FACULTATEA: Automatică si Calculatoare

SPECIALIZAREA: Calculatoare si Tehnologia Informației

DISCIPLINA: Ingineria Sistemelor

Indrumator laborator:
Marghescu Luminita Madalina

Realizator:
Mihalache Rares

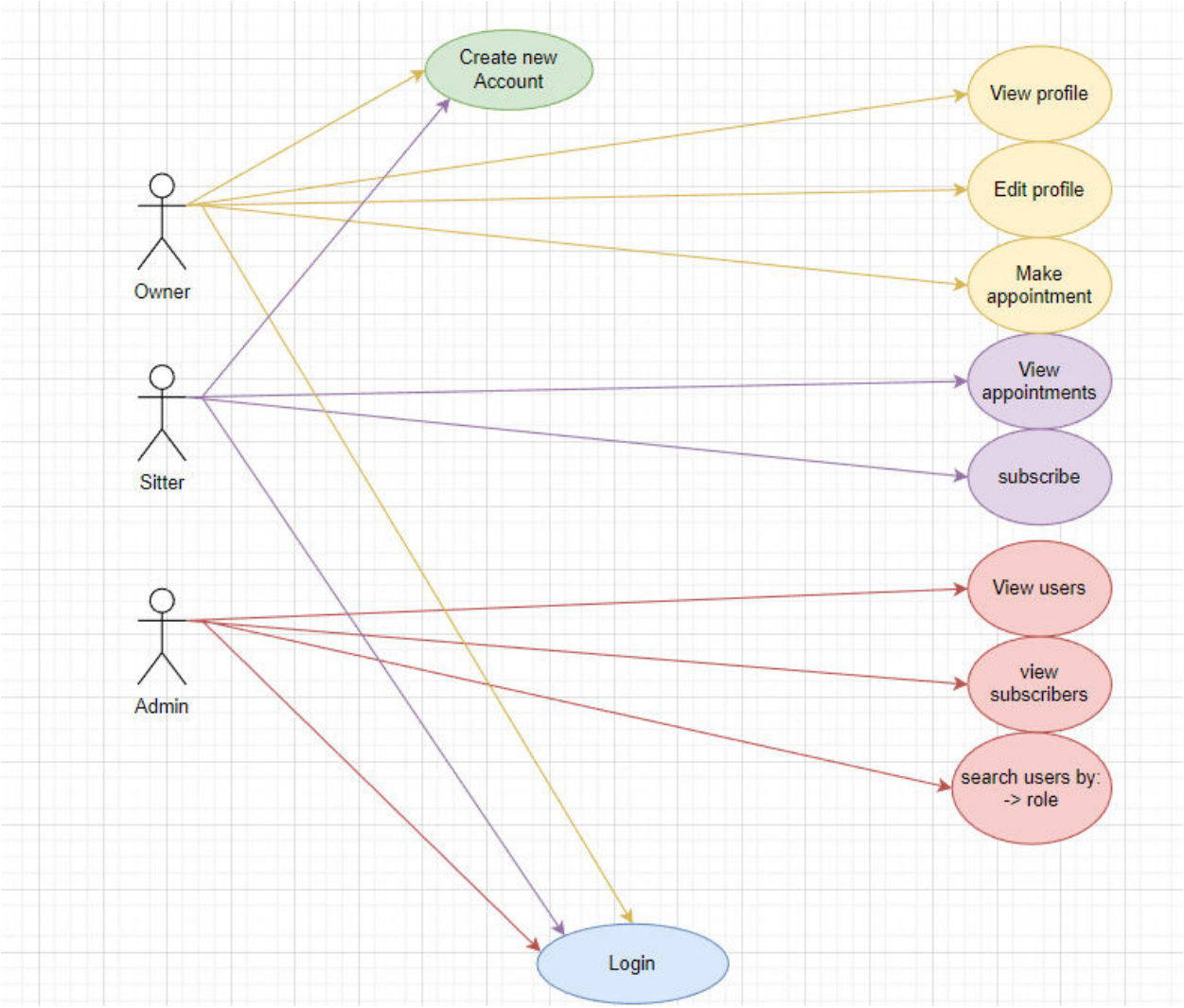
Descrierea proiectului:

Acest proiect are ca scop realizarea unei aplicatii Web, pentru consolidarea cunostiintelor de programare intr-un limbaj studiat, respectiv folosirea unor tehnici de programare si a unor Design Patterns, cat si a diferitelor framework-uri atat pentru front-end, cat si pentru back-end.

In cadrul acestui proiect, am incercat sa realizez o aplicatie destinata oamenilor care doresc sa caute ingrijitori pentru animalele lor de companie. Aceasta problema apare des cand vrei sa pleci undeva si nu ai pe cine sa lasi sa aiba grija de animalele tale.

Aplicatia are 3 actori principali: Admin, Owner si Sitter. Fiecare dintre acesti actori are anumite functionalitati, descrise mai amanunti in capitolul 2 (diagrama Use Case-uri). Inainte de a utiliza aplicatia, utilizatorul trebuie sa isi creeze un cont, iar mai apoi sa dea “login”.

Diagrama Use Case-uri:

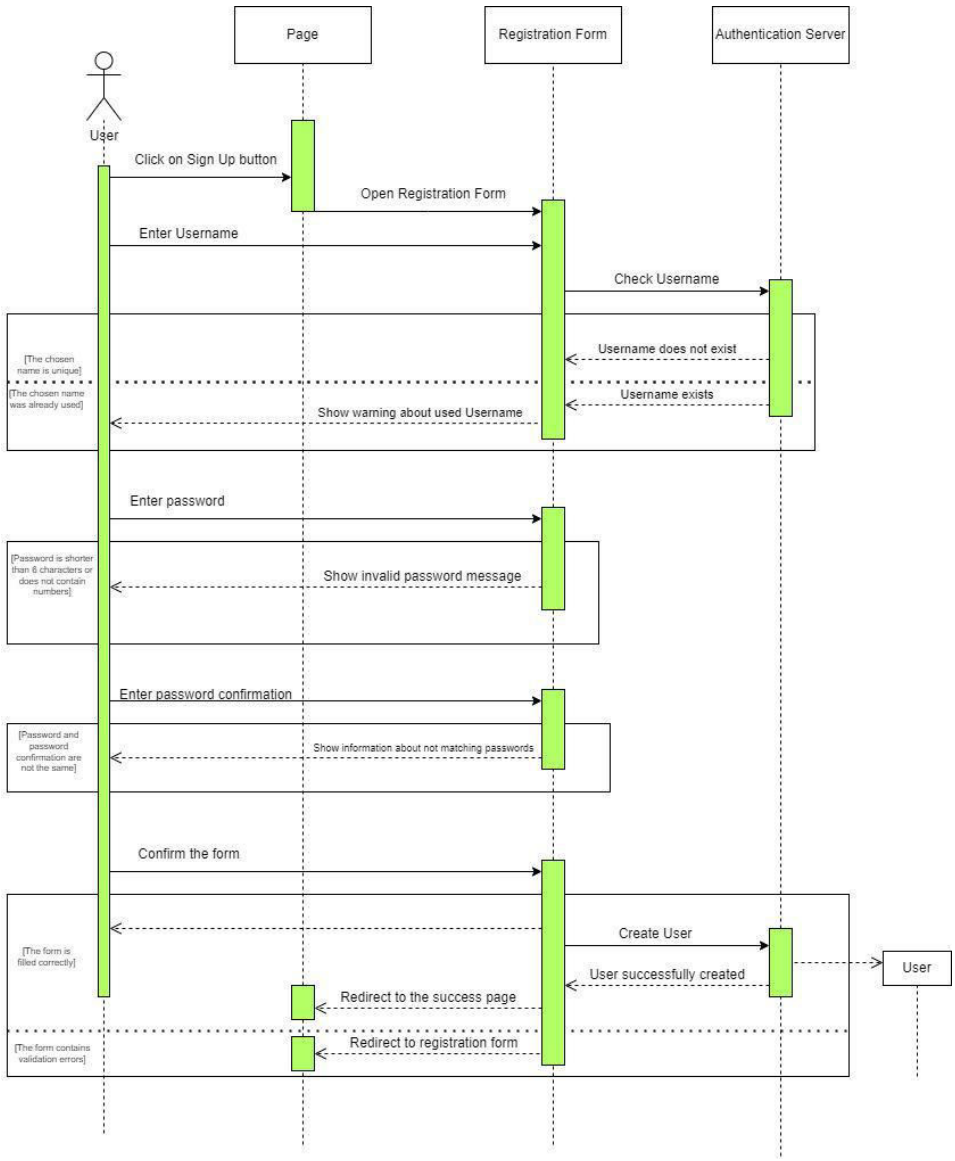


Indrumator laborator:
Marghescu Luminita Madalina

Realizator:
Mihalache Rares

Diagrama de secventa:

Mai jos este evidentiata diagrama de secventa pentru crearea unui cont:



Indrumator laborator:
Marghescu Luminita Madalina

Realizator:
Mihalache Rares

Design Patterns utilizate

Ca si Design Patterns, am folosit Builder (creational DP), pentru initializarea atributelor obiectelor noi create. Acest DP se poate observa in pachetul de modele.

De asemenea, am folosit Factory DP, pentru crearea unor obiecte care implementaza o anumita interfata, nestiind cu exactitate clasa din care provin acele obiecte. Astfel, clase obiectelor se stabileste la run time, si acest Design Pattern este foarte bun pentru ca respecta 2 principii SOLID, si anume Open-Closed Principle, care spune ca modulele software ar trebui sa fie deschise pentru a fi extinse si inchise pentru a fi modificate, si mai ales Liskov Principle, deoarece obiectele superclasei pot fi inlocuite cu obiectele subclaselor fara a strica aplicatia. Subclasele nu reprezinta altceva, decat niste extensii specializate ale superclasei. Astfel putem scrie ceva de genul: Superclasa obj1 = new Subclasa();

Ca si stil architectural, am mers pe o abordare Layered-architecture combinata cu MVC (Model-View-Controller). Astfel aplicatia mea este separata in cateva pachete generale:

- ➔ View - pentru crearea de pagini .html/ ceea ce vede utilizatorul
- ➔ Controller - aici are loc interactiunea si schimbul de informatii intre back-end si front-end, cat si stabilirea de end-pointuri si tipul de metoda HTTP pentru fiecare end-point.
- ➔ Service - aici este partea de business logic a aplicatiei.
- ➔ Repository – partea care se ocupa de comunicarea aplicatiei cu baza de date. Aici se scriu Query-uri JPQL sau SQL, dupa caz. Multe metode sunt deja implementate, deoarece toate clasele din acest pachet implementeaza JpaRepository.
- ➔ Model – aici sunt definite entitatile aplicatiei.

Tehnologii utilizate

Ca si tehnologii, limbajul folosit a fost Java, versiunea 17. Pe partea de back-end am utilizat mai multe frameworkuri si librarii. Printre acestea se numara: Spring MVC, Lombok, Spring Security, Spring Data JPA, etc.

Baza de date utilizata a fost PostgreSQL.

Pe partea de front-end am folosit thymeleaf si thymeleaf-security, iar pentru generarea templateurilor am folosit Bootstrap 4.0.0.

Indrumator laborator: Realizator:
Marghescu Luminita Madalina Mihalache Rares

Concluzie

Chiar daca mai e de muncit la aplicatie, pot zice ca am invatat foarte multe lucruri despre securitatea aplicatiilor Web. Am folosit Http Forms Authentication, am generator roluri pentru utilizatorii aplicatiei, am invatat despre autorizare si cum sa permit access doar la anumite pagini/resurse, in functie de rolul utilizatorului, etc.

Ca si functionalitati, am reusit sa dezvolt anumite functionalitati care tin mai mult de CRUD, si mai putine lucruri sofisticate.

Nu in ultimul rand, am invatat cum sa lucrez cu thymeleaf si cum sa imi import template-uri pe care sa le leg de controllerele aplicatiei.

Bibliografie

1. https://www.youtube.com/watch?v=EPd-u8ibXBY&list=PLVApX3evDwJ1d0lKKHssPQvzv2Ao3e__Q&index=1
2. https://www.thymeleaf.org/doc/tutorials/2.1/usingthymeleaf.html
3. https://www.baeldung.com/spring-mvc-and-the-modelattribute-annotation
4. https://spring.io/guides/tutorials/rest/

Indrumator laborator:
Marghescu Luminita Madalina

Realizator:
Mihalache Rares