

UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAȘI

FACULTATEA DE INFORMATICĂ



LUCRARE DE LICENȚĂ

DigitalVet

**propusă de**

Denisa - Alexandra Iacob

Sesiunea: *iunie/iulie, 2023*

**Coordonator științific**

Olariu Florin

UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI  
FACULTATEA DE INFORMATICĂ

## **DigitalVet**

Denisa - Alexandra Iacob

Sesiunea: *ianie/iulie, 2023*

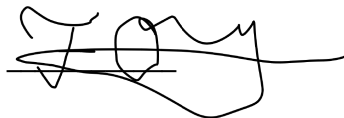
**Coordonator științific**

Olariu Florin

**Avizat,**  
**Îndrumător Lucrare de Licență**  
Titlul, Numele și prenumele: Olariu Florin

Data 20.06.2023

Semnătura



## **DECLARAȚIE PRIVIND AUTENTICITATEA CONȚINUTULUI LUCRĂRII DE LICENȚĂ**

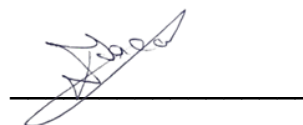
Subsemnata Iacob Denisa-Alexandra domiciliată în localitatea Izvoarele, comuna Răchiteni, județul Iași, născută la data de 25.02.2001, identificată prin CNP 6010225270829, absolventă a **Universității „Alexandru Ioan Cuza” din Iași, Facultatea de Informatică** specializarea Informatică, promoția 2020-2023, declar pe propria răspundere, cunoscând consecințele falsului în declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr. 1/2011 art.143 al. 4 și 5 referitoare la plagiat, că lucrarea de licență/diplomă/disertație/absolvire cu titlul: “DigitalVet” elaborată sub îndrumarea domnului Olariu Florin, este autentică, îmi aparține și îmi asum conținutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licență să fie verificată prin orice modalitate legală pentru confirmarea autenticității, consimțind inclusiv la introducerea conținutului său într-o bază de date în acest scop.

Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări științifice în vederea facilitării falsificării de către cumpărător a calității de autor al unei lucrări de licență, de diplomă sau de disertație și în acest sens declar pe proprie răspundere că lucrarea de față nu a fost copiată, ci reprezintă rodul cercetării pe care am întreprins-o.

**Iași, 20.06.2023**

**Absolvent Iacob Denisa-Alexandra**



## **ACORD PRIVIND PROPRIETATEA DREPTULUI DE AUTOR**

Facultatea de Informatică este de acord ca drepturile de autor asupra programelor-calculator, în format executabil și sursă, să aparțină autorului prezentei lucrări, ***Denisa-Alexandra Iacob***.

Încheierea acestui acord este necesară deoarece doresc ca în cadrul examenului de disertație să prezint o continuare a acestei lucrări

**Iași, 20.06.2023**

**Decan**

---

**Absolvent Iacob Denisa-Alexandra**



---

# Abstract

Această lucrare propune implementarea unei aplicații web pentru clinicile veterinare cu scopul de a facilita compararea acestora. Aplicația oferă utilizatorilor posibilitatea de a consulta recenziile lăsate de alți utilizatori, permițându-le astfel să ia decizii informate. Totodată, prin intermediul aplicației, utilizatorii pot realiza programări într-un mod simplu și eficient.

Pe parcursul lucrării, sunt prezentate detalii despre utilizarea unor framework-uri solide, precum React.js și Java Spring Boot, dar și modul în care acestea sunt integrate în realizarea unei aplicații scalabile și ușor de utilizat.

# Cuprins

|  |           |
|--|-----------|
| <b>Introducere.....</b>                    | <b>5</b>  |
| <b>1.Descrierea problemei.....</b>         | <b>7</b>  |
| <b>2. Soluții existente.....</b>           | <b>8</b>  |
| <b>3. Abordarea mea.....</b>               | <b>9</b>  |
| <b>4.Tehnologii utilizate.....</b>         | <b>10</b> |
| 4.1. Frontend.....                         | 10        |
| 4.1.1. React.js.....                       | 10        |
| 4.1.2. Material UI.....                    | 11        |
| 4.2. Backend - Java Spring Boot.....       | 11        |
| 4.3. Baze de date - MySQL.....             | 13        |
| 4.4. Stocarea în cloud - Firebase.....     | 14        |
| 4.5. Cererile HTTP.....                    | 15        |
| 4.6. Rutarea.....                          | 15        |
| <b>5. Schema aplicației.....</b>           | <b>18</b> |
| 5.1. Diagrama de flux.....                 | 18        |
| 5.2. Diagrama use case.....                | 19        |
| <b>6. Caracteristicile aplicației.....</b> | <b>20</b> |
| Pagina de start.....                       | 20        |
| Înregistrarea și autentificarea.....       | 22        |
| Afișarea clinicilor în urma filtrării..... | 23        |
| Serviciile.....                            | 24        |
| Programările.....                          | 25        |
| Pagina clinicii.....                       | 26        |
| Recenziile.....                            | 26        |
| Formularul de recenzii.....                | 27        |
| Pagina medicului veterinar.....            | 27        |
| Adăugarea clinicilor.....                  | 28        |
| Administrarea paginii clinicii.....        | 29        |
| Accesul neautorizat.....                   | 30        |
| URL greșit.....                            | 30        |
| <b>Concluzii și contribuții.....</b>       | <b>31</b> |
| Contribuții.....                           | 31        |
| Concluzii.....                             | 31        |
| Direcții viitoare.....                     | 32        |
| <b>Referințe.....</b>                      | <b>33</b> |

# Introducere

Lucrarea mea de licență s-a concentrat pe crearea unei pagini web dedicate clinicilor veterinare.

Ca multe alte persoane, am animale de companie, iar cea mai mare provocare a fost să găsesc o clinică veterinară și un medic veterinar de încredere pentru acestea. Prin urmare, am început să caut și să compar informații pe internet. Această căutare a fost o provocare, deoarece doream să fac o comparație rapidă și simplă a clinicilor. Aveam nevoie de o aplicație care să colecteze toate informațiile necesare și care să îmi ofere o perspectivă cuprinzătoare, astfel încât să pot lua cea mai bună decizie posibilă. În consecință, am creat o aplicație care să satisfacă aceste nevoi și să fie utilă atât pentru mine, cât și pentru alți iubitori de animale.

Prin intermediul paginii web construite, utilizatorii pot găsi informații despre diferite clinici veterinare, pot compara serviciile lor și pot vedea evaluări și recenzii ale altor proprietari de animale. Prin urmare, fiecare proprietar de animale de companie poate găsi cu ușurință clinica și medicul veterinar potrivit. Pentru a simplifica și mai mult procesul vizitelor la veterinar, am adăugat și opțiunea de programare online, iar programările vor putea fi vizualizate atât de către proprietari, cât și de către administratorii clinicilor. Astfel, întregul proces devine mai eficient pentru toți utilizatorii.

În acest moment nu există nicio aplicație în domeniul veterinar care să aibă toate caracteristicile menționate anterior. Majoritatea aplicațiilor software ce există în acest moment în domeniu sunt create pentru a gestiona o singură clinică.

“Digitail”[1] este una dintre cele mai complexe aplicații pentru proprietarii de animale. Aceasta are multe avantaje, cum ar fi posibilitatea de a crea un dosar medical digital individual pentru fiecare animal, în care pot stoca datele de contact, istoricul vaccinărilor și tratamentelor. De asemenea, oferă o gamă de articole informative și permite comunicarea cu medicul veterinar. Cu toate acestea singurul lucru în comun cu aplicația creată de mine este posibilitatea de a face programări prin intermediul aplicației.

Pagina web ce mi-a servit ca sursă de inspirație, atât în ceea ce privește designul, cât și conceptul este “Stailer”[2]. Aceasta este o aplicație dezvoltată pentru saloane și stilisti, cu scopul de a facilita compararea ușoară a serviciilor, vizualizarea recenziilor și a experienței altor clienți, dar și realizarea cu ușurință a programărilor. Datorită acestor caracteristici ce se apropiau mult de ideea de bază de la care am plecat în a crea aplicația, am ales să o folosesc ca sursă de inspirație.

Unul din principalele obiective ale aplicației au fost de a permite vizualizarea unor informații relevante indiferent dacă utilizatorul este autentificat sau nu. Așadar am permis utilizatorilor neautentificați să acceseze funcționalități precum căutarea de clinici sau veterinari, filtrarea acestora în funcție de locație, servicii și data disponibilității. Pentru acestea ei pot vizualiza informații precum descrierea, programul, recenziile, dar și serviciile puse la dispoziție de veterinarii respectivelor clinici.

Pentru utilizatorii care decid să se înregistreze și să se autentifice în cont, aplicația pune la dispoziție funcționalități precum adăugarea de recenzii, posibilitatea realizării unei programări, dar și salvarea clinicilor într-o listă de favorite.

Pentru administratorii de clinici, aplicația furnizează suport pentru crearea cu ușurință a paginilor clinicii și veterinarilor, actualizarea cu succes a acestora, dar și vizualizarea programărilor făcute de utilizatori.

Întrucât ideea lucrării mele a apărut de la o problemă pe care am întâmpinat-o și propune totodată o soluție eficientă în rezolvarea acesteia, lucrarea este structurată pe mai multe capitole ce prezintă o descriere detaliată a problemei, soluțiile deja existente pentru aceasta, dar și descrierea soluției mele, împreună cu tehnologiile utilizate în scopul realizării acesteia.

**Capitolul 1 - Descrierea problemei** prezintă o descriere detaliată a acesteia, cum am ajuns în contact cu aceasta, dar și ce m-a determinat să-mi doresc găsirea unei soluții și implementarea acesteia.

**Capitolul 2 - Soluții existente** oferă exemple de abordări existente pe piață pentru problema relatată în primul capitol, de ce a fost nevoie de o altă abordare și cum am ajuns la o conturarea ideii mele pornind de la acestea.

**Capitolul 3 - Abordarea mea** descrie ideea mea de rezolvare a problemei relatate în primul capitol, factorii ce au dus la aceasta și impactul pe care îl are implementarea acesteia.

**Capitolul 4 - Tehnologiile utilizate** prezintă framework-urile și bibliotecile folosite pentru implementarea soluției, dar și factorii ce au dus la alegerea folosirii acestora.

**Capitolul 5 - Schema aplicației** ilustrează scenariile de utilizare ale aplicației prin intermediul diagramelor de flux și use case.

**Capitolul 6 - Caracteristicile aplicației** este capitolul în care se explică în detaliu caracteristicile aplicației, sunt prezentate imagini cu cele mai importante componente ale acesteia și modul în care funcționează.

În final sunt prezentate concluziile, contribuțiile, dar și direcțiile viitoare ale aplicației.



## 1.Descrierea problemei

În ziua de astăzi, tehnologia a ajuns să ne simplifice viața în multe feluri, iar totul a ajuns la un click distanță. Acest lucru a făcut multe persoane să-și dorească să fie informate înainte de a lua o decizie, iar internetul reprezintă principala formă de informare. De la marile instituții și până la micile afaceri, multe au început să înțeleagă importanța unei pagini web [3]. Astfel, informarea nu mai reprezintă o problemă. Adevărata provocare a devenit luarea deciziilor. Întrucât toată lumea își dorește să ia cu ușurință cele mai bune decizii, avem nevoie de pagini web care să faciliteze acest proces.

În contextul problemei întâmpinate de mine, dificultățile au apărut în compararea informațiilor despre clinici veterinare. Fiind proprietară de animale de companie și dorindu-mi să găsesc un medic veterinar de încredere pentru acestea, am fost nevoită să compar serviciile oferite de fiecare clinică din oraș și să citesc recenzii de la alți proprietari de animale. Însă, această căutare și evaluare a clinicilor a fost și este consumatoare de timp. Această experiență m-a făcut să-mi dau seama că aveam nevoie de o platformă care să îmi aducă toate informațiile într-un singur loc, aveam nevoie de o soluție care să faciliteze comparația rapidă și să automatizeze procesul de luare a deciziilor, iar un prim pas este aducerea tuturor informațiilor din domeniu într-un singur loc.

Având în vedere că numărul proprietarilor de animale de companie este mare [4], prin dezvoltarea unei astfel de pagini web, oferim tuturor utilizatorilor cu o problemă asemănătoare o experiență mult mai simplă și eficientă. Prin accesarea tuturor informațiilor necesare într-un singur loc, aceștia, vor putea compara opțiunile rapid și totodată vor putea lua decizii informate cu mai multă ușurință.

În concluzie, într-o lume în care timpul și informațiile sunt prețioase, iar numărul de proprietari de animale este mare, crearea unei pagini web care să facă reuniunea și să summarizeze informațiile din mai multe pagini din acest domeniu devine indispensabilă pentru a îmbunătăți procesul de luare a deciziilor, dar și pentru a satisface nevoile utilizatorilor.

## 2. Soluții existente

În acest moment, au început să apară tot mai multe pagini web care adună într-un singur loc informații ce în mod normal ar fi necesitat parcurgerea mai multor pagini sau un efort mai mare. Aceste abordări oferă soluții convenabile și eficiente pentru foarte mulți utilizatori.

Analizând soluțiile existente în domeniul veterinar, am ajuns la concluzia că există platforme online pentru gestionarea clinicilor veterinare, însă majoritatea se axează pe partea de administrare și programare internă. Aceste soluții nu sunt orientate spre nevoile menționate și nu oferă o experiență eficientă pentru compararea clinicilor și a medicilor veterinari și în consecință luarea unei decizii.

Ca soluții reprezentative pentru lucrarea mea de licență, am ales să vorbesc despre aplicațiile “Digitail” și “Stailer”. Acestea deși abordează domenii total diferite, ambele au condus la conturarea soluției găsite de mine.

Digitail este o aplicație care integrează mai multe elemente importante din domeniul veterinar, cum ar fi articolele ce ajută la informare, comunicarea rapidă și eficientă cu medicul veterinar, dar și un spațiu în care poți stoca toate informațiile medicale ale animalului tău. Cu toate acestea, din păcate nu oferă decât o parte din rezolvarea problemei menționate. Mai exact, oferă o gamă largă de informații și aplicații din domeniu într-un singur loc, însă nu oferă suport pentru compararea eficientă între clinici și nici împărtășirea experienței prin recenzii.

Stailer este o aplicație care a adus la un loc saloanele de înfrumusețare din mai multe orașe, oferind o modalitate mai ușoară de a face programări pentru serviciile oferite de acestea, dar, în același timp, posibilitatea de a-ți împărtăși experiența prin intermediul recenziilor și făcând alegerea mai ușoară, aceasta fiind mai relevantă pentru problema relatată și abordată de mine.

Ca și concluzie a acestui capitol, aplicația reprezentativă pentru domeniul veterinar, Digitail, nu are implementată o soluție la problema propusă de mine, ea axându-se mai mult pe alte nevoi, dar, există o aplicație care implementează o astfel de soluție într-un domeniu total opus.

### 3. Abordarea mea

Ideea abordării mele a venit de la nevoia de a găsi o clinică veterinară și un medic veterinar de încredere pentru animalele mele de companie. Deși aplicația Digitail oferă ajutor în multe aspecte din domeniul veterinar, nu a putut oferi o soluție la problema mea specifică. Așadar, am fost nevoită să caut informații pe diferite pagini despre fiecare clinică veterinară din oraș. În căutarea mea am observat că unele pagini aveau un design atrăgător, dar nu neapărat și cele mai bun personal medical, în timp ce altele, deși arătau mai modest, aveau servicii oferite de medici cu mai multă experiență. Această diversitate în prezentarea clinicilor m-a determinat și mai mult să mă gândesc la cât de utilă ar fi existența unei platforme care să centralizeze și să ofere informații complete despre toate clinicile veterinare, fără a fi influențați de alți factori.

Experiența mea cu aplicația Stailer, pe care o utilizez de mult timp și care m-a ajutat să iau decizii informate în privința serviciilor de la salon, m-a făcut să-mi dau seama cât de ușor ar fi fost să pot accesa o pagină web care să ofere același suport pentru clinicile veterinare. Astfel, am fost inspirată să dezvolt o pagină web care să ofere suport similar în luarea deciziilor pentru clinicile veterinare. Scopul aplicației fiind de facilitare a procesului de selectare a unei clinici și a unui medic veterinar potrivit, oferind informații complete și recenzii de la alți proprietari de animale de companie.

Pagina web propusă de mine oferă un șablon comun pentru toate clinicile veterinare, astfel încât utilizatorii să nu mai fie distrași de aparență, ci să se concentreze pe esență. În cadrul paginii, clinicile își pot gestiona singure și cu ușurință pagina din meniul de setări din cont, și totodată pot vizualiza programările făcute de utilizatorii autentificați în aplicație. Utilizatorii pot vizualiza cu ușurință informațiile filtrate ale clinicilor, pot vedea recenzii care, de asemenea, pot fi filtrate după numărul de steluțe primite de la alți utilizatori și pot vizualiza, dacă doresc, doar recenziile unui anumit veterinar. Toate acestea funcționalități pot utilizate fără a fi nevoie de un cont, însă pentru a utiliza și celelalte funcționalități implementate, va fi nevoie de înregistrare și ulterior de autentificarea în cont. Beneficiile disponibile utilizatorilor cu cont includ adăugarea de clinici în lista de favorite, realizarea de programări, vizualizarea acestora, dar și adăugarea de recenzii. Toate acestea servesc la rezolvarea problemei menționate în primul capitol.

În concluzie, soluția propusă abordează în mod eficient problema menționată în cadrul primului capitol, aducând toate informațiile și funcționalitățile relevante într-un singur loc. Prin intermediul acestei abordări, obiectivul meu este să facilitez luarea deciziilor informate și să aduc confort iubitorilor de animale în alegerea serviciilor veterinare potrivite.

## 4. Tehnologii utilizate

### 4.1. Frontend

Pentru realizarea interfeței paginii web, am utilizat framework-ul React.js [5] și biblioteca Material UI [6].

#### 4.1.1. React.js

React.js este un framework JavaScript open-source, ce oferă utilizatorilor posibilitatea de a crea interfețe interactive bazate pe conceptul de componente. Aceste componente sunt blocuri de cod independente și reutilizabile, ce gestionează logica și aspectul unei anumite părți din interfață. Acest framework utilizează un model de programare bazat pe starea, dar și o sintaxă simplă și declarativă utilizată pentru a descrie actualizările interfeței în funcție de schimbările de stare ale aplicației.

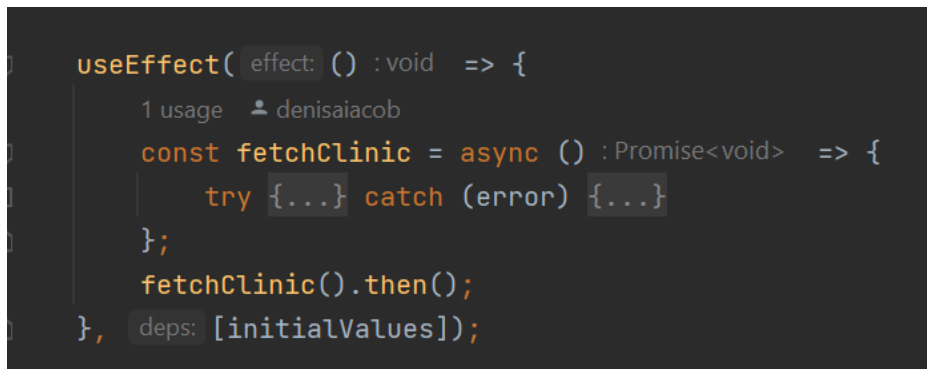
Un prim motiv pentru care am ales acest framework este posibilitatea de reutilizare a componentelor. Acest lucru mi-a permis să folosesc ori de câte ori am avut nevoie componente precum căsuțele pentru servicii, cardurile pentru clinici, dar și cardurile pentru recenzii. Aceste componente apar de foarte multe ori în pagină, în multe componente de rutare, iar React mi-a permis ca prin intermediul proprietăților cu care am instanțiat aceste componente, să le personalizez în funcție de cum am dorit să le afișez. Tot prin reutilizarea componentelor am creat și formularul pentru adăugarea clinicilor, dar și meniul de setări pentru clinici. La realizarea acestor două părți importante din pagină, am utilizat aceleași componente apelate cu proprietatea “update” pentru a stabili contextul în care este utilizată. Această proprietate React.js , pe lângă faptul că facilitează dezvoltarea, simplifică și procesul de întreținere și extindere a aplicației.

```
switch (activeStep) {  
  case 0:  
    stepContent = <CreateClinicPage clinic={clinic} setClinic={setClinic} update={false}/>;  
    break;  
  case 1:  
    stepContent = <AddClinicInfo info={info} setInfo={setInfo}/>;  
    break;  
  case 2:  
    stepContent = <AddProgram program={program} setProgram={setProgram}/>;  
    break;  
}
```

*Figura 1: Reutilizarea componentelor în meniul de setări*

Un alt motiv pentru care am ales să lucrez cu această tehnologie este capacitatea de actualizare eficientă doar a porțiunilor din interfață ce au suferit modificări. Pentru aceasta operațiune am utilizat hook-ul “useEffect”. Acesta poate controla momentul în care este apelată funcția de efect, dar și resursele utilizate și totodată poate gestiona dependențele pentru a evita efectele nedorite sau buclele infinite. Majoritatea componentelor din interfață folosesc acest hook împreună cu apelurile API pentru a actualiza constant informațiile din

interfață. Acest lucru duce la o performanță mai bună a aplicației, dar și la o experiență de utilizare mai fluidă.



```
useEffect( effect: () :void => {  
  1 usage  denisaiacob  
  const fetchClinic = async () : Promise<void> => {  
    try {...} catch (error) {...}  
  };  
  fetchClinic().then();  
}, deps: [initialValues]);
```

*Figura 2: “useEffect” la filtrarea clinicilor*

În imaginea anterioară am prezentat utilizarea funcției “useEffect”, împreună cu un apel asincron ce actualizează lista de clinici afișate în funcție de valoarea dependenței. Acest apel se execută ori de câte ori “initialValues” se modifică.

Nu în ultimul rând, am ales React.js datorită suportului extins, a documentației bogate, dar și a flexibilității și scalabilității ce ajută la menținerea codului pe termen lung.

#### 4.1.2. Material UI

Material UI este o bibliotecă de componente React ce implementează Material Design și este de asemenea open-source. Această bibliotecă oferă o colecție foarte mare de componente prefabricate realizate la cele mai înalte standarde de formă și funcție, dar care totodată pot fi personalizate și adaptate în funcție de nevoi, ce simplifică realizarea interfețelor de utilizator.

Material UI mi-a oferit posibilitatea de a face pagina să arate bine din punct de vedere estetic fiind formată din componente complexe pe care le-am personalizat în funcție de nevoi, fără a încărca codul. Totodată aceasta mi-a oferit suport pentru a realiza o pagină web responsive. Din această bibliotecă am importat foarte multe componente, dar cele mai utilizate ar fi butoanele, stilurile text și containerele .

În concluzie, prin utilizarea acestor două tehnologii, am reușit construirea unei interfețe scalabile, fluide, dar și estetice pentru pagina mea web.

## 4.2. Backend - Java Spring Boot

Pentru partea de backend, am ales să utilizez Java Spring Boot [7]. Acesta este un framework Java, open-source, care ajută la dezvoltarea rapidă a aplicațiilor bazate pe principii convenționale de programare și totodată oferă o gamă largă de componente și funcționalități. Gestionarea cererilor HTTP, gestionarea dependențelor, gestionarea persistenței datelor prin integrarea cu baze de date, securitatea aplicației, testarea unitară și multe altele se numără printre aceste funcționalități. Un mare avantaj al utilizării Java Spring

Boot este că oferă o modalitate simplă și eficientă de a construi API-uri RESTful. Adnotările specifice Spring ne permit să definim rapid rutele API-ului, acțiunile legate de el și modul în care datele sunt transmise între client și server.

Am ales să utilizez această tehnologie datorită tuturor avantajelor și funcționalităților menționate mai sus. Mai exact, prin intermediul acestei tehnologii, am putut defini cu ușurință API-urile necesare și am putut implementa funcționalitățile acestora prin marcarea adecvată a claselor corespunzătoare cu adnotări, dar și efectua apelurilor cu ajutorul Spring Data JPA către baza de date MySQL folosită.

Pentru implementarea serverului de backend am creat câte o entitate pentru fiecare tip de informație ce necesita stocarea în baza de date. Clasele de tip Entity sunt adnotate cu aceasta, dar și cu “@Table”. Aceste clase sunt utilizate pentru a defini attributele, constrângerile, dar și relațiile dintre entități.

Pentru fiecare entitate am creat câte o clasă controller ce gestionează cererile HTTP primite din interfață și interacționează cu clasele de tip services asociate, pentru a obține și manipula datele primite. Controlerele sunt responsabile și de rutele endpoint-urilor aplicației. Adnotarea “@RequestMapping” definește prefixul comun pentru toate rutele definite în cadrul controlerului, în cazul serverului meu, folosesc pentru toate controlerele prefixul “/api/v1/digitalVet”. În interiorul controlerului am definit metodele ce preiau datele și le trimite către metodele corespunzătoare din clasele services. Aceste metode sunt adnotate în funcție de tipul cererii HTTP. Cu alte cuvinte, controlerele servesc ca intermediare între cererile primite din interfață și serviciile ce manipulează datele din baza de date.

Clasele services reprezintă logica aplicației. Acestea se ocupă de de procesarea și administrarea datelor prin intermediul repository-ului. În cazul în care datele nu sunt valide ele returnează excepții, altfel efectuează operațiile și returnează răspunsul corespunzător.

După cum am menționat, clasele services nu manipulează efectiv datele din baza de date, ci utilizează repository-urile. Repository reprezintă o interfață care oferă metode pentru accesarea și manipularea datelor din baza de date. Aceste metode pot include operații de citire, scriere, actualizare și ștergere a datelor. Cele mai utilizate metode au fost “save” pentru scriere și “findBy” pentru căutare după coloană. În implementarea acestor interfețe am extins interfața “JpaRepository” pentru a moșteni operațiile CRUD și de găsim după criterii date, astfel încât să nu mai fi nevoie de implementarea lor manuală. Unele dintre ele implementează și interfața “JpaSpecificationExecutor” pentru a oferi suport pentru specificații JPA în operațiile de interogare a bazei de date. Cu alte cuvinte aceasta oferă capacitatea de a defini interogări complexe și personalizate pe entități. Pentru a defini interogări personalizate în Java Persistence Query Language am folosit adnotarea “@Query” și interogarea necesară.

```
1 usage  denisaiacob
@Query("SELECT review FROM ReviewEntity review WHERE review.vetId = ?1 ORDER BY review.day DESC")
Optional<List<ReviewEntity>> findByIdVetId(Long vetId);
```

Figura 3: Interogare personalizată pentru căutarea recenziilor după id-ul veterinarului

Imaginea de mai sus prezintă o interogare personalizată pentru a selecta recenziile unui anumit veterinar, ordonate descrescător după data adăugării.

Implementarea acestor operații este realizată automat de către framework-ul Spring Data JPA.

Pe lângă implementarea claselor explicate anterior, Java Spring Boot mi-a permis și asigurarea securității aplicației prin metodele de criptare ale parolei, dar și setarea nivelului de vizibilitate al acesteia. Am făcut acest lucru prin crearea unei clase de configurare ale aspectelor legate de securitatea aplicației. În această clasă am definit un bean ce furnizează un obiect de tipul PasswordEncoder. Acesta criptează parole, dar le și compară cu alte parole criptate din baza de date. Pentru criptarea parolilor cu algoritmul BCrypt, am utilizat implementarea BCryptPasswordEncoder.

Prin definirea ca bean a acestei metode, am putut să o injectez în alte componente ale aplicației și utiliza la înregistrarea, autentificarea și administrarea parolilor utilizatorilor. Cu alte cuvinte, prin definirea și utilizarea bean-ului “passwordEncoder”, m-am asigurat că parolele utilizatorilor sunt criptate într-un mod sigur și că pot fi verificate în mod corespunzător în cadrul procesului de autentificare.

Tot pentru securitatea parolilor am utilizat și adnotarea “@JsonProperty” pentru a marca accesul ca “WRITE\_ONLY” pentru toate parolele. Aceasta este utilizată pentru a controla accesul la parolă atunci când se realizează serializarea acesteia în format JSON.

Prin specificarea valorii “WRITE\_ONLY”, am indicat faptul că proprietatea “password” din “UserEntity” poate fi utilizată pentru a seta valoarea acesteia, dar nu va fi inclusă în răspunsul JSON returnat de API, iar astfel se evită expunerea neintenționată a parolilor.

În concluzie, această abordare m-a ajutat la dezvoltarea unui backend robust, care să poată gestiona cererile și să ofere răspunsuri eficiente cererilor din interfață într-un mod sigur.

### 4.3. Baze de date - MySQL

Pentru a sprijini stocarea și administrare eficientă a datelor am ales utilizarea unei baze de date MySQL [8]. Este un sistem pentru bazele de date relaționale recunoscut pentru performanța sa. El oferă o varietate de funcționalități, cum ar fi: suport pentru tranzacții, interogări complexe, indexare avansată și securitate a datelor.

Pentru a stoca informații despre clinici, veterinari, servicii, recenzii și celelalte informații am creat și gestionat tabele structurate folosind MySQL. Pentru a asigura o comunicare eficientă între aplicație și baza de date, dar și pentru ale accesa și modifica, am folosit interogări SQL. Totodată, MySQL mi-a oferit capacitatea de implementare a securității datelor, modificare accesului acestora, dar mi-a permis și definirea relației dintre tabele și constrângerilor acestora. Relațiile dintre tabele sunt definite prin intermediul câmpurilor id ce sunt marcate ca și chei primare ce sunt autoindexate. Pe lângă constrângerile oferite de cheia primară am folosit și constrângeri pentru “not null”. Aceste constrângeri asigură integritatea datelor și permit realizarea interogărilor într-un mod coerent și constant. Atât relația dintre tabele cât și constrângerile sunt marcate în clasele “entity” cu adnotările specifice. Cele mai utilizate adnotări pentru constrângeri folosite în implementare sunt “@Id” pentru cheia primară și “@NotBlank” pentru câmpurile care nu pot fi nule. Împreună

cu adnotarea pentru cheia primară am utilizat și adnotarea “@GeneratedValue” pentru autoindexare. Pentru relația dintre tabele am utilizat adnotările “@ManyToOne”, “@OneToMany” [9], dar și “@JoinColumn” în entități precum “FavoriteClinicsEntity”.

Un alt motiv pentru care am ales MySQL o reprezintă evenimentele. Pentru a păstra în baza de date doar programările ce nu au avut încă loc, am creat un eveniment care șterge toate programările mai vechi de data curentă. Acest eveniment are recurență zilnică și a fost creat utilizând interogarea:

```
CREATE EVENT IF NOT EXISTS `delete_old_appointments` ON SCHEDULE EVERY
1 DAY STARTS '2023-06-10 14:52:20' ON COMPLETION PRESERVE DO DELETE
FROM appointments WHERE day < CAST(GETDATE() AS Date);
```

Figura 4: Interogare folosită la crearea evenimentului de ștergere a programărilor vechi

#### 4.4. Stocarea în cloud - Firebase

Pentru a stoca fotografiile încărcate pentru profilul clinicilor veterinare, dar și cel al veterinarilor, am utilizat Firebase Storage [10]. Aceasta oferă o gamă largă de servicii și instrumente care permit stocarea în cloud și controlul fișierelor media, cum sunt fotografiile. Firebase Storage permite încărcarea și descărcarea fișierelor, modificarea permisiunilor de acces și furnizează URL-uri unice pentru a accesa și afișa aceste fișiere în aplicație.

În pagina mea, încărcarea fotografiilor este realizată folosind o funcție care se execută atunci când utilizatorul încarcă în pagină o imagine folosind un element input de tip fișier. Aceasta preia fișierul încărcat și se crează un obiect “FileReader” pentru a citi conținutul fișierului.

```
const imageRef : StorageReference = ref(storage, url: `vetImages/${imageUpload.name + v4()}`);
uploadBytes(imageRef, imageUpload).then((snapshot : UploadResult) : void => {
  getDownloadURL(snapshot.ref).then((url : string) : void => {
    setVet(prevVet => {...});
  });
});
```

Figura 5: Încărcarea imaginilor în Firebase Storage

În imaginea de mai sus avem partea de cod ce realizează încărcarea în Firebase, după ce este finalizată citirea. Cu ajutorul funcției ‘ref’ ce obține referința către directorul Firebase Storage, iar cu ‘uploadBytes’ ce face încărcarea efectivă în baza de date. Acestea sunt funcții din biblioteca Firebase Storage SDK. Funcția “v4” face parte din pachetul “uuid” [11] și are rolul de a genera un sufix unic ce este adăugat la finalul numelui imaginii pentru a evita conflictele de nume.

După ce încărcarea este finalizată, se obține URL-ul imaginii utilizând funcția ‘getDownloadURL’, iar acesta este ulterior stocat în baza de date a veterinarilor, respectiv clinicilor din MySQL.



## 4.5. Cererile HTTP

Pentru efectuarea cererilor HTTP , am utilizat biblioteca Axios [12]. Aceasta este o bibliotecă JavaScript ce ajută la comunicarea cu API-urile, dar și preluarea și trimiterea de date prin sintaxa simplă și intuitivă pentru efectuarea cererilor HTTP.

Pe lângă metodele “get, post, put, delete”, folosite pentru a trimite cereri la serverul de backend și returnarea răspunsurilor primite, am utilizat și suportul promisiunilor pentru a gestiona răspunsurile asincrone primite. Acestea au ajutat la tratarea cu ușurință a cazurilor de eroare.

```
const fetchData = async () : Promise<void> => {  
  try {  
    const response = await ClinicService.getProgramByClinicId(clinicId);  
    setProgram(response.data);  
  } catch (error) {  
    console.log(error);  
  }  
};
```

*Figura 6: Apel asincron pentru obținerea programului*

În imaginea de mai sus avem apelul asincron ce face cererea către serverul de backend, folosind un apel către serviciul “ClinicService” ce are implementată logica de obținere a programului clinicii veterinare în funcție de id-ul acesteia. Apelul axios din interiorul “ClinicService” arată astfel:

```
getProgramByClinicId(clinicId) {  
  return axios.get( url: CLINICS_PROGRAM_API_URL + "/" + clinicId);  
}
```

*Figura 7: Apel axios folosind metoda get*

Cererea HTTP este realizată apelând “axios.get” către API-ul corespunzător ce a fost construit folosind adresa de bază și adăugând la final id-ul clinicii.

## 4.6. Rutarea

Rutarea în React este reprezentată de procesul de navigare între pagini sau secțiuni ale unei aplicații web în cadrul unei singure pagini principale. Acest lucru se poate face folosind biblioteca React Router [13]. Această bibliotecă oferă componente precum “Router” și “Route”, dar și metode care facilitează crearea și administrarea rutelor aplicației. Router este componenta principală care înconjoară aplicația și gestionează rutarea. Aceasta poate fi de tipul BrowserRouter ce utilizează istoricul browserului pentru a gestiona rutarea, sau HashRouter ce folosește URL-urile hash pentru a realiza rutarea.

Pentru rutarea din cadrul aplicației create de mine, am utilizat tipul `BrowserRouter` pe care l-am definit în interiorul clasei `“index.js”`, după cum se poate vedea și în următoarea imagine.

```
root.render(  
  <React.StrictMode>  
    <BrowserRouter>  
      <AuthProvider>  
        <Routes>  
          <Route path="/" element={<App/>} />  
        </Routes>  
      </AuthProvider>  
    </BrowserRouter>  
  </React.StrictMode>  
);
```

*Figura 8: Rutarea aplicației*

Route este componenta ce definește o rută specifică prin intermediul proprietății `“path”`, împreună cu componenta ce trebuie afișată atunci când ruta se potrivește cu URL-ul curent și pe care am dat-o prin intermediul proprietății `“element”`. Toate rutele au fost definite în interiorul componentei principale `“Routes”`. Aceasta este utilizată pentru a grupa și organiza rutele într-un singur loc.

Pentru rutarea în funcție de roluri [14], am utilizat componenta `“Route”`, care, de data aceasta a primit doar proprietatea `element` și componenta ce verifică rolurile autorizate pentru vizualizarea paginii.

```
const RequireAuth = ({allowedRoles}) => {  
  const {auth} = useAuth();  
  const location : Location = useLocation();  
  
  return (  
    auth?.roles?.find(role => allowedRoles?.includes(role))  
    ? <Outlet/>  
    : auth?.user  
      ? <Navigate to="/unauthorized" state={{from: location}} replace/>  
      : <Navigate to="/login" state={{from: location}} replace/>  
  );  
};
```

*Figura 9: Verificarea accesului*

În imaginea de mai sus avem partea de cod ce verifică dacă utilizatorul autentificat are atribuit unul din rolurile permise pentru această rută. Rolurile permise sunt primite prin intermediul proprietății `“allowedRoles”`.

După verificarea accesului, această componentă decide dacă userul va naviga la pagina pentru care s-a inițiat cererea sau către pagina de `“Login”`, dacă nu este autentificat, respectiv cea de acces neautorizat dacă nu are acces la pagină.

Navigarea între rutele aplicației este realizată cu ajutorul componentei “Navigate” și a hook-ului “useNavigate” tot din biblioteca React Router.

“Navigate” este o componentă utilizată pentru a naviga la o anumită rută și poate fi folosită în interiorul unui alte componente pentru a declanșa navigarea de la o rută la alta. Prin intermediul proprietăților, am specificat ruta către care am dorit navigarea, dar am utilizat și alte opțiuni cum ar fi redirecționarea sau înlocuirea istoricului de navigare în cazul în care utilizatorului nu i-a fost aprobat accesul. “useNavigate” poate fi apelat direct pentru a naviga la o anumită rută.

Pentru cazul în care URL-ul nu se potrivește cu nici una din rutele declarate, am definit o rută de capturare a acestora. Aceasta are alocată o componentă ce afișează codul de eroare 404 și textul “Pagina nu a fost găsită”.

## 5. Schema aplicației

### 5.1. Diagrama de flux

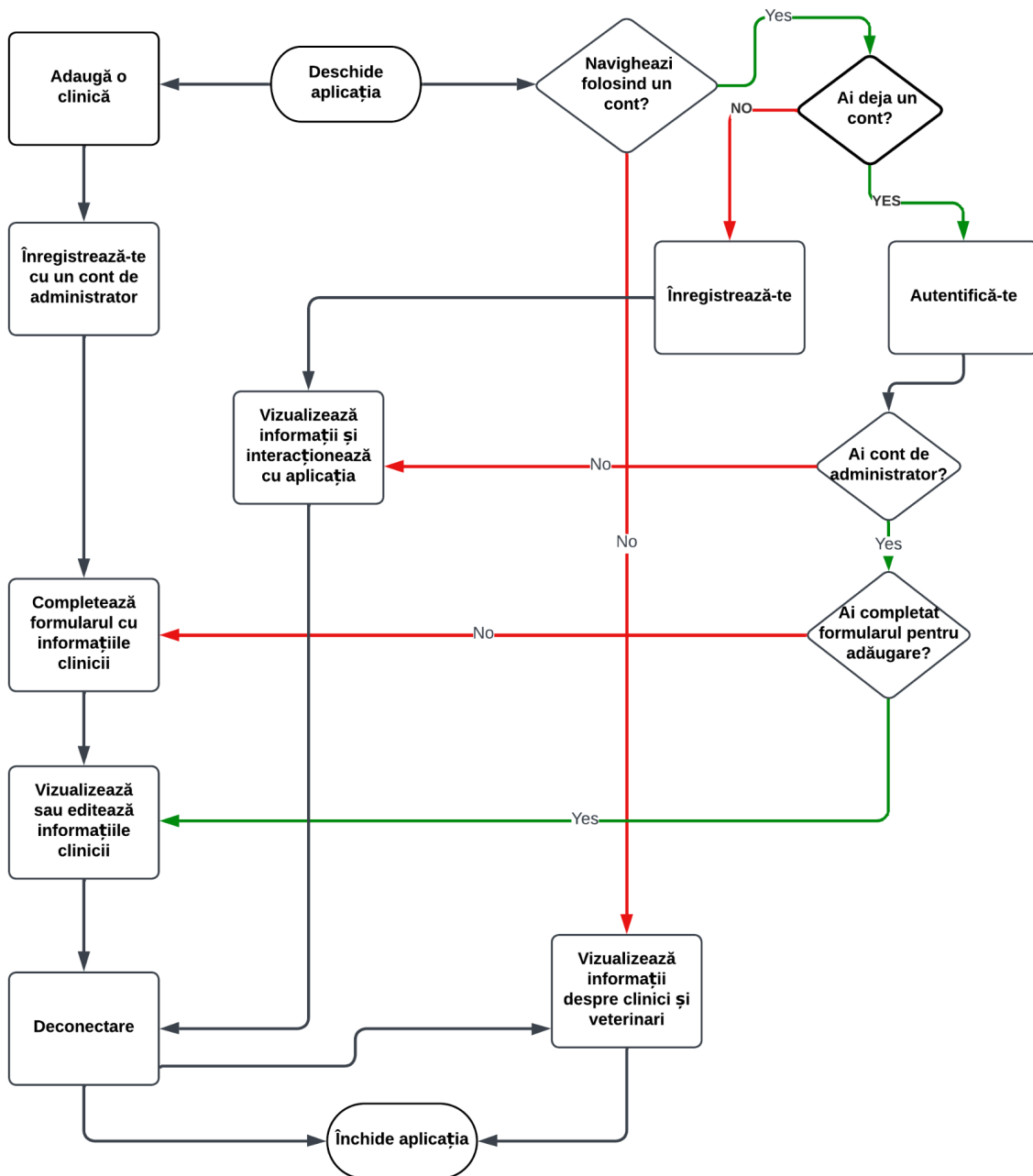


Figura 10: Diagrama de flux a aplicației

În diagrama de mai sus am ilustrat scenariile posibile ale utilizării aplicației. Aceasta prezintă fluxul de acțiuni pe care îl poate avea un utilizator de la deschiderea aplicației și până la închiderea acesteia, prin pașii și deciziile luate.

## 5.2. Diagrama use case

În cadrul următoarei diagrame putem observa o reprezentare grafică a interacțiunii dintre utilizatori și aplicația web implementată.

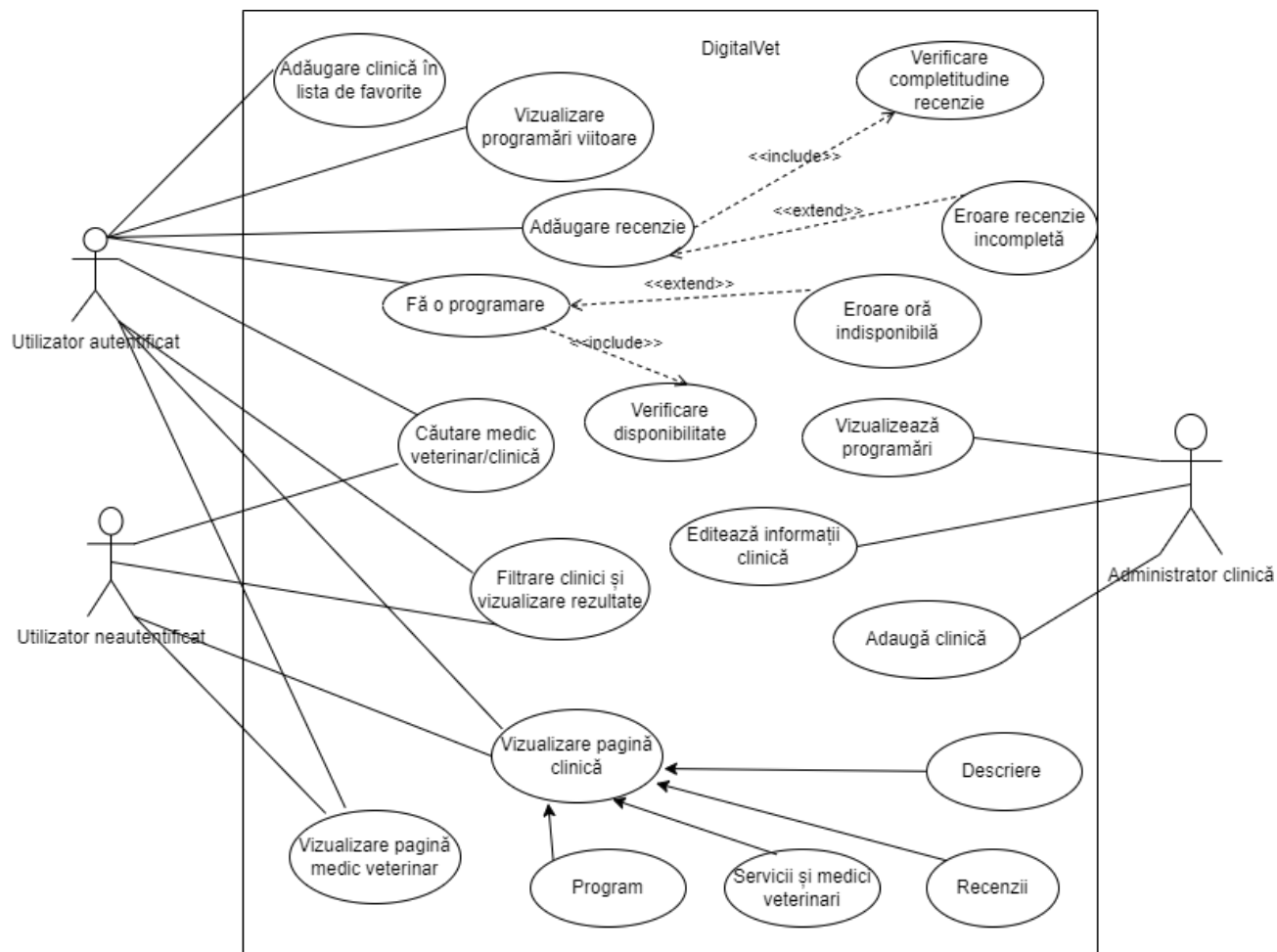


Figura 11: Diagrama use case a aplicației

Această diagramă prezintă scenarii specifice în care utilizatorii autentificați, utilizatorii neautentificați, dar și administratorul unei clinici pot interacționa cu aplicația. Aceste scenarii descriu acțiunile și funcționalitățile pe care acești utilizatori le pot realiza în cadrul aplicației pentru a vedea informații despre clinici veterinare, a face programări, a vizualiza recenzii și chiar adăuga, precum și pentru a modifica informațiile paginii clinicii.

## 6. Caracteristicile aplicației

Aplicația este una de tipul "guest", adică permite utilizatorilor să navigheze prin anumite funcționalități fără a fi nevoie de autentificare.

### Pagina de start

Una dintre paginile cu acces nerestricționat este chiar pagina de start. Aceasta este o fereastră simplă, ce include bara de activități și o casetă de filtrare sau căutare a clinicilor și veterinarilor.

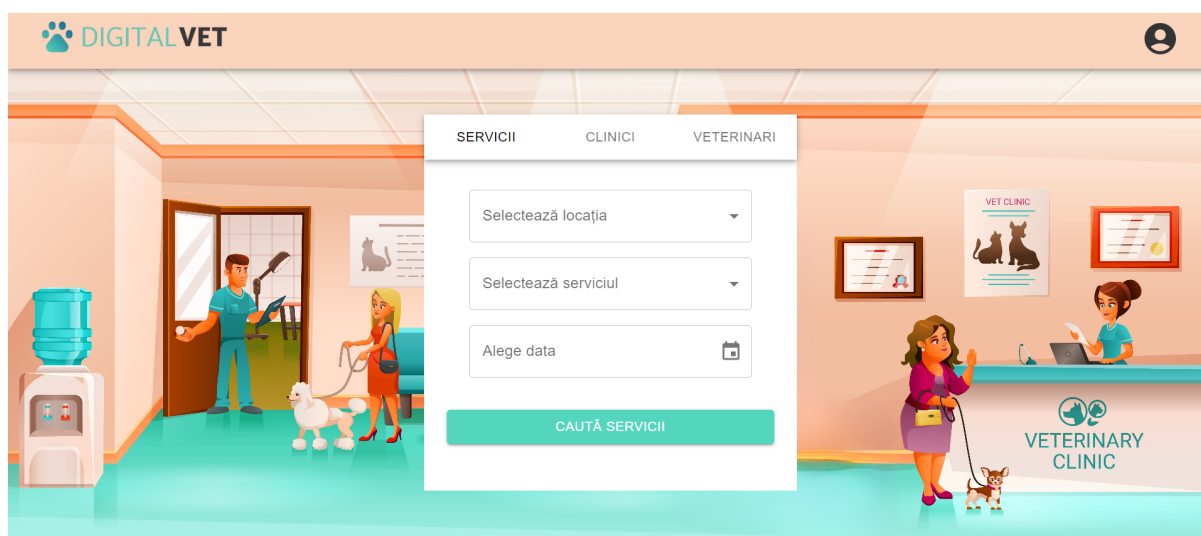


Figura 12: Pagina de start

Bara de activități va fi vizibilă și pe toate celelalte subpagini și are în componență mai multe butoane. Unul dintre acestea este logo-ul aplicației ce servește și ca buton de "home". La apăsarea acestuia utilizatorul va fi redirecționat către pagina de start.

Tot în bară putem observa și o iconiță ce simbolizează contul. La apăsarea acesteia, în primă fază, vom avea două butoane, unul pentru redirecționarea spre pagina de înregistrare și unul pentru pagina de autentificare în cont. După autentificare sau înregistrare aceasta va fi formată dintr-un meniu diferit în funcție de tipul de utilizator.

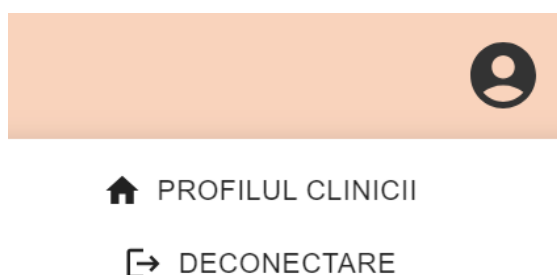
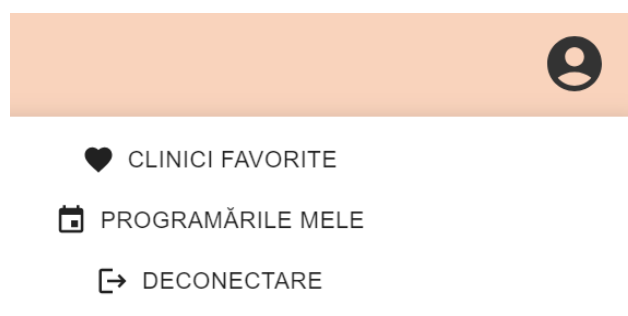


Figura 13: Meniul unui administrator de clinică

Pentru utilizatorii ce au un cont de administrare al clinicii, meniul va fi format dintr-un buton ce îi va redirecționa spre profilul clinicii, de unde se vor putea face și setările acestuia, dar și un buton pentru deconectarea de la cont. Acestea pot fi observate în figura 13.



*Figura 14: Meniul unui utilizator autentificat*

Pentru utilizatorii ce au un cont obișnuit, acesta va fi format din trei butoane. Primul buton permite vizualizarea listei de clinici favorite, al doilea permite vizualizarea programărilor făcute, iar al treilea este pentru deconectare de la cont. Acestea pot fi observate în imaginea de mai sus.

În bara de activități mai avem și un buton pentru adăugarea de clinici. Acesta este vizibil doar utilizatorilor ce nu sunt autentificați. Butonul pentru adăugarea clinicii îi va redirecționa pe utilizatori spre fereastra de înregistrare pentru un cont de administrator.

Cea mai importantă componentă din pagina de start este caseta de filtrare și căutare. Aceasta este formată din trei butoane pentru vizualizare alternativă. Primul buton, ce este surprins și în figura 12, afișează în casetă trei componente de filtrare. Componentele de filtrare sunt reprezentate de un buton de tip select pentru locația clinicii, unul pentru serviciul pe care îl căutăm, dar și unul pentru a selecta data la care serviciul este disponibil. Putem selecta doar unul sau mai multe dintre filtrele pentru servicii, după care prin apăsarea butonului de căutare vom fi redirecționați spre pagina cu rezultate. Pentru ca acestea să fie afișate, trebuie să avem selectat cel puțin un filtru, în caz contrar se va afișa un mesaj de eroare.

Cel de-al doilea buton din caseta de căutare este foarte asemănător cu cel de-al treilea. Ambele conțin o bară de căutare [15]. Diferența dintre acestea este dată de ce putem căuta în fiecare. În una putem căuta clinici veterinare (figura 15), iar în cealaltă un anumit medic veterinar (figura 16). Această funcționalitate a fost adăugată pentru persoanele care cunosc deja un medic sau clinica la care doresc doar să facă o programare sau să adauge o recenzie.

Figura 15: Căutarea clinicilor

Figura 16: Căutarea veterinarilor

## Înregistrarea și autentificarea

Casetele de autentificare [16] și înregistrare vor fi afișate în locul celei de filtrare și căutare. Acestea sunt formate din câmpuri de introducere text validate [17].

Înregistrează-te'."/>

Figura 17: Autentificarea

Autentifică-te'."/>

Figura 18: Înregistrarea

Pentru adresele de email verificare se face printr-un regex specific acestora, iar pentru parole avem o validare în același format și care impune utilizatorului o parolă de minim 8 caractere dintre care minim o literă și o cifră. În formularul de înregistrare, pe lângă această condiție, mai avem și condiția ca cele două parole să fie identice.

În figura 17 avem surprins modul în care sunt afișate erorile în cazul în care criteriile nu sunt respectate. Dacă toate criteriile de validare au fost respectate însă utilizatorul încearcă să se autentifice cu un cont inexistent, se va reseta formularul și se va afișa în colțul din stânga jos mesajul de eroare din figura 19.



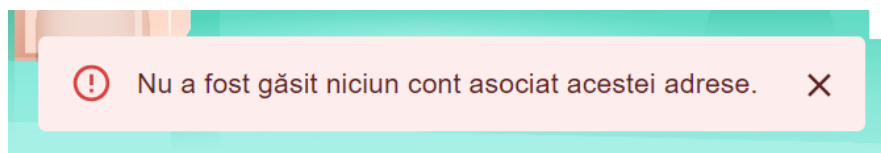


Figura 19: Mesaj de eroare pentru introducerea unei adrese greșite la autentificare

În cazul în care un utilizator dorește înregistrarea cu o adresă de mail ce are un cont asociat, va primi de asemenea un mesaj de eroare.

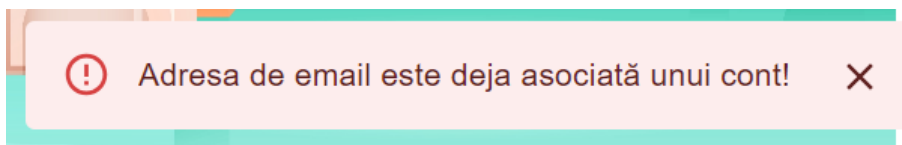
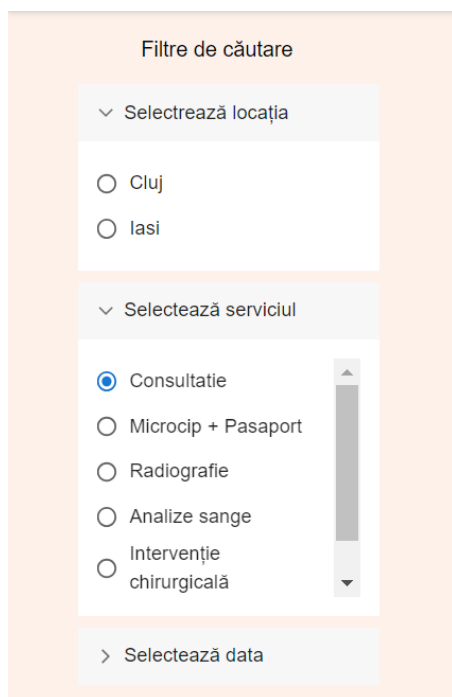


Figura 20: Mesaj de eroare pentru înregistrarea cu o adresă deja utilizată

## Afișarea clinicilor în urma filtrării



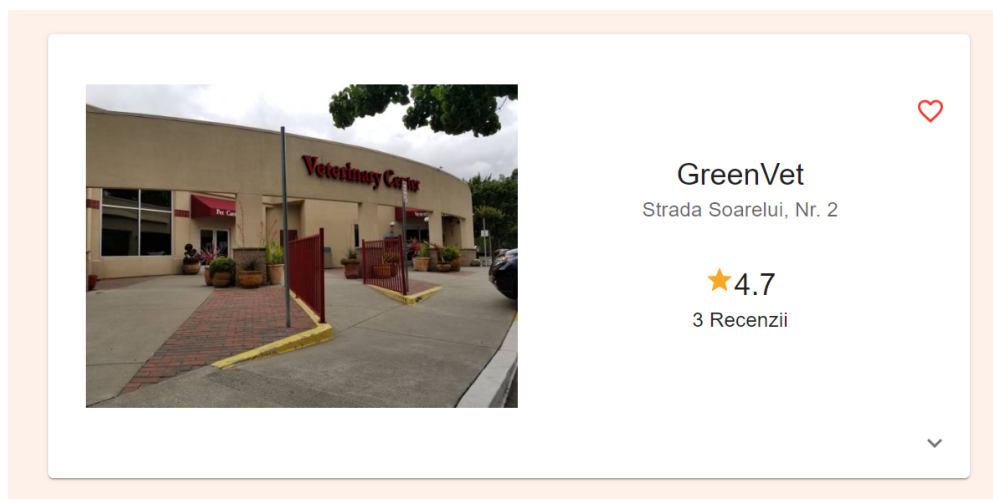
Pagina de afișare a rezultatelor filtrării este împărțită în două. În partea stângă vom avea afișate filtrele selectate în momentul în care am făcut filtrarea, pentru a le putea modifica, iar în restul paginii vom avea afișate clinicile care corespund filtrării.

După cum se poate observa și în figura 12, filtrele pe care nu le utilizăm pot fi restrânse. Totodată acestea au o dimensiunea maximă, iar în cazul unui număr mai mare de servicii sau locații se va afișa o bară pentru derularea acestora.

În cazul ecranului de telefon acest meniu va fi înlocuit de un buton ce la apăsare va deschide lista de filtre.

Figura 21: Filtrele

Clinicile afișate în urma filtrării vor fi actualizate imediat și vor apărea încadrate în câte o casetă. În figura 22 putem observa un exemplu de casetă al unei clinici.



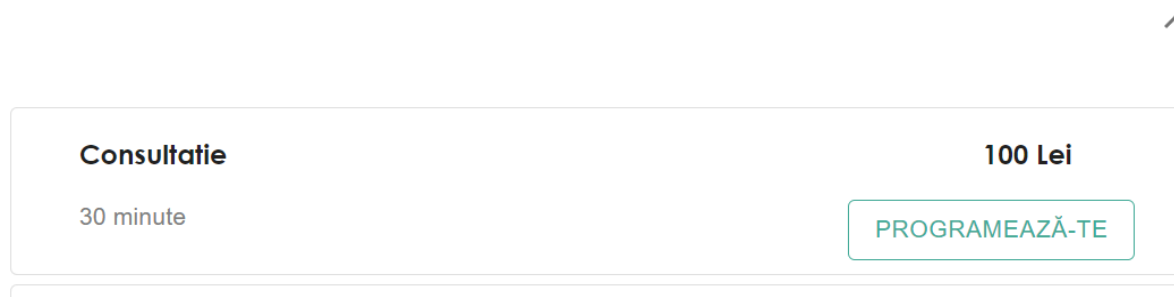
*Figura 22: Afișarea clinicilor*

Caseta este împărțită în două părți. În partea stângă va conține fotografia clinicii sau un avatar dacă aceasta nu a fost adăugată de către administrator. În partea dreaptă aceasta va avea numele clinicii, adresa, media recenziilor și numărul de recenzii. În cazul utilizatorilor autentificați vom avea și o casetă de selectare în formă de inimă, pentru adăugarea clinicii în lista de favorite. Orice click pe imagine sau text ne va duce pe pagina clinicii.

În partea de jos a casetei putem observa și o săgeată. Aceasta are rolul de a expanda caseta și de a afișa lista de servicii disponibilă pentru clinica respectivă.

## Serviciile

Fiecare serviciu va fi afișat de asemenea într-un container ce va arăta ca cel din figura 23.

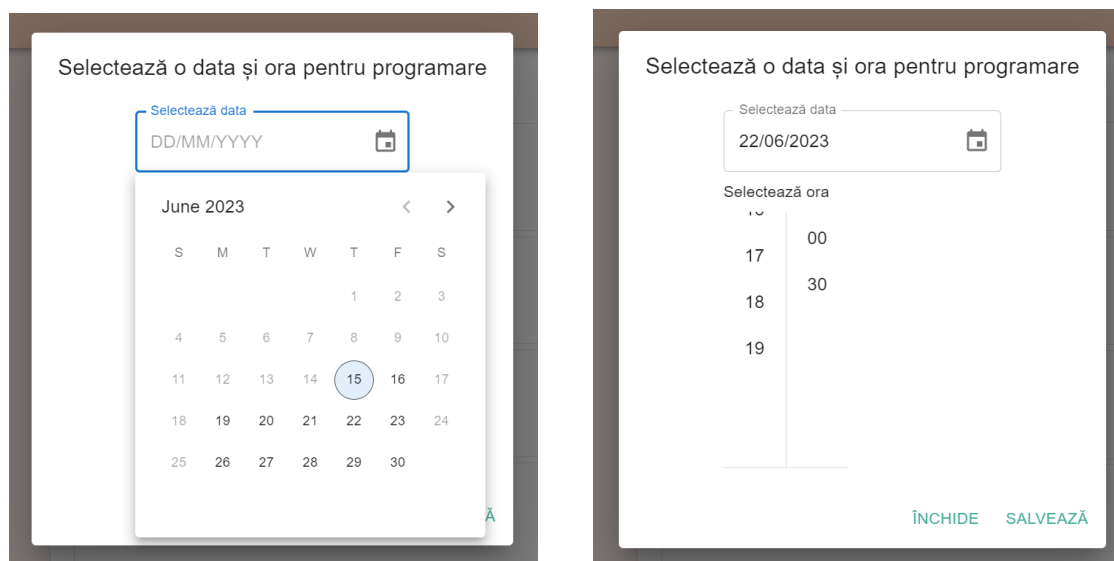


*Figura 23: Afișarea serviciilor*

Acesta va conține numele serviciului, durata, prețul, iar pentru utilizatorii autentificați și un buton pentru programări. Butonul va deschide o fereastră pop-up unde vor fi completate datele necesare realizării programării.

## Programările

În formularul pentru programări nu vor trebui adăugate veterinarul sau serviciul, în funcție de serviciul la care dăm click pe buton, acestea vor fi gestionate automat. Utilizatorul va trebui să-și selecteze doar data și ora. Zilele și orele în care clinica este închisă vor fi dezactivate automat. Pentru zilele și orele la care intervalul este deja ocupat va fi afișat un mesaj de eroare, iar programarea nu va fi salvată, însă va rămâne deschisă pentru a selecta alt interval orar sau altă zi. Putem face o programare pentru o zi ce va fi cel târziu peste 3 luni. Orele disponibile vor apărea în funcție de durata serviciului.



The figure consists of two side-by-side screenshots of a web form titled "Selectează o data și ora pentru programare".

The left screenshot shows the "Selectează data" section with a calendar for June 2023. The date 15 is selected. The right screenshot shows the "Selectează ora" section with a dropdown menu displaying 17:00 and 18:30. At the bottom right, there are two buttons: "ÎNCHIDE" and "SALVEAZĂ".

*Figura 24: Formularul pentru programări*

În imaginile de mai sus putem observa un exemplu de programare la o clinică ce este închisă în zilele de sâmbătă și duminică, iar în ziua selectată programul clinicii se încheie la ora 19:00.

Atât în contul de utilizator, cât și în cel al clinicii, programările se vor afișa sub formă de carduri ce conțin serviciul pentru care s-a făcut programarea, numele medicului, data și ora programării, după cum se poate observa și în figura 25.



The figure shows a single appointment card with a light orange border. The card contains the following text:

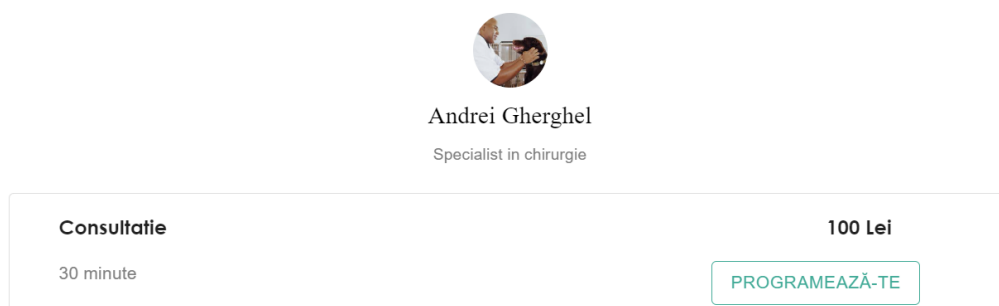
Consultatie  
Andrei Gherghel  
21/06/2023  
11:30:00

*Figura 25: Cardul programării*

Programările mai vechi de data curentă sunt șterse automat din baza de date, așadar nu vor mai apărea în cont după ce au trecut [18].

## Pagina clinicii

În pagina clinicii vom avea mai întâi caseta ce a fost afișată și în pagina de rezultate, după care se vor afișa patru butoane de vizualizare alternativă. Primul buton afișează lista medicilor veterinar, urmați de serviciile acestora. Căsuța medicului veterinar ne redirecționează în pagina medicului.



Andrei Gherghel  
Specialist in chirurgie

Consultatie 100 Lei  
30 minute

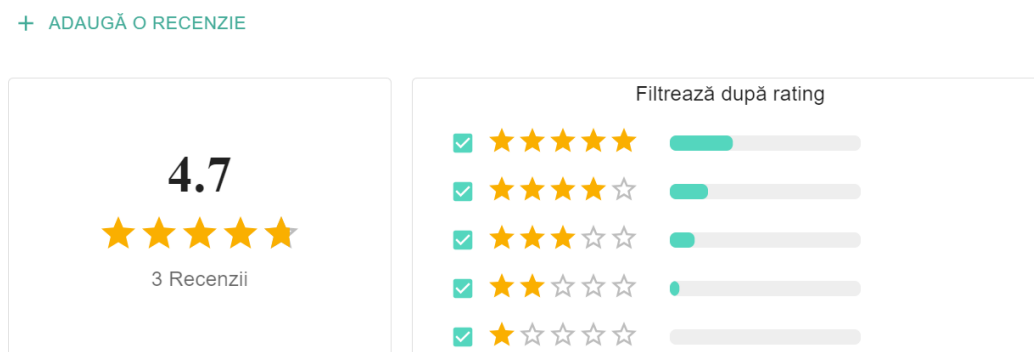
PROGRAMEAZĂ-TE

Figura 26: Lista de servicii

Cel de-al doilea buton de vizualizare alternativă ne va afișa descrierea clinicii, cel de-al treilea va afișa programul acesteia, iar cel de-al patrulea va afișa secțiunea de recenzii.

## Recenziile

În imaginea următoare putem observa prima parte a secțiunii de recenzii. Aceasta are în componență butonul pentru adăugarea de recenzii ce este vizibil doar utilizatorilor autentificați, o coloană de sumarizare a numărului de recenzii, împreună cu media acestora, dar și o coloană pentru filtrarea recenziilor după numărul de steluțe.



+ ADAUGĂ O RECENZIE

4.7  
★★★★★  
3 Recenzii

Filtrează după rating

| Rating | Check                               | Progress |
|--------|-------------------------------------|----------|
| ★★★★★  | <input checked="" type="checkbox"/> | 100%     |
| ★★★★☆  | <input checked="" type="checkbox"/> | 100%     |
| ★★★☆☆  | <input checked="" type="checkbox"/> | 100%     |
| ★★☆☆☆  | <input checked="" type="checkbox"/> | 100%     |
| ★☆☆☆☆  | <input checked="" type="checkbox"/> | 100%     |

Figura 27: Filtrarea recenziilor

În a doua parte a secțiunii de recenzii sunt afișate recenziile în funcție de data adăugării. Imaginea următoare prezintă un exemplu de recenzie.

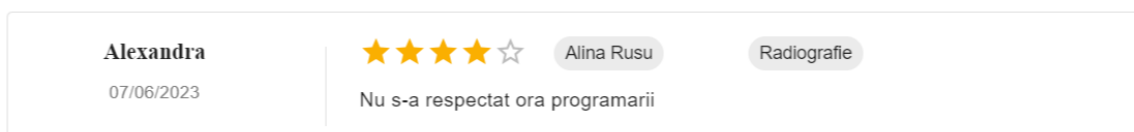


Figura 28: Afișarea recenziilor

O recenzie are în componența ei prenumele utilizatorului ce a lăsat recenzia, data la care a fost adăugată, numărul de steluțe acordat, medicul, serviciul, dar și o scurtă descriere a experienței.

## Formularul de recenzii

Formularul de recenzii este o fereastră pop-up ce apare la apăsarea butonului de adăugare. La deschidere, acesta va arăta ca în figura 29. El va avea setate datele clinicii la care se adaugă recenzia automat, urmând ca utilizatorul să poată selecta unul din medicii clinicii. După ce utilizatorul a selectat și un medic va putea selecta și unul dintre serviciile acestui medic, iar formularul va arăta ca în figura 30. După salvare recenzia va fi adăugată automat în listă, iar media împreună cu numărul de recenzii va fi actualizat în timp real.

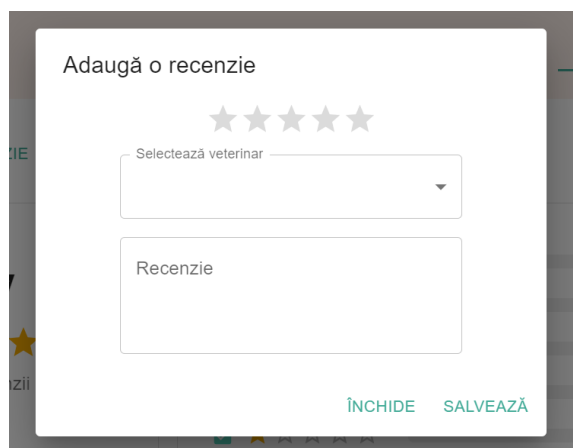


Figura 29: Adăugarea recenziei 1

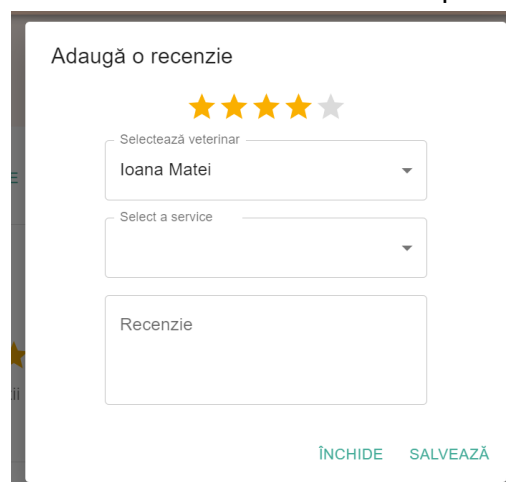


Figura 30: Adăugarea recenziei 2

Formularul de recenzii este de asemenea validat. În cazul în care selectăm un număr de steluțe, însă medicul veterinar sau serviciul nu este selectat, vom primi un mesaj de eroare, iar recenzia nu va fi salvată.

## Pagina medicului veterinar

Aceasta conține informațiile medicului ce apar și în secțiunea servicii din pagina clinicii, iar pe lângă acestea mai avem descrierea acestuia și o secțiune de recenzii, asemănătoare cu cea din pagina clinicii, în care apar doar recenziile acestui medic.

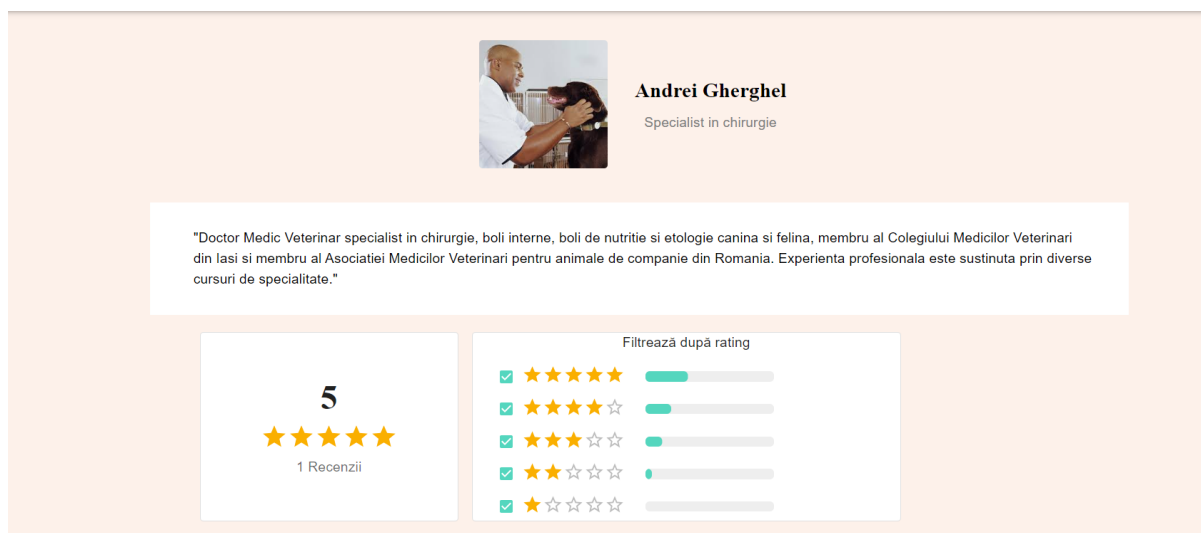


Figura 31: Pagina medicului veterinar

## Adăugarea clinicilor

Adăugarea clinicilor se face printr-un formular format din mai mulți pași. Acesta se va deschide imediat după înregistrarea administratorului.

În imaginea următoare putem observa structura formularului, informațiile ce trebuiesc completate la unul din pași, dar și bara pentru navigarea între între aceștia.

**DIGITAL VET**

✓ Pagina clinicii ✓ Informațiile clinicii ✓ Adăugă program ● Adăugă medici ○ Adăugă servicii

**Adăugă veterinari**

Nume:

Funcție:

O scurtă descriere:

Nu ai ales niciun fișier

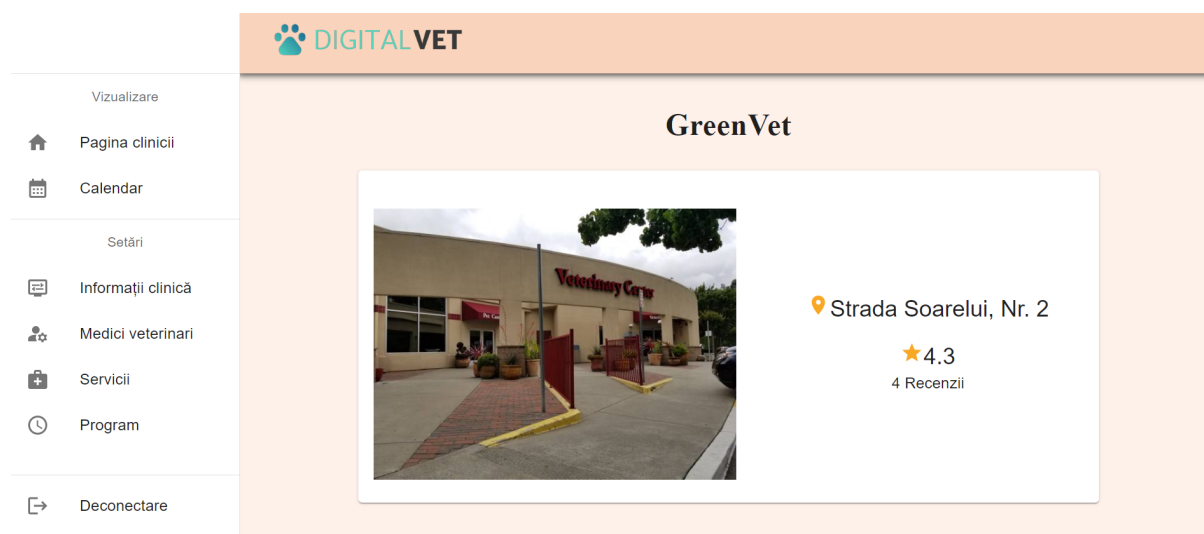
Figura 32: Adăugarea unei clinici

La primul pas, administratorul va completa informațiile necesare pentru pagina clinicii. Aceste informații includ numele, adresa, orașul, o scurtă descriere a clinicii, dar și o fotografie. În cel de-al doilea pas, vor fi adăugate informațiile fiscale ale clinicii. Pasul al treilea presupune adăugarea programului acesteia, iar în ultimii doi pași vor fi adăugați veterinarii și serviciile oferite de către clinică.

Formularul este validat, iar submisia acestuia se va putea face doar după completarea tuturor informațiilor. În cazul în care administratorul nu trimite formularul după înregistrare, la următoarele autentificări în cont va fi redirecționat către acesta.

## Administrarea paginii clinicii

După completarea formularului de adăugare a clinicii, administratorul va fi redirecționat în pagina ce conține meniul de administrarea al acesteia. Acesta poate fi observat în următoarea imagine.



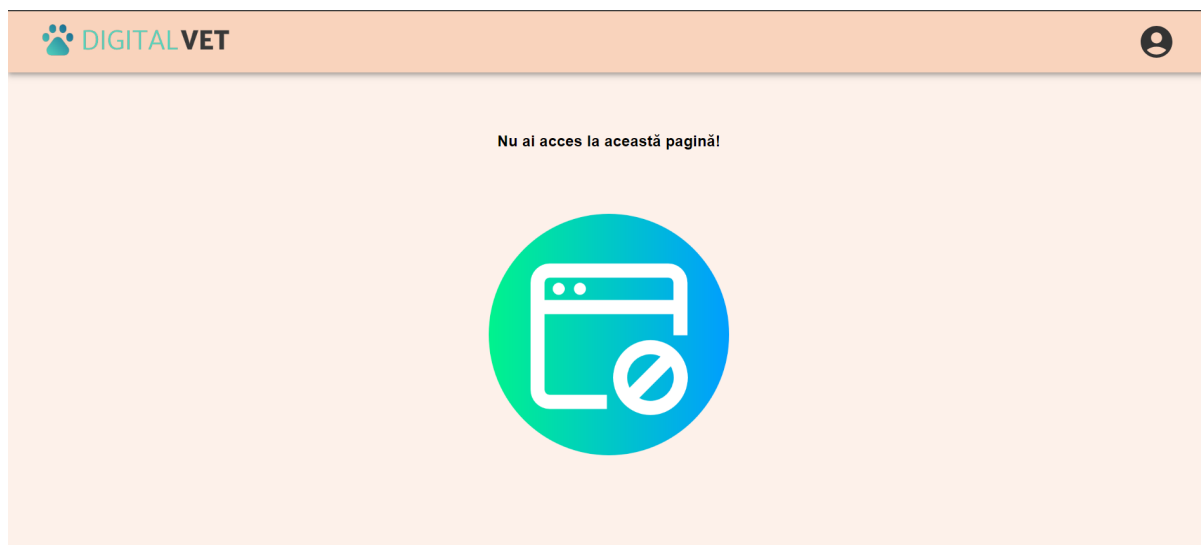
*Figura 33: Administrarea paginii clinicii*

În prima parte a meniului aveam butoanele pentru vizualizare. Primul buton este pentru vizualizarea paginii clinicii exact cum apare pentru orice utilizator. Al doilea buton este pentru a vizualiza programările făcute de utilizatori.

În a doua parte a meniului avem partea de setări. Aici putem edita informațiile ce apar pe pagina clinicii, serviciile, dar și paginile medicilor veterinari.

Din contul de administrator, utilizatorul poate naviga și prin celelalte funcționalități ale aplicației, ce sunt disponibile fără autentificare, și se va putea întoarce mereu în pagina de administrare sau deconecta de la cont din bara de activități.

## Accesul neautorizat



*Figura 34: Fereastra pentru acces neautorizat*

În imaginea de mai sus putem vizualiza fereastra ce se va deschide în cazul în care un utilizator logat într-un cont pentru care nu are rol de administrator va încerca să acceseze URL-ul către o pagină de administrare clinică sau dacă un administrator de clinică va încerca să acceseze link-ul pentru pagina de clinici favorite sau programări.

## URL greșit

În cazul în care un utilizator va încerca accesarea unui URL greșit, acesta va fi redirecționat în fereastra din imaginea următoare.



*Figura 35: Fereastra pentru URL greșit*



# Concluzii și contribuții

## Contribuții

Ideea aplicației DigitalVet a venit de la o problemă întâmpinată de mine în găsirea unui medic veterinar pentru animalutele mele de companie. Procesul de căutare a fost unul destul de greu întrucât am fost nevoită să verific pe rand pagina fiecărei clinici veterinare din oraș. Îmi doream mult ca informațiile relevante despre clinici, recenzii și servicii să fie disponibile într-un singur loc și totodată să fie ușor de accesat. Cu toate acestea am constatat că în acest moment nu există o astfel de aplicație în domeniul veterinar. Această experiență m-a inspirat să dezvolt o aplicație care să eficientizeze procesul de căutare și selecție al clinicilor veterinare, dar totodată și realizarea programărilor.

Deoarece numărul de persoane ce dețin un animaluț de companie este unul mare, iar nevoia de a lua decizii bune într-un timp scurt a devenit o prioritate pentru mulți, m-am gândit că o astfel de aplicație nu m-ar ajuta doar pe mine ci și pe alți iubitori de animale. Așadar am implementat aplicația ideală în viziunea mea, pentru a-mi face viața mai ușoară mie, dar și altor persoane.

În cadrul realizării acestei lucrări am dezvoltat o aplicație web ce oferă suport pentru compararea de clinici și medici veterinari printr-o serie de filtre și vizualizarea de recenzii. De asemenea, aplicația permite utilizatorilor autentificați să adauge clinicile în lista lor de favorite, dar și să lase recenzii. Un alt aspect important al aplicației este dat de funcționalitatea pentru programări. Aceasta optimizează procesul de realizare a programărilor prin intermediul aplicației fără a mai fi nevoie de contactarea personalului clinicilor.

În realizarea acestei aplicații am folosit o serie de tehnologii solide din mai multe puncte de vedere. Pe partea de frontend am ales să utilizez un framework popular ce oferă caracteristici precum componentele reutilizabile și gestionarea eficientă a stării aplicației. De asemenea, am luat în considerare și aspectul securității, astfel că am ales să utilizez pentru serverul de backend Java Spring Boot, un framework ce oferă suport de securitate și protecție a datelor utilizatorilor.

## Concluzii

În concluzie, lucrarea mea, DigitalVet își propune să simplifice procesul de căutare și compararea al clinicilor veterinare, dar și al medicilor veterinari, prin aducerea acestora într-un singur loc. Aplicația oferă suport pentru filtrarea clinicilor și evaluarea lor prin intermediul recenziilor, dar facilitează și procesul de realizare al programărilor, oferind o funcționalitate pentru realizarea acestora prin intermediul aplicației, fără a mai fi nevoie de contactarea personalului clinicii. Totodată aplicația este concepută ca fiind în continuă dezvoltare, având potențial pentru direcții viitoare. Acest aspect este abordat și prin utilizarea

unui set de tehnologii, ce permite implementarea de noi caracteristici prin reutilizabilitatea și modularitatea componentelor, performanța ridicată, dar și nivelul de securitate.

## Direcții viitoare

Direcțiile de dezvoltare ale aplicației sunt date de serviciile pe care clinicele le pot oferi utilizatorilor. O direcție mai importantă ar putea fi implementarea unui chat interactiv între medici și utilizatori, care să permită consultațiile sau sfaturile online. Pentru a motiva medicii să-și dorească o asemenea funcționalitate, aceste consultații ar fi plătite. Așadar, utilizatorii vor trebui să achite mai întâi o sumă prestabilită pentru a accesa acest serviciu. După efectuarea plății, aceștia ar putea iniția și purta conversații cu medicii prin intermediul chat-ului, până când unul dintre ei decide să încheie discuția.

# Referințe

- [1] <https://digitail.io/ro/>
- [2] <https://stailer.ro/>
- [3] <https://www.oracle.com/ro/cx/marketing/automation/what-is-marketing-automation/statistics/>
- [4] <https://www.healthforanimals.org/reports/pet-care-report/global-trends-in-the-pet-population/>
- [5] <https://react.dev/>  
<https://legacy.reactjs.org/docs/getting-started.html>
- [6] <https://mui.com/material-ui/getting-started/overview/>
- [7] <https://spring.io/projects/spring-boot>  
<https://www.baeldung.com/spring-boot>  
<https://www.baeldung.com/spring-security-registration-password-encoding-bcrypt>  
<https://www.bezkoder.com/spring-jpa-query/>  
<https://www.youtube.com/watch?v=EHDlebVv6zw> -ultima vizită 20.06.2023
- [8] <https://dev.mysql.com/>  
<https://www.mysqltutorial.org/>
- [9] <https://www.bezkoder.com/jpa-one-to-many/>
- [10] <https://firebase.google.com/docs/firestore/client/libraries>  
<https://www.youtube.com/watch?v=YOAEBSCKArA> -ultima vizită 20.06.2023
- [11] <https://www.npmjs.com/package/uuid>
- [12] [https://axios-http.com/docs/api\\_intro](https://axios-http.com/docs/api_intro)
- [13] <https://reactrouter.com/en/main>
- [14] <https://www.youtube.com/watch?v=oUZjO00NkhY&list=RDCMU CY38RvRIxYOD O4penyxUwTg&index=2> - ultima vizită 19.06.2023

- [15] <https://www.youtube.com/watch?v=x7niho285qs> -ultima vizită 20.06.2023
- [16] [https://www.youtube.com/watch?v=X3qyx0\\_UTR4&t=1459s](https://www.youtube.com/watch?v=X3qyx0_UTR4&t=1459s) -ultima vizită 19.06.2023
- [17] <https://www.youtube.com/watch?v=EYpdEYK25Dc> -ultima vizită 20.06.2023
- [18] <https://www.mysqltutorial.org/mysql-triggers/working-mysql-scheduled-event/>
- Sursă fotografii utilizate: <https://www.freepik.com/>