Универзитет у Београду

Електротехнички факултет



Веб апликација за негу и побољшање квалитета живота кућних љубимаца

Мастер рад

|  |  |
| --- | --- |
| Ментор: | Кандидат: |
| Проф. др Бошко Николић | Стефан Вукашиновић 3101/2017 |

Београд, Јул, 2018.

Садржај

[Садржај i](#_Toc12621574)

[1. Увод 1](#_Toc12621575)

[2. Захтеви за реализацијом система 3](#_Toc12621576)

[2.1. Кориснички захтеви апликације 3](#_Toc12621577)

[2.1.1. Ветеринар 3](#_Toc12621578)

[2.1.2. Власници кућних љубимаца 4](#_Toc12621579)

[2.2. Веб технологије и њихова примена у раду 5](#_Toc12621580)

[2.2.1. Веб апликацијe и разменa података на вебу 5](#_Toc12621581)

[2.2.2. Коришћене технологије 6](#_Toc12621582)

[3. Опис рада система 12](#_Toc12621583)

[3.1. Опис рада система за ветеринара 12](#_Toc12621584)

[3.1.1. Додавање текстова о кућним љубимцима 12](#_Toc12621585)

[3.1.2. Распоред заказаних термина и извршавање прегледа 13](#_Toc12621586)

[3.1.3. Преглед личних информација пријављеног корисника 17](#_Toc12621587)

[3.2. Опис рада система за власника љубимаца 17](#_Toc12621588)

[3.2.1. Додавање новог профила за љубимца 18](#_Toc12621589)

[3.2.2. Избор активног профила и детаљи љубимца 18](#_Toc12621590)

[3.2.3. Преглед распореда заузетих термина и заказивање прегледа 20](#_Toc12621591)

[4. Реализација система 22](#_Toc12621592)

[4.1. Почетна страница, компоненте и руте апликације 22](#_Toc12621593)

[4.2. Реализација компоненте навигационог менија 23](#_Toc12621594)

[4.3. Реализација компоненте за одабир активног профила за љубимца 25](#_Toc12621595)

[4.4. Преглед профила љубимца и промена активног профила 27](#_Toc12621596)

[4.5. Реализација компоненте календара 29](#_Toc12621597)

[4.6. Валидација међусобно зависних поља форме у *Angular*-у 31](#_Toc12621598)

[*4.7.* Промена стилова дефинисаних у *Bootstrap*–у 32](#_Toc12621599)

[5. Закључак 33](#_Toc12621600)

[Литература 35](#_Toc12621601)

[Списак скраћеница 36](#_Toc12621602)

[Списак слика 37](#_Toc12621603)

1. Увод

Предмет рада представља развој веб апликације која се нуди на коришћење власницима кућних љубимаца и која им нуди стручну помоћ и савете у вези са здрављм и квалитетним животом њихових љубимаца. Поменуте активности садрже планирање и заказивање одлазака на ветеринарске прегледе, кућну негу, савете ветеринара, лекове и редовну вакцинацију, као и страницу са популарним текстовима о кућним љубимцима.

Апликација има 2 врсте корисника, а то су ветеринари и власници кућних љубимаца. Када се региструју, власницима се нуди опција да додају профил свог кућног љубимца. Број профила кућних љубимаца за једног власника је неограничен. Власници љубимаца, уколико имају више љубимаца, могу да мењају активан профил љубимца и да прегледају његов профил, где се налазе све информације и здравствена историја љубимца. На страници за заказивање прегледа на ветеринарској клиници приказан је преглед свих слободних термина, као и историја прегледа свих власникових љубимаца.

Ветеринари су задужени да обављају прегледе љубимаца у заказаним терминима. Они имају приступ страници на којој је приказан календар са свим заказаним прегледима за тренутни дан, месец или годину. Ветеринари по избору жељеног прегледа из календара, пишу извештај, преписују потребну терапију, вакцинишу или додају болести прегледаном љубимцу. Могу и да ажурирају почетну страну апликације са текстовима о кућним љубимцима...

Документ се састоји из 5 целина које чине: увод, захтеви за реализацијом система, опис рада система, реализација система и заскључак.

У уводу је дат кратак опис теме мастер рада и преглед документа по поглављима. У наставку приказани су добијени резултати.

У другом поглављу „Захтеви за реализацијом система“, описани су кориснички захтеви апликације и технологије које су коришћене за реализацију корисничког интерфејса (*front-end*), програмирања сервера и базе (*back-end*) и размене података међу њима. Размена података на интернету и појам веб апликације. Технике реактивног програмирања у *Angular-у*. *PHP*. *JSON*.

Треће поглавље „Опис рада система“ садржи детаљан опис рада реализоване апликације са корисничким упутством, које се састоји од упутства за ветеринара и упутства за власника љубимаца. Упутства садрже опис свих функционалности у апликацији, објашњене корак по корак, уз приложене слике корисничког интерфејса.

У четвртом поглављу „Реализација система“, наведени су неки од проблема при реализацији апликације и описана су њихова решења, укључујући делове програмског кода. Компоненте и Рутирање у *Angular*-у. Сервиси за слање *get/post http* захтева помоћу класе *HttpClient*. Сервиси за комуникацију међу компонентама. *Angular* календар. *PDO* –класа за конекцију са базом у *PHP*-у. Валидатори у *Angular*-у. Редефинисање стилова из *Bootstrap*-а.

Закључак садржи кратку рекапитулацију реализоване апликације, као и део о могућој надоградњи.

Размена података између клијента и сервера засниваће се на *post* или *get* *HTTP* захтевима које *Angular* реализује у позадини помоћу реактивних технологија које користе *AJAX.* *PHP* ће бити коришћен искључиво у сврхе дохватања података из базе и прослеђивања података путем *JSON*-a у сировом облику. Дакле, кориснички интерфејс биће у потпуности реализован у *Angular*-у.

1. Захтеви за реализацијом система

У овом поглављу ће бити описани кориснички захтеви апликације, као и коришћене технологије.

У првом потпоглављу, у којем су описани кориснички захтеви апликације, говори се о различитим типовима корисника и њиховим привилегијама и правима приступа апликацији.

У другом потпоглављу, на самом почетку, описани су појмови веб апликације, веб технологије, као и поступак комуникације и размене података између клијента и веб сервера. Потом се говори о веб технологијама коришћеним за потребе реализације саме апликације и о примерима коришћења.

* 1. Кориснички захтеви апликације

Постоје две врсте регистрованих корисника, а то су ветеринари и власници кућних љубимаца. Апликацији могу приступати и нерегистровани корисници али само страници са популарним текствовима о кућним љубимцима и промоцијама ветеринарских клиника које нуде овакву врсту апликације.

На почетној страници доступно је пријављивање регистрованих корисника на систем, као и регистровање нових корисника. При регистрацији корисник попуњава основне информације о себи, а по успешнем регистровању на ситем приказује му се страница за додавање профила првог љубимца. У случају пријављивања на систем, потребно је у зависности од тога да ли је пријављени корисник ветеринар или власник љубимаца, приказати одговарајућу страну и доделити кориснику одређене привилегије. У наставку овог поглавља објашњено је које су то привилегије ветеринара односно власника љубимаца.

* + 1. Ветеринар

Ветеринари су корисници, представници ветеринарске клинике и професионални ветеринари по струци. Њима су кориснички налози додељени од стране саме клинике и они, као такви, немају могућност самосталног регистровања на систем. Њихова основна улога је да прегледају кућне љубимце и воде евиденцију о њиховој историји лечења и начину живота. Имају и опцију додавања вести и текстова на страну са популарним текстовима о љубимцима.

* + - 1. Прегледи љубимаца

Ветеринарима се на почетном екрану апликације приказује календар са тренутним даном и распоредом прегледа који су заказани за тај дан. При обављању прегледа, потребно је кликнути на заказани термин где се у искакајућем прозору приказују основни детаљи љубимца који има заказани термин, као и 2 опције. Једна за одлазак на страну са детаљним подацима љубимца и прегледом његовог картона, друга за одлазак на страну на којој се извршава преглед изабраног љубимца.

Само извршавање прегледа подразумева попуњавање извештаја са прегледа, који чине:

* Извештај са прегледа (обавезно поље)
* Вакцинација (није обавезно)
* Терапија (није обавезно)
* Додавање болести за кућног љубимца (није обавезно)

По извршењу прегледа потребно је означити дати преглед као успешно обављен и не приказивати га у дневном распореду прегледа за текући дан.

* + - 1. Ажурирање почетне странице са популарним текстовима о љубимцима

Ветеринари имају опцију додавања нових текстова о кућним љубимцима на почетну страну апликације која је доступна свим корисницима апликације, било пријављеним или не. Уз текст је пожељно окачити и одговарајућу слику.

* + 1. Власници кућних љубимаца

Власници љубимаца имају неколико основних опција, а то су: додавање профила за новог љубимца, детаљан преглед профила и здравственог картона, као и могућност додавања детаљних карактеристика љубимца. Такође, корисницима је омогућена навигација кроз постојеће профиле љубимаца, промена активног профила и заказивање прегледа на ветеринарској клиници.

* + - 1. Додавање профила новог кућног љубимца

Кориснику се приказује форма са основним подацима о љубимцу, као што су име, врста, раса, тежина и датум рођења. Слика је опциона. Када се профил сачува, он аутоматски постаје активан и кориснику се приказује страница детаљима о љубимцу.

* + - 1. Преглед профила љубимца

На овој страници, у падајућој листи, корисник има могућност одабира профила кућног љубимца који ће бити активан. Уколико корисник има само једног љубимца онда је тај профил увек активан.

Кориснику су потом приказане информације о кућном љубимцу чији је профил активан. Информације су подељене у 3 категорије, а то су:

* основне информације
* детаљне информације
* здравствени картон љубимца.

Након креирања профила само су основне информације попуњене, док су детаљне информације и лекарски картон празни.

Кориснику је омогућена опција додавања или промене детаљних информација. Неке од информација су: количина оброка, број оброка, врста оброка, физичка активност...

Здравствени картон је подељен у 4 подкатегорије, а то су:

* болести љубимца
* вакцинације
* терапија
* лекарски извештаји са прегледа.

Болести љубимца могу додавати и ветеринари приликом прегледа али и власници приликом креирања профила. У оквиру вакцинација и терапија, приказана је листа са датумима и детаљима конкретних вакцинација и терапија које је љубимац добио. Лекарски извештаји су сортирани по најскоријем датуму креирања и приказани у листи са линком ка страници са детаљима о сваком извештају понаособ.

* + - 1. Заказивање прегледа

На овој страници корисницима је иницијално приказан календар са распоредом прегледа и преосталих теримина за текућу недељу. Кликом на слободан термин нуди се опција резервисања термина за кућног љубимца чији је профил активан у том тренутку. Када се резервација сачува, у календар се уписује име љубимца у заказаном термину. Кликом на заказани термин свог љубимца, корисник има опцију отказивања заказаног прегледа.

* 1. Веб технологије и њихова примена у раду

У оквиру овог потпоглавља биће објашњене веб технологије примењене при изради апликације, као и шта су то веб апликације и технологије.

* + 1. Веб апликацијe и разменa података на вебу

Једна веб апликација састоји се из 2 целине:

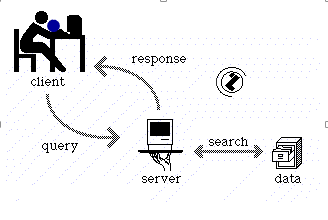
* кориснички интерфејс илити презентациони слој - *front-end*
* серверски слој – *back-end*

Кориснички интерфејс – *front-end*  подразумева целокупан садржај веб странице који је видљив кориснику и са којим корисник има директан контакт, укључујући и дизајн странице. Циљ доброг вебсајта је да се обезбеди да корисници који њему приступе виде информације у формату који је лако читљив и пре свега релевантан. Информације су подаци који се добијају од стране сервера. Улога корисничког интерфејса је да те податке прикаже кориснику.

Серверски слој – *back-end* дефинише како један веб сајт треба да функционише. Ту спада све што корисник не може да види у веб читачу, као што су база података и сервер. Серверска страна апликације је задужена за безбедност, структуру и дозволу приступа подацима од стране клијента (корисника). Сајтови који имају серверски слој називају се динамичким сајтовима јер се њихов садржај константно мења и апдејтује. Подаци се чувају у базама података.

Најједноставније објашњене како се одвија комуникација корисничког интерфејса и сервера, тј. размена података, је кроз следеће кораке:

* клијент приступом веб страници у позадини шаље захтев за подацима ка серверу
* на серверу се шаље упит ка бази података
* подаци се читају из базе података и процесирају на серверу
* када је одговор на серверу спреман шаље се назад клијенту
* клијенту се приказује пристигли одговор са сервера у виду веб странице



Слика 2.2.1.1 Размена података између клијента и сервера

* + 1. Коришћене технологије

За потребе израде апликације коришћене су савремене технологије. За реализацију корисничког интерфејса коришћен је *Angular* фрејмворk (један од најпопуларнијих фрејмворка за *JavaScript*). *Angular* је написан у потпуности у језику *TypeScript* који представља проширење језика JavaScript. У оквиру фрејмворка коришћени су *HTML5* и *CSS3*.

За потребе прилагођења апликације уређајима са различитим величинама екрана, као и за потребе дизајна страница коришћен је *Bootstrap4*, фрејмворк за *CSS*.

За размену података између клијента и сервера коришћен је *JSON* формат, док је серверски код писан у програмском језику *PHP*.

Као развојна окружења коришћени су *VS Code*, за кориснички слој апликације и *NetBeans* за потребе писања кода у језику *PHP*.

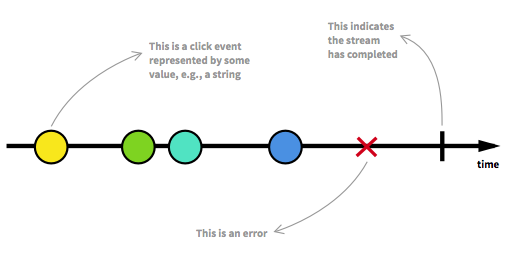
* + - 1. Angular фрејмворк

*Angular* је део JavaScript екосистема и данас је један од најпопуларнијих инструмената за развој софтвера. Гугл га је представио 2009. године и добио је топлу добродошлицу од развојне заједнице. Према истраживању сајта *StackOverflow* за 2018, 36,9% софтверских инжењера сада примењује неку од верзија *Angular*-а 2-7 за креирање корисничких интерфејса.

У наставку ћемо говорити о главним алатима које инжењери обично користе са *Angular*-oм да би добили пуну корист од овог фрејмворка:

*RxJs*  (Реактивнo проширење за *JavaScript*). Мало је вероватно да ћете моћи да радите са *Angular*-ом без *RxJs*, реактивном библиотеком за програмирање, која има за циљ да обради асинхроне податке са више догађаја.

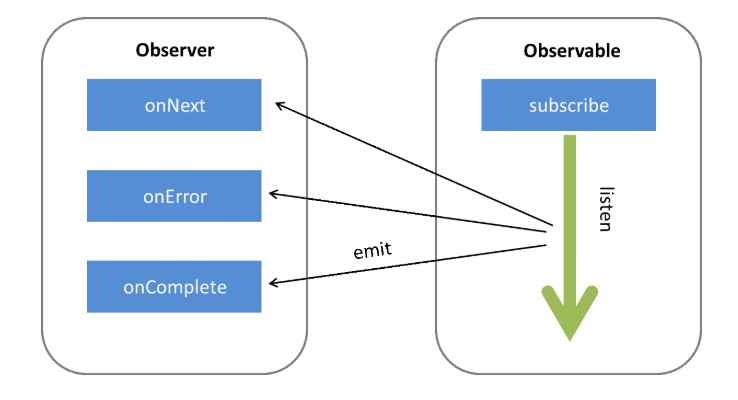
Реактивно програмирање је парадигма асинхроног програмирања која се бави токовима података и пропагацијом промена. *RxJs* је библиотека за реактивно програмирање која користи *Observables,* које се могу објаснити као колекције различитих вредности током времена (Слика 2.2.2.1)*.*



Слика 2.2.2.1 *Observable* као ток података током времена

*Observables* су у широкој употреби у *Angular*-у за руковање асинхроним кодом. На пример, за рад са Ајаксом, када ослушкујете догађаје и крећете се кроз странице апликације (тзв. *Routing*). Да бисмо разумели зашто треба да користимо асинхрони код уместо обичног *JavaScript*-а, размислите о коду који чека одговор од удаљеног сервера. Не постоји начин да се сазна када ће се одговор вратити, и да ли ће се икада вратити, тако да није добра идеја престати са извршавањем скрипте док не добијете одговор. Асинхрони код је много боље решење.

Да бисмо регистровали промену вредности на току података, коју емитује *Observable*, неопходно је претплатити јој се помоћу *subscribe* metode и за то је задужен *Observer.* Тек тада је могуће успешно очитавати податке са тока који се емитују и процесирати их пре приказивања у оквиру апликације. Постоје 3 опције/методе које *Observer* има на располагању, а то су: обрада следећег податка са тока података, обрада грешке и акција која се извршава када се ток података комплетира, тј. заврши (Слика 2.2.2.2).



Слика 2.2.2.2 Интеракција и процесирање пристиглих података са тока података

Најчешћа примена реактивног програмирања и *RxJs* библиотеке у апликацији је кроз *Http* захтеве за подацима на серверу. Потребне методе за рад са *Http* захтевима налазе се у модулу *HttpClient*. Коришћене су методе *get()* и *post()* за слање гет и пост захтева за подацима на сервер. Повратна вредност ових метода је управо *Observable*, којој се претплаћује и једном када подаци стигну са сервера, спремни су за обраду и приказ кориснику.

*Angular CLI*. Интерфејс командне линије (*CLI*) се користи за креирање пројеката, додавање датотека, извршавање исправки, бављење тестирањем за отклањање грешака и имплементацију.

*Two-way data binding.* Двосмерно повезивање података. *Angular* је направљен са *MVC* (*Model-View-Controller)* архитектуром. И фрејмворк је синхронизовао Модел и Поглед. Како се подаци у моделу мењају, тако се мења и поглед, тј. садржај странице.

*Angular* директиве. Директиве су можда најзначанији део примене *Angular*-а. Неке од најчешћих примера коришћења директива у апликацији су следећи:

Компонента, као основни градивни елемент сваке *Angular* апликације. Уколико је потребно исти *HTML* код приказивати на више стране апликације, потребно је направити посебну компоненту и издвојити код у одвојену целину. Потом се та компонента укључује у друге фајлове у којима ју је потребно приказати, навођењем њеног селектора. *Angular* је и замишљен да се састоји од различитог броја компоненти које чине целину и комуницирају по потреби међусобно.

Атрибутске директиве. Служе за манипулацију модела документа – *DOM (Document Object Model)* и променом његовог понашања и изгледа. Ове директиве се користе за примену условног стила на елементе, приказивање или сакривање елемената...Пример коришћења је за бојење оквира поља форме у црвено у случају да постоји грешка при валидацији тог поља помоћу директиве *[ngClass]=”{ ’име\_класе’ : услов }”,* која динамички додељује класу пољу уколико је услов задовољен.

Структурне директиве. Намењене су за креирање или брисање елемената из модела ДОМ-а*.* Неке атрибутске директиве - као што су *hidden*, које показују или скривају елемент - у ствари одржавају ДОМ таквим каквим јесте. Али структурне *Angular* директиве додају или потпуно уклањају елементе из ДОМ-а, заправо мењајући *HTML* структуру. Пример је директива \**ngIf=”услов”* која додаје у ДОМ одређени садржај уколико је услов задовољен, у супротном га брише из ДОМ-а (Слика 2.2.2.3).



Слика 2.2.2.3. Структурна директива \**ngIf*

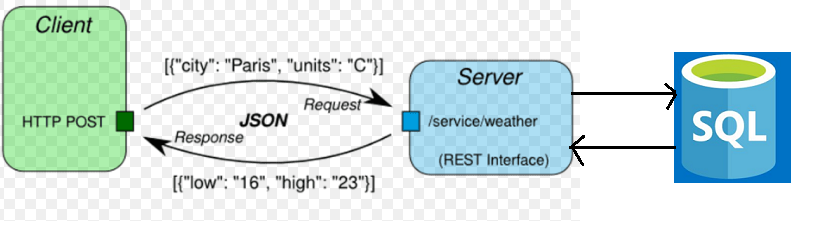
*Depedency injection*. Ове зависности дефинишу како различити делови кода међусобно комуницирају и како промене у једној компоненти утичу на друге. Обично, зависности су директно дефинисане у самим компонентама. Тако да свака промена зависности захтева и промену компоненти. Убризгавање зависности је учинила компоненте више употребљивим, лакшим за управљање и тестирање.

Пример коришћења убризгавања зависности је код потребе компоненте да дохвата податке са сервера помоћу *Http* захтева. Захтеве не шаљу саме компоненте већ то за њих чине сервиси који се у компоненте убризгавају и тако бивају повезани. Компоненте користе само методе тих сервиса које као повратне вредности враћају податке помоћу претходно описаних токова података*.*

*Рутирање.* Рутер је моћна *JavaScript* алатка који је изградио и одржава *Angular* тим који се може инсталирати из пакета *@angular/router*. Он обезбеђује комплетну библиотеку за рутирање са могућношћу да има различите стратегије за проналажење руте, лак приступ параметрима руте и чуварима за руту како би заштитили компоненте од неовлашћеног приступа.

* + - 1. PHP

С обзиром да *Angular* фрејмворк имплементира рутирање апликације, довољлно је користити *PHP* за дохватање података из базе података и прослеђивање у сировом облику клијенту путем ЈСОН-а. Дакле, *PHP* није задужен за генерисање било каквог *HTML* садржаја јер је то посао *Angular* фрејмворка. Пример комуникације клијента и сервера путем ЈСОН-а приказан је на Слици 2.2.2.4.



Слика 2.2.2.4. Комуникација клијента и сервера путем *JSON*-а

За конекцију са базом коришћен је *PDO,* што доноси неколико предности при писању кода, као што су:

* сигурност (употреба *Prepared Statements*)
* употребљивост (многе помоћне функције за аутоматизацију честих проблема)
* реупотребљивост (интерфејс независан од врсте базе података)

*Prepared Statements* предстаљају једини исправан начин извршавања упита над базом. Захваљујући њима онемогућени су напади на базу, тзв. *SQL Injections.* Њихова примена је потербна када је неку променљиву потребно укључити у упит. На пример променљиву са садржајем корисничког поља форме. Потом је потребно само помоћу 2 методе објекта *PDO*, прво припремити упит са *prepare(),* а потом га извршити над базом са *execute().* На овај начин је база заштићена од малициозних напада (Слика 2.2.2.5).

За извршавање упита коришћене су *PDO* трансакције, где се у случају било какве грешке приликом извршавања неког од упита у оквиру трансакције, ради *rollback*, тј. враћање базе података у стање пре почетка трансакције. У случају да трансакција прође без грешака, *PHP* штампа одговор клијенту путем ЈСОН-а.



Слика 2.2.2.5. *Prepared Statement* за пријаву корисника

* + - 1. JSON

*JSON* (*JavaScript Object Notation*) је лаган формат за размену података. Људима је лако да га читају и пишу. Лак је и за машине да га анализирају и генеришу. Заснован је на подскупу *JavaScript* програмског језика, Стандард *ECMA-262* - 3. издање - децембар 1999. *JSON* је текстуални формат који је потпуно независан од језика, али користи конвенције које су познате програмерима *C*-породице језика, укључујући *C , C++, C#*, *Java*, *JavaScript*, *Pearl, Python* и многи други. Ова својства чине *JSON* идеалним језиком за размену података.

*JSON* се заснива на 2 структуре:

* Колекција парова кључ/вредност. У различитим језицима оваква структура се реализује као објекат, структура, речник (*dictionary*), хеш табела, листа кључева или асоцијативни низ
* Сортирана листа вредности. У многим језицима оваква структура се реализује као низ, вектор или листа

Ово су универзалне структуре и практично сви модерни програмски језици их подржавају у једном од наведених начина.

* + - 1. Bootstrap

*Bootstrap* је бесплатан *CSS* фрејмворк усмерен на флексибилан, прилагођен мобилним телефонима, фронт-енд веб развој. Садржи *CSS* и опционо *JavaScript* темплејте за типографију, форме, дугмад, навигацију и друге компоненте интерфејса.

За потребе реализације жељеног дизајна апликације, као и прилагођавања приказа садржаја апликације на екранима различитих димензија, коришћен је *Bootstrap*-ов *CSS*. Примена овог фрејмворка у апликацији такође је уочљива код искакајућих прозора и порука, које се приказују кориснику у оквиру тих прозора, тзв. *Pop-up*. За њихову имплементацију *Bootstrap* користи, горе поменуте, *JavaScript* темплејте.

За потребе редефинисања неких стилова *Bootstrap*-а или креирања потпуно нових правила која нису предвиђена у овом фрејмворку, писани су кориснички *CSS* фајлови. Више о примерима коришћења и реализације ових фајлова биће речено у 4. поглављу: „Реализација система“.

Дакле, када клијент приступа различитим страницама илити рутама апликације, *Angular* у позадини шаље *HTTP* захтеве за подацима на сервер. Ови захтеви су у ствари *AJAX* позиви. Када је одговор на серверу спреман, он се шаље путем *JSON*-а назад клијенту, тј. у *Angular*, где се кориснику сервира *HTML* садржај странице генерисане помоћу пристиглих података.

1. Опис рада система

Ово поглавље садржи детаљан опис самог рада система и састоји се из корисничког упутства за ветеринара и администратора.

Сви корисници апликације, укључујући и оне непријављене, могу да приступају почетној страници апликације (Слика 3.1). На овој страници налазе се популарни текстови о кућним љубимцима. Ту се могу пронаћи разни савети о здравом начину живота, исхрани, као и промције које нуде ветеринарске клинике.

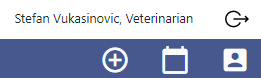


Слика 3.1 Почетна страна са популарним текстовима о љубимцима

* 1. Опис рада система за ветеринара

Ветеринару се након пријаве на систем у навигационом менију приказују 3 опције (Слика 3.1.1.). То су:

* иконица за додавање нових текстова о кућним љубимцима на почетну страну
* календар иконица за преглед распореда заказаних термина и извршавање прегледа
* профилна иконица за преглед личних информација о пријављеном кориснику



Слика 3.1.1. Навигациони мени ветеринара

* + 1. Додавање текстова о кућним љубимцима

Кликом на иконицу из менија приказује се страница (Слика 3.1.2) са пољима форме која је потребно поунити: наслов, текст и слика (опционо). Кликом на дугме *Save* текст се чува у бази и ветеринару се приказује почетна страница апликације са ново додатим текстом.



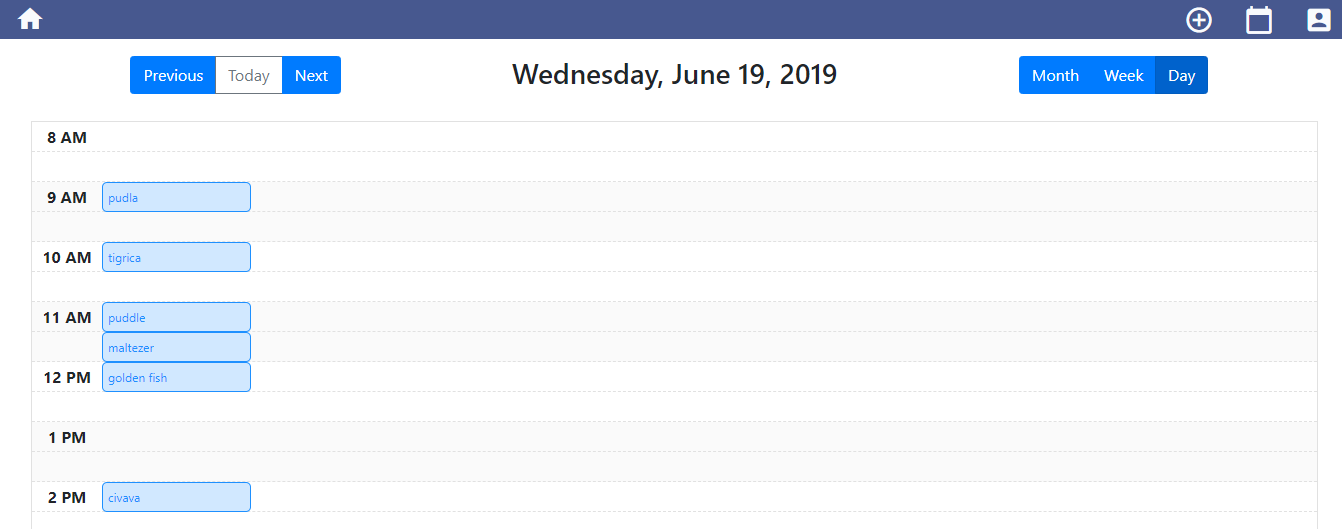
Слика 3.1.2. Додавање текстова о кућним љубимцима

* + 1. Распоред заказаних термина и извршавање прегледа

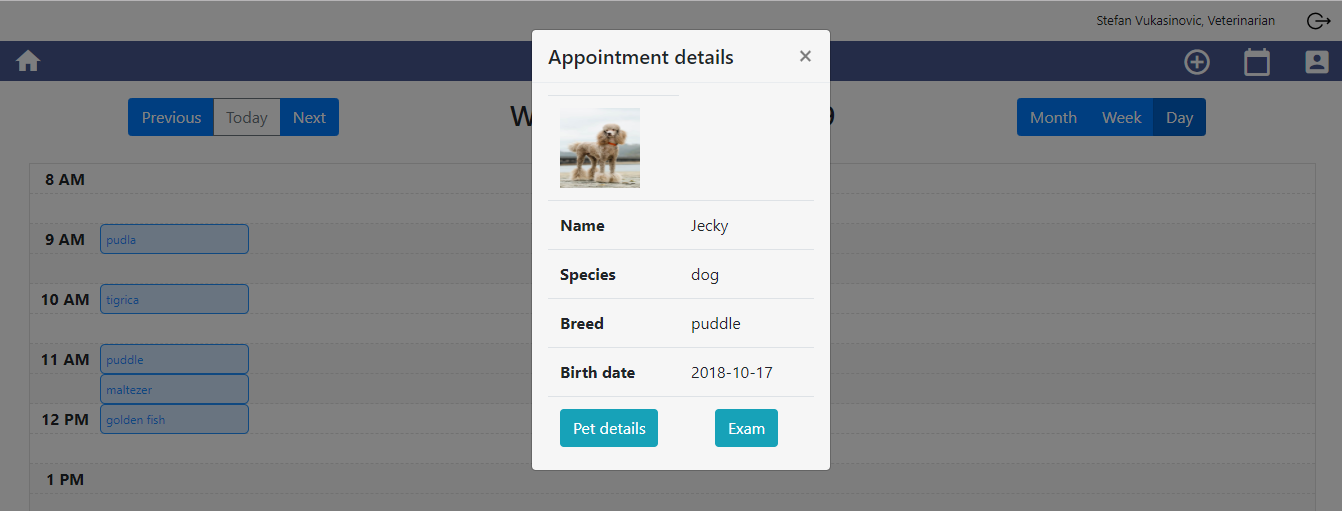
Након пријаве на систем, ветеринару се приказује страна са календаром, где је иницијално приказан распоред заказаних прегледа за текући дан (Слика 3.1.2.1). Ветеринару је омогућено да прегледа распоред прегледа још и на недељном и месечном нивоу. Приказана је само врста љубимца који се прегледа.

Кликом на неки од заказаних термина, приказују се детаљи о љубимцу, као и 2 опције (Слика 3.1.2.2):

* одлазак на страницу са детаљним информацијама о љубимцу - *Pet details*
* извршавање прегледа изабраног љубимца - *Exam*



Слика 3.1.2.1. Дневни приказ заказаних прегледа

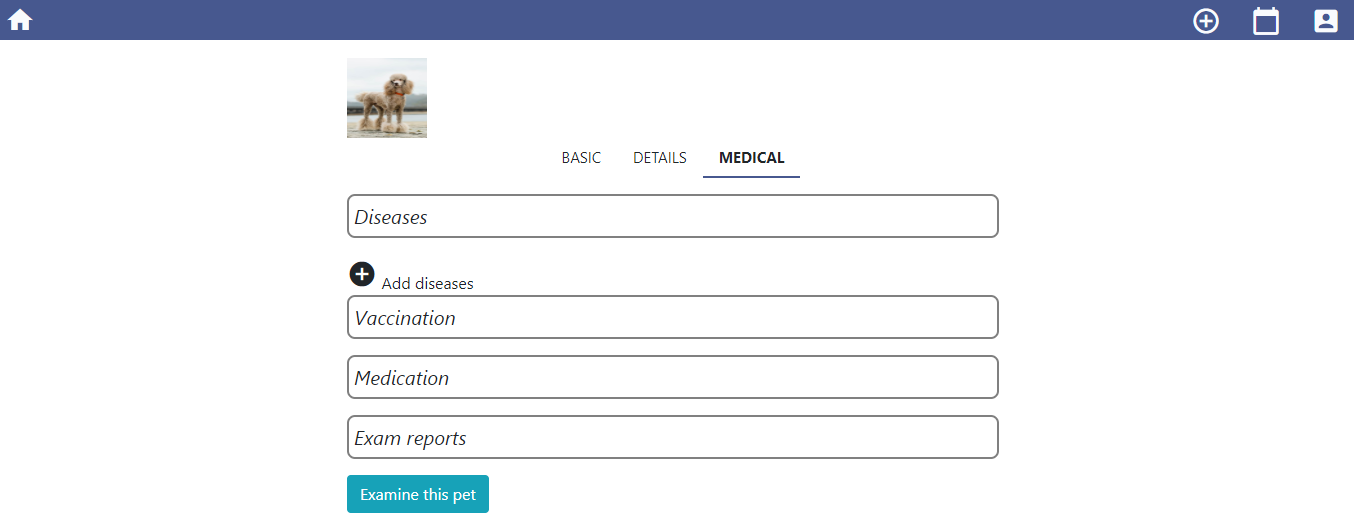


Слика 3.1.2.2. Искакајући прозор са детаљима заказаног прегледа

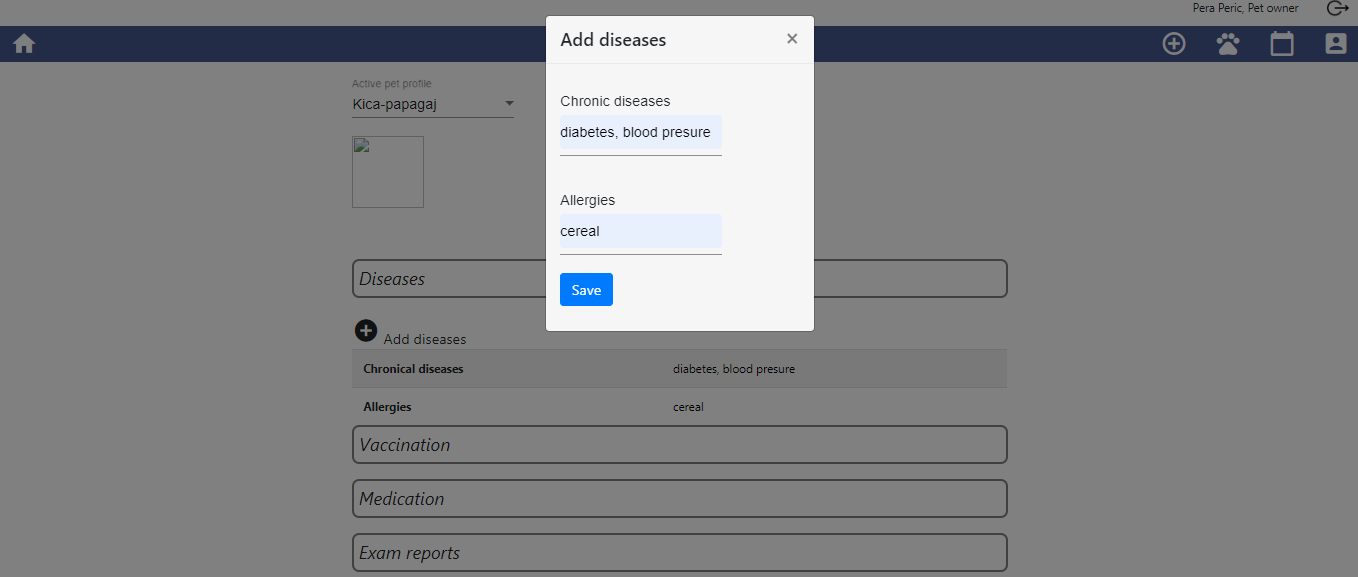
* + - 1. Страница са детаљима о љубимцу – Pet Details

Ово је страница којој могу приступати и ветеринари и власници љубимаца, с тиме што су им привилегије различите. Ветеринари могу прегледати основне, као и детаљне информације о љубимцу без права измене истих. Такође, доступан им је преглед здравственог картона љубимца, са историјом болести, вакцинација, терапија које је љубимац добијао, као и листом извештаја са свих претходних прегледа. Ветеринари могу додавати болести у картон по потреби, кликом на поље *Diseases*, па потом на *Add disease* (Слика 3.1.2.3). Потом се у искакајућем прозору додају болести или алергије и сачувају измене (Слика 3.1.2.4).

Ветеринар на овој страници има и опцију да прегледа иазабраног љубимца кликом на дугме *Examine this pet*  (Слика 3.1.2.3). Обављање прегледа је објашњено у следећем пододељку.



Слика 3.1.2.3 Опције за додавање болести – *Add disease* и обављање прегледа – *Examine this pet*

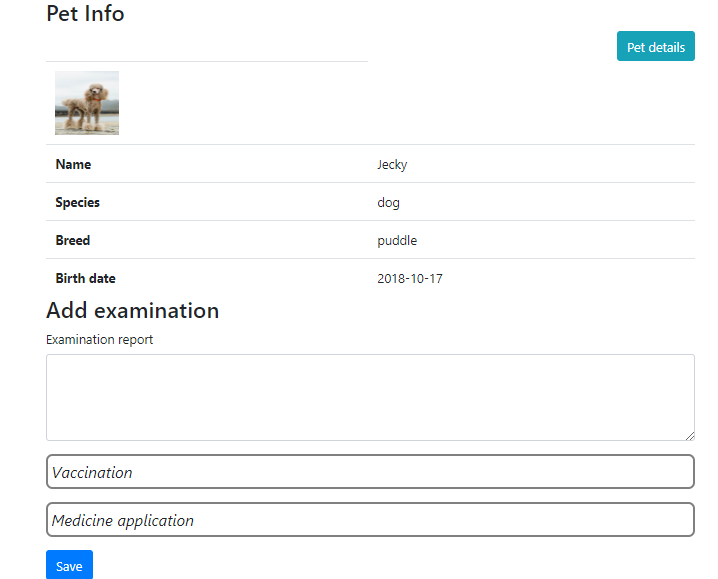


Слика 3.1.2.4. Искакајући прозор за додавање болести и алергија

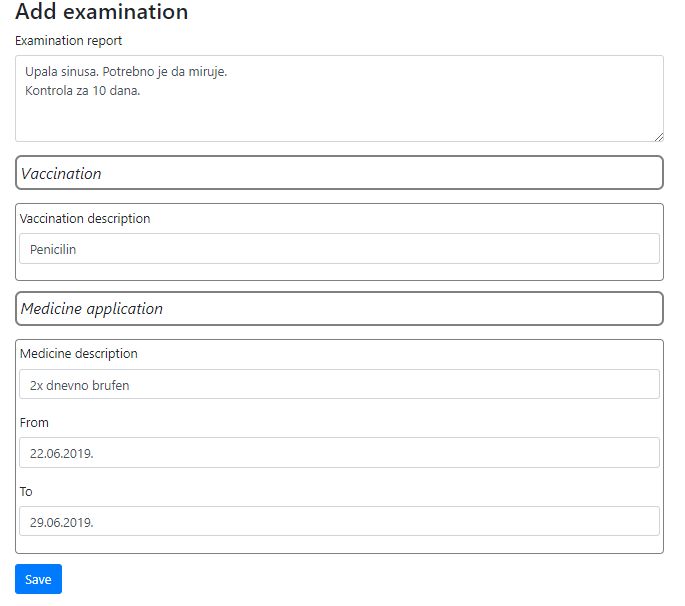
* + - 1. Извршавање прегледа

На овој страници ветеринару се приказује форма за попуњавање извештаја са прегледа (Слика 3.1.2.5). Опционо ветеринар може да попуни рубрике за вакцинацију и преписану терапију кликом на жељена поља. Ветеринару се приказују додатна поља форме која се у том случају морају попунити (Слика 3.1.2.6).

По успешном завршетку прегледа ветеринару се приказује календар са прегледом термина за текући дан. Обављени преглед је обојен зеленом бојом као знак да је успешно обављен.

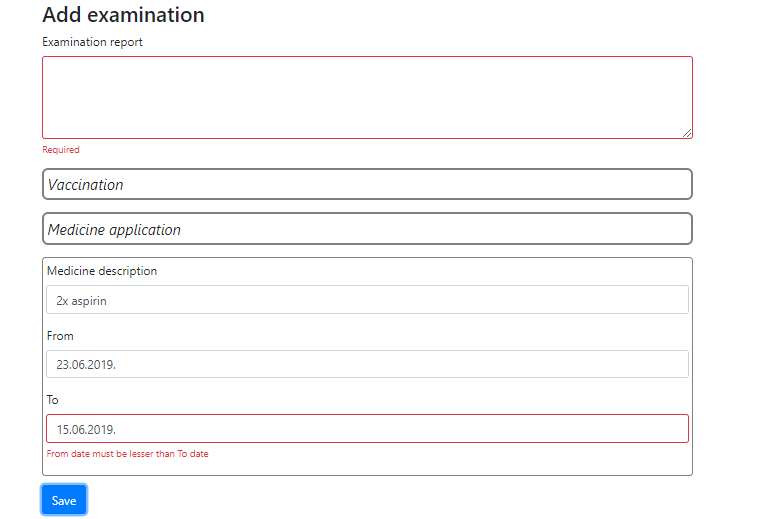


Слика 3.1.2.5. Страница за извршавање прегледа љубимца



Слика 3.1.2.6. Форма за попуњавање извештаја са ветеринарског прегледа

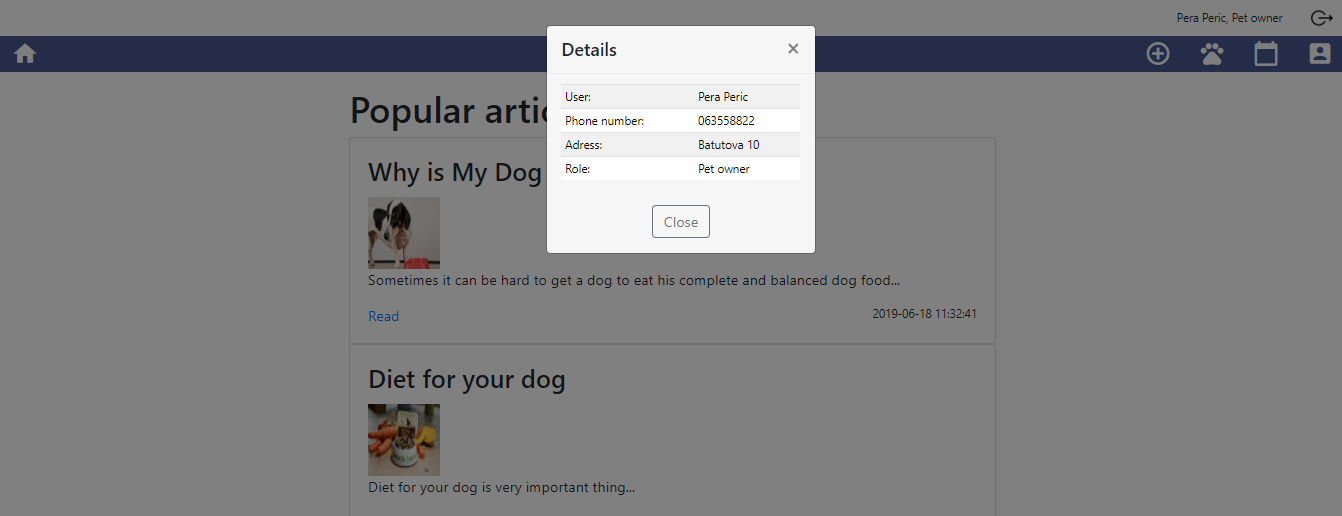
У случају да се при уносу података направе грешке, сигнализира се порука о грешци и форма се не чува све док се не унесу валидне вредности (Слика 3.1.2.7).



Слика 3.1.2.7. Приказ грешке при уносу невалидних података

* + 1. Преглед личних информација пријављеног корисника

Кликом на профилну иконицу из менија приказују се подаци пријављеног корисника u искакајућем прозору (Слика 3.1.3.1).

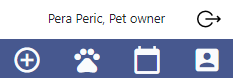


Слика 3.1.3.1. Преглед личних информација пријављеног корисника

* 1. Опис рада система за власника љубимаца

Власницима се након пријаве на систем у навигационом менију приказују 4 опције (Слика 3.2.1.). То су:

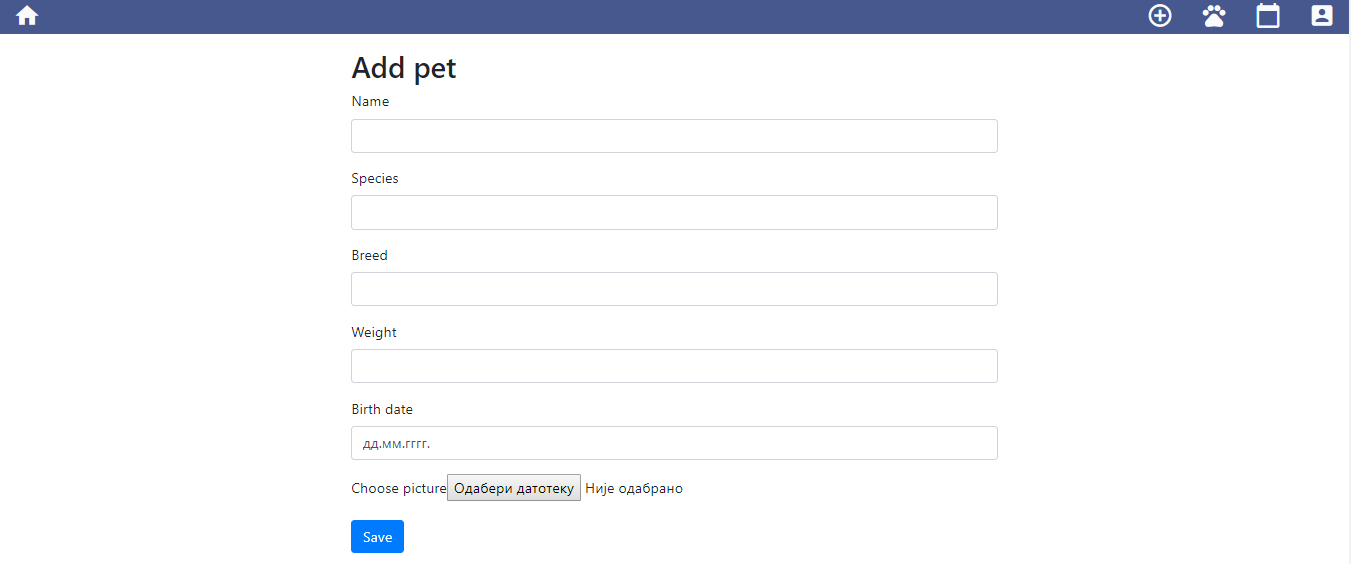
* плус иконица за додавање новог профила за кућног љубимца
* шапица иконица за преглед постојећих профила и избор активног профила
* календар иконица за преглед распореда заузетих термина и заказивање прегледа
* профилна иконица за преглед личних информација о пријављеном кориснику



Слика 3.2.1. Навигациони мени за власника љубимаца

* + 1. Додавање новог профила за љубимца

Власници, као што је раније поменуто, имају право да направе неограничен број профила за онолико љубимаца колико их поседују. Приказује им се форма са основним информацијама о љубимцу: име, врста, раса, тежина, датум рођења и слика која је опциона (Слика 3.2.1.1).



Слика 3.2.1.1 Форма за додавање новог профила кућног љубимца

* + 1. Избор активног профила и детаљи љубимца

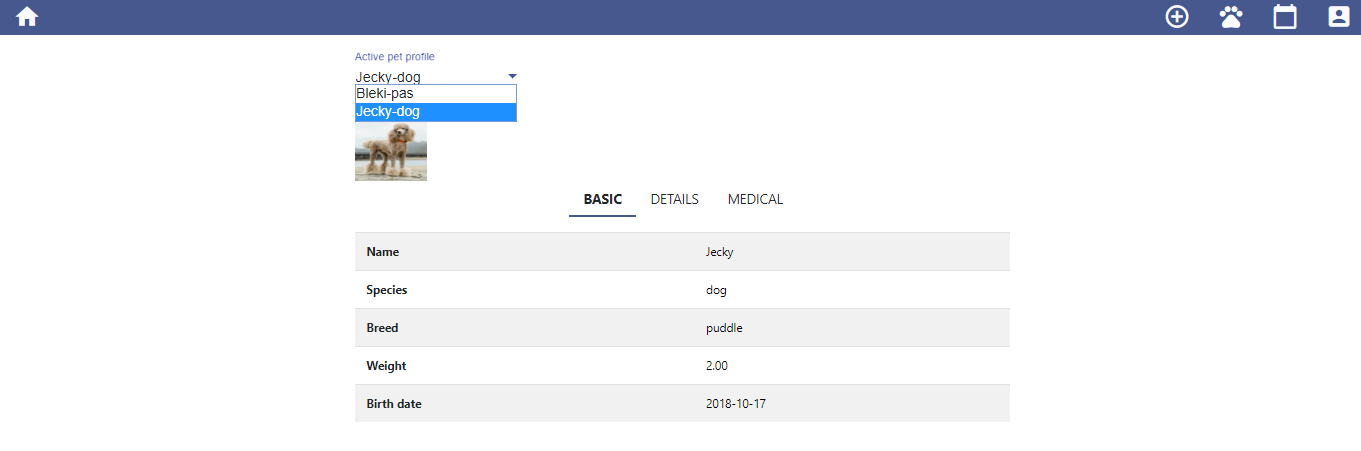
По успешном креирању профила, он аутоматски постаје активан и кориснику се приказује страница са детаљима ново креираног љубимца.

На врху странице кориснику се приказује који профил је тренутно активан и у падајућем менију му се нуди промена активног профила (Слика 3.2.1.2).

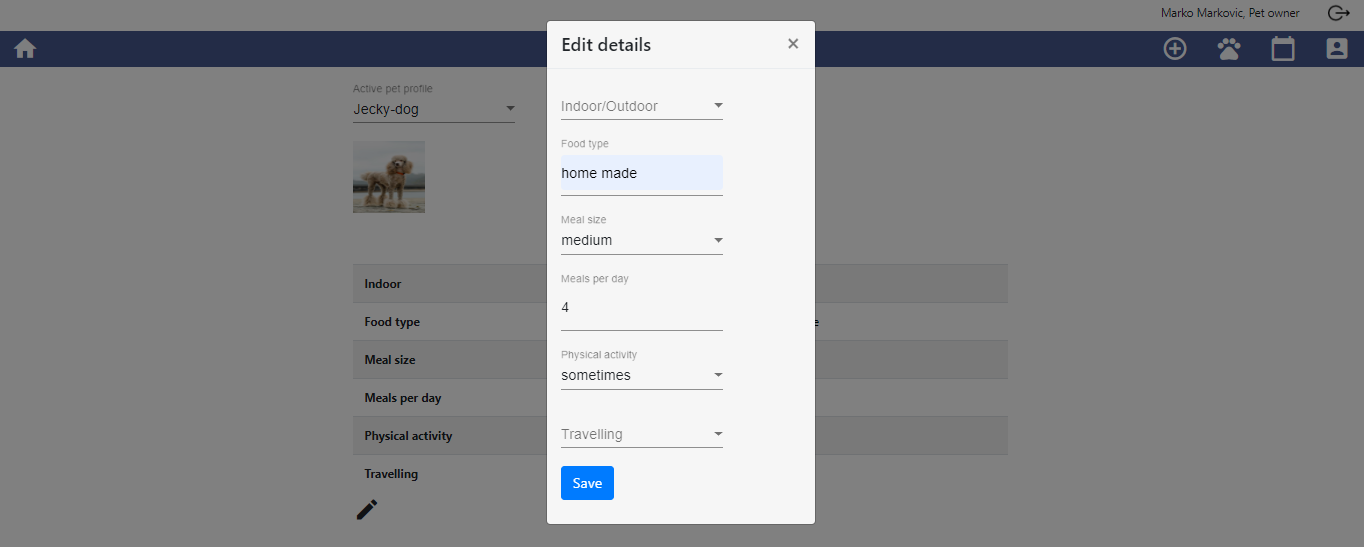
Детаљне карактеристике могу се унети кликом на *Details*, па потом кликом на оловчицу на дну странице. Приказује се форма са пољима у искакајућем прозору. Сва поља су опциона и не морају бити попуњена при чувању измена (Слика 3.2.1.3).

Преглед здравственог картона налази се у трећој категорији – *Medical*. Чине га 4 подкатегорије (Слика 3.2.1.4):

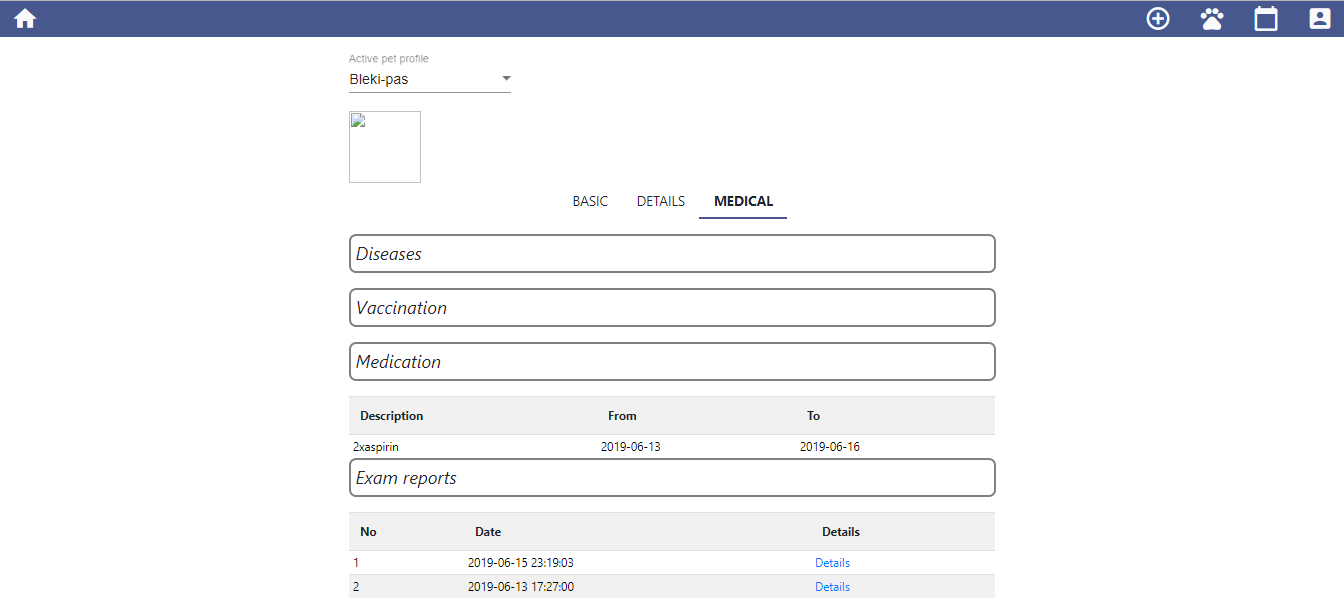
* болести и алергије, може их додавати и власник љубимца поред ветеринара
* вакцинације
* преписана терапија
* извештаји са претходних прегледа



Слика 3.2.1.2 Страница са детаљима о профилу љубимца



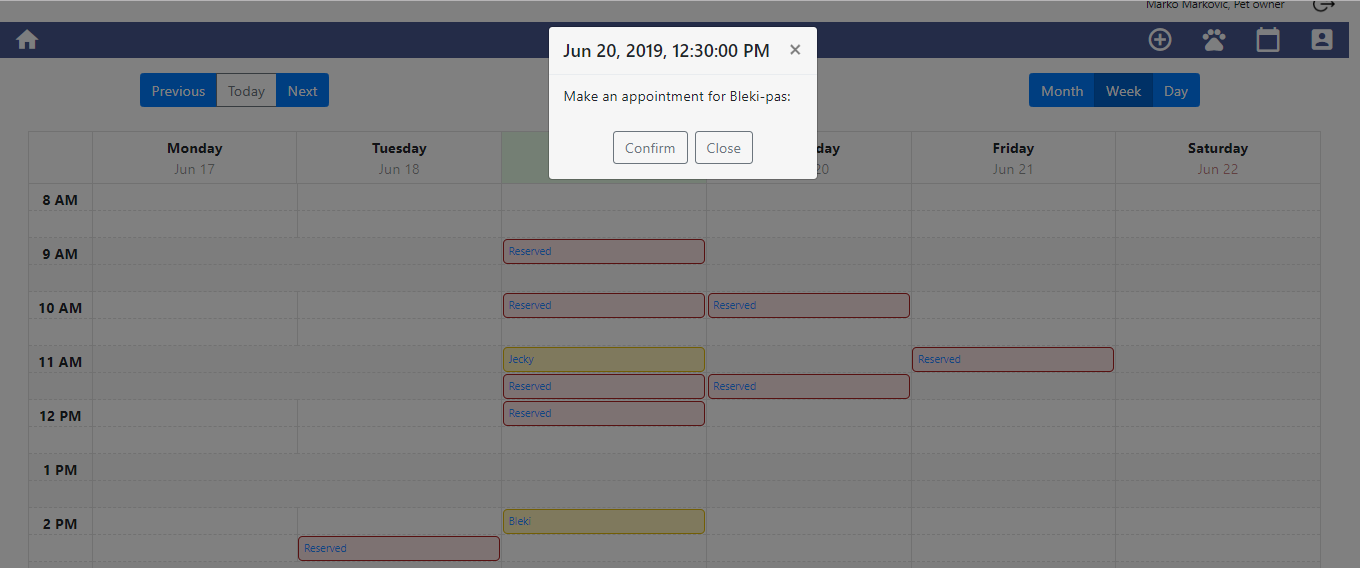
Слика 3.2.1.3 Искакајући прозор за додавање детаљних информација за љубимца



Слика 3.2.1.4 Здравствени картон љубимца

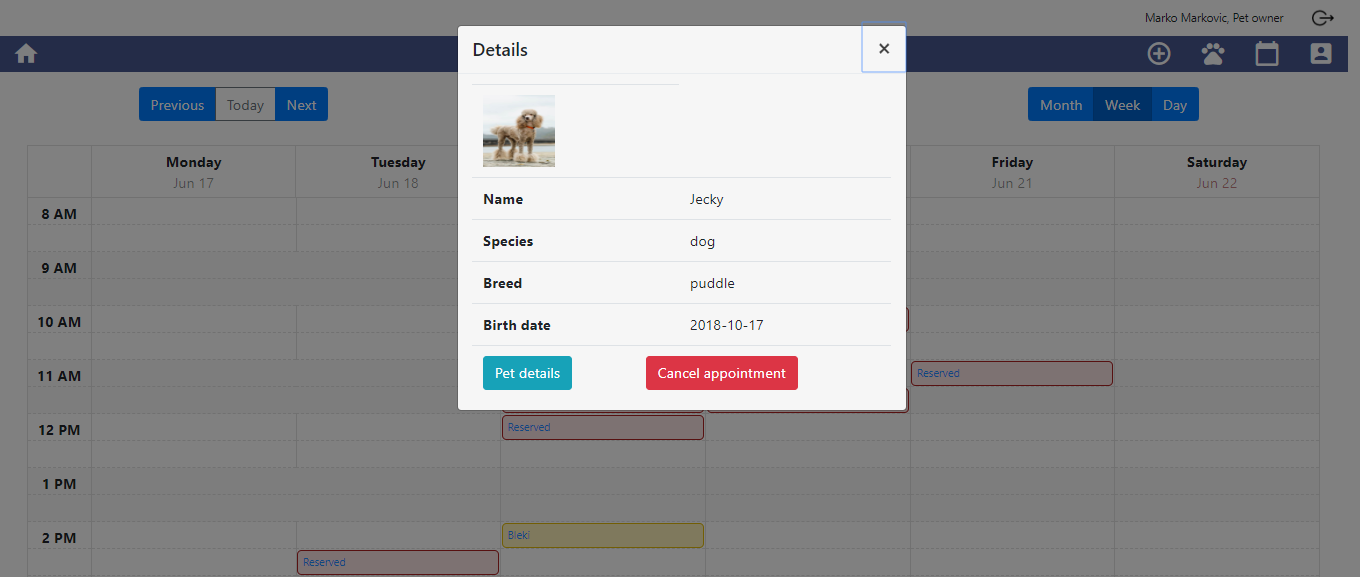
* + 1. Преглед распореда заузетих термина и заказивање прегледа

Кориснику се приказује календар, са иницијално приказаном текућом недељом. Црвеном бојом су приказани заузети термини. Кликом на било који слободан термин може се заказати преглед за тренутно активног љубимца (Слика 3.2.3.1).



Слика 3.2.3.1. Заказивање термина прегледа

Заказани прегледи пријављеног корисника обојени су жутом бојом. Кликом на неки од таквих прегледа корисник може да откаже преглед кликом на дугме - *Cancel appointment* (Слика 3.2.3.2).



Слика 3.2.3.2. Отказивање прегледа

1. Реализација система

У овом поглављу биће објашњено како су реализовани одређени делови апликације, као и проблеми који су се јављали током рада и њихова решења. Уз свако решење биће приложени и делови програмског кода како би се стекао што бољи увид у проблематику решења.

* 1. Почетна страница, компоненте и руте апликације

Предност *Angular* фрејмворка је у томе што има доста сложенију структуру организације фајлова, који комуницирају и заједно чине целину која перфектно функционише. У овом потпоглављу биће у кратким цртама објашњено како функционише једна *Angular* апликација.

На слици приказана је почетна страница апликације *index.html*, на којој је важно уочити да се у њеном телу, помоћу селектора *<app-root>*, укључује нова компонента која представља основу илити оклоп апликације – *app.component.html* (Слика 4.1.1).

Код ове компоненте (Слика 4.1.2), битно је уочити део кода где се укључује садржај компоненте тренутне руте апликације, а то је селектор *<router-outlet>.* На основу тренутно активиране руте, која се чита из урл-а, на основу фајла за конфигурацију рута (Слика 4.1.3), приаказаће се темплејт одабране компоненте на месту *<router-outlet>* селектора.

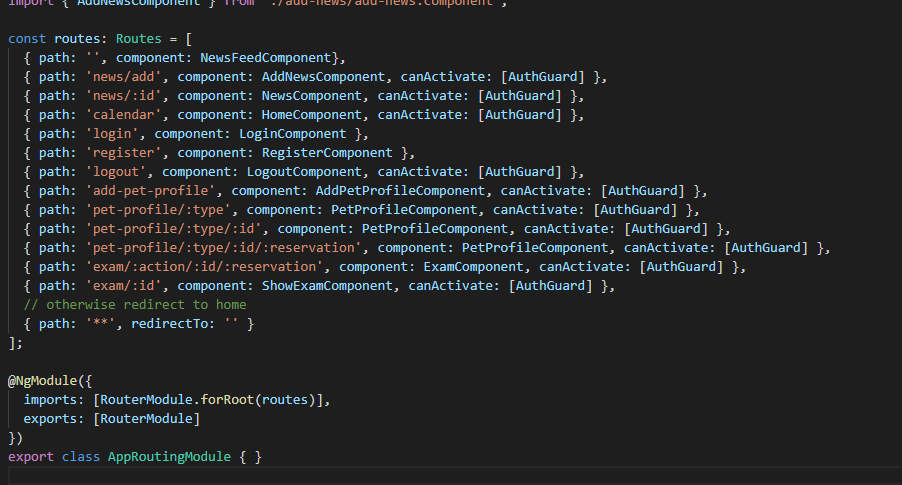
На претходно приказани начин *Angular* раздваја логичке целине и дозвољава нам да реализујемо компоненте које се могу посматрати као потпуно независне међусобно и тиме нам омогућава писање бољег кода који је употребљив за коришћење у више апликација - *reusability*. Свака компонента има свој темплејт, контролер, као и *CSS* фајл, о чему ће бити речи нешто касније у тексту.



Слика 4.1.1. Почетна страница апликације – *index.html*



Слика 4.1.2. *app.component.html*

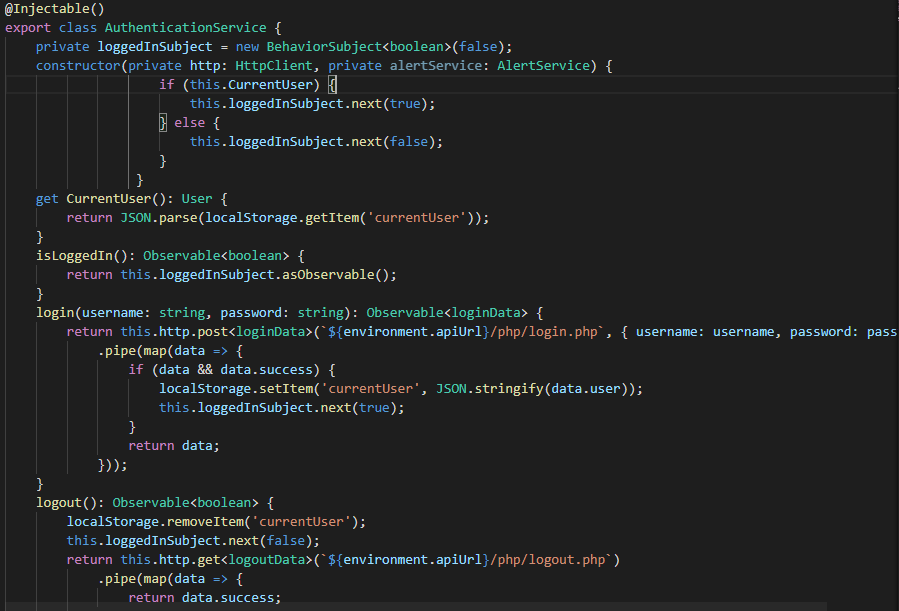


Слика 4.1.3. Фајл за конфигурацију рута – *app-routing.module.ts*

* 1. Реализација компоненте навигационог менија

Навигациони мени потребно је приказивати на свим страницама апликације, у сваком тренутку, било да је корисник пријављен или не. Стога је мени реализаован као компонента и укључен је у основну компоненту апликације помоћу селектора *<app-navbar>* (Слика 4.1.3). Проблем који се јавио приликом реализације, био је динамичка промена ставки менија у зависности од тога да ли је корисник пријављен или не, као и да ли је ветеринар или власник љубимаца.

За решење овог проблема, потребно је сигнализирати компоненти менија када год се корисник пријави или одјави са система, како би се ажурирала променљива са подацима корисника који је пријављен на систем. То се решава помоћу сервиса за аутентикацију корисника, који имплементира методу *isLoggedIn()*, која ће слати информацију сваки пут када се корисник пријави или одјави са система (Слика 4.2.1). У оквиру овог сервиса имплементиране су и 2 основне методу за пријаву и одјаву корисника са ситема, *login()* и *logout()*.



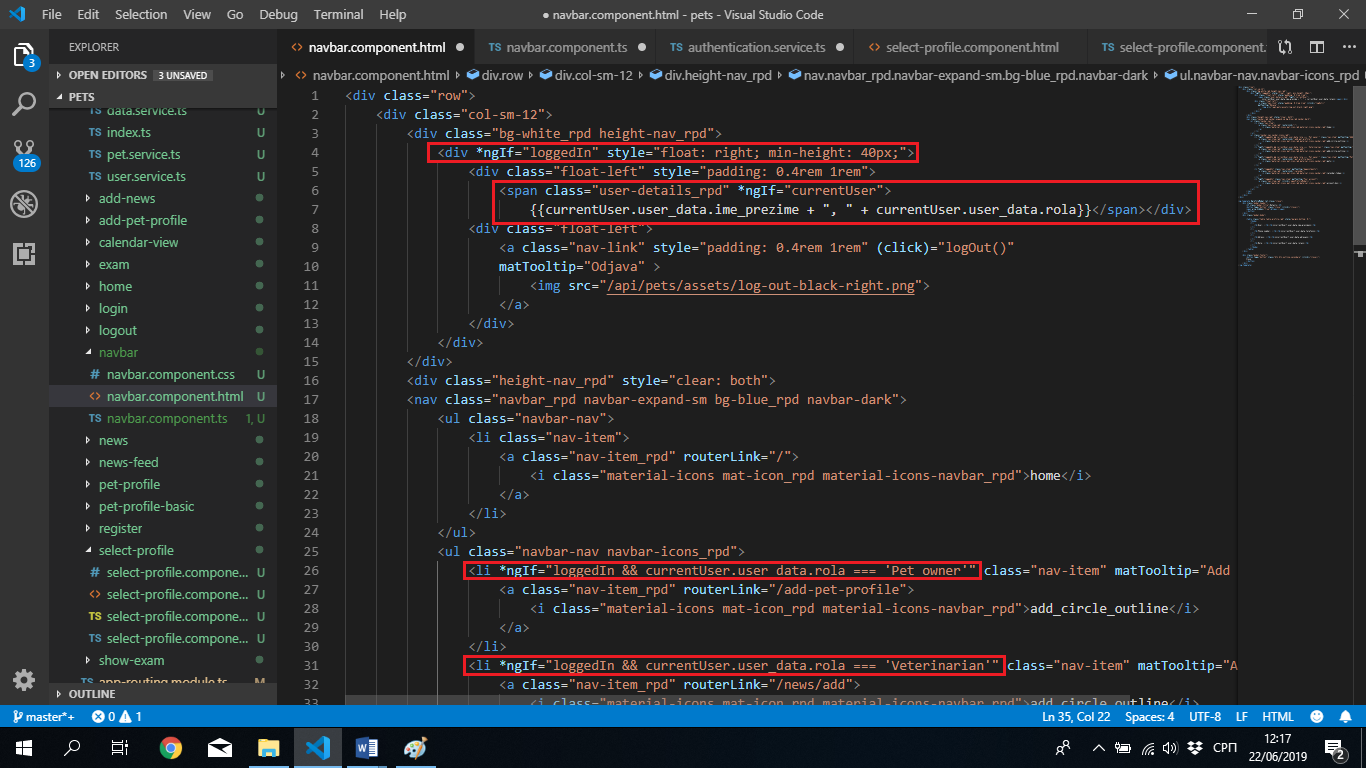
Слика 4.2.1. Сервис за аутентикацију корисника

Овај сервис се убризгава у контролер компоненте помоћу навођења у коснтруктору контролера компоненте менија (Слика 4.2.2). Информација ће бити враћена као *Observable*, а све што је потребно је да контролер ослушкује промену и да је обради када се она догоди – *subscribe*. У случају када се корисник пријави на систем, потребно је прочитати његове податке и сачувати их у променљиву тренутног корисника, док је у случају одјаве са система, потребно ту променљиву обрисати, постављањем на вредност *null*.

Темплејт компоненте менија (Слика 4.2.3) садржаће одговарајући садржај у зависности од тога да ли је корисник пријављен или не, као и да ли је пријављени корисник ветеринар или власник љубимаца. Ово се постиже коришћењем структурне наредбе *\*ngIf=”*услов*”*, где се услов проверава на основу вредности променљивих *loggedIn* и *currentUser* из контролера. Када је корисник пријављен у горњем десном делу менија, приказују се име и презиме корисника и његова рола - ветеринар или власник. У доњем делу менија, у зависности од роле, кориснику се приказују различите опције.



Слика 4.2.2. Контролер компоненте менија



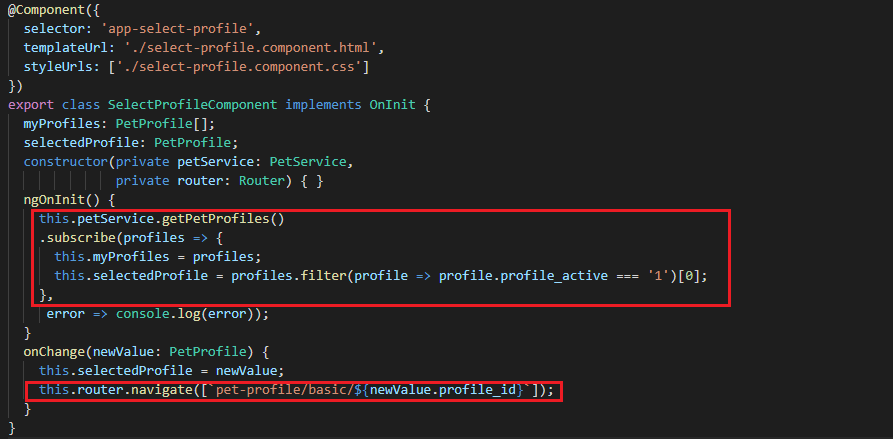
Слика 4.2.3. Темплејт компоненте менија

* 1. Реализација компоненте за одабир активног профила за љубимца

За потребе реализације ове компоненте коришћено је селект поље, у којем је на почетку селектован тренутно активан профил. Прво је потребно дохватити из базе података све постојеће профиле љубимаца пријављеног корисника, а потом је потребно филтрирати из низа профила онај профил који је активан и поставити га као тренутну вредност селект поља.

Дохватање профила из базе података обавља се помоћу претходно поменутих *HTTP* захтева који се шаљу на сервер и на серверу се извршава *PHP* скрипта која враћа одговор у *JSON* формату. Овај захтев се шаље приступом рути „*/pet-profile/basic/id”*, где је параметар ид љубимца потребан само уколико ветеринар прегледа профил неког љубимца. У случају да власник љубимца прегледа профил приликом посете овој рути, ид није потребан јер се подразумева да се подаци дохватају за тренутно активног љубимца.

Ово је реализовано помоћу сервиса *PetService* и његове методе *getPetProfiles(),* који се позивају у иницијализационом блоку компоненте – *ngOnInit()* (Слика 4.3.1). Сервис је компонента која нам служи за комуникацију са сервером тако што шаље поменуте *HTTP* захтеве помоћу модула *HttpClientModule* који се мора импортовати у основном модулу апликације – *AppModule*, а потом користи у компоненти сервиса помоћу инстанце класе *HttpClient*. У овој класи подржани су основни *get* и *post* *http* захтеви помоћу истоимених метода (Слика 4.3.2).



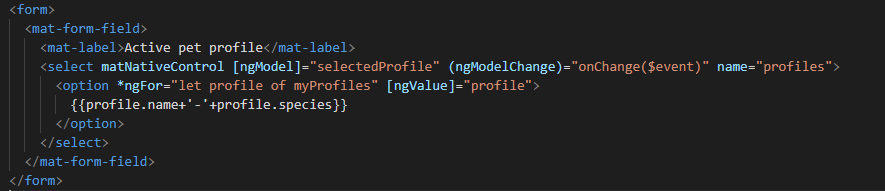
Слика 4.3.1. Контролер компоненте за избора активног профила љубимца

У случају промене изабраног профила, позива се метода *onChange($event)* и прослеђује јој се ново изабрана вредност. Кориснику се тада приказује страница са детаљима о љубимцу чији је профил активиран – „*/pet-profile/basic/id”*. У овом случају, када власник приступа рути са параметром ид, тада се на серверу памти промена активног профила у бази података. Више о томе у наредном потпоглављу.

Темплејт компоненте приказан је на слици 4.3.3. Треба приметити да је помоћу директиве *[ngModel]=”selectedProfile”* извршено повезивање селект поља из темплејта са одговарајућом вредности активног профила помоћу једнострок везивања – *One way binding*. За потребе попуњавања осталих опција овог селект поља, коришћена је још једна веома корисна структурна директива - *\*ngFor=”let profile of myProfiles”*, која динамички уграђује *HTML* код опција селекта. И то ће урадити онолико пута колика је дужина низа профила - *myProfiles* и која представља низ свих профила пријављеног корисника. Важно је приметити да се вредности променљивих из контролера у самом темплејту могу приказати помоћу двоструких витичастих заграда – *{{ }}*.



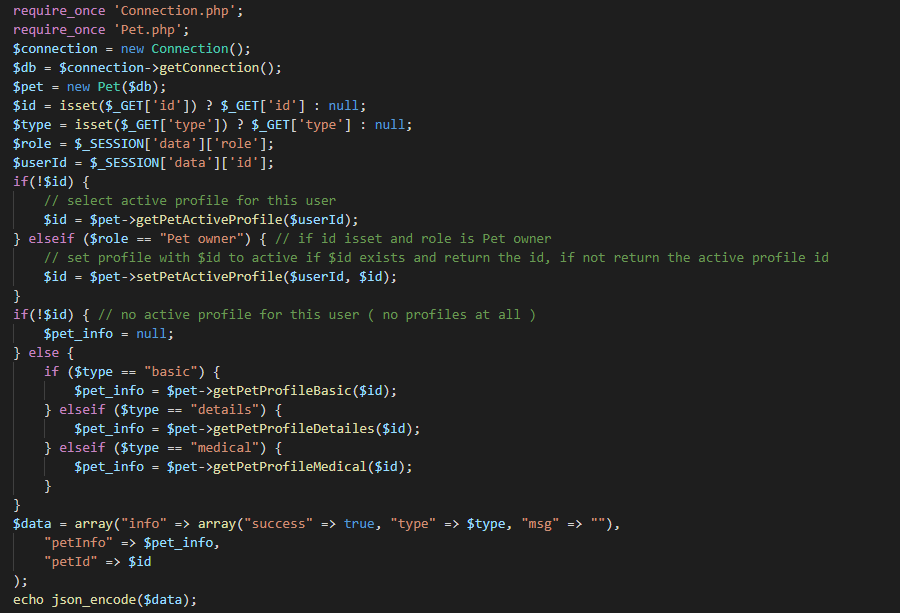
Слика 4.3.2. Слање *http* и *get* захтева помоћу класе *HttpClient* из модула *HttpClientModule*



Слика 4.3.3. Темплејт компоненте за избор активног профила љубимца

* 1. Преглед профила љубимца и промена активног профила

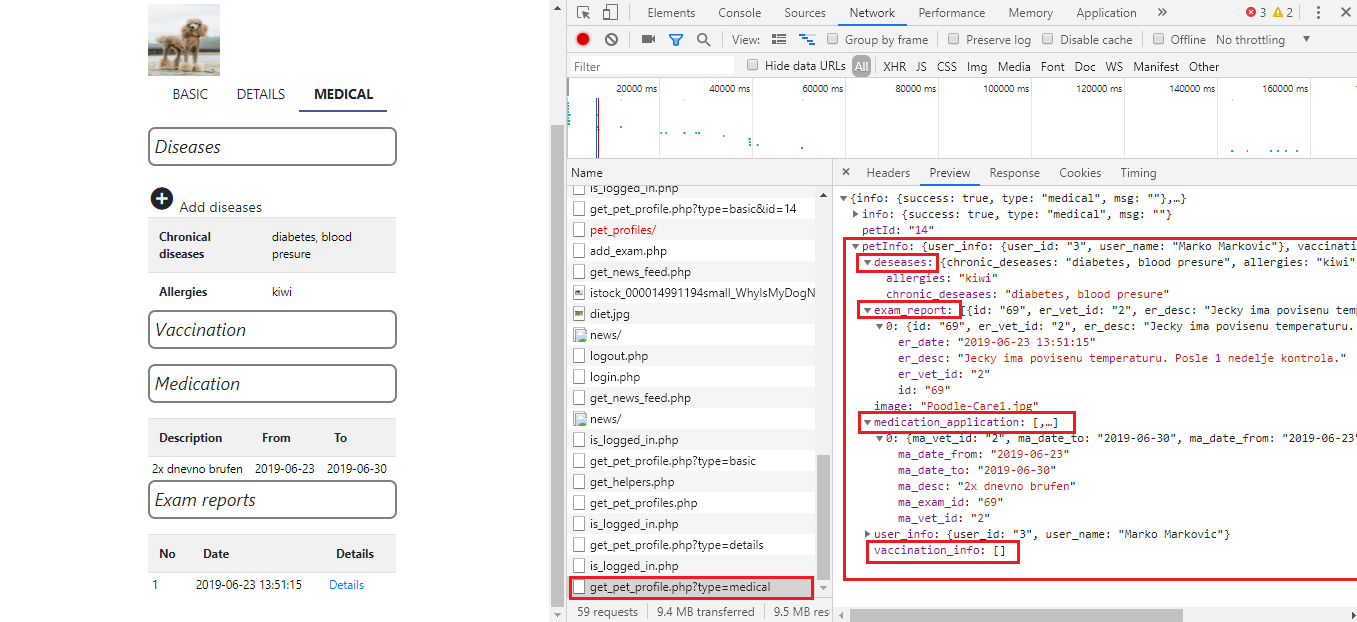
Могућност прегледа профила љубимаца имају власник тог љубимца и ветеринари. Профил једног љубимца чине: основне информације, детаљне информације и медицински картон (сви прегледи, терапије, вакцинације и болести). Све 3 наведене категорије података о љубимцу дохватају се посебно са сервера када се кликом на одређену категорију иницира захтев. *PHP* скрипта, која се извршава при овим захтевима и која враћа као одговор одговарајуће податке, састоји се од метода које су дефинисане у одвојеном фајлу „*Pet.php*“ (Слика 4.4.1). Имплементације тих метода нису приложене због свог обима. У наредном параграфу објашњено је како се ови подаци преносе до клијента, тј. каква је структура *JSON* одговора са сервера (Слика 4.4.2).



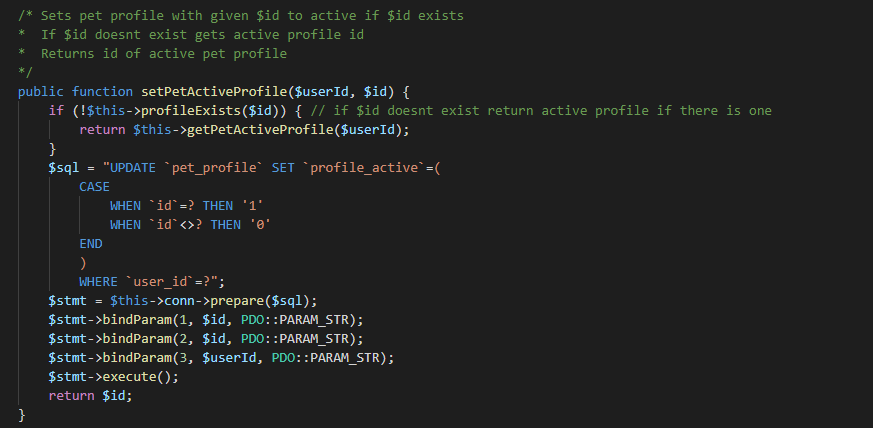
Слика 4.4.1. *PHP* фајл за дохватање профила и сетовање активног профила љубимца

Одговор са сервера представља *JSON* објекат који садржи 3 поља: информације о усцпешности операције – *info*, ид љубимца чији је профил и информације о љубимцу – *petInfo*, који нам је од највећег значаја. Ово поље је објекат који се састоји од 5 поља, која су редом низови објеката: болести, извештаји са прегледа, терапија, власникове информације и вакцинације.

У претходном потпоглављу напоменуто је да се и одабир активног профила љубимца реализује на рути „*/pet-profile/basic/id”*, тј. с тиме што у овом случају корисинк који приступа овој рути мора бити власник љубимца (Слика 4.4.1). Дакле, уколико је прослеђен параметар ид на ову *PHP* страницу, после провере да је корисник власник љубимца, позива се метода класе *Pet.php* која сетује активан профил за власника тако што прослеђује ид власника (чита се из сесијске променљиве) и ид љубимца чији профил постаје активан. На слици 4.4.3 приказана је имплементација те методе.



Слика 4.4.2. *JSON* објекат са информацијама о профилу љубимца



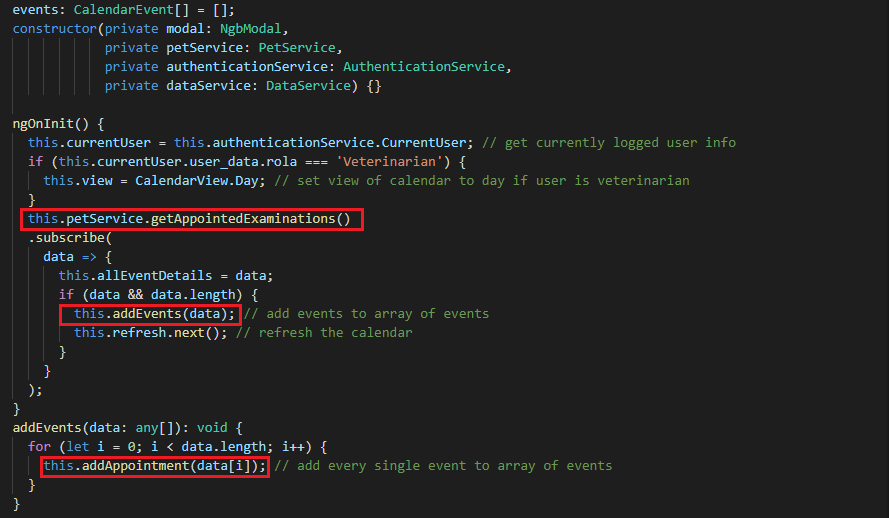
Слика 4.4.3. *PHP* фајл за сетовање активног профила љубимца

* 1. Реализација компоненте календара

Компонента за приказ календара је најважнија и најсложенија компонента у апликацији. Када су у питању власници љубимаца, у оквиру ове компоненте реализована је функционалност увида у слободне термине, резервисања прегледа за љубимце, детаља о претходним прегледима... Ветеринари имају и опцију уписивања конкретног прегледа када се преглед догоди. У наставку текста биће објашњене само реализације најважнијих функционалности ове компоненте.

При иницијализацији компоненте потребно је са сервера дохватити све заказане прегледе, а потом их процесирати на одговарајући начин пре самог приказивања у календару, у зависности од тога да ли је корисник власник љубимца или ветеринар.

Дохватање података, тј. заказаних прегледа, са сервера је као и до сада реализовано помоћу *HTTP* захтева, који шаље метода сервиса „*getAppointedExaminations()*“. Потом је све заказане прегледе потребно учитати у меморији на одговарајући начин како би били успешно приказани у календару (Слика 4.5.1). Прегледи се чувају у низу термина/догађаја календара „*events: CalendarEvent[]*“, а за додавање прегледа задужена је функција „*addAppointment(data)*“ (Слика 4.5.2).

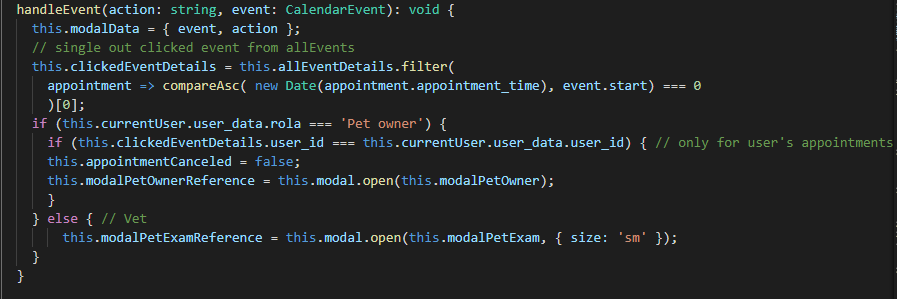


Слика 4.5.1. Иницијализација компоненте календара



Слика 4.5.2. Форматирање заказаних термина за приказ у календару

Све опције које се могу извршити на неким прегледом,, било да је у питању власник или ветеринар, доступне су кликом на појединчан догађај календара, тј. заказани термин прегледа. Приликом обраде клика на неки од догађаја, прво је потребно издвојити све информације о том догађају из низа догађаја календара. Потом приказати одговарајуће опције над кликнутим догађајем у зависности од тога да ли је корисник власник или ветеринар. Приметити да власник има опције само над догађајима везаним за своје љубимце (Слика 4.5.3).



Слика 4.5.3. Обрада клика над догађајем календара

Опције које има власник су преглед профила љубимца и отказивање изабраног прегледа, док ветеринар има опцију одласка на страну за извршавање, тј. уписивање прегледа у базу. Поменуте опције нису приказане у наставку текста.

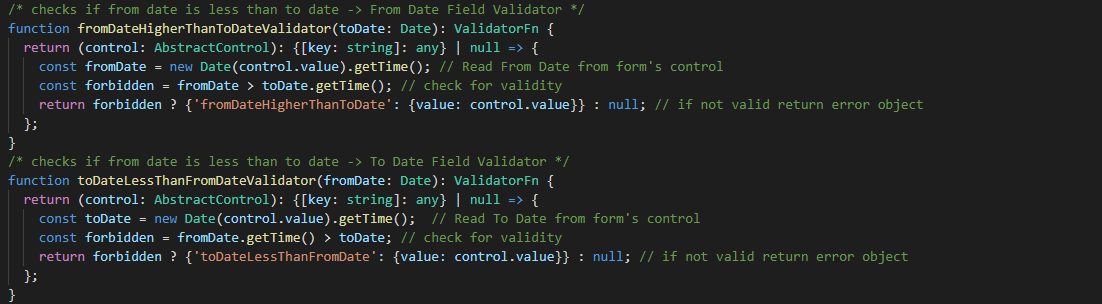
* 1. Валидација међусобно зависних поља форме у *Angular*-у

Валидација поља форме у *Angular*-у представља проверу валидности кориснички унетих података пре слања форме на сервер. За имплементацију ових провера користе се валидатори, који су имплементирани као засебна класа у *Angular*-у и налазе у библиотеци за форме. Када је реч о апликацији, најчешће су коришћени предефинисани валидатори за проверу обавезних поља – „*Validators.required“*. Оваква врста валидатора додељује се пољима форме и они се односе искључиво на проверу садржаја тих поља форме.

Проблем настаје када је потребно вршити проверу више међусобно зависних поља у исто време. Пример у апликацији је при додели почетног и крајњег датума преписане терапије љубимцу. Као што је и логично, није дозвољено да почетни датум буде после крајњег датума.

Да би се оваква функционалност постигла, потребно је креирати кориснички валидатор и применити га на овим пољима. Ово се постиже тако што се дефинише функција која се додељује пољу форму и чија је повратна вредност валидаторска функција. Валидаторска функција је функција која као параметар прихвата поље форме – *AbstractControl*, док је њена повратна вредност или објекат који садржи врсту грешке или „*null“*, у случају да грешка привалидацији не постоји. Тело валидатора садржи потребну логику за проверу исправности.

У нашем случају потребно је креирати два валидатора, за оба поља датума по један. У телу валидаторске функције проверава се да ли је датум „*from*“ мањи од датума „*to*“ у валидатору за поље „*from*“ и обрнуто за валидатор поља „*to*“ (Слика 4.6.1).



Слика 4.6.1. Валидаторске функције за валидацију датума при преписивању терапије

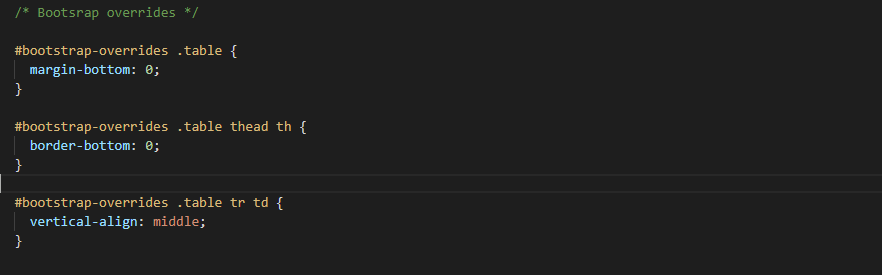
* 1. Промена стилова дефинисаних у *Bootstrap*–у

Проблем се јавља уколико има потребе применити другачије особине стилова од оних дефинисаних у *Bootstrap*-у на одређене елементе који користе класе из *Bootstrap*-а. Притом је потребно задржати све остале особине које елемент поседује и чита из *Bootstrap* фајла.

Најбоље решење овог проблема је применом *CSS* приоритета. Сваки селектор има своју тежину, тј. број поена које доноси елементу:

* 100 поена за ид – *id*
* 10 поена за класу или псуедо-класу – *class/pseudo-class*
* 1 поен за таг селектор
* 1000 поена ако је стил дефинисан у оквиру тага елемента – *inline*

У случају преклапања дефиниција особина стила за исти елемент, побеђује онај стил са највећим бројем поена. С тога, најбоље решење овог проблема је додавање ид-а основном елементу на страници: „<body id="bootstrap-overrides">“. На овај начин, када год желимо да редефинишемо неку особину стила из *Bootstrap*-а, потребно је испред дефиниције стила за жељени елемент навести ид селектор, који ће јој доделити 100 поена и аутоматски бити у предности у односу на дефиницију у *Bootstrap*-у. Пример употебе оваг решења у апликацији, уочљив је код редефинисања неких особина из *Bootstrap* класе за табеле – „ class=’table’ “ (Слика 4.7.1).



Слика 4.7.1. Редефинисање особина класа из *Bootstrap*-а

1. Закључак

Реализована веб апликација нуди се на коришћење власницима кућних љубимаца и ветеринарским клиникама. Ова апликација власницима љубимаца нуди стручну помоћ и савете у вези са здрављем и квалитетним животом њихових љубимаца. У то спадају планирање и заказивање одлазака на ветеринарске прегледе, кућна нега, савети ветеринара, лекови и редовна вакцинација. Апликација представља ветеринарски портал који ће се уступати ветеринарским клиникама, а које ће својим корисницима наплаћивати услуге коришћења.

За потребе реализације апликације коришћене су савремене технологије – за реализацију корисничког интерфејса коришћен је *Angular* фрејмворк, као и технике реактивног програмирања које су подржане у њему. За размену података између клијента и сервера коришћен је *JSON* формат, док је као серверски језик коришћен *PHP*. За потребе прилагођивања страница различитим димензијама екрана, коришћен је *Bootstrap*.

Добијени резултати показали су да је *Angular* фрејмворк идеално решење када се ради о реализацији корисничког интерфејса. Прикупљање података у позадини и динамичка промена страница када су ти подаци спремни за приказ, олакшани су применом структурних, атрибутских и многих других директива у *Angular*-у. *Angular* рутирање практично је обеспотребило коришћење PHP фрејмворка, а самим и тим и убрзање извршавања серверског кода и перформанси саме апликације. Валидација поља форме и компактан приказ порука о грешкама, компонента календара, само су неке од предности које овај фрејмворк доноси.

Комуникација између клијента и сервера се заснива на *POST* или *GET*  *HTTP* захтевима које *Angular* реализује у позадини помоћу *AJAX* технологије*.* Ово је постигнуто помоћу техника реактивног програмирања, које *Angular* користи помоћу библиотеке *rxjs*.

С обзиром да *Angular* фрејмворк имплементира рутирање апликације, *PHP* је коришћен искључиво у сврхе дохватања података из базе и прослеђивања података путем *JSON*-a у сировом облику. За конекцију са базом коришћена је *PDO* класа из *PHP*-а, којом је значајно унапређена сигурност и заштита од малициозних напада на базу – *Sql Injection*.

Дакле, за генерисање комплетног *HTML* садржаја задужен је *Angular*, у којем се пристигли подаци са сервера процесирају и приказују у жељеној форми. На овај начин су у потпуности раздвојене две логичке целине:

* дизајн странице и приказ података - *Angular*
* дохватање и управљање подацима – *PHP*

Добијени резултати показали су да реализована апликација поседује све потребне карактеристике једне модерне веб апликације. Под тим се мисли на:

* реализацију потребних функционалности једног ветеринарског портала
* брзину дохватања тражених података и динамичку промену садржаја страница тек када су подаци спремни за приказ
* прилагођеност апликације уређајима свих величина екрана

Када је реч о евентуалној надоградњи апликације, могло би се разматрати о додавању опције комуникације власника љубимаца и ветеринара путем *chat*-а. Такође, све врсте услуга које се нуде корисницима, као и коришћење апликације, могле би се наплаћивати преко интернета путем саме апликације.

Литература

1. <https://angular.io/>
2. <https://material.angular.io/>
3. <https://mattlewis92.github.io/angular-calendar>
4. <https://www.php.net/manual/en/manual.php>
5. <https://getbootstrap.com/>
6. <https://www.quackit.com/bootstrap/bootstrap_4/tutorial/>
7. [http://reactivex.io/rxjs/manual/](http://reactivex.io/rxjs/manual/overview.html)
8. <https://stackoverflow.com/>
9. <https://www.w3schools.com/>
10. Adam Freeman, *Pro Angular 6*, 2018.
11. Antonio Lopez, *Learning PHP 7*, 2016

Списак скраћеница

|  |  |
| --- | --- |
| AJAX | *Asynchronous JavaScript and XML* |
| CSS | *Cascading Style Sheets* |
| JSON | *JavaScript Object Notation* |
| HTML | *Hypertext Markup Language* |
| HTTP | *HyperText Transfer Protocol* |
| PDO | *PHP Data Objects* |
| PHP | *PHP: Hypertext Preprocessor* |
| SQL | *Structured Query Language* |

Списак слика

[Слика 2.2.1.1 Размена података између клијента и сервера 6](#_Toc12619816)

[Слика 2.2.2.1 *Observable* као ток података током времена 7](#_Toc12619817)

[Слика 2.2.2.2 Интеракција и процесирање пристиглих података са тока података 7](#_Toc12619818)

[Слика 2.2.2.3. Структурна директива \**ngIf* 8](#_Toc12619819)

[Слика 2.2.2.4. Комуникација клијента и сервера путем *JSON*-а 9](#_Toc12619820)

[Слика 2.2.2.5. *Prepared Statement* за пријаву корисника 10](#_Toc12619821)

[Слика 3.1 Почетна страна са популарним текстовима о љубимцима 12](#_Toc12619822)

[Слика 3.1.1. Навигациони мени ветеринара 12](#_Toc12619823)

[Слика 3.1.2. Додавање текстова о кућним љубимцима 13](#_Toc12619824)

[Слика 3.1.2.1. Дневни приказ заказаних прегледа 13](#_Toc12619825)

[Слика 3.1.2.2. Искакајући прозор са детаљима заказаног прегледа 14](#_Toc12619826)

[Слика 3.1.2.3 Опције за додавање болести – *Add disease* и обављање прегледа – *Examine this pet* 14](#_Toc12619827)

[Слика 3.1.2.4. Искакајући прозор за додавање болести и алергија 15](#_Toc12619828)

[Слика 3.1.2.5. Страница за извршавање прегледа љубимца 15](#_Toc12619829)

[Слика 3.1.2.6. Форма за попуњавање извештаја са ветеринарског прегледа 16](#_Toc12619830)

[Слика 3.1.2.7. Приказ грешке при уносу невалидних података 17](#_Toc12619831)

[Слика 3.1.3.1. Преглед личних информација пријављеног корисника 17](#_Toc12619832)

[Слика 3.2.1. Навигациони мени за власника љубимаца 18](#_Toc12619833)

[Слика 3.2.1.1 Форма за додавање новог профила кућног љубимца 18](#_Toc12619834)

[Слика 3.2.1.2 Страница са детаљима о профилу љубимца 19](#_Toc12619835)

[Слика 3.2.1.3 Искакајући прозор за додавање детаљних информација за љубимца 19](#_Toc12619836)

[Слика 3.2.1.4 Здравствени картон љубимца 20](#_Toc12619837)

[Слика 3.2.3.1. Заказивање термина прегледа 20](#_Toc12619838)

[Слика 3.2.3.2. Отказивање прегледа 21](#_Toc12619839)

[Слика 4.1.1. Почетна страница апликације – *index.html* 22](#_Toc12619840)

[Слика 4.1.2. *app.component.html* 23](#_Toc12619841)

[Слика 4.1.3. Фајл за конфигурацију рута – *app-routing.module.ts* 23](#_Toc12619842)

[Слика 4.2.1. Сервис за аутентикацију корисника 24](#_Toc12619843)

[Слика 4.2.2. Контролер компоненте менија 25](#_Toc12619844)

[Слика 4.2.3. Темплејт компоненте менија 25](#_Toc12619845)

[Слика 4.3.1. Контролер компоненте за избора активног профила љубимца 26](#_Toc12619846)

[Слика 4.3.2. Слање *http* и *get* захтева помоћу класе *HttpClient* из модула *HttpClientModule* 27](#_Toc12619847)

[Слика 4.3.3. Темплејт компоненте за избор активног профила љубимца 27](#_Toc12619848)

[Слика 4.4.1. *PHP* фајл за дохватање профила и сетовање активног профила љубимца 28](#_Toc12619849)

[Слика 4.4.2. *JSON* објекат са информацијама о профилу љубимца 29](#_Toc12619850)

[Слика 4.4.3. *PHP* фајл за сетовање активног профила љубимца 29](#_Toc12619851)

[Слика 4.5.1. Иницијализација компоненте календара 30](#_Toc12619852)

[Слика 4.5.2. Форматирање заказаних термина за приказ у календару 30](#_Toc12619853)

[Слика 4.5.3. Обрада клика над догађајем календара 31](#_Toc12619854)

[Слика 4.6.1. Валидаторске функције за валидацију датума при преписивању терапије 32](#_Toc12619855)

[Слика 4.7.1. Редефинисање особина класа из *Bootstrap*-а 32](#_Toc12619856)