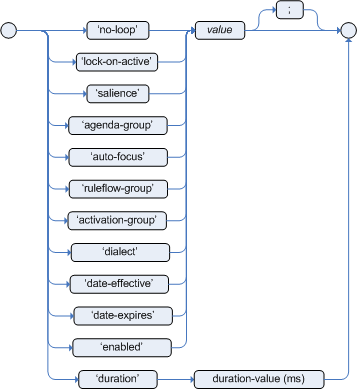
За имплементацију експертских система базираних на правилима одабран је *Drools*. *Drools* је напредан систем за управљање пословним правилима (*Business Rule Management System*) који укључује машину за правила (eng. *Rule engine*) која аутоматски спроводи правила користећи чињенице смештене у радној меморији и базу знања [[56]](#Reference56). Написан је у програмском језику *Java*. Доступан је као бесплатан софтвер. Подржава механизме уланчавања унапред (eng. *Forward-chaining*) и уланчавања уназад (eng. *Backward-chaining*) [[56]](#Reference56). Помоћу њих и на основу расположивих чињеница доносе се одлуке и изводе закључци. *Drools* такође подржава обраду комплексних догађаја (*CEP* - *Complex Event Processing*) [[8]](#Reference8), а ради утврђивања да ли чињенице из радне меморије испуњавају услове правила користи унапређен *Rete* алгоритам [[56]](#Reference56).

Правила у *Drools*-у су структуирана у два дела тј. две стране: леву страну (*LHS* - *left-hand side*), познату као *when* део, и десну страну (*RHS*  - *right-hand side*), познату као *then* део. *When* део представља услове који се морају испунити како би се извршиле акције дефинисане у *then* делу (Слика 3.4). У *then* делу могуће је додавати нове објекте у меморију (*insert*), мењати их (*modify, update*) или уклањати (*delete*). Свако правило започиње кљуном речју *rule* након које следи јединствени назив правила, а завршава се са кључном речју *end* која представља крај дефиниције правила.



Слика 3.4 [[57]](#Reference57)

Атрибути (eng. *Attributes*) представљају опционе смернице које утичу на извршавање правила на различите начине. Ови атрибути укључују: *agenda-group, salience, no-loop* и *lock-on-active* (Слика 3.5 )*.*



Слика 3.5 [[58]](#Reference58)

* *Agenda-group* (група агенде) групише правила омогућавајући извршавање само одабране групе правила
* *Salience* (приоритет) одређује редослед извршавања правила у случају постојања више важећих правила
* *No-loop* (без поновног извршавања) спречава да се већ примењено правило поново изврши у истом циклусу извршавања правила
* *Lock-on-active* (закључавање током активности) спречава активирање других правила док се одређено правило обрађује

Упити (енг. *Query*) су метод за претрагу радне меморије и добијање чињеница на основу задатих услова [[59]](#Reference59). Користе се за уланчавање уназад. Листинг 3.1 илуструје класичан пример упита који користи механизам уланчавања уназад.

Листинг . [59]

query isContainedIn( String x, String y )

Location( x, y; )

or

( Location( z, y; ) and isContainedIn( x, z; ) )

end

# Спецификација

Систем за препоруке при раду азила за животиње пружа веб апликацију која служи за помоћ при раду радницима азила за животиње. Један модул тог система јесте подсистем за препоруку кућних љубимаца. Он је у понуди за потенцијалне будуће власнике кућних љубимаца који траже оптималног кућног љубимца за њихове услове и жеље. Спецификација целокупног система је представљена различитим *UML* (*Unified Modeling Language*) [[60]](#Reference60) дијаграмима: дијаграмом случајева коришћења [[60]](#Reference60), дијаграмом класа [[60]](#Reference60) и дијаграмом секвенце [[60]](#Reference60).

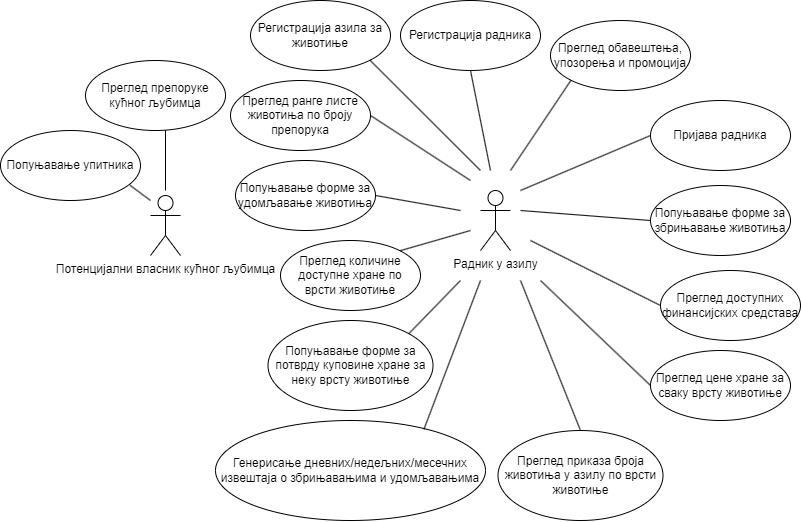
## Дијаграм случајева коришћења

Дијаграм случајева коришћења приказује како различити корисници интерагују са системом. Користи се како би се представиле функционалности система. Корисници система су повезани са функционалностима које користе. Омогућава лакше разумевање система и често се користи у раним фазама пројекта ради дефиниције и документације захтева. Слика 4.1 приказује дијаграм случајева коришћења система за препоруке при раду азила за животиње.

Корисници овог система су радници у азилу за животиње и потенцијални власници кућних љубимаца који траже препоруке.

Потенцијални власници кућних љубимаца који траже препоруке могу да попуњавају упитник од петнаест питања. На основу одговора на та питања, добијају препоруке кућних љубимаца.

Радник азила за животиње може да региструје азил за животиње за чије управљање су потребне препоруке. Затим може да се региструје и пријави на систем. Тада добија преглед контролне табле азила за животиње. Може да прегледа ранг листу животиња по броју препорука добијених на анкетама. Може да прегледа листу обавештења, упозорења и промоција. Може да попуњава форме за збрињавање и удомљавање животиња. Може да генерише дневне, недељне и месечне извештаје збрињавања и удомљавања животиња. Има преглед доступних финансијских средстава. Може да гледа приказ броја животиња у азилу по врсти животиња, заједно са количином доступне хране за ту врсту и цене те хране. Има могућност попуњавања форме за потврду куповине одређене количине хране за животиње.



Слика 4.1

## Дијаграм класа

Дијаграм класа представља структуру система кроз класе, њихове атрибуте, методе и међусобне односе. Атрибути представљају својства класа док методе представљају њихове функционалности. Међусобни односи класа, између осталог, могу бити: наслеђивање, асоцијација и агрегација. Дијаграм класа помаже у дизајнирању и разумевању структуре софтвера. Слика 4.2 представља дијаграм класа система за препоруке при раду азила за животиње.

Енумерација *AnimalType* представља врсту животиње (пас, мачка, птица...). Енумерација *AnimalBreed* представља расу животиње (пудла, чивава, ротвајлер...). Енумерација *PromotionOrResettlementType* означава да ли се ради о збрињавању (животиња улази у азил) или удомљавању (животиња одлази из азила, у нечији дом) животиња. Енумерација *ReportType* представља врсту извештаја, а може бити дневна, недељна или месечна.

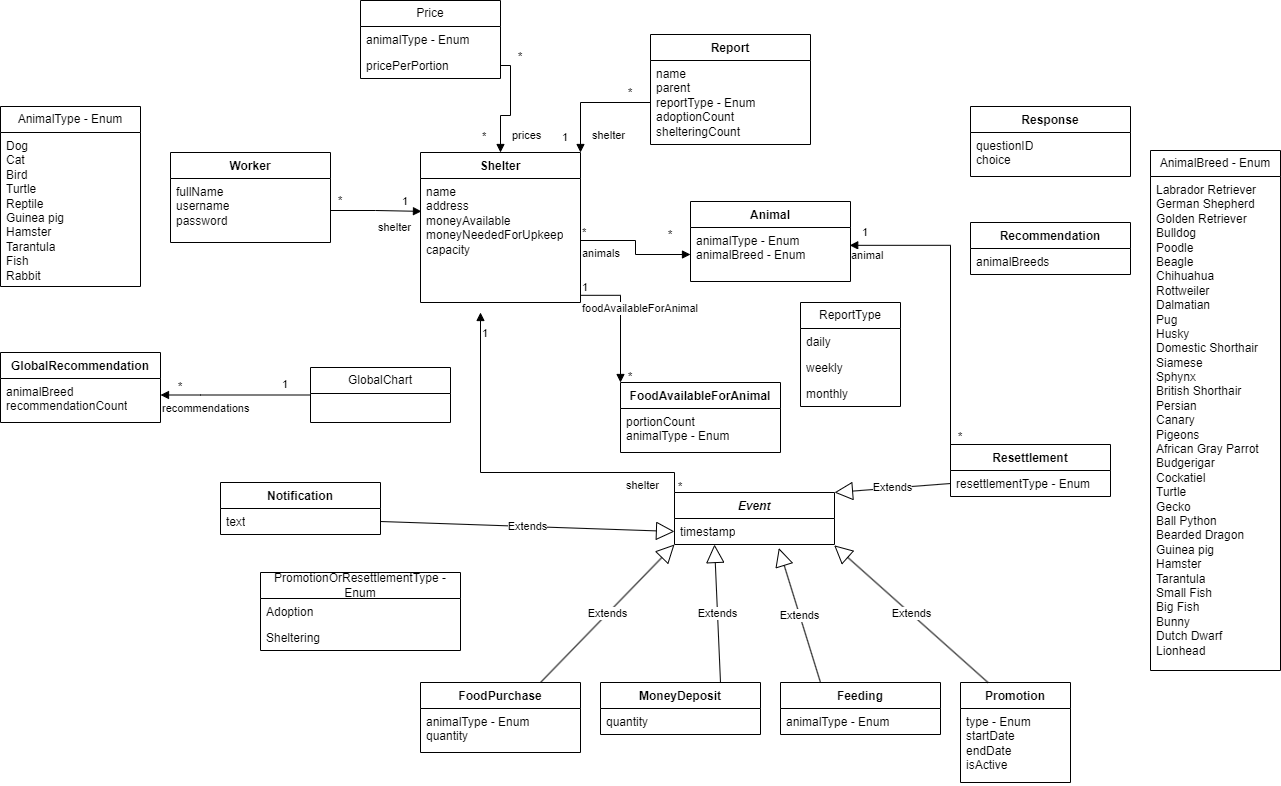
Класа *Shelter* представља азил за животиње. Садржи име и адресу азила, количину доступних финансијских средстава, срачунату количину финансијских средстава потребних за одржавање рада азила у следећа 3 дана и капацитет азила за животиње. Класе *Worker*, *Report*, *Animal*, *FoodAvailableForAnimal* и *Price* су директно везане за азил животиња.

Класа *Worker* представља радника и садржи његово пуно име, корисничко име и лозинку. Класа *Report* представља извештај о раду азила за животиње и садржи назив извештаја, „родитељски“ извештај (у случају дневног извештаја, родитељски је недељни, док је у случају недељног извештаја, родитељски извештај месечни), врсту извештаја, број удомљавања и број збрињавања.

Класа *Animal* представља животињу која је дефинисана својом врстом, расом и именом. Класа *FoodAvailableForAnimal* представља број порција хране који је доступан у азилу за одређену врсту животиње. Класа *Price* представља цену по порцији хране за одређену врсту животиње.

Класа *Event* представља догађај и садржи временску ознаку догађаја. Наслеђују је класе *Notification*, *Resettlement*, *FoodPurchase*, *MoneyDeposit*, *Feeding*, *Promotion*. Класа *Notification* представља обавештење и садржи текст обавештења. Класа *Resettlement* представља „пресељење“ животиње и описује да ли је у питању збрињавање или удомљавање. Класа *FoodPurchase* представља куповину хране и описује количину и врсту купљене хране. Класа *MoneyDeposit* представља пребацивање финансијских средстава азилу и садржи количину пребаченог новца. Класа *Feeding* представља догађај храњења животиња и садржи врсту животиња које су нахрањене. Класа *Promotion* представља промоцију збрињавања или удомљавања животиња и садржи свој почетни и крајњи датум заједно са ознаком која говори да ли је тренутно активна.

Класа *Response* представља одговор на питање у анкети за препоруку кућних љубимаца. Класа *Recommendation* представља препоруку кућног љубимца и укључује све препоручене расе. Класа *GlobalRecommendation* садржи број препорука које је добила одређена врста животиње, а класа *GlobalChart* садржи тај број препорука за сваку врсту животиње.

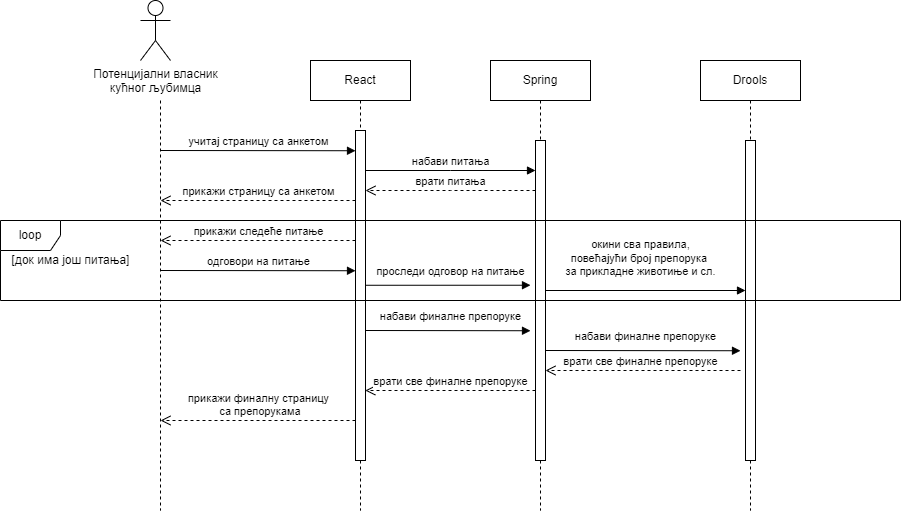


Слика 4.2

## Дијаграм секвенци

Дијаграм секвенци приказује редослед комуникације између објеката у систему током времена. Тиме демонстрира како објекти међусобном комуникацијом остварују одређену функционалност. Вертикалне линије представљају животни век објекта, док хоризонталне стрелице означавају поруке које се шаљу између објеката. Тиме помаже у визуелизацији тока извршавања одређеног сценарија или случаја коришћења. Користи се за боље разумевање и дизајн динамичког понашања система.

Слика 4.3 представља један дијаграм секвенци система описаног у овом раду. У питању је процес попуњавања упитника при добијању препоруке кућних љубимаца. Корисник учитава анкету и одговара на питања. На крају тог процеса добија приказ свих врста кућних љубимаца који му одговарају. Највећи део посла се дешава у позадини, у раду система базираног на знању. Тај систем окида различита правила и тиме одређује најпогодније кућне љубимце и ажурира листу најпопуларнијих кућних љубимаца међу посетиоцима сајта азила.



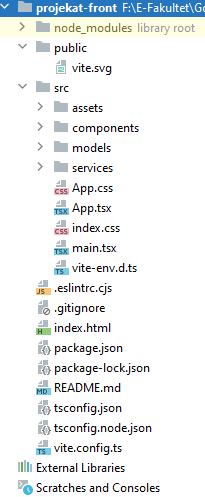
Слика 4.3

# Имплементација

У овом поглављу биће представљена имплементација система за препоруке при раду азила за животиње. Сви делови архитектуре система ће бити детаљно описани. Апликација се састоји од три дела: клијентског дела који користи *React*, серверског дела који користи *Spring* и система базираног на знању који користи *Drools* а интегрисан је у серверски део. Изворни код апликације се налази на *GitHub* [[61]](#Reference61) репозиторијуму <https://github.com/VukRadmilovic/animal-shelter-system>.

## Клијентска апликација

Клијентска апликација је имплементирана помоћу *React* радног оквира. Слика 5.1 представља структуру фолдера изворног кода. Фолдер *node\_modules* садржи инсталиране зависности и библиотеке наведене у *package.json* фајлу. Фолдер *public* садржи статичне фајлове доступне клијенту. Фајл *.eslintrc.cjs* је конфигурациони фајл за *ESLint* [[62]](#Reference62), алат за анализу и одржавање квалитета *TypeScript* [[63]](#Reference63) кода који је коришћен*.* Фајл *.gitignore* наводи фајлове и фолдере које *Git* [[64]](#Reference64)*,* који је коришћен за верзионисање, треба да игнорише*.* Фајл *index.html* је основни *HTML* фајл у који *React* апликација учитава и рендерује све компоненте*.* Фајл *package.json* садржи информације о пројекту попут назива, верзија и зависности које су потребне за покретање апликације*.* Фајл *package-lock.json* закључава тачне верзије инсталираних зависности како би се осигурала доследност између различитих окружења*.* Фајл *README.md* служи за опис упутстава и информација о пројекту*.* Фајл *tsconfig.json* дефинише правила и опције за *TypeScript* тј. процес претварања *TypeScript* кода у *JavaScript.*



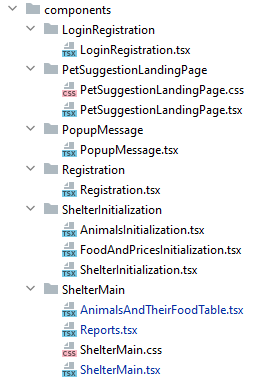
Слика 5.1

У фолдеру *src* налази се више фолдера и фајлова. Фајл *App.tsx* (Листинг 5.1) јесте главна *React* компонента апликације, где се дефинишу основне структуре и логика корисничког интерфејса. Ово је почетна тачка апликације где се компоненте и сервиси учитавају и комбинују, заједно са својим прикладним путањама. Стилови специфични за апликацију који се примењују на елементе унутар *app.tsx* се налазе у *App.css.* Фајл *index.css* садржи глобалне стилове који се примењују на читаву апликацију, укључујући фонтове, маргине и друге дизајн елементе*.* Фајл који представља улазну тачку *React* апликације и спој главне компоненте App.tsx и *HTML* елемента из *index.html* назива се *main.tsx.*

function *App*() {  
 const userServiceSingleton = new UserService();  
 const suggestionServiceSingleton = new PetSuggestionsService();  
 const shelterServiceSingleton = new ShelterService();  
 return (  
 <>  
 <BrowserRouter>  
 <Routes>  
 <Route path="/" element={<PetSuggestionLandingPage suggestionService={suggestionServiceSingleton}/>}/>  
 <Route path="/Shelter" element={<LoginRegistration userService={userServiceSingleton} shelterService={shelterServiceSingleton}/>}/>  
 <Route path="/ShelterMain" element={<ShelterMain shelterService={shelterServiceSingleton}/>}/>  
 <Route path="/ShelterInit" element={<ShelterInitialization shelterService={shelterServiceSingleton}/>}/>  
 </Routes>  
 </BrowserRouter>  
 </>  
 )  
}  
  
export default *App*

Листинг 5.1

Фолдер *assets* обично садржи статичке ресурсе попут слика, фонтова и икона.Фолдер *components* садржи *React* компоненте тј. основне градивне блокове корисничког интерфејса апликације. Свака компонента је део интерфејса која се може поново употребити и одвојена је у засебан фолдер и фајл (Слика 5.2). Фолдер компоненте увек садржи *.tsx* фајл који садржи логику и структуру компоненте, а некада садржи и *.css* фајл са стиловима те компоненте.



Слика 5.2

Фолдер *models* садржи дефиниције типова и интерфејса који представљају структуре података у апликацији. Служе за типизовање података и осигуравање да подаци који се користе у апликацији одговарају предвиђеним структурама података. Листинг 5.2 представља пример интерфејса који описује једну животињу у азилу за животиње.

Листинг .

export interface Animal {  
 animalBreed: string,  
 animalType: string,  
 name: string  
}

Фолдер *services* садржи класе и функције које се баве комуникацијом са сервером помоћу *API* позива као и логичким операцијама невезаним за специфичне компоненте. Листинг 5.3 садржи пример функције за слање одговора на питање упитника серверу ради препоруке кућних љубимаца.

Листинг .

public sendResponse(response : UsersQuestionResponse): Promise<void> {  
 return axios({  
 method: 'POST',  
 url: `${this.api\_host}/api/questionnaire/response`,  
 data: response  
 }).then(() => {}).catch((err) => {  
 console.log(err);  
 throw err;  
 });  
}

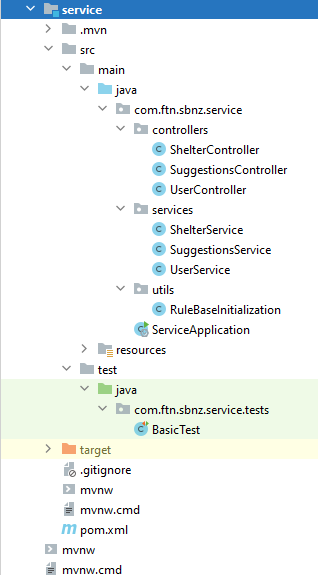
## Серверска апликација

Серверска апликација је централна тачка система. Имплементирана је помоћу *Spring* радног оквира. Комуницира са клијентском апликацијом путем *HTTP* протокола. Сем тога, комуницира и са моделским делом апликације и са системом базираним на знањем имплементираним помоћу *Drools*-а.

Слика 5.3 представља структуру серверске апликације. Кључно за функционисање *Spring Boot* апликације јесте неки начин изградње софтвера и увезивања зависности. У случају овог система користи се *Maven* [[65]](#Reference65) који помаже са управљањем пројекта и који је популаран у *Java* екосистему. Све зависности се дефинишу у *pom.xml* фајлу где се наводе назив, група и верзија библиотеке, а *Maven* онда аутоматски преузима те библиотеке и укључује их у пројекат. Аутоматски изграђује пројекат помоћу одређених процеса изградње попут компајлирања кода, покретања тестова и креирања извршних фајлова. Присутни *mvnw* и *mvnw.cmd* фајлови користе се за покретање *Maven* команди без инсталације *Maven-*а на различитим системима. Листинг 5.4 приказује пример дефинисања зависности у *pom.xml* фајлу.

<dependency>  
 <groupId>com.ftn.sbnz</groupId>  
 <artifactId>kjar</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
</dependency>  
  
<dependency>  
 <groupId>org.kie</groupId>  
 <artifactId>kie-api</artifactId>  
 <version>${drools.version}</version>  
</dependency>

Листинг .



Слика 5.3

Фолдер *controllers* садржи контролере који су прва додирна тачка клијента са сервером. Одговорни су за руковање *HTTP* захтевима ка апликацији. Обрађују долазне захтеве, позивају одговарајуће сервисе за извршење пословне логике и враћају одговоре кориснику. Листинг 5.5 представља пример сажете верзије контролера за руковање захтева везаних за азил животиња.

package com.ftn.sbnz.service.controllers;  
import com.ftn.sbnz.model.models.AnimalWithName;  
...  
  
@CrossOrigin("http://localhost:5173")  
@RestController  
@RequestMapping(value = "/api/shelter")  
public class ShelterController {  
 private final ShelterService shelterService;  
 public ShelterController(ShelterService shelterService) {  
 this.shelterService = shelterService;  
 }  
 @GetMapping(value = "/notifications")  
 public ResponseEntity<?> getNotificationsAndPromotions() {  
 try {  
 return new ResponseEntity<>(shelterService.getNotificationsAndPromotions(), HttpStatus.*OK*);  
 } catch (ResponseStatusException ex) {  
 return new ResponseEntity<>(ex.getReason(), ex.getStatus());  
 }  
 }  
 @PostMapping(value = "/shelter-animal", consumes = "application/json")  
 public ResponseEntity<?> shelterAnimal(@RequestBody AnimalWithName animal) {  
 try {  
 shelterService.shelterAnimal(animal);  
 return new ResponseEntity<>("Successful sheltering", HttpStatus.*OK*);  
 } catch (ResponseStatusException ex) {  
 return new ResponseEntity<>(ex.getReason(), ex.getStatus());  
 }  
 }

...  
}

Листинг 5.5

Фолдер *services* садржи сервисне класе које имплементирају пословну логику апликације. При комуникацији налазе се између контролера и слоја који се бави самим подацима. Обрађују податке, извршавају пословну логику и враћају тражене податке контролеру. Листинг 5.6 представља пример неких функција из сервиса који се бави азилом за животиње. Уместо узимања података из базе података, користи се систем базиран на знању. Тиме се узимају чињенице складиштене у њему, па се оне обрађују и враћају контролеру. Алтернативно, нове чињенице и подаци се убацују у базу знања, као у прве две функције приказаног примера.

Листинг .

public void shelterAnimal(AnimalWithName animal) {  
 Resettlement resettlement = new Resettlement(getSessionClock().getCurrentTime(), getShelter(), PromotionOrResettlementType.*SHELTERING*, animal);  
 kieSession.insert(resettlement);  
 kieSession.fireAllRules();  
}  
  
public void adoptAnimal(AnimalWithName animal) {  
 Resettlement resettlement = new Resettlement(getSessionClock().getCurrentTime(), getShelter(), PromotionOrResettlementType.*ADOPTION*, animal);  
 kieSession.insert(resettlement);  
 kieSession.fireAllRules();  
}

public ReportDTO getDailyReport(String date) {  
 QueryResults results = kieSession.getQueryResults("findLeaf", date);  
 Report r = new Report();  
  
 if (results.size() == 0) {  
 return new ReportDTO(-1, -1);  
 }  
  
 for (QueryResultsRow row : results) {  
 r = (Report) row.get("$r");  
 }  
  
 return new ReportDTO(r.getAdoptionCount(), r.getShelteringCount());  
}

Фолдер *utils* садржи само фајл *RuleBaseInitialization* који представља везу између сервера и *Drools*-а. Иницијализује правила по одређеним шаблонима за различите сценарије у домену препоруке кућних љубимаца. Различити скупови правила се иницијализују, компајлирају и складиште како би се могли користити у тестовима и у току рада саме апликације. Овде се налази веза између различитих одговора корисника на упитник и препорука које ће систем да донесе. Постоје и шаблони правила који се динамички прилагођавају улазним подацима и тиме пружају ефикасна и флексибилна правила при препорукама. Шаблони се, између осталог, користе за питања која имају континуалне вредности одговора. Због тога, препоруке се доносе у зависности од тога у ком опсегу се налази одговор корисника. Шаблони се користе и за питања са дискретним одговорима и за правила везана за *CEP* функционалност система јер се такав начин показао лакши и флексибилнији за имплементацију. Учитавају се и обична правила и она за уланчавање уназад, користи се реално време и све се то учитава ради стварања једне сесије која се користи при функционисању сервера. Код за стварање те сесије приказује Листинг 5.7.

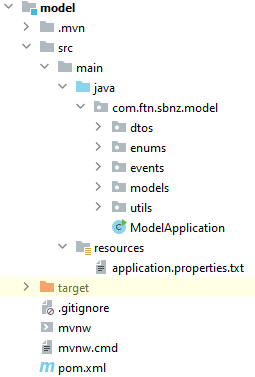
Фолдер *tests* садржи тестове који се користе ради верификације функционалности апликације. Постоји један фајл са тестовима који покрива већину функција система и тиме осигурава да се ништа од функционалности није покварило при изменама система. Иницијализује се посебна сесија за тестове а затим се тестирају различити случајеви коришћења и функције. Користи се псеудо-сат који симулира протицање реалног времена. Различите чињенице се уносе у систем а затим се покрећу правила и тестира се да ли су се прикладна правила покренула и да ли је стање система онакво какво би требало бити након одређених догађаја.

Листинг .

public KieSession createKieSession() {  
 KieHelper kieHelper = new KieHelper();  
 kieHelper.addContent(basicTemplateCompiled, ResourceType.*DRL*);  
 kieHelper.addContent(continuousTemplateCompiled, ResourceType.*DRL*);  
 kieHelper.addContent(cepTemplateCompiled, ResourceType.*DRL*);  
 Results results = kieHelper.verify();  
 if (results.hasMessages(Message.Level.*WARNING*, Message.Level.*ERROR*)){  
 List<Message> messages = results.getMessages(Message.Level.*WARNING*, Message.Level.*ERROR*);  
 for (Message message : messages) {  
 System.*out*.println("Error: "+message.getText());  
 }  
 throw new IllegalStateException("Compilation errors were found. Check the logs.");  
 }  
 InputStream basicRules = RuleBaseInitialization.class.getResourceAsStream("/rules/basic/basic.drl");  
 Resource ruleResource = ResourceFactory.*newInputStreamResource*(basicRules);  
 kieHelper.addResource(ruleResource, ResourceType.*DRL*);  
  
 InputStream backwardRules = RuleBaseInitialization.class.getResourceAsStream("/rules/backward/backward.drl");  
 Resource backwardResource = ResourceFactory.*newInputStreamResource*(backwardRules);  
 kieHelper.addResource(backwardResource, ResourceType.*DRL*);  
  
 KieSessionConfiguration config = KieServices.Factory.*get*().newKieSessionConfiguration();  
 config.setOption(ClockTypeOption.*get*("realtime"));  
 return kieHelper.build().newKieSession(config,null);  
}

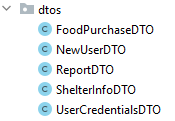
## Моделски део апликације

Моделски део апликације је спона између серверске и *Drools* апликације и кључна је за рад обе. Садржи модел података апликације тј. састоји се од структуре података целе апликације, а на те структуре се ослањају сви остали елементи система. Слика 5.4 приказује структуру фолдера моделског дела апликације. Фајл *ModelApplication* није веома значајан за рад апликације јер се она не покреће директно већ је само зависност за остале апликације. Остали фајлови су објашњени у претходним одељцима. Фолдери *dtos*, *enums*, *events*, *models* и *utils* садрже модел података и описани су у наставку.



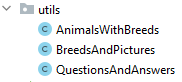
Слика 5.4

Фолдер *dtos* садржи *DTO* објекте. Ти објекти служе за пренос података између различитих слојева апликације, најчешће између клијента и сервера. Контролер у серверу прима податке у формату *DTO* фајлова јер су они прикладнији за пренос али их затим конвертује јер нису најпогоднији за обраду података и њихово трајно чување. Контролер их прима али их често и враћа назад клијенту као одговор. Слика 5.5 приказује коришћене *DTO*-ове у апликацији.



Слика 5.5

Фолдер *utils* садржи класе и методе са предефинисаним вредностима које се враћају клијенту када су неопходне попут информација о доступним животињама, слике животиња, питања и одговори анкета и слично. Слика 5.6 приказује све коришћене *utils* класе у овој апликацији.



Слика 5.6

Фолдер *models* садржи доменске моделе или ентитете које представљају објекте који се заправо обрађују и памте у систему. Њих обрађује сервисни слој при имплементацији пословне логике и они садрже поља која су прикладна за чување у бази података или у овом случају у бази знања система базираног на знању. Листинг 5.8 приказује класу која моделује радника азила.

Листинг .

package com.ftn.sbnz.model.models;  
public class Worker {  
 private String fullName;  
 private String username;  
 private String password;  
 private Shelter shelter;  
 public Worker(String username, String fullName, String password, Shelter shelter) {  
 this.username = username;  
 this.fullName = fullName;  
 this.password = password;  
 this.shelter = shelter;  
 }  
 public String getFullName() {  
 return fullName;  
 }  
 public void setFullName(String fullName) {  
 this.fullName = fullName;  
 }

}

Фолдер *events* садржи моделе везане за догађаје у систему базираном на знању. Догађаји су представљени класом *Event* и њеним наследницама. Њихова главна карактеристика јесте што су везане за специфичан тренутак у времену када су се десиле и за азил који представља место где су се десиле. Остале информације о догађају зависе од специфичне класе, попут *Feeding* (догађај храњења животиња), *Promotion* (догађај промоције збрињавања или удомљавања) и *Notification* (обавештење). Листинг 5.9 приказује сажету верзију кода базне класе догађаја.

Листинг .

package com.ftn.sbnz.model.events;

import com.ftn.sbnz.model.models.Shelter;

...

public abstract class Event {

private LocalDateTime timestamp;  
 private Shelter shelter;  
 public Event(LocalDateTime timestamp, Shelter shelter) {  
 this.timestamp = timestamp;  
 this.shelter = shelter;  
 }  
 public Event(long timestamp, Shelter shelter) {  
 this(LocalDateTime.*ofInstant*(Instant.*ofEpochMilli*(timestamp), ZoneId.*systemDefault*()), shelter);  
 }  
 public LocalDateTime getTimestamp() {  
 return timestamp;  
 }  
 public void setTimestamp(LocalDateTime timestamp) {  
 this.timestamp = timestamp;  
 }

...

}

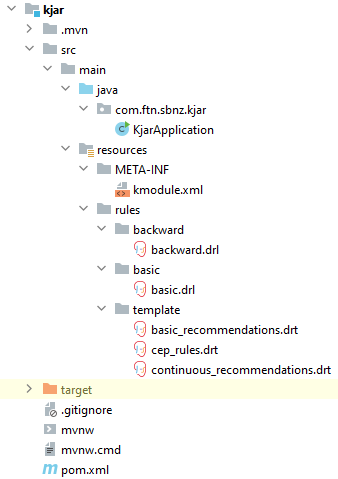
Фолдер *enums* се састоји од енумерација. Енумерације су објекти који могу да имају фиксан скуп вредности које се не мењају. Користе се за дефинисање различитих категорија, статуса и типова. Побољшавају читљивост и једноставност кодирања ограничавањем вредности које одређено поље може имати и обележавањем тих поља на читљив начин. Листинг 5.10 описује пример енумерације која описује све врсте животиња које су подржане системом.

Листинг .

package com.ftn.sbnz.model.enums;  
  
public enum AnimalBreed {  
 *GERMAN\_SHEPHERD*, *GOLDEN\_RETRIEVER*, *BULLDOG*, *POODLE*, *BEAGLE*, *CHIHUAHUA*, *ROTTWEILER*, *DALMATIAN*, *LABRADOR\_RETRIEVER*,  
 *PUG*, *HUSKY*, *DOMESTIC\_SHORTHAIR\_CAT*, *SIAMESE\_CAT*, *SPHYNX*, *BRITISH\_SHORTHAIR\_CAT*, *PERSIAN\_CAT*, *CANARY*, *PIGEON*, *AFRICAN\_GRAY\_PARROT*,  
 *BUDGERIGAR*, *COCKATIEL*, *TURTLE*, *GECKO*, *BALL\_PYTHON*, *BEARDED\_DRAGON*, *GUINEA\_PIG*, *HAMSTER*, *TARANTULA*, *SMALL\_FISH*,  
 *BIG\_FISH*, *BUNNY*, *DUTCH\_DWARF\_RABBIT*, *LIONHEAD*}

## Drools апликација

Последњи описани али кључни део система за препоруке при раду азила за животиње јесте *Drools* апликација. У питању је систем базиран на знању који је интегрисан у серверску апликацију и комуницира са њом и моделском како би пружао препоруке у склопу целокупног система веб апликације. Слика 5.7 представља структуру фолдера ове апликације. *KjarApplication* није веома значајан фајл сходно томе што се ова апликација, слично моделској, не покреће директно већ је само зависност серверске апликације. Фајл *kmodule.xml* унутар фолдера *META-INF* јесте конфигурациони фајл за дефинисање различитих ресурса и модула правила. Може укључивати више база знања и сесија, као и разне конфигурације. Кључан је за интеграцију са *Spring*-ом.



Слика 5.7

Фолдери који укључују правила налазе се у *rules* фолдеру и укључују: *backward*, *basic* и *template* фолдере. Фолдер *basic* садржи фајл *basic.drl*. Тај фајл садржи већину правила система за препоруке при раду азила за животиње. Садржи део правила везаних за препоруку кућних љубимаца и највећи део правила везаних за управљање самог азила. Листинг 5.11 приказује пример правила које служи за обавештавањe радника о недостатку финансијских средстава. При ажурирању попуњености азила, ажурира се и количина новца потребног за његово одржавање. Приказано правило се окида када год новац који је доступан азилу падне испод новца потребног за његово одржавање током следећа три дана. Ствара се обавештење које наглашава потребу за набављањем финансијских средстава. То обавештење укључује и информације о тренутној количини доступног новца као и о количини која је неопходна за функционисање азила током следећа три дана. Затим се то обавештење убацује у радну меморију, како би се приказало корисницима система.

Листинг .

rule "If more funds are needed, send notification"  
 when  
 $clock: SessionClock()  
 $shelter: Shelter($moneyNeeded: moneyNeededForUpkeep, $moneyAvailable: moneyAvailable, moneyAvailable < moneyNeededForUpkeep)  
 then  
 Notification notification = new Notification($clock.getCurrentTime(), $shelter, "Money available (" + $moneyAvailable + ") is less than money needed for upkeep (" + $moneyNeeded + "). Seek funds!");  
 insert(notification);System.*out*.println("More funds needed!");  
end

Листинг 5.12 приказује пример правила које покреће промоцију удомљавања када попуњеност азила пређе 90%, како би се убрзао процес ослобађања места. Правило се активира након што се преброје све животиње у азилу и утврди да је попуњеност већа од 90%, под условом да тренутно не постоји активна промоција. У том случају, креира се нова промоција удомљавања која почиње од тренутног времена и уноси се у радну меморију, чиме постаје видљива корисницима кроз панел са обавештењима.

Листинг .

rule "If shelter 90%+ full, start adoption promotion"  
 when  
 $clock: SessionClock()  
 $shelter: Shelter($capacity: capacity, $animals: animals)  
 Number(intValue >= $capacity \* 0.9) from accumulate(  
 AnimalWithName() from $animals,  
 count()  
 )  
 not Promotion(shelter == $shelter, isActive == true)  
 then  
 Promotion promotion = new Promotion($clock.getCurrentTime(), $shelter, PromotionOrResettlementType.*ADOPTION*);  
 insert(promotion);  
 System.*out*.println(promotion);  
 System.*out*.println("Adoption promotion started!");  
end

Листинг 5.13 приказује правило које аутоматски шаље обавештење у 9, 13 и 18 часова како би подсетило раднике азила да је време за храњење животиња. Правило се активира у тим тачно одређеним терминима током дана, а тада ствара ново обавештење које наглашава потребу храњења животиња радницима. То обавештење се убацује у радну меморију како би се приказало радницима и помогло им да обаве свој посао коректно.

Листинг 5.14 приказује једноставан пример упита који се користи за добављање свих обавештења. Добављају се сва обавештења у радној меморији и сједињују у једну листу. Она се касније приказују у панелу обавештења кориснику.

Листинг .

Листинг .

query "getNotifications"  
 $notifications: List() from collect(Notification())  
end

rule "Send notification to feed the animals at 09:00, 13:00 and 18:00"  
 timer( cron: 0 0 9,13,18 \* \* ? )  
 when  
 $clock: SessionClock()  
 $shelter: Shelter()  
 then  
 Notification notification = new Notification($clock.getCurrentTime(), $shelter, "Time to feed the animals.");  
 insert(notification);  
end

Фолдер *backward* садржи фајл *backward.drl* унутар којег су правила везана за уланчавање уназад. Уланчавање уназад је коришћено унутар система за генерисање и претрагу извештаја. Тај систем функционише на следећи начин: прво се празни извештаји за сваки дан генеришу или на почетку месеца или за текући месец у тренутку покретања програма уколико већ не постоје. Затим се при сваком збрињавању или удомљавању (Листинг 5.16) окида правило које ажурира извештај за тај дан. У случају приказаног правила, налази се догађај удомљавања и извештај који има одговарајуће име. Имена извештаја су форматирана као датуми за које извештај важи. Тај пронађени извештај се затим ажурира како би му се повећао број удомљавања за тај дан. Извештаји сем свог назива имају и свој „родитељски“ извештај. Родитељски извештаји за дневне извештаје су недељни а за недељне су месечни. Листинг 5.15 приказује упите који су кључни за овај систем јер се баве добављањем свих потребних извештаја кроз рекурзију. Упит под називом *isContainedIn* користи рекурзију како би пронашао да ли се извештај под називом *x* налази хијерархијски испод извештаја под називом *y*. Испод месечних извештаја би се хијерархијски налазили недељни и дневни, а испод недељних дневни. Упит прво проверава да ли је извештај *x* директно испод извештаја *y*, а затим да ли постоји неки други извештај који се налази између извештаја *x* и *y.* Прецизније, проверава да ли постоји извештај који се налази испод *y* али изнад *x*. *Drools* омогућава коришћење упита без навођења свих параметара. Упит под називом *isContainedInNonLeaf* служи како би се упит *isContainedIn* позвао на такав начин да пронађе све извештаје који се налазе хијерархијски испод извештаја *y*. Ово је урађено тако што није наведен извештај *x,* и тиме извештај за који се тражи да буде испод извештаја *y* може бити било који извештај. Сви извештаји који испуњавају услов се враћају. Упит *isContainedInForLeaf* је једноставнија верзија упита *isContainedIn*. Проверава само да ли се извештај са називом *x* налази директно испод извештаја са називом *y.* Упит *findLeaf* позива упит *isContainedInForLeaf*, али не наводи аргумент *y*. Тиме се добављају сви извештаји са одређеним називом, без обзира ком родитељском извештају припадају. Сервисни слој комуницира са упитима *findLeaf* и *isContainedInNonLeaf.* Ти упити се користе како би се покупили потребни дневни извештаји из радне меморије. Затим се ти дневни извештаји директно враћају кориснику или агрегирају у недељне или месечне који се затим враћају кориснику. На пример, ако је потребан недељни, упитима се пронађу сви извештаји који су хијерархијски испод недељног те недеље (дневни извештаји) и агрегирају се у један недељни извештај.

query isContainedIn(String x, String y, Report r)  
 r:= Report( x, y; )  
 or  
 (Report( z, y; ) and isContainedIn( x, z, r; ) )  
end  
  
query isContainedInForLeaf(String x, String y, Report r)  
 r:= Report( x, y; )  
end  
  
query findLeaf(String x)  
 isContainedInForLeaf(x, y, $r; )  
end  
  
query isContainedInNonLeaf(String y)  
 isContainedIn(x, y, $r; )  
end

Листинг 5.15

rule "If animal adopted, update report for that day"  
 lock-on-active true  
 when  
 $clock : SessionClock()  
 $resettlement: Resettlement(resettlementType == resettlementType.ADOPTION, $shelter: shelter)  
 $report: Report(name == $resettlement.getReportName())  
 then  
 modify($report){incrementAdoptionCount()};  
end

Листинг 5.16

У фолдеру *template* налазе се 3 фајла која дефинишу правила која ће бити генерисана по шаблону користећи *template* могућности *Drools*-а. Фајл *cep\_rules* бави се *CEP* функционалношћу система за препоруке при раду азила за животиње. Наиме, циљ правила јесте да прекине промоцију удомљавања или збрињавања када она више није потребна. Да ли је и даље потребна рачуна се на основу броја удомљавања или збрињавања у претходних недељу дана урачунавајући тренутни капацитет азила. Листинг 5.17 садржи код фајла *cep\_rules*. Шаблони се користе када желимо да генеришемо више сличних правила користећи исти шаблон. У овом случају користе се да би писали један код који важи и за збрињавање и за удомљавање животиња. Дакле, ако је активна промоција удомљавања, а у 7 дана од почетка промоције је већ удомљено преко 20% капацитета склоништа док је збринуто мање од 10% капацитета склоништа, промоција удомљавања се прекида. У исто време, уколико је активна промоција збрињавања, а у 7 дана од почетка промоције збринуто је преко 20% капацитета склоништа док је удомљено мање од 10% капацитета склоништа у истом периоду, промоција збрињавања се прекида. Број удомљавања и збрињавања се броји из радне меморије, а последица окидања правила јесте измена промоције постављајући јој крајњи датум као данашњи и њен статус на неактиван.

template header  
promotionType  
lessThanAction  
  
package template;  
  
import com.ftn.sbnz.model.events.Promotion;  
import com.ftn.sbnz.model.events.Resettlement;  
import com.ftn.sbnz.model.models.Shelter;  
import com.ftn.sbnz.model.enums.PromotionOrResettlementType;  
import org.kie.api.time.SessionClock;  
  
template "Adoption/sheltering termination"  
  
rule "@{promotionType} Termination"  
when  
 $clock: SessionClock()  
 Shelter($capacity: capacity)  
 $promo: Promotion(type == PromotionOrResettlementType.@{promotionType}, isActive == true)  
 Number(doubleValue > $capacity \* 0.2) from accumulate (  
 $resettlementLikePromo: Resettlement(resettlementType == PromotionOrResettlementType.@{promotionType},  
 this after[0s,7d] $promo),  
 count($resettlementLikePromo)  
 )  
 Number(doubleValue < $capacity \* 0.1) from accumulate (  
 $oppositeResettlement: Resettlement(resettlementType == PromotionOrResettlementType.@{lessThanAction},  
 this after[0s,7d] $promo),  
 count($oppositeResettlement)  
 )  
then  
 System.*out*.println("Inside @{promotionType} termination");  
 modify($promo){  
 setEndDate($clock.getCurrentTime()),  
 setIsActive(false)  
 };  
end  
  
end template

Листинг 5.17

Фајлови *basic\_recommendations* и *continuous­\_recommendations* укључују шаблонска правила за препоруку кућних љубимаца при попуњавању упитника. Фајл *basic\_recommendations* (Листинг 5.18) садржи једно правило које при уносу одговора од стране корисника уноси у систем прикладне препоруке. У овом случају се шаблони користе како би се велик број различитих правила која су веома слична и рађена по истом шаблону могао генерисати на много једноставнији начин од писања сваког правила. За различита питања и одговоре на та питања уносе се различите препоруке кућних љубимаца за корисника у радну меморију.

template header  
ruleName  
questionId  
choice  
recommendations  
  
package template;  
  
import com.ftn.sbnz.model.models.Recommendation;  
import com.ftn.sbnz.model.models.Response;  
  
template "Pet recommendation rules"  
  
rule "@{ruleName}"  
when  
 Response($userId: userId, questionId==@{questionId}, choice==@{choice})  
then  
 insert(new Recommendation($userId, "@{recommendations}"));  
end  
  
end template

Листинг 5.18

Фајл *continuous\_recommendations* (Листинг 5.19) садржи три правила за унос препорука. Овде су потребна три јер корисник за нека питања уноси континуалну нумеричку вредност, па препоруке функционишу на принципу три различита опсега одговора где сваки има себи придружене одговарајуће препоруке. Постоји посебно правило за одговоре на питање који су испод неке доње границе, правило за одговоре који су изнад горње границе, и правило за одговоре који се налазе између. Свако правило уноси прикладне препоруке у радну меморију.

template header  
ruleName  
questionId  
lowLimit  
highLimit  
recommendationsLow  
recommendationsMid  
recommendationsHigh  
...  
rule "@{ruleName} <l"  
when  
 Response($userId: userId, questionId==@{questionId}, $choice: choice, choice < @{lowLimit})  
then  
 insert(new Recommendation($userId, "@{recommendationsLow}"));end  
rule "@{ruleName} >=l, <=h"  
when  
 Response($userId: userId, questionId==@{questionId}, $choice: choice, choice >= @{lowLimit}, choice <= @{highLimit})  
then  
 insert(new Recommendation($userId, "@{recommendationsMid}"));end  
rule "@{ruleName} >h"  
...

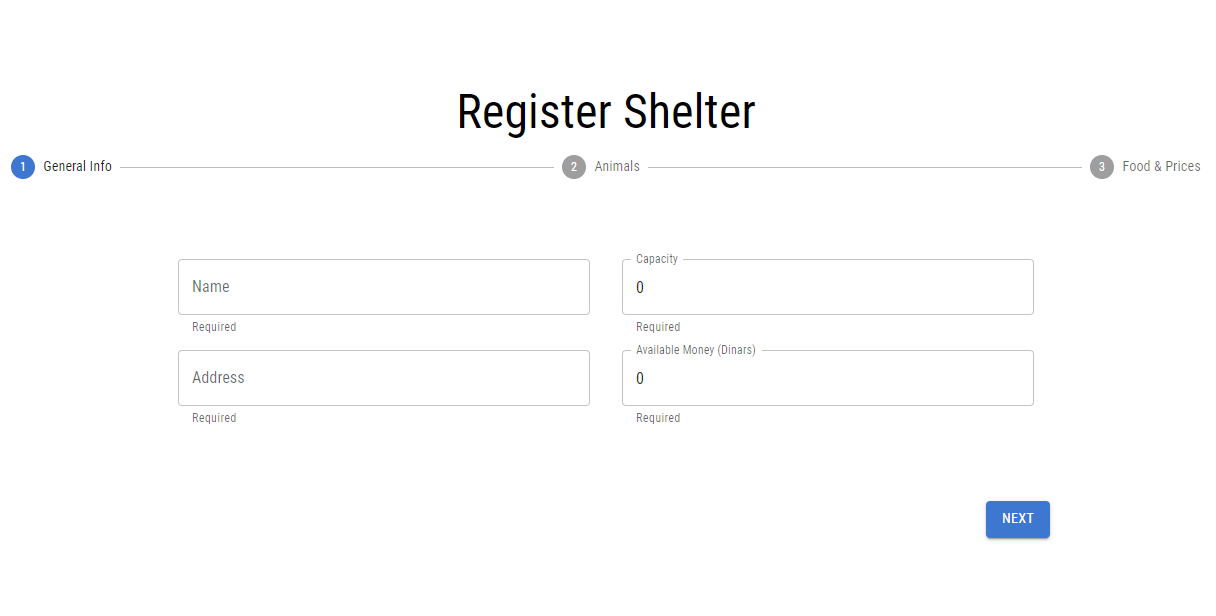
Листинг 5.19

# Приказ имплементираног решења

У овом поглављу је представљена апликација из перспективе њених корисника. Радници у азилу за животиње могу да користе апликацију како би олакшали свој рад и учинили га ефикаснијим и ефектнијим. Апликацију могу користити и потенцијални власници кућних љубимаца у жељи за препоруком врсте кућног љубимца које ће њима највише одговарати.

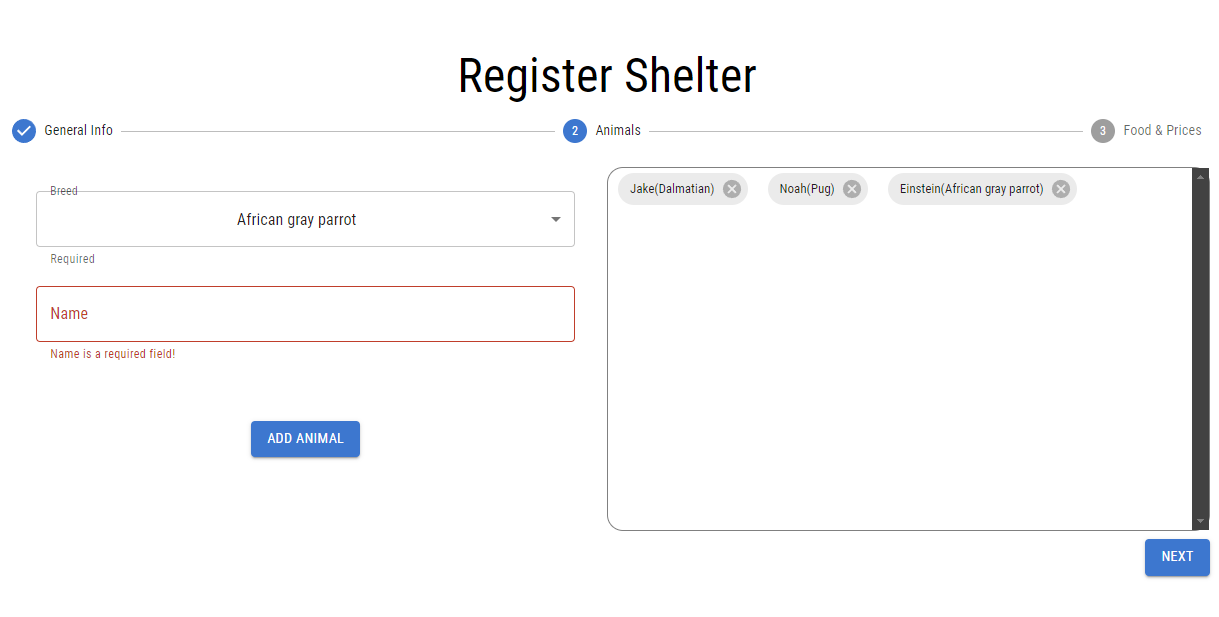
## Радник у азилу за животиње

Пре него што се азилом може управљати, он се мора регистровати (Слика 6.1). Уноси се назив азила, адреса на којој се налази, количина доступног новца и број животиња које азил прима – капацитет.



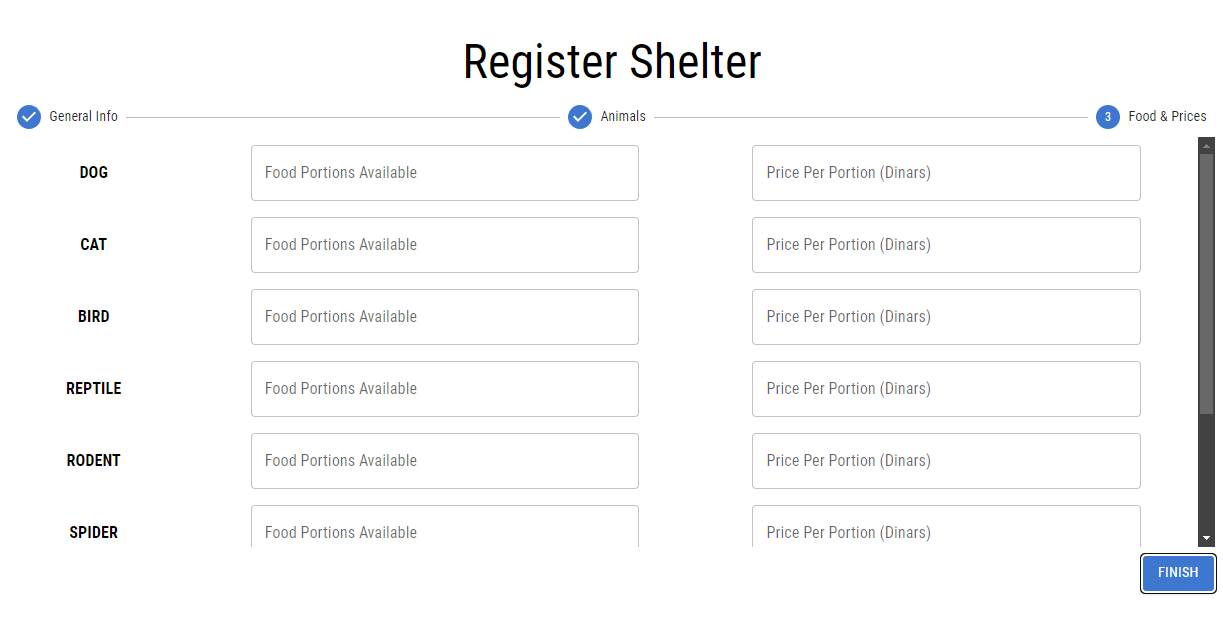
Слика 6.1

Након тога, уносе се информације о свим животињама које се налазе у азилу (Слика 6.2). Информације које се дају су назив и врста.



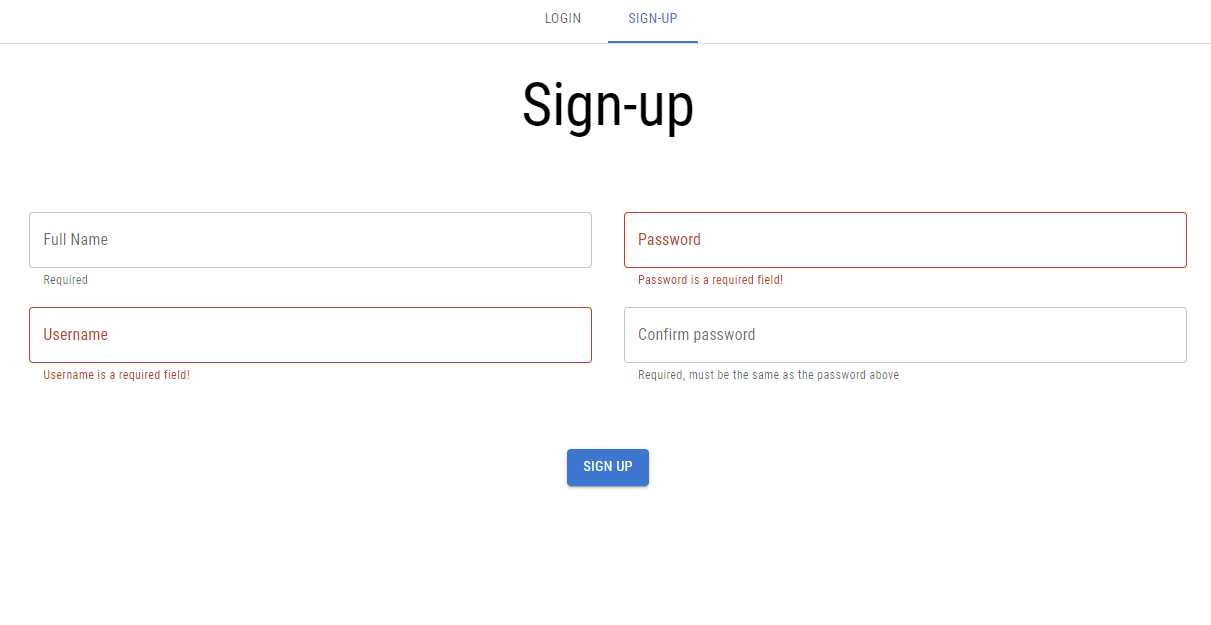
Слика 6.2

На крају, уносе се информације о доступној количини хране за сваку врсту животиње, као и о цени по порцији хране (Слика 6.3).

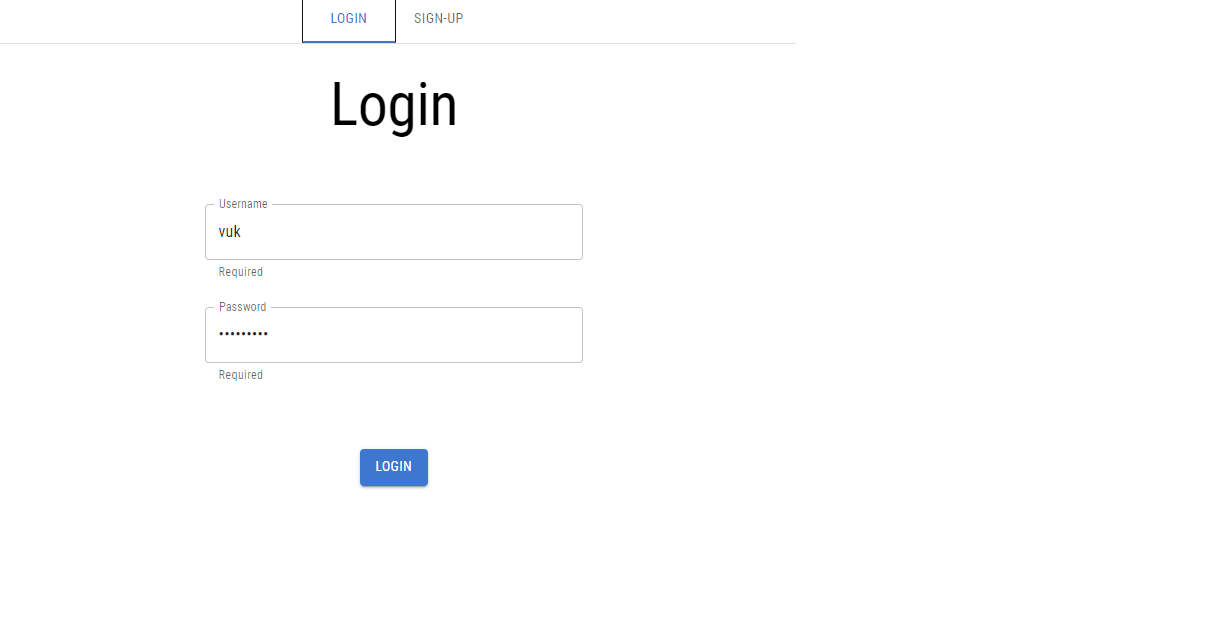


Слика 6.3

Након тога, регистрација азила је завршена. Да би се приступило контролној табли азила неопходно је регистровати (Слика 6.4) бар једног радника који ће затим да се пријави (Слика 6.5) како би управљао азилом и добијао препоруке.

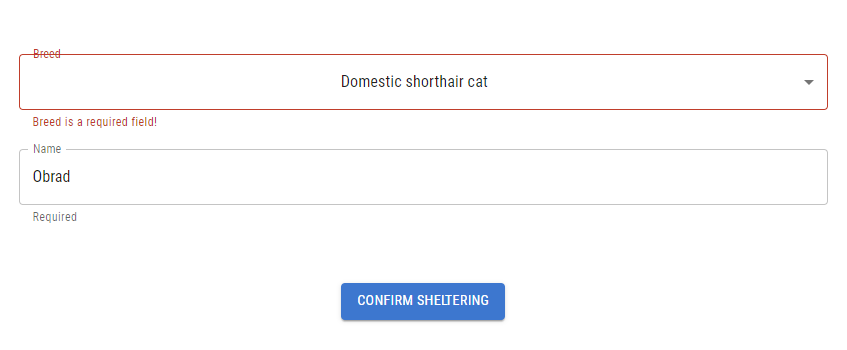


Слика 6.4



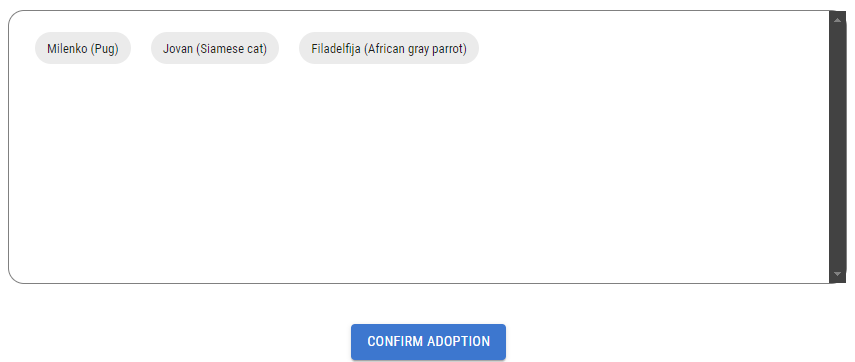
Слика 6.5

Када се корисник пријави, видеће контролну таблу са низом различитих табова који имају различите функције за помоћ и препоруке при управљању азила за животиње. Форма за збрињавање животиња укључује унос информација о врсти и имену животиње која је збринута (Слика 6.6).



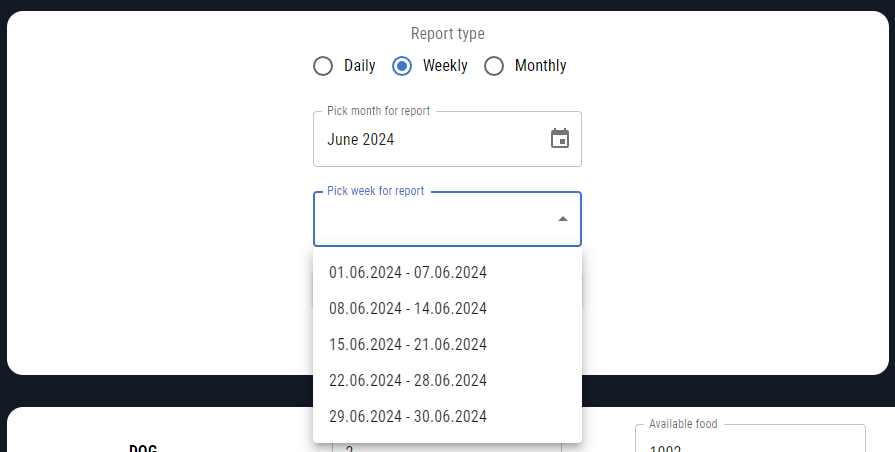
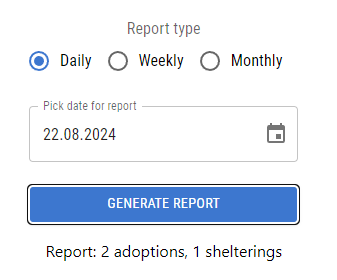
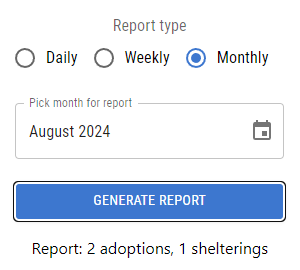
Слика 6.6

Контролна табла садржи и листу свих животиња које тренутно живе у азилу. Листа садржи имена и врсте животиња, а могуће је путем форме потврдити да је било која од њих удомљена (Слика 6.7).



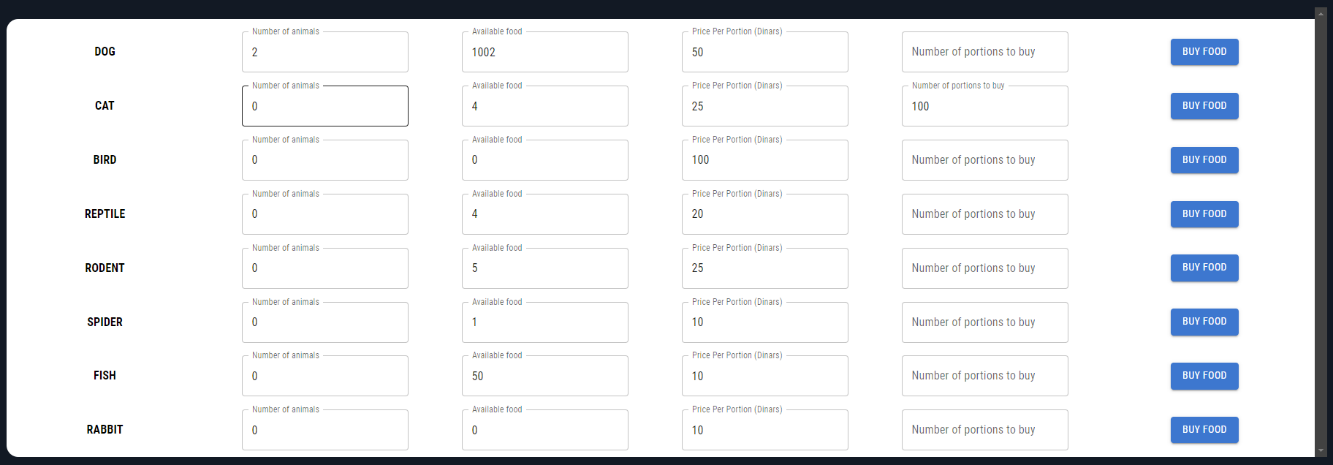
Слика 6.7

Могуће је генерисати извештаје који укључују информације о броју удомљавања и збрињавања животиња за било који дан, недељу, или месец у години (Слика 6.8).

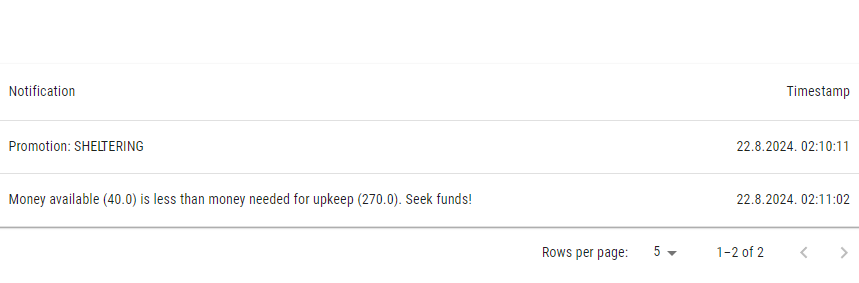
Слика 6.8

Постоји и приказ који представља количину сваке врсте животиње у азилу, колико порција хране за њих се чува у азилу, цена порције хране за ту животињу и нуди могућност бележења куповине одређене количине хране (Слика 6.9).



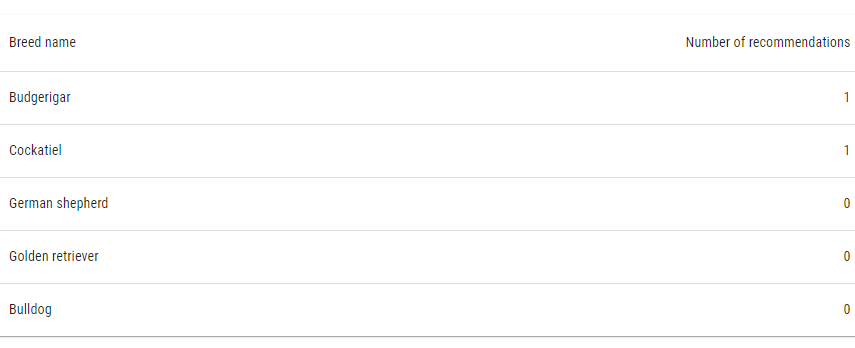
Слика 6.9

Током рада азила, радници добијају обавештења везана за шта се тренутно дешава у азилу, аларме за храњење животиња, упозорења због недостајућих финансијских средстава, препоруке промоција и слично. Та обавештења су приказана, сортирана од најновијих до најстаријих, са промоцијама фиксираним на врху, у једном од табова контролне табле (Слика 6.10).



Слика 6.10

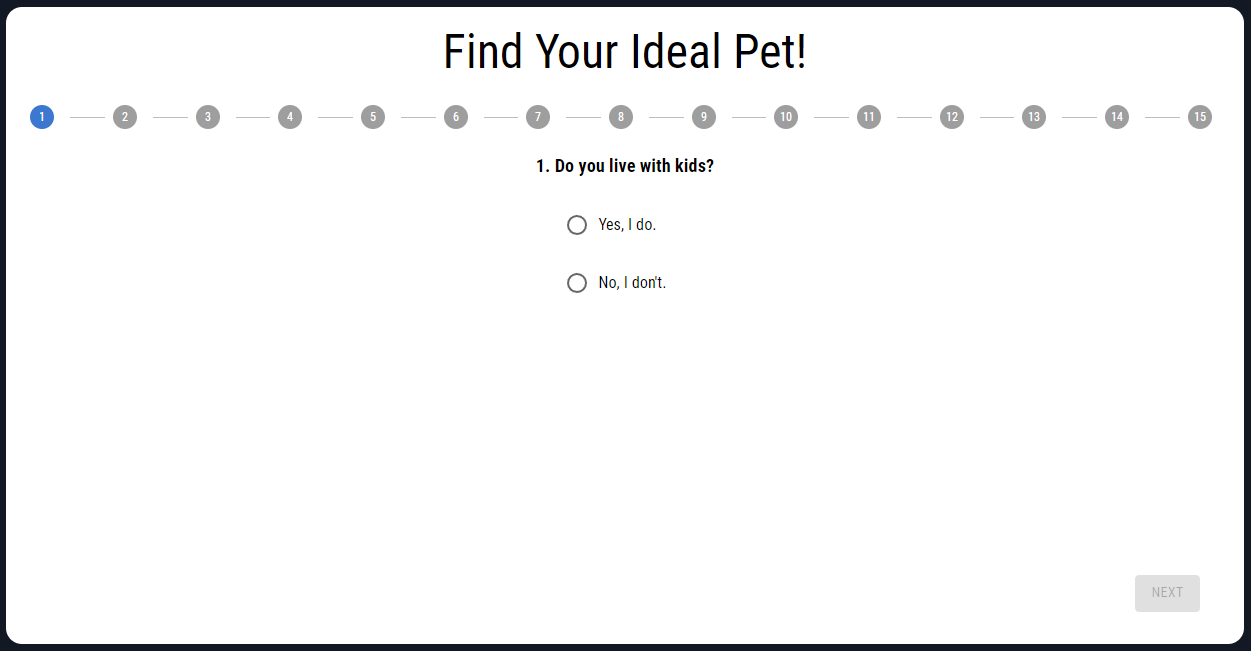
Веза између система за препоруке при раду азила за животиње и модула за препоруку кућних љубимаца је манифестована кроз таб који приказује колико пута је свака врста животиња препоручена на сајту (Слика 6.11). Прегледом таба а и путем аутоматских препорука које систем прави, радници се могу одлучити за стварање нових промоција и потраживањем одређених врста животиња које су чешће потребне од других.



Слика 6.11

## Потенцијални власник кућног љубимца

Корисник без регистрације може директно приступити страници за предлог кућних љубимаца. Концепт апликације јесте да корисник попуњава упитник на чијем крају добије препоруку. Упитник се састоји од 15 питања. Првих десет питања је конципирано тако да свако питање има више понуђених одговора од којих се бира један (Слика 6.12, Слика 6.13).

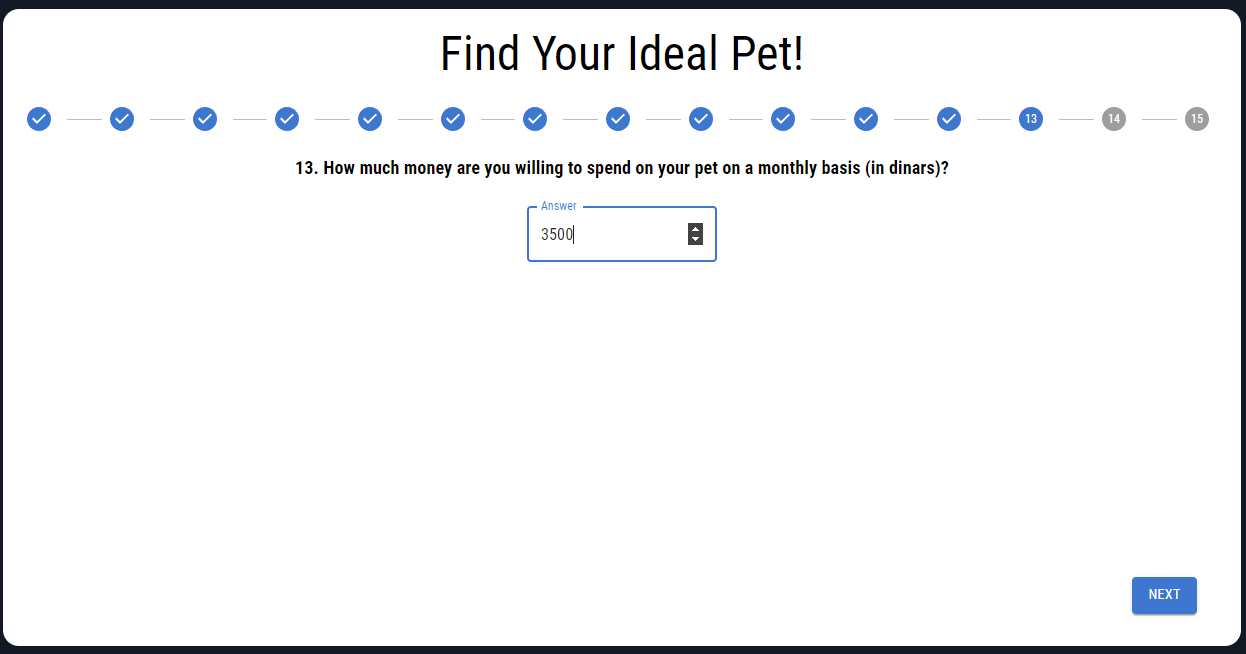


Слика 6.12



Слика 6.13

Последњих пет питања омогућавају кориснику да унесе нумеричку вредност као одговор (Слика 6.14, Слика 6.15).

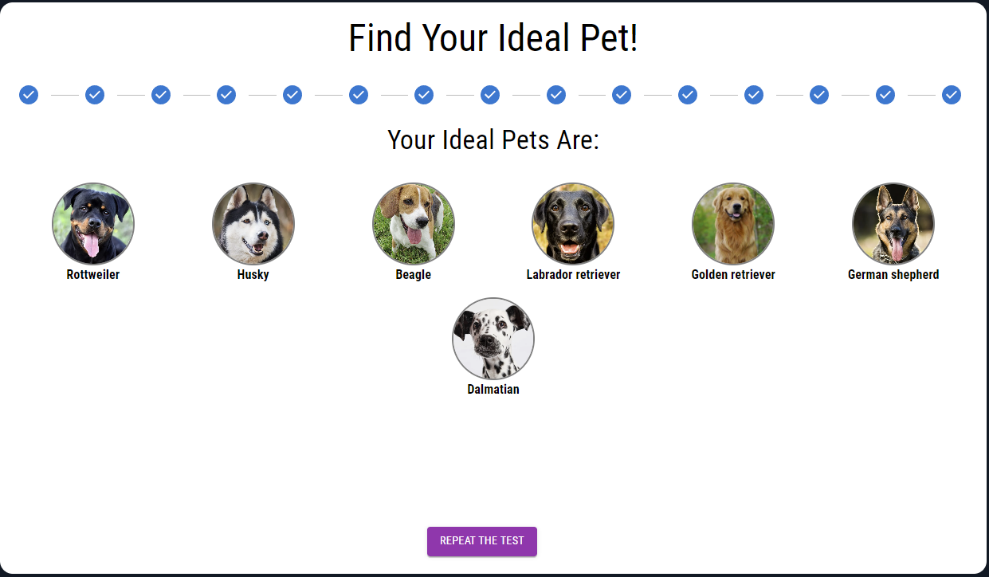


Слика 6.14

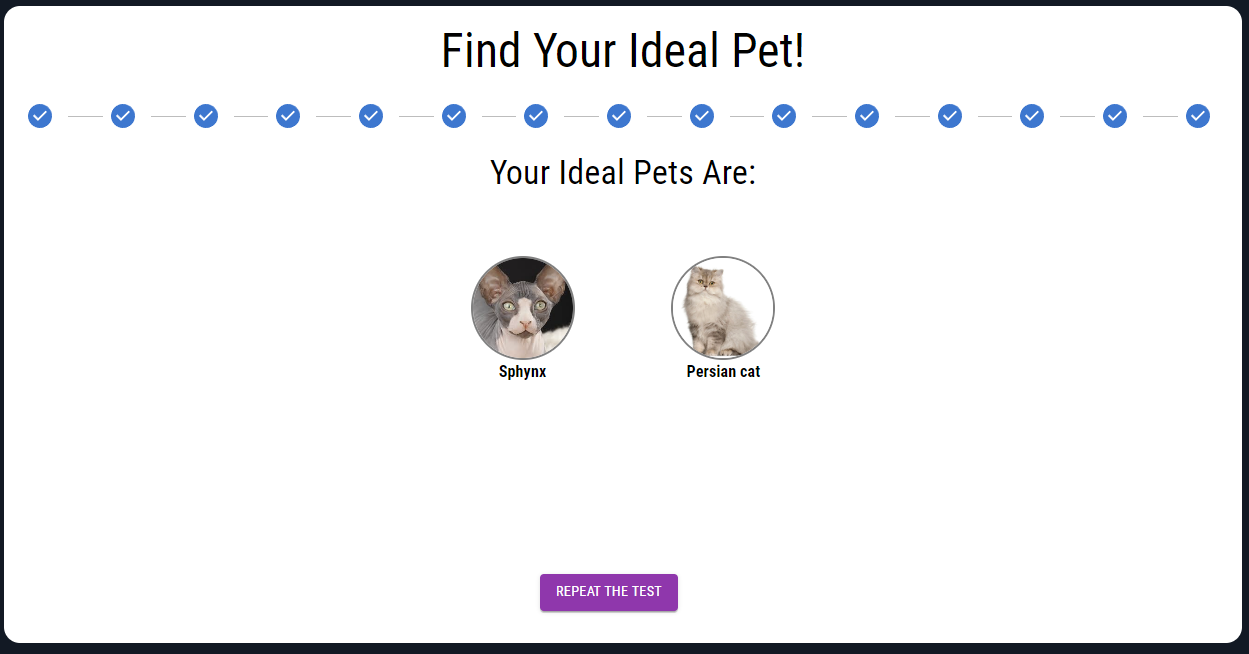


Слика 6.15

Након одговора на последње питање, корисник добија приказ са листом животиња које су најпогодније за њега (Слика 6.16, Слика 6.17). Број животиња који ће бити приказан није фиксиран већ зависи од тога колико животиња је утврђено да је пригодно за корисника.



Слика 6.16



Слика 6.17

# Закључак

У овом раду представљен је развој једног система базираног на правилима. Развијена је веб апликација чији је серверски део урађен у *Spring Boot-*у, док је клијентски део направљен помоћу *React*-а. Експертско знање уведено је путем система базираног на правилима имплементираном у *Drools*-у и интегрисаном у серверском делу апликације. Систем омогућава радницима да лакше управљају азилом за животиње и добијају различите препоруке за њено управљање. Олакшана је набавка и управљање доступном храном за животиње, управљање финансијама, израда извештаја у вези рада азила и вршење промоција удомљавања и збрињавања животиња у прикладим тренуцима. Такође, кроз модул за препоруку животиња, потенцијални власници кућних љубимаца сада имају поуздану апликацију која им може препоручити врсту кућног љубимца који је стварно прикладан за њих.

Извршена је анализа тренутног стања у области и објашњене су технологије и технике коришћене при изради апликације. Описана је спецификација система кроз одређене дијаграме и представљена је његова имплементација.

У будућности могуће је побољшати ову апликацију проширењем могућности управљања азила. Тренутно се не уносе никакве медицинске информације о животињама, нити се чувају неке детаљније информације о појединачним животињама у азилу. Овде се налази потенцијално проширење, јер би радницима у азилу било корисно да забележе те информације на једном месту. Ово би поготово било корисно када дођу нови радници који не знају ништа о животињама у азилу. На тај начин би лако могли да се информишу о њима и упознају их.

Било би добро да постоји већа спрега између сајта за управнике азила и сајта за потенцијалне власнике, где би корисници могли да виде тренутне животиње у азилу које су спремне за удомљавање. Могли би да прегледају њихове профиле са њиховим личностима, сликама и другим информацијама о њима, што би олакшало процес удомљавања. То би се могло спојити са системом за препоруку кућног љубимца где би и специфични кућни љубимци из азила могли бити препоручени потенцијалним власницима. У случају проширења дела сајта који користе потенцијални власници, било би корисно креирати и мобилну апликацију како би људи лакше могли да се информишу и добију препоруке без потребе коришћења рачунара. Функционалност генерисања извештаја би се могла проширити како би укључила више информација о пословању азила.

# Литература

|  |  |
| --- | --- |
| [1] | S. Buecker, M. Marcus, C. Sandy, S. Melina и L. Maike, „Is loneliness in emerging adults increasing over time? A preregistered cross-temporal meta-analysis and systematic review,“ *Psychological Bulletin,* p. 787, 2021. |
| [2] | J. M. Twenge, H. Jonathan, A. B. Blake, M. Cooper, L. Hannah и A. Le Roy, „Worldwide increases in adolescent loneliness,“ *Journal of adolescence,* pp. 257-69, 2021. |
| [3] | R. Purewal, R. Christley, K. Kordas, C. Joinson, K. Meints, N. Gee и C. Westgarth, „Companion animals and child/adolescent development: A systematic review of the evidence,“ *International journal of environmental research and public health,* p. 234, 2017. |
| [4] | H. Christian, F. Mitrou, R. Cunneen и S. R. Zubrick, „Pets are associated with fewer peer problems and emotional symptoms, and better prosocial behavior: Findings from the longitudinal study of Australian children,“ *The Journal of Pediatrics,* pp. 200-6, 2020. |
| [5] | T. Zhang, „Advancing Non-profit Organizations Through Technology and Informatization An analysis on improving the underdeveloped animal shelters and rescues industry,“ *E3S Web of Conferences,* pp. 1-4, 2020. |
| [6] | „State of pet homelessness,“ [На мрежи]. Available: https://stateofpethomelessness.com/. [Последњи приступ 20 8 2024]. |
| [7] | R. Akerkar и P. Sajja, Knowledge-based systems, Jones & Bartlett Publishers, 2009. |
| [8] | M. Proctor, „Drools: a rule engine for complex event processing,“ *Applications of Graph Transformations with Industrial Relevance: 4th International Symposium, AGTIVE 2011, Budapest, Hungary, October 4-7, 2011, Revised Selected and Invited Papers 4,* p. 2, 2012. |
| [9] | Mohanty, C. Ranjan, R. V. Radhakrishnan, M. Jain, P. K. Sasmal, U. Hansda, S. K. Vuppala и S. K. Doki, „A Study of the Pattern of Injuries Sustained from Road Traffic Accidents Caused by Impact with Stray Animals,“ *Journal of Emergencies, Trauma, and Shock,* т. 14, бр. 1, pp. 23-7, 2021. |
| [10] | Y. Zhang, „Statistical Analysis of the Influence of Stray Animals on People's Lives,“ *International Journal of Education and Humanities,* т. 5, бр. 3, pp. 161-4, 2022. |
| [11] | L. Wu, M. Shao, S. Wei, R. Lu и B. Huang, „Widespread of stray animals: design a technological solution to help build a rescue system for stray animals,“ *International Conference on Human-Computer Interaction,* pp. 376-96, 2022. |
| [12] | C. Grosan и A. Ajith, „Rule-based expert systems,“ *Intelligent systems: A modern approach,* pp. 149-85, 2011. |
| [13] | S. S. Abu-naser и A.-Z. A. Ola, „An Expert System for Diagnosing Eye Diseases Using CLIPS,“ *Journal of Theoretical and Applied Information Technology,* pp. 923-30, 2008. |
| [14] | S. S. Abu-naser и M. W. Alawar, „An expert system for feeding problems in infants and children,“ *International Journal of Medicine Research,* pp. 79-82, 2016. |
| [15] | S. S. Abu-naser и M. H. Al-Bayed, „Detecting Health Problems Related to Addiction of Video Game Playing Using an Expert System,“ *World Wide Journal of Multidisciplinary Research and Development,* pp. 7-12, 2016. |
| [16] | S. S. Abu-naser и M. A. Al-Nakhal, „A ruled based system for ear problem diagnosis and treatment,“ *World Wide Journal of Multidisciplinary Research and Development,* pp. 25-31, 2016. |
| [17] | S. S. Abu-naser и R. M. AlDahdooh, „Lower Back Pain Expert System Diagnosis And Treatment,“ *Journal of Multidisciplinary Engineering Science Studies (JMESS),* т. 2, бр. 4, pp. 441-46, 2016. |
| [18] | P. Turner, J. Berry и S. MacDonald, „Animal shelters and animal welfare: Raising the bar,“ *Can Vet J.,* pp. 893-6, 2012. |
| [19] | D. S. Tuber, D. D. Miller, K. A. Caris, R. Halter, F. Linden и M. B. Hennessy, „DOGS IN ANIMAL SHELTERS: Problems, Suggestions, and Needed Expertise,“ *Psychological Science,* pp. 379-86, 1999. |
| [20] | A. Magdum, A. Magdum, G. Chavan и S. Jadhav, „Mobile Application of Pet Adoption System,“ *International Research Journal of Engineering and Technology (IRJET),* pp. 740-3, 2023. |
| [21] | T. Bzdick, „Digital dog shelter application,“ *Thesis. Rochester Institute of Technology,* pp. 1-54, 2010. |
| [22] | B. S. Campanilla, J. O. Etcuban, A. P. Maghanoy, P. A. P. Nacua и N. S. Galamiton, „Pet Adoption App To Free Animal Shelters,“ *Journal of Positive School Psychology,* т. 6, бр. 8, pp. 5993-6006, 2022. |
| [23] | R. ŞENER и T. TÜRK, „Developing An Android-based Mobile Application for Temporary Animal Shelter Activities,“ *Eurasian Journal of agricultural research,* pp. 1-8, 2019. |
| [24] | C.-H. Huang, P.-Y. Shen и Y.-C. Huang, „IoT-Based Physiological and Environmental Monitoring System in Animal Shelter,“ *Seventh International Conference on Ubiquitous and Future Networks,* pp. 317-22, 2015. |
| [25] | A. Jalayer, M. Jalayer, M. Khakzand и M. Faizi, „AUTOMATED OPTIMAL LAYOUT GENERATOR FOR ANIMAL SHELTERS: A FRAMEWORK BASED ON GENETIC ALGORITHM, TOPSIS AND GRAPH THEORY,“ *arXiv preprint arXiv:2405.14172,* pp. 1-15, 2024. |
| [26] | M. Munch, „DESIGN OF AN ANIMAL MANAGEMENT TOOL TO SUPPORT LIFESAVING PRACTICES IN AN ANIMAL SHELTER,“ *Doctoral dissertation. Georgia Institute of Technology,* pp. 1-76, 2020. |
| [27] | P. Jeyaraj и A. Aponso, „A Review of Techniques for Image Classification to Enhance Online Animal Adoption Speed,“ *Proceedings of the 2020 12th International Conference on Computer and Automation Engineering (ICCAE 2020),* pp. 114-8, 2020. |
| [28] | A. McBride, *Appropriate pet selection: which owner, which animal?,* University of Southampton, 2005. |
| [29] | S. Kajbaje, R. Sawant, R. Loke и V. Patil, „AI-Based Pet Adoption System,“ *International Research Journal of Engineering and Technology (IRJET),* pp. 2567-71, 2022. |
| [30] | Buzz, „Buzz to the rescues,“ Buzz, 2024. [На мрежи]. Available: https://buzztotherescues.com/. [Последњи приступ 17 8 2024]. |
| [31] | „Petfinder,“ [На мрежи]. Available: https://www.petfinder.com/. [Последњи приступ 17 8 2024]. |
| [32] | „Adopt a pet,“ [На мрежи]. Available: https://web.archive.org/web/20240816190716/https://www.adoptapet.com/. [Последњи приступ 17 8 2024]. |
| [33] | „Pawlytics,“ [На мрежи]. Available: https://pawlytics.com/. [Последњи приступ 17 8 2024]. |
| [34] | „Shelterluv,“ [На мрежи]. Available: https://pawlytics.com/. [Последњи приступ 17 8 2024]. |
| [35] | „PDSA,“ [На мрежи]. Available: https://www.pdsa.org.uk/. [Последњи приступ 18 8 2024]. |
| [36] | „PDSA Get PetWise,“ [На мрежи]. Available: https://www.pdsa.org.uk/pet-help-and-advice/choosing-a-pet. [Последњи приступ 18 8 2024]. |
| [37] | „Chewy,“ [На мрежи]. Available: https://www.chewy.com/. [Последњи приступ 18 8 2024]. |
| [38] | „Be Chewy,“ [На мрежи]. Available: https://be.chewy.com/. [Последњи приступ 18 8 2024]. |
| [39] | „Be chewy pet quiz,“ [На мрежи]. Available: https://be.chewy.com/what-pet-should-i-get-take-our-quiz-and-find-out/. [Последњи приступ 18 8 2024]. |
| [40] | C. Gackenheimer, Introduction to React, Apress, 2015. |
| [41] | S. Sharma, Modern API Development with Spring and Spring Boot: Design highly scalable and maintainable APIs with REST, gRPC, GraphQL, and the reactive paradigm, Packt Publishing Ltd, 2021. |
| [42] | E. Wilde и C. Pautasso, REST: from research to practice, Springer Science & Business Media, 2011. |
| [43] | D. Gourley и B. Totty, HTTP: the definitive guide, O'Reilly Media, Inc, 2002. |
| [44] | D. Crockford, JavaScript: The Good Parts: The Good Parts, O'Reilly Media, Inc, 2008. |
| [45] | C. Musciano и B. Kennedy, HTML & xhtml: The definitive guide: The definitive guide, O'Reilly Media, Inc, 2002. |
| [46] | „React.dev,“ [На мрежи]. Available: https://react.dev/. [Последњи приступ 24 8 2024]. |
| [47] | H. Schildt, Java: the complete reference, 2007. |
| [48] | H. Y. Yang, E. Tempero и H. Melton, „An empirical study into use of dependency injection in java,“ *19th Australian Conference on Software Engineering,* pp. 239-47, 2008. |
| [49] | G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J.-M. Loingtier и J. Irwin, „Aspect-oriented programming,“ *ECOOP'97—Object-Oriented Programming: 11th European Conference Jyväskylä, Finland, June 9–13, 1997 Proceedings 11,* pp. 220-42, 1997. |
| [50] | „Spring docs,“ [На мрежи]. Available: https://docs.spring.io/spring-framework/reference/overview.html. [Последњи приступ 24 8 2024]. |
| [51] | I. Cosmina, R. Harrop, C. Schaefer и C. Ho, Pro Spring 5: An in-depth guide to the Spring framework and its tools, Apress, 2017. |
| [52] | J. Brittain и D. I. F, Tomcat: The Definitive Guide: The Definitive Guide, O'Reilly Media, Inc, 2007. |
| [53] | „Jetty,“ [На мрежи]. Available: https://jetty.org/index.html. [Последњи приступ 22 8 2024]. |
| [54] | P. Webb, D. Syer, J. Long, S. Nicoll, R. Winch, A. Wilkinson, M. Overdijk и C. Dupuis, „Spring boot reference,“ [На мрежи]. Available: https://docs.spring.io/spring-boot/docs/1.1.0.M2/reference/pdf/spring-boot-reference.pdf. [Последњи приступ 22 8 2024]. |
| [55] | „Spring boot architecture,“ [На мрежи]. Available: https://www.javatpoint.com/spring-boot-architecture. [Последњи приступ 24 8 2024]. |
| [56] | „Drools docs - rule engine,“ [На мрежи]. Available: https://docs.drools.org/5.2.0.M2/drools-expert-docs/html/ch01.html. [Последњи приступ 24 8 2024]. |
| [57] | „How to write a drools business rule,“ [На мрежи]. Available: https://logicdrop.com/guides/how-to-write-a-drools-business-rule/. [Последњи приступ 24 8 2024]. |
| [58] | „Drools docs rule attributes,“ [На мрежи]. Available: https://docs.drools.org/5.6.0.Final/drools-expert-docs/html\_single/#d0e5150. [Последњи приступ 24 8 2024]. |
| [59] | „Drools docs,“ [На мрежи]. Available: https://docs.drools.org/6.2.0.Final/drools-docs/html/ch20.html#d0e22934. [Последњи приступ 24 8 2024]. |
| [60] | D. Pilone и N. Pitman, UML 2.0 in a Nutshell, O'Reilly Media, Inc, 2005. |
| [61] | „Github,“ [На мрежи]. Available: https://github.com/. [Последњи приступ 27 8 2024]. |
| [62] | „ESLint,“ [На мрежи]. Available: https://eslint.org/. [Последњи приступ 27 8 2024]. |
| [63] | G. Bierman, M. Abadi и M. Torgersen, „Understanding typescript,“ *ECOOP 2014–Object-Oriented Programming: 28th European Conference, Uppsala, Sweden, July 28–August 1, 2014. Proceedings 28,* pp. 257-81, 2014. |
| [64] | J. D. Blischak, E. R. Davenport и G. Wilson, „A quick introduction to version control with Git and GitHub,“ *PLoS computational biology,* 2016. |
| [65] | R. Bharathan, Apache Maven Cookbook, Packt Publishing Ltd, 2015. |

# Биографија

Вук Радмиловић је рођен 20. јуна 2001. године у Новом Саду. Завршава основну школу „Васа Стајић“ у Новом Саду. Средњу електротехничку школу „Михајло Пупин“ завршава у Новом Саду. Године 2020. уписује Факултет техничких наука у Новом Саду, смер Софтверско инжењерство и информационе технологије. Све испите полаже и студије завршава у року 2024. године.