***Implementarea unei aplicații web***

***care gestionează informații despre animale***

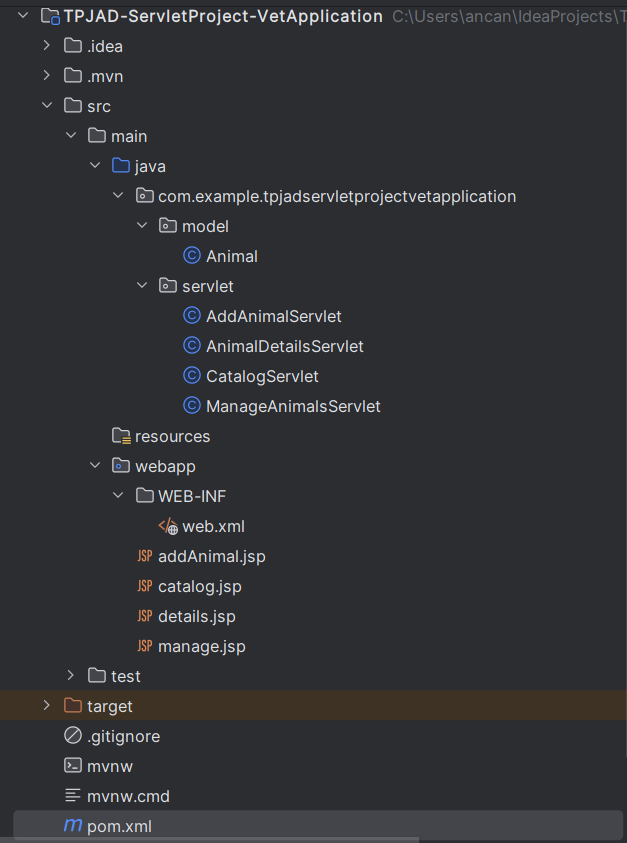
Nistor Anca Alexia, grupa 244

1. **Instalarea și configurarea serverelor Tomcat, Jetty și WildFly**
   1. *Tomcat*

* Descărcarea fișierului zip al versiunii dorite de pe site-ul oficial: [Apache Tomcat® - Apache Tomcat 9 Software Downloads](https://tomcat.apache.org/download-90.cgi)
* Instalarea Tomcat
* Configurarea fișierului tomcat-users.xml astfel încât să conțină un nume de utilizator și parolă
* În cazul în care nu e deja instalat, instalarea Java Development Kit și setarea variabilei de mediu JAVA-HOME
* Deschiderea unui terminal in folderul aferent Tomcat și utilizarea comenzii *bin\startup.bat*
* Dacă serverul pornește cu succes, <http://localhost:8080> va afișa prima pagină Tomcat
  1. *Jetty*
* Descărcarea fișierului zip al versiunii dorite de pe site-ul oficial: [Jetty Downloads :: Eclipse Jetty](https://jetty.org/download.html)
* Instalarea Jetty
* Crearea unui director separat pentru jetty.base, în cazul în care Jetty încearcă să fie pornit cu același director pentru jetty.home și jetty.base. Inițializarea jetty.base folosind *java -jar C:\Users\ancan\jetty-home-12.0.15\start.jar --create-startd* în același director
* Activarea modulelor necesare pentru a rula Jetty rulând *java -jar C:\Users\ancan\jetty-home-12.0.15\start.jar --add-to-start=http,server,deploy* în directorul jetty-base
* Pornirea Jetty folosind *java -jar C:\Users\ancan\jetty-home-12.0.15\start.jar*
* Dacă totul este configurat corect, <http://localhost:8080> va afișa pagina de bun venit a Jetty
  1. *WildFly*
* Descărcarea fișierului zip al versiunii dorite de pe site-ul oficial: [WildFly Downloads](https://www.wildfly.org/downloads/)
* Instalarea WildFly
* Crearea unui utilizator pentru WildFly navigând în directorul bin și rulând *add-user.bat*
* Pornirea WildFly utilizând *standalone.bat*
* Odată ce serverul este pornit, consola de administrare poate fi accesată folosind <http://localhost:9990>

1. **Crearea proiectului și structura acestuia**

* Instalarea Intelij IDEA sau a unui IDE capabil de a facilita dezvoltarea unui proiect Java care utilizează servleturi
* Crearea unui proiect Java de tipul aplicație web
* Structura proiectului este împărțită astfel:



Fișierele care alcătuiesc aplicația propriu-zisă sunt cele din pachetele model și servlet, precum și fișierele de tip .jsp aflate în directorul webapp.

* 1. *Arhitectura aplicației*

Clasa Animal (prezentă în ./src/main/java/model)

|  |  |  |
| --- | --- | --- |
| Nume atribut | Tip | Descriere |
| Id | Int | Identificatorul unic al animalului. |
| Name | String | Numele animalului. |
| Specie | String | Specia de care animalul aparține (ex. câine, pisică). |
| Gender | String | Genul animalului (masculin / feminin). |
| Age | Int | Vârsta animalului, exprimată în ani. |
| OwnerName | String | Numele proprietarului animalului |

Pentru fiecare dintre aceste atribute, clasa prezintă getteri și setteri.

* getId(): Returnează identificatorul unic al animalului.
* getName(): Returnează numele animalului.
* getSpecie(): Returnează specia animalului.
* getGender(): Returnează genul animalului.
* getAge(): Returnează vârsta animalului în ani.
* getOwnerName(): Returnează numele proprietarului animalului.
* setName(String name): Setează numele animalului.
  + Parametru**:** name - Noul nume al animalului.
* setSpecie(String specie): Setează specia animalului.
  + Parametru**:** specie - Noua specie a animalului.
* setGender(String gender): Setează genul animalului.
  + Parametru**:** gender - Noul gen al animalului.
* setAge(Integer age): Setează vârsta animalului.
  + Parametru**:** age - Noua vârstă a animalului.
* setOwnerName(String ownerName): Setează numele proprietarului animalului.
  + Parametru**:** ownerName - Noul nume al proprietarului.

Clasa CatalogServlet (prezentă în ./src/main/java/servlet)

Aceasta un servlet care gestionează afișarea unei liste de animale într-un format paginat. Este responsabilă pentru inițializarea datelor, manipularea cererilor HTTP de tip GET și trimiterea datelor către un fișier JSP pentru afișare.

* init(): Este o metodă apelată automat de server la inițializarea servletului, care creează o listă de obiecte de tip Animal și o stochează în contextul aplicației (ServletContext), astfel încât datele să fie accesibile pe durata sesiunii.
* doGet(HttpServletRequest request, HttpServletResponse response): Este responsabilă pentru gestionarea cererilor HTTP de tip GET și implementează logica de paginare pentru lista de animale, trimițând apoi datele către fișierul JSP.

*Parametri:*

* + HttpServletRequest request: este cererea HTTP trimisă de client, care conține informații despre parametri, de exemplu “page”. El setează atribute care vor fi accesibile în fișierul JSP pentru generarea răspunsului, precum “animals” sau “totalPage”.
  + HttpServletResponse response: permite serverului să răspundă la cererea clientului.

Clasa AnimalDetailsServlet (prezentă în ./src/main/java/servlet)

Este un servlet care gestionează afișarea detaliilor pentru un anumit animal selectat din catalog. Aceasta procesează cererile de tip GET, caută un animal pe baza numelui său în lista disponibilă și trimite datele către un fișier JSP pentru afișare.

* doGet(HttpServletRequest request, HttpServletResponse response): metoda gestionează cererile HTTP de tip GET, primind numele unui animal ca parametru și căutându-l în lista de animale stocată în contextul aplicației. Dacă găsește un animal cu numele respectiv, trimite detaliile acestuia către fișierul details.jsp pentru afișare.

*Parametri:*

* HttpServletRequest request: conține detalii despre cererea HTTP primită, inclusiv parametrul “name”, a cărei valoare este extrasă.
* HttpServletResponse response: permite trimiterea unui răspuns către client. În cazul în care animalul nu este găsit, response va trimite un mesaj de eroare.

Clasa ManageAnimalsServlet (prezentă în ./src/main/java/servlet)

Acest servlet gestionează două tipuri principale de acțiuni asupra listei de animale: editarea și ștergerea.

* doGet(HttpServletRequest request, HttpServletResponse response): metoda este utilizată pentru afișarea detaliilor unui animal pentru editare. Dacă acțiunea preluată din URL-ul cererii este “action”, metoda încearcă să găsească animalul cu id-ul specificat. Se folosește de animalele din contextul aplicației, filtrează animalul dorit și, dacă este găsit, adaugă detaliile acestuia în cererea HTTP (atributul “animal” în request). Dacă nu este găsit, trimite o eroare 404 Not Found. În final, răspunsul este trimis către pagina JSP ./manage.jsp, care afișează formularul pentru editarea animalului.

*Parametri:*

* HttpServletRequest request: este folosit pentru a extrage parametrii “action” și “id” din URL-ul cererii, și adaugă un obiect în cerere pentru a fi utilizat mai târziu în pagina JSP, mai exact ”animal”.
* HttpServletResponse response: reprezintă răspunsul HTTP pe care serverul trimite înapoi către client. În cazul unei erori, trimite un mesaj Not Found.
* doPost(HttpServletRequest request, HttpServletResponse response): metoda este utilizată pentru a gestiona cererile HTTP de tip POST, cum ar fi editarea sau ștergerea unui animal. Dacă acțiunea dorită este cea de editare, funcția găsește animalul al cărui id corespunde cu cel primit în cerere, din lista de animale, și actualizează atributele lui cu valorile trimise. Dacă acțiunea este de ștergere, elimină animalul cu id-ul aferent din listă. În final, după ce opțiunea de editare sau de ștergere a fost efectuată, utilizatorul este redirecționat către catalogul cu lista de animale.

*Parametri:*

* HttpServletRequest request: este folosit pentru a extrage acțiunea (editare / ștergere) și atributele noi ale animalului, în cazul editării lui. De asemenea, mai este utilizat pentru a găsi calea aferentă pentru a redirecționa utilizatorul înapoi la catalog.
* HttpServletResponse response: folosit pentru a redirecționa utilizatorul către catalogul cu animale.

Clasa AddAnimalServlet (prezentă în ./src/main/java/servlet)

Acest servlet este responsabil pentru adăugarea unui nou animal în listă și gestionează cererile de tip GET și POST.

* doGet(HttpServletRequest request, HttpServletResponse response): metoda afișează pagina addAnimal.jsp, care conține un formular pentru introducerea detaliilor unui nou animal.
* doPost(HttpServletRequest request, HttpServletResponse response): metoda extrage datele trimise prin formular, creează un obiect Animal și îl adaugă în lista de animale gestionată la nivel de context.

*Parametri:*

* HttpServletRequest request: este folosit pentru a extrage atributele animalului.
* HttpServletResponse response: folosit pentru a redirecționa utilizatorul către catalogul cu animale, după ce animalul a fost adăugat.

Pagina catalog.jsp (prezentă în ./src/main/webapp)

Această pagină JSP afișează un catalog de animale într-un tabel dinamic, unde datele sunt furnizate de CatalogServlet printr-o listă de obiecte Animal. Fiecare rând din tabel include detalii precum id-ul, specia, și numele animalului, alături de acțiuni pentru vizualizarea detaliilor sau editarea acestuia. Navigarea între pagini este implementată folosind butoane pentru paginarea anterioară și următoare, ascunse atunci când utilizatorul se află la marginea intervalului.

Pagina details.jsp (prezentă în ./src/main/webapp)

Pagina afișează detaliile unui animal selectat din catalog. Structura include un antet simplu și o secțiune centrală care conține informațiile detaliate despre animal, cum ar fi numele, genul, specia, vârsta, și numele proprietarului. Datele sunt furnizate de un obiect Animal, setat ca atribut în request de către AnimalDetailsServlet. Pagina oferă, de asemenea, un buton pentru a reveni la catalog, facilitând navigarea între detalii și lista de animale.

Pagina manage.jsp (prezentă în ./src/main/webapp)

Această pagină JSP oferă un formular pentru editarea detaliilor sau ștergerea unui animal existent. Valorile câmpurilor formularului sunt precompletate cu datele deja existente ale animalului. Formularul permite utilizatorului să editeze numele, specia, genul, vârsta și numele proprietarului. Pe lângă funcționalitatea de salvare a modificărilor, există și un buton pentru ștergerea animalului selectat, iar o acțiune (edit sau remove) este transmisă către servletul ManageAnimalsServlet. În partea de jos a paginii, un link facilitează revenirea la catalogul cu animale.

Pagina addAnimal.jsp (prezentă în ./src/main/webapp)

Pagina oferă un formular pentru adăugarea unui nou animal în catalog. Acesta include câmpuri obligatorii pentru detalii esențiale, precum nume, specie, gen, vârstă și numele proprietarului, fiecare fiind validat prin atributul required. După completarea datelor, utilizatorul trimite formularul către AddAnimalServlet pentru a adăuga animalul în sistem.

* 1. *Comunicarea între servleturi și paginile JSP*

Servleturile sunt responsabile pentru manipularea datelor, cum ar fi adăugarea, editarea și ștergerea animalelor, și pentru transmiterea acestora către paginile JSP care sunt folosite pentru prezentarea datelor, prin metode precum setAttribute.

De exemplu, în metoda *doGet* din servletul *AnimalDetailsServlet*, id-ul animalului este preluat din request. Folosind întreaga listă de animale, aceasta caută animalul cu id-ul primit. Dacă acesta este găsit, atributul “animal” este setat având valoarea aferentă. Apoi, servletul redirecționează cererea către pagina JSP corespunzătoare (*details.jsp*), folosind RequestDispatcher.

În pagina JSP, datele animalului sunt accesibile prin expresii JSP, care se află între %. Aceste expresii vor afișa valorile specifice pentru animalul selectat.

Un alt exemplu ar fi când un utilizator dorește să adauge un animal nou în catalog. Un formular în JSP (prezent în *addAnimal.jsp*) trimite datele către *AddAnimalServlet*, care preia informațiile din cerere și apoi creează un animal nou, pe care îl adaugă în lista (“animals”) din context. Această logică este prezentă în metoda *doPost*.

1. **Deploy și execuția aplicației**
   1. Tomcat

* În terminal, se navighează în directorul proiectului și se rulează comanda *mvn package*
* Se copiază fișierul generat (de tip .war) în directorul webapps al serverului Tomcat
* Se pornește Tomcat navigând la directorul principal, apoi rulând *bin\startup.bat*
* Se accesează *localhost:8080/numele-fișierului* .war
  1. Jetty
* În terminal, se navighează în directorul proiectului și se rulează comanda *mvn package*
* Se copiază fișierul generat (de tip .war)
* Se pornește Jetty rulând *java -jar C:\Users\ancan\jetty-home-12.0.15\start.jar* în directorul jetty-base!
* Se accesează *localhost:8080/numele-fișierului* (.war)
  1. WildFly
* În terminal, se navighează în directorul proiectului și se rulează comanda *mvn package*
* Se copiază fișierul generat (de tip .war) în directorul webapps al serverului WildFly
* Se pornește serverul rulând fișierul standalone.bat în folderul bin
* Se accesează *localhost:9990*, se autentifică utilizând numele de utilizator și parola
* Se navighează către *deployments*, unde se adaugă fișierul .war creat
* Se pornește aplicația din interfață