Universitatea Tehnică ”Gheorghe Asachi” din Iași
Facultatea de Automatică și Calculatoare
Domeniul Calculatoare și Tehnologia Informației
Specializare Tehnologia Informației

Aplicație web bazată pe servicii ce facilitează interacțiunea pacienților cu medicii

Coordonator științific:
Ș.l. dr. inf. Dumitriu Tiberius

Student:
Munteanu Letiția-Ioana

2021-2022

Cuprins

Introducere

Contextul pandemiei a avut un impact semnificativ asupra tuturor domeniilor, generând schimbări majore în viețile oamenilor. Fiind nevoiți să reducem pe cât de mult posibil interacțiunea fizică cu alte persoane, deplasările în spații publice și să respectăm alte reguli specifice acestei perioade, activitatea în mediul online a luat amploare în majoritatea domeniilor. Odată cu această schimbare, multe dintre serviciile de care oamenii au nevoie s-au dezvoltat pe partea virtuală, facilitându-ne astfel utilizarea lor.

Toți acești factori ai vieții cotidiene precum și interesul în a ajuta alte persoane, au dus la alegerea temei de licență dezvoltarea unei aplicații web ce reprezintă o platformă online pentru spitalizare, făcând astfel mai ușor de accesat serviciile medicale. Astfel, oamenii pot folosi această platformă pentru a beneficia de consultații online, fără a fi nevoiți să se deplaseze la centrele medicale. Își pot primi rezultatele, interpretate în prealabil de către un medic și își pot introduce simptomele în vederea alegerii unei specializări medicale spre care ar trebui să se îndrepte.

1. Fundamente teoretice

~~Terminologie~~ specifică

Lucrarea de licență va fi dezvoltată cu ajutorul următoarelor tehnologii: partea de backend va fi implementată folosind o arhitectură bazată pe servicii. Pentru facilitarea acestui lucru se vor folosi Spring Boot, utilizând ca limbaj de programare Java și FastApi, utilizând Python. Partea de frontend va fi realizată utilizând libraria ReactJS, codul fiind scris in JavaScript. În ceea ce priveşte procesarea datelor introduse de către utilizator pentru obţinerea unei recomandări de specializare medicală s-a folosit un algoritm de Machine Learning, Random Forest. Pentru stocarea datelor s-au folosit ca baze de date MySQL şi MongoDB.

Spring Boot este un tool care facilitează dezvoltarea de aplicații web, servicii și microservicii utilizând framework-ul **Spring,** făcând acest proces mai ușor și rapid prin cele trei capabilități oferite:

➤ autoconfigurare: aplicațiile sunt inițializate cu dependențe prestabilite, pe care utilizatorul nu trebuie să le configureze manual

➤ posibilitatea de a configura dependențele în funcție de cerințele proiectului

➢ crearea unei aplicații de sine stătătoare care rulează de una singură, fără a se baza pe surse externe, acest lucru fiind datorat încorporării unui server web la inițializarea aplicației (Tomcat 7 este serverul utilizat by default de Spring Boot) [3].

De asemenea, securitatea este asigurată într-o astfel de aplicaţie, Spring Boot având ca standard de securitate **Spring Security**.

Spring Security este un framework care se ~~focusează~~ pe controlul accesului într-o aplicaţie dezvoltată în Java. Acesta oferă funcţionalităţi atât din punct de vedere al autentificării, cât şi din punct de vedere al autorizării, fiind foarte flexibil în ceea ce priveşte parametrizarea, personalizarea şi configurarea în vederea satisfacerii cerinţelor dezvoltatorilor de aplicaţii [12].

Infrastructura web a Spring Security se bazează în întregime pe filtre standard de servleţi. Acesta nu foloseşte intern servleţi sau alte framework-uri bazate pe servleţi, aşadar, nu are legături către o tehnologie web particulară. Lucrul cu HttpServletRequests şi HttpServletResponses asigură flexibilitate în primirea cererilor de la diferiţi clienţi, cum ar fi: browsere, servicii web client, HttpInvoker sau aplicații AJAX. Spring Security menţine intern o înlănţuire de filtre, fiecare având responsabilitate proprie. Acestea sunt adăugate sau şterse din configuraţie în funcţie de serviciile cerute. De asemenea, ordinea filtrelor este importantă, acest lucru fiind dat de dependenţele dintre filtre [13].

Aşadar, Spring Security asigură funcţionalităţi pentru: autentificare, autorizare şi filtrarea servleţilor.

Autentificarea reprezintă procesul prin care se validează identitatea unui utilizator, pe când **autorizarea** este procesul prin care utilizatorului autentificat i se permite accesul la diferite resurse. Spring Security are o arhitectură proiectată să separe autentificarea de autorizare.

În contextul Spring Security, autentificarea este reprezentată de tokenul de autentificare sau pur şi simplu, autentificarea. Acesta conţine informaţiile necesare procesului de autentificare, ele fiind:

- **principalul**, prin care este identificat userul, fiind de obicei o instanţă a UserDetails atunci când autentificarea este făcută cu username şi parolă
- **credenţialele**, de obicei parola
- **permisiunile**, o colecţie de roluri sau scopuri care vor fi folosite mai apoi în configurarea proceselor de autorizare

Autentificarea poate fi privită ca fiind input-ul către **AuthenticationManager**, API-ul care defineşte cum realizează filtrele Spring Security autentificarea. În urma unei

astfel de cereri care a fost procesată de către AuthenticationManager prin metoda **authenticate**, tokenul de autentificare este salvat în **SpringContextHolder**, aici fiind salvate detaliile celor autentificaţi. Acesta foloseşte un thread local pentru a salva informaţiile necesare, iar pentru a afla userul curent autentificat se foloseşte **SecurityContext**, care conţine un obiect Authentication. Deşi se foloseşte un singur thread pentru salvarea acestor informaţii, **ProviderManager**, va şterge informaţiile sensibile din Authentication, obiectul returnat atunci când o cerere de autentificare este făcută cu succes. Provider Manager este cea mai folosită implementare a AuthenticationManager. Acesta iterează o cerere de autentificare printr-o listă de **AuthenticationProvider**, care realizează diferite tipuri de autentificare, până când găseşte tipul de autentificare căutat [14], [15].

Există mai multe mecanisme de autentificare, cele mai populare fiind:

- autentificarea cu nume şi parolă
- OAuth 2.0 Login
- SAML 2.0 Login
- Central Authentication Server (CAS)
- Remember me

Cel mai des întâlnit mecanism de autentificare este cel care foloseşte credenţialele unui client, parolă şi nume de utilizator, pentru a-i verifica identitatea. Pentru acest tip de autentificare există câteva metode prin care poate fi utilizat:

- Form Authentication
- Basic Authentication
- Digest Authentication

Majoritatea aplicaţiilor folosesc Form Authentication, sau mai simplu spus, autentificarea făcută cu ajutorul unui formular HTML în care clientul îşi poate introduce numele de utilizator și parola. Flow-ul procesului de autentificare prin această metodă este următorul:

1. Clientul îşi introduce credenţialele şi le trimite, acest lucru concretizându-se într-o cerere care ajunge la server
2. Este ales filtrul corect care procesează cererea venită. Având în vedere că cererea trimisă este de tipul POST, conţinând parola şi numele utilizatorului, filtrul ales de către Spring Security este **UsenamePasswordAuthenticationFilter.**
3. Mai apoi, acest filtru este trimis mai departe către AuthenticationManager, care, prin implementarea sa, iterează prin lista de AuthenticationProvider şi încearcă să facă autentificarea pe baza obiectului de tip Authentication primit

4. Dacă autentificarea nu s-a făcut cu succes, atunci SecurityContextHolder este eliberat şi este invocat **AuthenticationFailureHandler**
5. Dacă autentificarea s-a făcut cu succes, **SessionAuthenticationStrategy** este notificat de acest lucru iar obiectul Authentication este setat în SecurityContextHolder [16], [17].

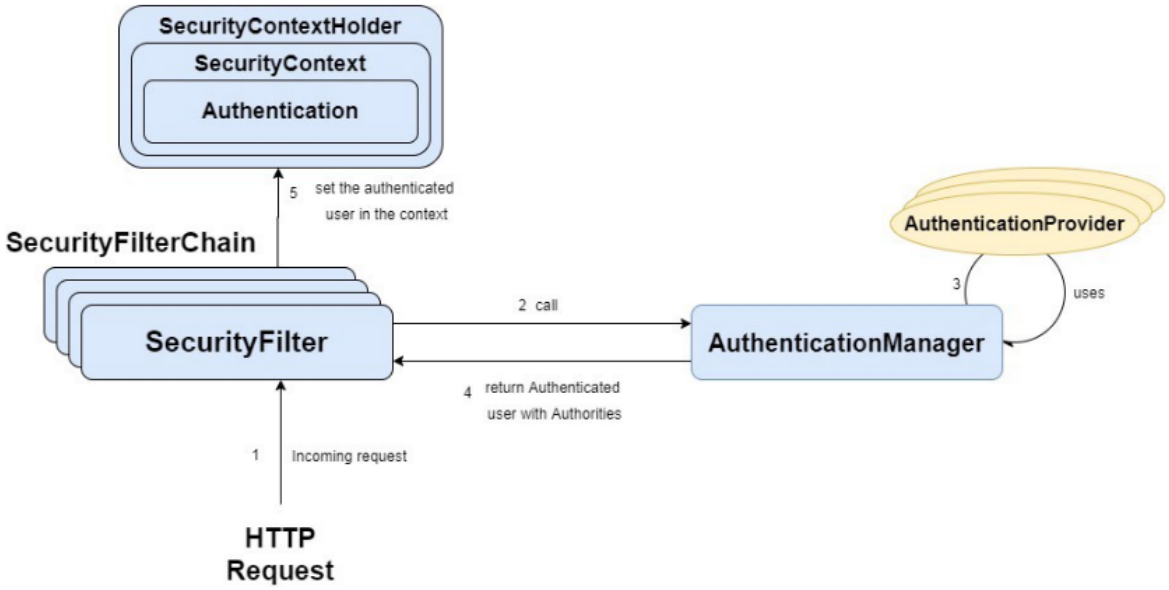

Fig. 1: Flow-ul procesului de autentificare

Detaliile utilizatorilor sunt stocate într-o bază de date. Pentru a putea accesa detaliile acestora este necesară folosirea **UserDetailsService**, o interfaţă formată dintr-o singură metodă care returnează un obiect de tipul **UserDetails**, obiect ce conţine toate informaţiile unui client. De asemenea, pentru a putea verifica parola, care este stocată în baza de date sub formă criptată, trebuie creat un **Bean** corespunzător, care nu este altceva decât un obiect instanţiat şi gestionat de către container-ul Spring, având ca scop furnizarea unui serviciu în acest caz. Bean-ul **PasswordEncoder** are ca scop stabilirea unui model de criptare a parolelor, de exemplu **BCryptPasswordEncoder**, oferind mai apoi funcţionalităţi de decriptare şi verificare a acestor date.

Autorizarea utilizatorilor către anumite resurse poate fi stabilită atât în configurarea web a aplicaţiei adăugând în metoda **antMatchers** rolul pentru care accesul este permis, cât şi la nivel de controller al aplicaţiei, adăugând adnotarea **PreAuthorize** cu specificarea rolului corespunzător.

FastAPI este un framework modern şi rapid folosit pentru dezvoltarea aplicaţiilor web în Python. Principalele facilităţi oferite de acesta sunt:

- rapiditate, înregistrând performanţe înalte, fiind unul dintre cele mai rapide framework-uri valabile pentru Python
- reducerea erorilor, reducând 40% din erorile induse de către programatori
- robusteţe, oferind cod pentru producţie, având documentaţie interactivă

Acesta oferă flexibilitatea de a utiliza diferite servere pentru dezvoltarea aplicaţiilor, unul dintre cele mai folosite fiind **Uvicorn** [18].

ReactJS este o librărie JavaScript folosită pentru a crea interfețe utilizator interactive. După cum este și denumirea, “React” reacționează la stările aplicației, făcând update-uri ori de câte ori se schimbă datele. React are o arhitectură bazată pe componente, fiecare administrându-și stările proprii. Acestea sunt compuse mai apoi pentru a construi un UI complex [8].

Componentele sunt segmente de cod independente şi în acelaşi timp, reutilizabile, reprezentând baza construirii unei aplicaţii în React. Acestea au acelaşi scop ca şi funcţiile JavaScript, dar în schimb, returnează elemente HTML. Există două tipuri de componente, iar acestea sunt:

- **componente funcţionale**
- **componente de tip clasă**

Componentele funcţionale sunt funcţii JavaScript care pot avea sau nu parametri, proprietăţi şi care returnează elemente JSX (HTML şi JavaScript). Acestea nu au stare proprie şi nici metode care să gestioneze ciclul de viaţă al acestora, deşi aceste lucruri pot fi adăugate folosind funcționalitățile oferite de către React, numite Hook-uri. Scopul principal al componentelor funcţionale este acela de a afişa informaţii.

Componentele de tip clasă sunt mai complexe decât cele funcţionale, oferind posibilitatea de a trimite date de la o clasă la alta. Spre deosebire de componentele funcţionale, acestea au stare proprie şi metode de gestionare a ciclului de viaţă, cum ar fi **componentDidMount, componentDidUpdate** sau **componentWillUnmount**. Există diferenţe şi din punct de vedere al sintaxei. Componentele de tip clasă trebuie sa

extindă React Component şi trebuie să aibă o metodă numită **render** pentru a returna elemente JSX [19].

Hook-urile sunt noi funcţionalităţi oferite de librăria React începând cu versiunea 16.8 având scopul de a gestiona starea componentelor funcţionale. Acestea nu pot fi utilizate în componentele de tip clasă. Cele mai utilizate astfel de funcţii sunt:

1. **Hook-urile de stare** prin care se poate seta sau se poate lua starea unei componente utilizând funcţia **useState**.
2. **Hook-urile efect** care permit efectuarea unor acţiuni la randarea componentei, cum ar fi actualizarea DOM-ului, extragerea datelor de la un server sau procesarea lor, setarea anumitor informaţii. Acestea sunt echivalente cu metodele de gestiune a ciclului de viaţă al componentelor de tip clasă compontDidMount, componentDidUpdate şi componentWillUnmount.
3. **Hook-uri personalizate** care sunt funcţii JavaScript având o sintaxă specifică, numele funcţiei începând cu **use.**
4. **Hook-urile de context** care sunt folosite pentru a returna valoarea curentă unui anumit context. Acestea pot fi trimise într-o ierarhie de tip părinte-copil de componente.
5. **Hook-urile useCallback** folosite pentru a memora starea curentă a unei componente având ca scop ne-randarea componentelor copil atunci când nu este necesar.
6. **Hook-urile useMemo** care returnează valoare memorată a unei componente ajutând în optimizarea performanţelor.
7. **Hook-urile useRef** ce returnează o referinţă către un obiect prin proprietatea **current**, fiind mutabile. Scopul principal este de a accesa o componentă copil.
8. **Hook-urile useParams** ce returnează un obiect de tip cheie-valoare a parametrilor dinamici pentru URL-urile curente. Componentele copil moştenesc aceşti parametri de la componentele părinte [20], [21], [22].

De asemenea, mai există şi **componente stilizate** care permit utilizarea CSS pentru a stiliza componentele React. Acestea oferă următoarele funcţionalităţi:

- CSS automat prin monitorizarea componentelor randate într-o pagină şi injectarea stilului specific acestora şi nimic mai mult.
- Înlăturarea conflictelor date de numele claselor deoarece acestea generează nume unice pentru fiecare componentă stilizată, neexistând mai apoi duplicate.
- Stilizare simplă şi dinamică, adaptând stilul unei componente bazat pe proprietăţile acesteia.

- Mentenanţă uşor de întreţinut
- Prefixare CSS automată [23].

Utilitatea arborilor de decizie poate fi observată în diferite contexte cum ar fi clasificarea textului, extragerea textului, compararea statistică a datelor, acestea fiind doar câteva exemple. Domeniul aplicabilităţii acestora este unul vast şi variat, fiind o ustensilă foarte bună mai ales în domeniul medical, unde apare nevoia de a diagnostica unele boli, acest lucru având un aport considerabil în evoluţia medicinei şi tratarea bolilor.

Algoritmii bazaţi pe arbori de decizie oferă reguli de clasificare uşor de înţeles pentru oameni. În ciuda acestui fapt, există şi unele dezavantaje, cel mai mare fiind sortarea atributelor atunci când arborele trebuie să separe un nod. Acest lucru devine costisitor, având ca efect scăderea eficienţei atunci când datele folosite sunt de dimensiuni mari.

În 2001, Leo Breiman a prezentat ideea de Random Forest (Păduri aleatoare), dovedindu-se a înregistra performanţe mai bune decât alţi algoritmi de clasificare precum Reţelele Neuronale sau Maşini Vectoriale de Sprijin (Support Vector Machine - SVM). Acest algoritm are la bază conceptul de selecţie aleatoare de subseturi de date îmbinat cu folosirea de diferiţi clasificatori, lucru ce oferă diversitate, dovedindu-se a fi mult mai eficient.

Random Forest este format dintr-o familie de vectori identic distribuiţi aleator. Fiecare arbore este format din subseturi aleatorii de date sau caracteristici, selectate din întregul set de date. Acest concept poartă denumirea de Bagging şi are ca efect îmbunătăţirea predicţiilor, rezultând eficienţe ridicate.

Avantajele acestei abordări sunt:

- depăşirea problemei de supraadaptare în procesul de antrenare a reţelei
- oferă o modalitate eficientă de gestiune a datelor inconsistente
- oferă o acurateţe mai mare decât arborii de decizie simpli
- în fiecare arbore, subsetul de caracteristici este selectat aleatoriu la formarea nodului [24].

Arborii de decizie din Random Forest sunt formaţi din subseturi aleatorii de date din întregul set de date, având trei componente: nodurile de decizie, nodurile frunză sau terminale şi nodul rădăcină. Algoritmul arborilor de decizie împarte setul de date în ramuri care la rândul lor sunt separate în alte ramuri, continuând în această manieră până se ajunge la nodul frunză care nu mai poate fi împărţit. Nodurile din arborii de decizie reprezintă atributele sau caracteristicile folosite pentru a prezice rezultatul [25].

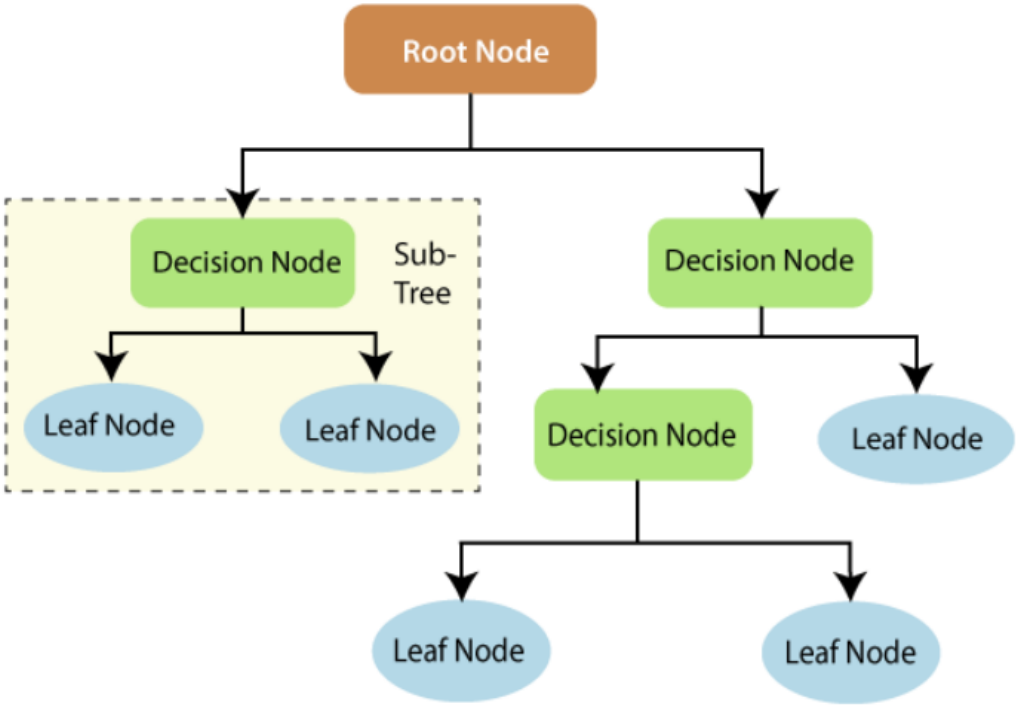

Fig. 2: Formarea arborilor de decizie

Pe baza unor metrici de calculare se poate măsura gradul de incertitudine în alegerea subseturilor de date, acest lucru putând fi redus. Cele mai cunoscute două metrici sunt:

- entropia
- indexul Gini

Entropia poate fi definită ca ordinul de dezordine, indicând ca şi selecţie optimă caracteristica cu valoarea cea mai mică. Aceasta se calculează ca o sumă de produse folosind probabilitatea elementelor de a aparţine unei clase, formula matematică fiind: [26].

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

Fig.3: Formula de calcul a entropiei

Indexul Gini reprezintă impuritatea, măsurând frecvenţa la care elementele alese în mod aleatoriu sunt etichetate greşit. Valoarea minimă a indexului Gini este 0, acesta fiind cel mai favorabil caz, atunci când toate elementele alese fac parte din clase distincte. Aşadar, acest nod nu va mai fi împărţit. Cu cât indexul Gini este mai mic, cu atât va creşte acurateţea. Acesta se calculează ca sumă între probabilităţile elementelor de a aparţine unei clase, având formula matematică egală cu: [27].

$$GiniIndex = 1 - \sum_j p_j^2$$

Fig.4: Formula de calcul a indexului Gini

MySQL este o bază de date dezvoltată de către Oracle, fiind o sursă deschisă, accesibilă folosirii de publicul larg. Aceasta este o bază de date relaţională, bazată pe limbajul structurat de interogare (Structured Query Language - SQL). Rulează pe platforme virtuale, cum ar fi Linux, UNIX sau Windows, fiind folosită mai ales în aplicaţii web, bazându-se pe un model client-server. Nucleul acesteia este serverul MySQL care gestionează toate comenzile sau instrucţiunile bazei de date. Este valabil ca un program separat într-o reţea de medii de lucru de tipul client-server online şi de asemenea ca o librărie care poate fi adăugată separat în aplicaţii.

Avantajele acestei baze de date sunt:

- permite stocarea şi accesarea datelor aflate în mai multe motoare de stocare precum InnoDB, CSV şi NDB
- replicarea şi partiţionarea tabelelor, oferind performanţă şi durabilitate
- suport pentru o gamă variată de date precum float, double, char, varchar, binary, varbinary, text, blob, date, time, datetime, timestamp, year, set, enum
- oferă securitate ridicată
- oferă suport pentru diferite protocoale de comunicaţie, cum ar fi TCP/IP pe diferite platforme [28].

MongoDB este o bază de date non-relaţională folosită pentru a stoca volume mari de date. Structura acesteia este formată din colecţii care conţin seturi de documente, ce sunt perechi de tip cheie-valoare. Oferă o flexibilitate mai mare spre deosebire de bazele de date relaţionale, avantajele acesteia fiind:

- documentele pot varia în ceea ce priveşte numărul de proprietăţi, mărime şi conţinut
- documentele nu trebuie să aibă o schemă predefinită, existând posibilitatea de a adăuga proprietăţi pe parcurs
- scalabilitate
- modelul de date pus la dispoziţie oferă posibilitatea de a crea ierarhii de relaţii pentru stocarea datelor complexe într-o manieră mai uşoară
- crearea indecşilor pentru îmbunătăţirea performanţelor căutărilor, fiecare proprietate putând fi indexată
- load balancing, concept ce se referă la împărțirea datelor către mai multe instanţe de MongoDB care pot rula pe diferite servere, aducând îmbunătăţiri la nivel de sistem şi totodată gestionând erorile hardware [29].

Proiecte similare

În urma căutării făcute, s-au găsit site-uri medicale care oferă o parte din funcționalitățile existente în acest proiect, cum ar fi: citirea rezultatelor după verificarea acestora de către un medic (funcționalitate oferită de Arcadia). În schimb, nu s-a găsit în nicio astfel de platformă funcționalitatea de ghidare a pacientului către o ramură medicală în urma introducerii simptomelor [9], [10], [11], [12].

2. Proiectarea aplicației

Arhitectura aplicației

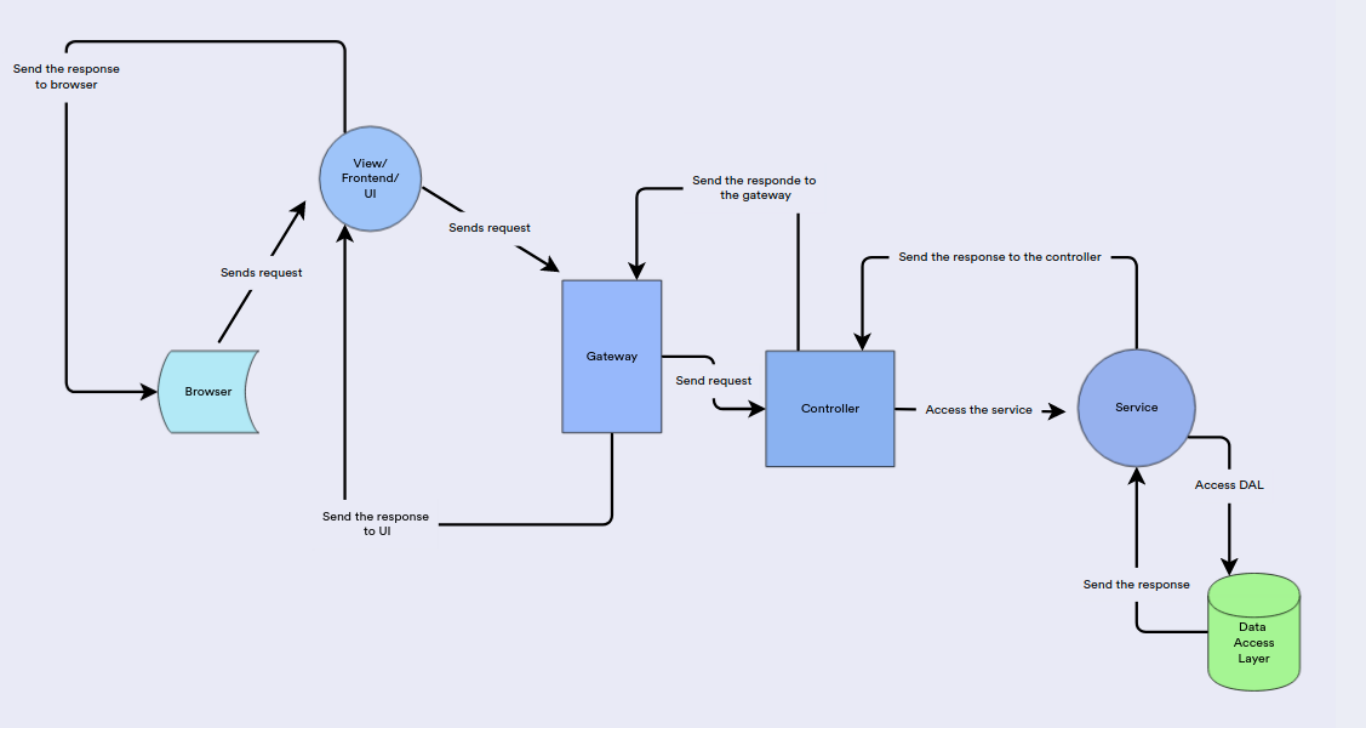

Fig 1: Arhitectura generală a aplicației

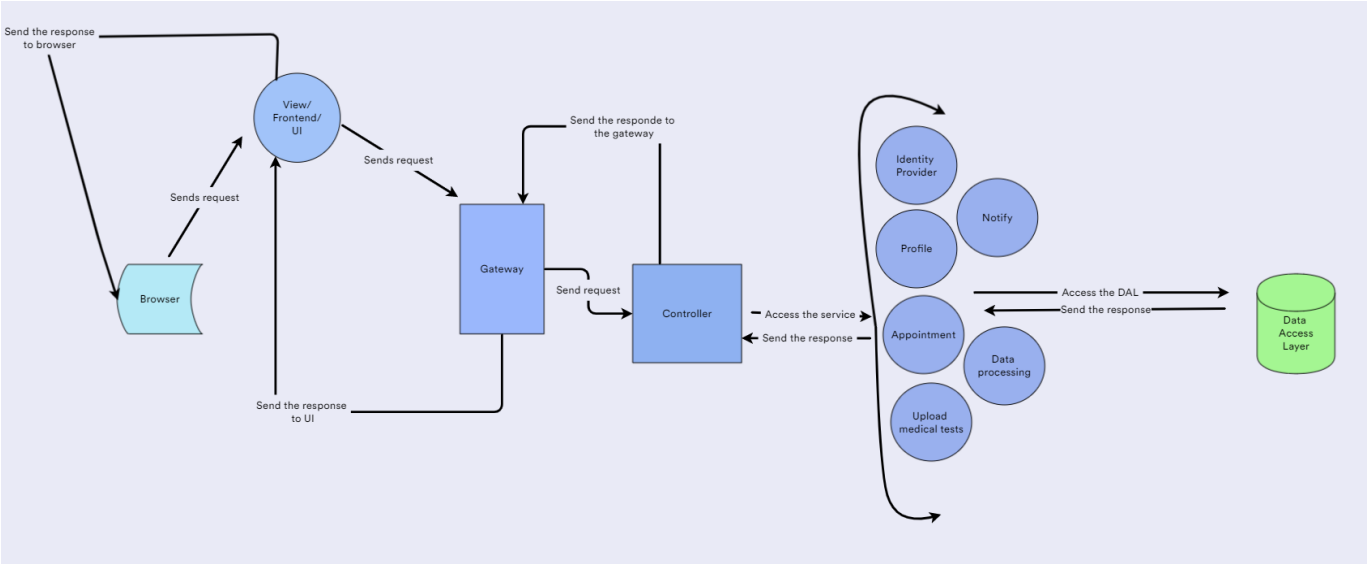

Fig 2: Arhitectura aplicaţiei cuprinzând serviciile exacte

Ca bază de date se va utiliza MySQL. Tabelele folosite sunt:

Table: doctori

Columns:

cuim	varchar(15) PK
nume	varchar(15)
prenume	varchar(15)
specializare	varchar(25)
sediu	varchar(25)
sala	varchar(10)

În tabela “doctori” sunt stocare informaţiile despre profilul doctorilor, cuprinzând datele de mai sus. Pentru a identifica în mod unic un doctor se foloseşte ca şi cheie primara câmpul cuim ce reprezintă codul unic de identificare al medicilor.

Table: boli

Columns:

id_boala	int AI PK
nume_boala	varchar(100)

Tabela “boli” cuprinde totalitatea bolilor din baza de date.

Table: boli_simptome

Columns:

id_boli_simptome	int AI PK
id_boala	int
id_simptom	int

Tabela “boli-simptome” este folosită pentru a face legătura între simptomele corespunzătoare unei anumite boli şi boala respectivă.

Table: simptome

Columns:

id_simptom	int AI PK
nume_simptom	varchar(45)

Tabela “simptome” cuprinde totalitatea simptomelor din baza de date.

Table: pacienti

Columns:

id	int AI PK
nume	varchar(45)
prenume	varchar(45)
cnp	varchar(13)
data_nastere	date
varsta	int
grupa_sange	varchar(5)
greutate	int
nr_telefon	varchar(10)
simptome	varchar(255)
specificatii	varchar(255)
inaltime	int

Tabela “pacienţi” cuprinde informaţiile referitoare la profilul pacienţilor. Pentru a identifica în mod unic pacienţii, aceştia au ca şi cheie primară un integer autoincremented.

Table: users_pacienti

Columns:

id	int AI PK
id_pacient	int
id_username	varchar(25)

Tabela “users-pacienţi“ este o tabelă de legătură dintre tabela “pacienţi” şi “users”, legând profilul unui pacient de contul asignat acestuia.

Table: users

Columns:

username	varchar(25) PK
email	varchar(255)
enabled	bit(1)
first_name	varchar(255)
last_name	varchar(255)
locked	bit(1)
password	varchar(255)
role	varchar(255)

Tabela “users” conţine informaţiile referitoare la conturile tuturor utilizatorilor. Cheia primară a acestei tabele este chiar username-ul utilizatorului deoarece acesta este unic.

Table: users_doctori

Columns:

id	int AI PK
id_cuim	varchar(15)
id_username	varchar(25)

Tabela “users-doctori” este o tabelă de legătură dintre tabela “doctori” şi “users”, legând profilul unui doctor de contul asignat acestuia.

Table: specializari

Columns:

id_specializare	int AI PK
nume	varchar(45)

Tabela “specializări” cuprinde totalitatea specializărilor existente în baza de date.

Comunicarea serviciilor:

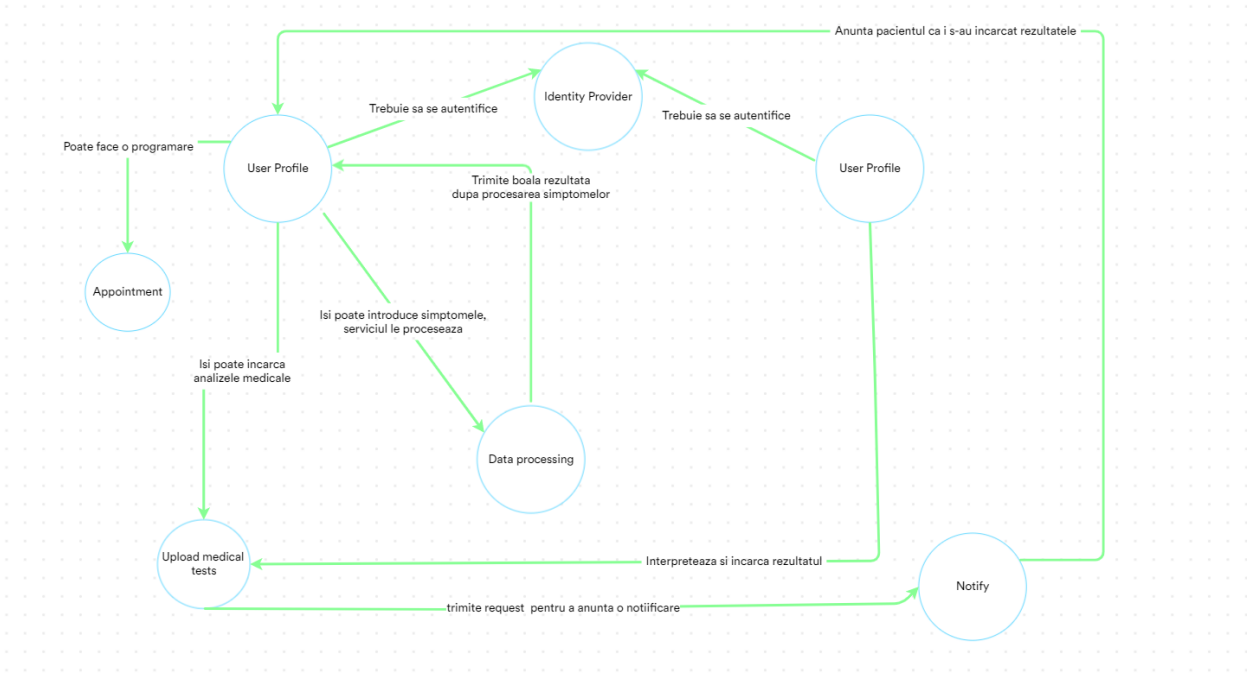

Fig 3: Diagrama de flow a serviciilor

Identity Provider este serviciul care are ca scop creare conturilor utilizatorilor aplicaţiei. Acesta conţine toată logica securității aplicației: configurarea web, filtrele de autentificare și autorizare, crearea de JWT-uri, decodificarea acestora și transmiterea răspunsurilor corespunzătoare. Este primul serviciu cu care utilizatorul interacționează, fie logându-se, fie creându-și un cont. Implementarea acestuia a fost făcută în Java folosind Spring-Boot, iar securitatea a fost făcută folosind Spring Security.

User Profile este serviciul care are ca scop crearea unui profil atât pentru doctor, cât și pentru pacienți. De asemenea în acest serviciu se află logica pentru operațiile pe care userii le pot face, cum ar fi: adăugarea în baza de date de către doctor sau admin

de noi boli, simptome sau specializări. Pentru a menține securitatea aplicației, acest serviciu comunică cu **Identity Provider**, verificările fiind făcute asupra rolului pe care utilizatorul îl are, pentru acest lucru făcându-se requesturi pentru a decodifica JWT-ul. Implementarea acestui serviciu s-a făcut folosind Java.

Data Processing este serviciul care are ca scop procesarea simptomelor introduse de către orice utilizator, este un serviciu ce poate fi accesat de oricine, nu este condiţionat de a avea cont. Pentru prelucrarea datelor a fost folosit algoritmul de Machine Learning Random Forest bazat pe indicele Gini minim. Motivul pentru care a fost ales acest algoritm este acuratețea ridicată în raport cu alți algoritmi, cum ar fi Rețelele Neuronale sau Rețele Bayesiene. Implementarea acestui serviciu a fost făcută în Python, utilizând framework-ul FastAPI.

Urmează implementarea următoarelor servicii: **Notification, Upload medical data** și **Appointment**. Notification va fi serviciul care va trimite notificări doctorului atunci când pacientul încarcă date medicale pe care acesta trebuie să le vadă, respectiv notificări pacientului atunci când doctorul publică un răspuns. Upload medical data va fi serviciul utilizat pentru a încărca date medicale sub format de imagine sau text și stocarea acestora într-o bază de date. Ca bază de date se va folosi MongoDB și platforma Firebase. Appointment va fi serviciul utilizat pentru a putea face o programare. Implementarea tuturor serviciilor va fi făcută în Java, utilizând Spring-Boot.

Dificultăți întâmpinate:

Majoritatea dificultăţilor întâmpinate pe parcurs au fost legate de securitate. Preponderente au fost problemele legate de CORS policy. Din cauza următoarei linii de cod din configurarea web, majoritatea requesturilor au fost blocate, având ca rezultat eroarea “403- forbidden”.

```
http.cors().configurationSource(
     request->newCorsConfiguration()
          .applyPermitDefaultValues());
```

Soluția a fost comentarea acesteia și dezactivarea corsului în configurarea web.

O altă dificultate întâmpinată a fost oprirea într-un while infinit în cadrul algoritmului Random Forest iar astfel nu era primit niciun rezultat. Acest lucru s-a întâmplat din cauza introducerii unui set de simptome inexistent în datele de antrenare.

Ca și soluție a fost aleasă introducerea unei variabile ce indică numărul maxim de iterații.

Listă specificații funcționale:

➢ pacientul își poate introduce simptomele pentru a vedea către ce specializare trebuie să se îndrepte

➢ pacientul poate încărca poze cu analize medicale

➢ doctorii au acces la datele încărcate de pacienți și le interpretează

➢ pacienții pot vizualiza interpretarea medicului după ce acesta a publicat rezultatele

➢ pacienții pot face o programare medicala

➢ pacienții sunt notificați atunci când s-au încărcat rezultatele

➢ pacienții/doctorii se pot autentifica

Listă specificații nonfuncționale:

➢ aplicația este securizată prin folosirea JWT-urilor

➢ procesarea simptomelor poate dura câteva secunde

Concluzii:

Tema aleasă este una interesantă, abordând o implementare bazată pe servicii combinată cu utilizarea inteligenței artificiale pentru procesarea simptomelor în vederea stabilirii unei ramuri medicale spre care pacientul ar trebui să se îndrepte pentru un consult.

Bibliografie:

Documentație tehnică:
[1].https://docs.oracle.com/javaee/6/firstcup/doc/gkhoy.html, ultima vizitare martie, 2022
[2]. https://www.ibm.com/cloud/learn/java-spring-boot, ultima vizitare martie, 2022
[3]. https://spring.io/projects/spring-boot, ultima vizitare martie, 2022
[4].https://www.youtube.com/watch?v=zvLZLFWrbAA&t=503s&ab_channel=SpringDeveloper, ultima vizitare martie, 2022
[5]. https://docs.spring.io/spring-framework/docs/current/reference/html/overview.html, ultima vizitare martie, 2022
[6].https://medium.com/java-epic/which-web-server-is-used-by-spring-boot-4e024106923f, ultima vizitare martie, 2022

[7].https://docs.spring.io/spring-boot/docs/2.0.9.RELEASE/reference/html/howto-embedded-web-servers.html, ultima vizitare aprilie, 2022
[8]. https://reactjs.org/, ultima vizitare martie, 2022
[12]. https://spring.io/projects/spring-security, ultima vizitare iunie, 2022
[13].https://docs.spring.io/spring-security/site/docs/3.0.x/reference/security-filter-chain.html, ultima vizitare iunie, 2022
[14].https://docs.spring.io/spring-security/site/docs/3.0.x/apidocs/org/springframework/security/core/Authentication.html, ultima vizitare iunie, 2022
[15].https://docs.spring.io/spring-security/reference/servlet/authentication/architecture.html#servlet-authentication-authentication, ultima vizitare iunie, 2022
[16].https://medium.com/@haytambenayed/how-does-spring-security-work-internally-525d359d7af, ultima vizitare iunie, 2022
[17].https://docs.spring.io/spring-security/reference/servlet/authentication/passwords/form.html, ultima vizitare iunie, 2022
[18]. https://fastapi.tiangolo.com/ , ultima vizitare iunie, 2022
[19]. https://levelup.gitconnected.com/types-of-react-components-a38ce18e35ab, ultima vizitare iunie, 2022
[20]. https://reactrouter.com/docs/en/v6/hooks/use-params, ultima vizitare iunie, 2022
[21]. https://www.freecodecamp.org/news/react-hooks-fundamentals/, ultima vizitare iunie, 2022
[22]. https://www.javatpoint.com/react-hooks , ultima vizitare iunie, 2022
[23]. https://styled-components.com/docs/basics , ultima vizitare iunie, 2022
[24].https://www.researchgate.net/publication/259235118_Random_Forests_and_Decision_Trees , ultima vizitare iunie, 2022
[25].https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/ , ultima vizitare iunie, 2022
[26].https://towardsdatascience.com/entropy-how-decision-trees-make-decisions-2946b9c18c8 , ultima vizitare iunie, 2022
[27].https://quantdare.com/decision-trees-gini-vs-entropy/ , ultima vizitare iunie, 2022
[28].https://www.techtarget.com/searchoracle/definition/MySQL, ultima vizitare iunie, 2022
[29].https://www.guru99.com/what-is-mongodb.html , ultima vizitare iunie, 2022

Căutări făcute pe tema licenței:

[9]. https://www.reginamaria.ro/clinici/campus-medical-iasi, ultima vizitare aprilie, 2022
[10]. https://www.medlife.ro/, ultima vizitare martie, 2022
[11]. https://www.arcadiamedical.ro, ultima vizitare aprilie, 2022
[12]. https://www.zenmedical.ro, ultima vizitare martie, 2022