UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAȘI

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

WELLNESSWAVE - PLATFORMĂ PENTRU DIGITALIZAREA SISTEMULUI PUBLIC MEDICAL

propusă de

Bălteanu Andrei

Sesiunea: *Iulie, 2023*

Coordonator științific

Asist. Dr. Scutelnicu Liviu-Andrei

UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

WELLNESSWAVE - PLATFORMĂ PENTRU DIGITALIZAREA SISTEMULUI PUBLIC MEDICAL

Bălteanu Andrei

Sesiunea: *Iulie, 2023*

Coordonator științific

Asist. Dr. Scutelnicu Liviu-Andrei

Cuprins

1. Introducere

1.1. Motivații

Tehnologia a evoluat în ultimul deceniu cu o viteză incredibilă, schimbând în bine multe aspecte ale vieții noastre de zi cu zi, de la mașini autonome la electrocasnice inteligente și până la medici ce pot opera de la distanță. Printre beneficiile aduse de dezvoltarea tehnologiilor se regăsește și digitalizarea unor servicii sau obiecte ce sunt folosite în viața noastră cotidiană.

Subiectul principal al lucrării de față este bazat pe îmbunătățirea funcționării defectuoase a sistemului informatic actual din spitale și cum acesta a rămas în urmă din punct de vedere al standardelor secolului actual și al așteptărilor cetățenilor dintr-o țară europeană.

Am ales acest domeniu deoarece am avut multiple experiențe cu sistemul de înregistrare a pacienților dintr-un spital de stat și am fost profund dezamăgit, văzând cum birocrația și funcționarea defectuoasă a sistemului curent încetinește considerabil procesul de tratarea a pacienților, cât și numărul de pacienți tratați într-o zi.

Această funcționare deficitară face ca atenția și concentrarea personalului medical să nu se axeze pe tratarea pacienților ci pe încercarea de folosire a sistemului, într-un mod în care să îl facă utilizabil. Pe lângă aceste probleme, se mai adaugă și faptul că există frecvent momente când aceste sisteme din spitale pur și simplu nu funcționează, momente în care atât pacienții cât și medicii nu pot face nimic altceva decât să aștepte [2].

De-a lungul experiențelor personale avute cu acest serviciu public, s-au remarcat următoarele incidente, cu repetitivitate:

- **Programări ce nu se respectă sau sunt anulate**: o persoană care este programată la ora 11:00 pentru o consultație este chemată de către medicul curant începând cu ora 08:00, pentru că, deși există un sistem de programări, acesta nu se respectă niciodată din multiple motive cum ar fi sisteme nefuncționale, pene de curent, timpi extinși a unor consultații. Astfel, se pierd avantajele unei sistem de programări precum ora exactă a consultației, planificarea zilei de lucru a medicului, timp mai puțin irosit de către pacient, existența unei ordini și a unei discipline, evitarea aglomerării spitalului;
- **Birocrație anevoioasă**: pentru ca pacientul să reușească să obțină consultația efectivă, acesta trebuie să urmeze anumiți pași, care presupun următoarele:
    - pacientul se prezintă cu biletul de trimitere de la medicul de familie, în ziua consultației la medicul specialist din unitatea sanitară. Dacă medicul recomandă

anumite analize ce trebuiesc efectuate în aceeași zi la spital, acesta listează documentele în care specifică ce analize trebuie recoltate. Acesta este primul timp de așteptare al pacientului, el fiind nevoit să aștepte la coadă pentru a fi consultat, respectiv pentru a primi cererea de efectuare a analizelor;

- după ce a primit documentele referitoare la analize, acestea trebuiesc prezentate la un birou de registratură, aferent serviciului care se ocupă cu înregistrarea acestora. Acesta este un al doilea timp de așteptare;
- după ce datele pacientului sunt prelucrate MANUAL, este listat un alt document ce conține toate aceste date, care va trebui prezentat la centrul de recoltare;
- odată ajuns la centru de recoltare, se preiau analizele cerute de medic. Intervine un alt timp de așteptare (randul de persoane care așteaptă recoltarea analizelor, neexistând un sistem de programări);
- după ce se recoltează analizele, pacientul revine la medicul curant la care a avut consultația, pentru a o finaliza, discutând detaliile finale, în cele mai multe cazuri, în lipsa rezultatelor analizelor. Revenind la cabinetul medicului, intervine un alt timp de așteptare (o ultimă coadă de așteptare, deoarece există și alți pacienți care așteaptă să intre pentru consultație sau discuții finale).

În toți acești pași se folosește sistemul actual, care este anevoios, atât din punct de vedere al timpului pierdut pentru obținerea documentelor listate și transmiterea lor fizică, cât și din punct de vedere al procesării manuale ale acestora. De foarte multe ori, acest sistem nu funcționează deci nu duce la digitalizare în adevăratul sens al cuvântului.

- **Documente anterioare:** deoarece medicina are ramuri ce se întrepătrund, este necesar ca, atunci când un pacient se prezintă la un consult medical de specialitate, să prezinte toate documentele de la consultațiile, tratamentele și intervențiile anterioare, de la diferite specialități medicale. Astfel, nevoia de a prezenta toate documentele pentru un consult nou, la un medic având o specialitate diferită, poate duce la probleme precum: degradarea sau pierderea documentelor, uitarea acestora acasă înaintea consultației etc.

Având în vedere aceste probleme care apar frecvent în procesul de tratare a pacienților, Ministerul Sănătății[1] a propus introducerea cardului de sănătate. Emiterea acestui card nu a decurs conform așteptărilor, deoarece la momentul actual cardul fiind folosit doar pentru verificarea dacă posesorul acestuia figurează ca asigurat medical sau nu. În ciuda faptului că

[1] https://www.ms.ro/

introducerea cardului de sănătate ar fi putut aduce toate beneficiile prezentate în continuare, din motive necunoscute publicului larg, nu s-a implementat cu succes un astfel de sistem.

Prin intermediul lucrării de față, propunem un sistem ce are rolul de a rezolva o parte din problemele enumerate anterior și totodată eliminarea timpilor de așteptare la cozile interminabile, aducând un plus de valoare consultației, medicul axandu-se pe problema pacientului și nu pe completarea și pasare de formulare de la un departament la altul. Scopul pus la dispoziție de această lucrare a fost gândit pentru funcționarea unui singur spital, însă acest sistem are potențial de implementare pe scară largă.

1.2. Obiective

Având ca șablon problemele expuse în subcapitolul precedent, pentru a rezolva problemele sau măcar o parte din acestea descrise în subcapitolul anterior, a fost aleasă dezvoltarea unei aplicații web, capabilă să asigure buna funcționare a unui spital, din punct de vedere al programărilor, consultațiilor și al documentelor partajate. Prezentarea amănunțită a implementării alese se regăsește în Capitolul 3.

O bună funcționare a unui spital duce la efecte benefice atât asupra pacienților, cât și a personalului angajat. Urmărind generarea acestor efecte, aplicația aduce următoarele funcționalități:

Acces la documente online:

După crearea unui cont, utilizatorul, în cazul de față fiind vorba despre pacient, poate încărca documentele în format electronic (PDF) pe platforma web (consultații anterioare, scheme de tratament etc.). Apoi, în timpul unei consultații, medicul înrolează pacientul, astfel având acces la fișierele acestuia.

Cum am prezentat anterior, acesta ar aduce beneficii ambelor părți. Pacienții, pe de o parte, au încredere că documentelor lor sunt toate în siguranță într-un singur loc și anume într-o bază de date securizată, iar pe de altă parte, medicii au acces facil la documente și la posibilitatea organizării zilei de lucru.

Înregistrare inteligentă:

Conform Romania: Health System Review [4], un lucru frecvent întâlnit în spitalele din România sunt cozile și listele lungi de așteptare. Această problemă este datorată faptului că este vorba despre un volum foarte mare de pacienți care doresc să se înregistreze și prea puțini care realizează acest lucru. Astfel se pierde timpul atât din punct de vedere al pacientului, așteptându-și rândul la coadă, cât și de către medici, așteptând pacienții să se înregistreze pentru a putea începe consultația.

Pentru a elimina această problemă, propunem implementarea unui serviciu care, prin încărcarea în sistem a unei simple fotografii a cărții de identitate, extrage informațiile necesare înregistrării (aceste informații sunt numele, prenumele, CNP-ul, județul, vârsta și sexul). În cazul în care algoritmul de recunoaștere dă greș, interfața aplicației, dă posibilitatea utilizatorului să corecteze greșelile eventual apărute.

Programări online:

Pentru a exista o ordine și un timp real de așteptare, am avut în vedere în implementare, dezvoltarea unui sistem de programări online. Pacienții ce doresc să creeze o programare, vor alege perioada calendaristică în care ar dori să aibă loc consultația (ziua și intervalul orar). Locurile disponibile aferente programării (ziua și intervalul orar) vor fi afișate împreună cu medicii care sunt disponibili pentru consultație. După ce toate aceste informații au fost selectate, programarea este înregistrată în sistem și urmată de o confirmare care va fi generată și trimisă pe e-mail pacientului.

Utilizatorii, în cazul de față fiind vorba despre pacienți, pot anula o programare, accesând lista de programări a acestora, și selectând-o pe cea pe care nu o mai doresc.

Generarea programului de lucru al medicilor

Pentru a eficientiza și pentru a testa platforma, am luat în considerare implementarea unui serviciu de generare a programului de lucru. Fiecare specializare va avea alocați doi medici în fiecare zi. Toți medicii sunt adăugați la începutul lunii într-o coadă în ordine alfabetică. Din această coadă se vor extrage pentru fiecare zi câte doi medici, după care se vor adăuga din nou la coadă pentru următoarea lor zi de consultații. În felul acesta se generează programul medicilor asigurându-se că fiecare dintre aceștia au asignate cel mult 3 zile de consultații în decurs de o săptămână.

Această implementare este necesară pentru sistemul de programări deoarece, pentru a avea la dispoziție intervalele orare ce definesc disponibilitatea medicilor, într-o primă etapă, trebuie să existe programul acestora de lucru.

Toate aceste transformări ale sistemului actual către un sistem informatic funcțional va fi un beneficiu atât pentru medici (câștigând timp prețios pentru a planifica ziua de lucru cât și pentru a trata pacienții într-un mod eficient și rapid) dar și pentru pacienți (deoarece aceștia nu mai irosesc timp așteptând la cozile interminabile pentru înregistrare, programare sau căutarea documentelor cerute de către medic în teancul de documente cu care s-ar prezenta fizic la consultație).

2. Stadiul cercetărilor în domeniu

În ultimii ani s-au realizat mai multe studii despre digitalizarea serviciilor medicale din România, precum "Studiu privind soluțiile digitale folosite în unitățile medicale private" [1]. Acesta evidențiază faptul că transformarea sistemului medical a devenit în zilele noastre o nevoie stringentă. Această nevoie este datorată și faptului că trăim în "secolul vitezei", iar oamenii așteaptă de la administrația publică un serviciu rapid, de calitate și eficient, care să le rezolve rapid problemele.

În plus, se vorbește despre faptul că transformarea nu constă doar în transpunerea actului medical ci și într-un sistem informatic care oferă mai multe beneficii printre care enumerăm câteva dintre cele mai importante:

- **încărcarea și transmiterea informațiilor de interes general în format digital:** pe perioada pandemiei de Sars-Cov2, recomandările pentru sporirea protecției împotriva virusului s-au transmis atât prin intermediul media (televiziune, radio etc.), cât și prin atenționarea telefonică cu ajutorul sistemului *Ro-Alert*[2] ("permite difuzarea de mesaje de tip Cell Broadcast [9] pentru avertizarea și alarmarea populației în situații de urgență, conform prevederilor legale"). Acest sistem a facilitat transmiterea informațiilor fără a exista pericolul apropierii fizice;
- **încurajarea prevenției în detrimentul tratării:** Țările mai dezvoltate din punct de vedere economic, educațional și sanitar (precum Olanda și Danemarca) au început să pună mai mult accent pe prevenție decât pe tratarea bolilor. Această prevenție ajută la întârzierea simptomelor sau chiar la evitarea în totalitate a bolii, astfel îmbunătățind calitatea vieții și reducând costurile asistenței medicale. Prevenția are ca scop identificarea și eliminarea factorilor de risc ce pot declanșa una sau mai multe boli. Ministerul Sanatatii[3], SRMP[4] (Societatea Română de Medicină Preventivă), ARSPMS[5] (Asociația Română de Sănătate Publică și Management Sanitar), CPSS[6] (Centrul pentru Politici și Servicii de Sănătate) sunt doar câteva instituții care au implementat platforme ce promovează prevenția în detrimentul tratării.

[2] https://ro-alert.ro/
[3] https://www.ms.ro/ro/
[4] https://srmp.ro/
[5] https://arspms.ro/
[6] https://cpss.ro/ro/index.php

Un alt studiu care oferă o perspectivă importantă asupra nivelului de digitalizare a instituțiilor românești este "Digitalizarea administrației publice din România - între nevoile și aspirațiile unei societăți moderne a secolului XXI" [5]. Acest studiu evaluează la nivel european, în baza indicelui economiei și societății digitale (DESI[7]) stadiul informatizării societății românești în spațiul administrativ național. Totodată, articolul analizează cauzele ce împiedică dezvoltarea și folosirea unui sistem informatic adecvat nevoilor unei societăți moderne dar și eventual soluții care ar putea genera beneficii reale pentru cetățeni.

Se pare că în ultimii doi ani România a avut o "performanță" ieșită din comun, clasându-se consecutiv pe ultimul loc în *clasamentul*[8] realizat de UE în funcție de indicele DESI. Cele mai avansate țări în acest clasament sunt Finlanda, Suedia, Danemarca și Olanda. Încep totuși să se ia măsuri pentru creșterea nivelului de digitalizare din țară, precum introducerea cărților de identitate cu cip, introducerea platformei *Ghișeul.ro*[9] pentru declarații și obligații fiscale, existența unui card de sănătate etc. Cu toate acestea, serviciile românești lasă de dorit în comparație cu fruntașele Europei și cererea cetățenilor.

Concluziile acestui studiu sunt comune: modernizarea țării din punct de vedere informatic este absolut necesară, beneficiile digitalizării nu pot fi ignorate iar România ar putea lua lecții importante legat de bunele practici ale statelor europene, ceea ce ar duce și la succesul procesului de modernizarea al țării.

Pe tema digitalizării serviciilor publice din România mai există studii recente, precum *The degree of digitalization of public services in Romania* [6]. Acest studiu scoate în evidență fiecare aspect analizat în cadrul determinării indicelui DESI, evidențiind totodată principalele cauze care duc la o dezvoltare precară a serviciilor publice în format electronic. Printre acestea se regăsesc:

- lipsa unei arhitecturi IT eficiente și eficace;
- lipsa sistemelor informatice necesare instituțiilor publice centrale pentru operare online;
- insuficiența guvernării electronice și a specialiștilor în departamentele IT ale instituțiilor publice și a autorităților;
- lipsa unei legislații adecvate și a unor proceduri bine definite pentru a dezvolta și menține un sistem care să acopere nevoile cetățenilor în materie de instituții publice.

[7] https://digital-strategy.ec.europa.eu/ro/policies/desi
[8] https://digital-strategy.ec.europa.eu/en/policies/desi-romania
[9] https://www.ghiseul.ro/ghiseul/public/

Concluzii au fost similar celor menționate din studiile anterioare, România are potențial (industria IT fiind în continuă creștere în ultimii ani) însă guvernarea trebuie să acorde mai multă importanță și atenție modernizării țării din punct de vedere informatic, alocând timpul, banii și resursele umane necesare.

3. Arhitectura aplicației

În cadrul acestui capitol, se prezintă platforma web, numită în continuare *WelnessWave*, care face obiectul lucrării de față, ce își propune să implementeze un set de soluții pentru anumite probleme ce se întâlnesc în viața de zi cu zi a pacienților ce accesează serviciile medicale publice.

Platforma se bazează pe un proiect alcătuit din mai multe subsisteme: backend, frontend și o bază de date. Componenta backend este implementată folosind limbajul de programare Java[10], care este un limbaj de programare open-source ce oferă posibilitatea rulării sistemului pe orice tip de sistem de operare. Baza de date este relațională de tip MySQL[11] iar componenta frontend folosește atât limbajul clasic HTML[12] cât și componente JavaScript[13] și CSS[14], pentru a creea o interfață prietenoasă, practică și intuitivă utilizatorilor.

Arhitectura aplicației este orientată pe servicii și oferă numeroase funcționalități ce vor fi descrise în cadrul acestui capitol.

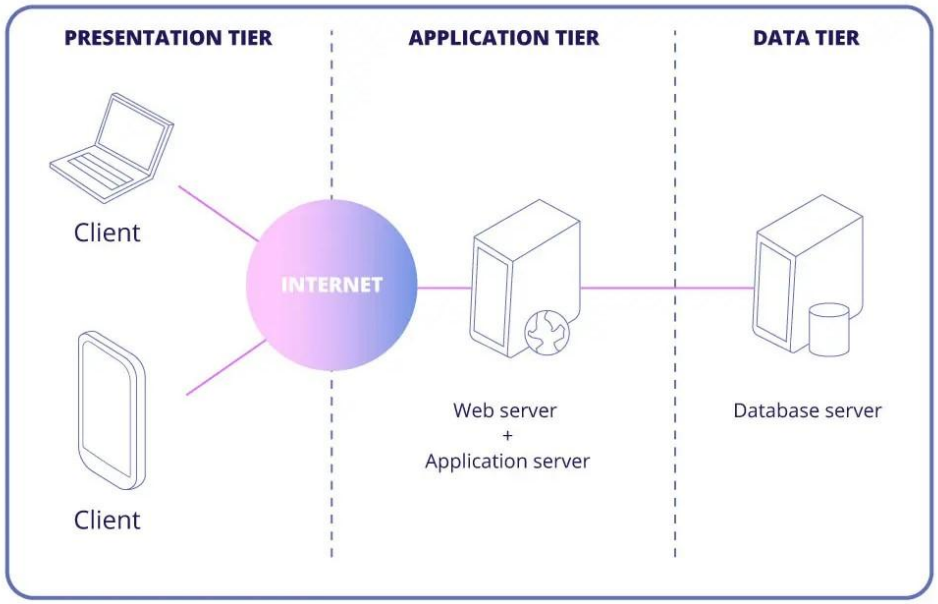

Figura 1. Arhitectura aplicației

[10] https://www.java.com/en/
[11] https://www.mysql.com/
[12] https://html.com/
[13] https://javascript.info/
[14] https://www.w3schools.com/css/

Componentele principale ale aplicației, după cum se poate observa din Figura 1, sunt următoarele: aplicația back-end (Application Tier), baza de date (Data Tier) și aplicația front-end (Presentation Tier).

3.1. Componenta baza de date

Aplicația WellnessWave are ca prim avantaj posibilitatea de stocare a informațiilor din cadrul unei unități medicale în format digital, informații referitoare la pacienți, istoricul medical al acestora și activitatea spitalului. Pentru a putea salva informațiile, este nevoie de o formă de stocare special concepută pentru acest lucru. Astfel, au fost concepute bazele de date. Câteva dintre avantajele folosirii bazelor de date sunt:

- permit organizarea și structurarea informațiilor, sub formă de tabele, într-un mod eficient și sigur, ceea ce face mai ușoară accesarea și gestionarea lor;
- asigură realizarea copiilor de rezervă ale datelor, ceea ce previne pierderea lor;
- accesarea și manipularea datelor de dimensiuni foarte mari într-un timp foarte scurt.

O bază de date relațională este preferată în cazul unei aplicații pentru un spital, deoarece aceasta oferă o structură bine definită și organizată pentru date. Într-o bază de date relațională, datele sunt organizate în tabele, iar relațiile dintre acestea sunt definite prin chei primare și străine.

Această structură permite o gestionare eficientă și mai precisă a datelor, precum și o interogare mai ușoară și mai rapidă a acestora. De asemenea, o bază de date relațională oferă mecanisme de securitate și de integritate a datelor mai avansate, ceea ce este important în cazul datelor medicale, pe care le putem categorisi ca făcând parte din suita datelor cu caracter sensibil.

Pentru baza de date a aplicației a fost ales un server de tipul MySQL din următoarele motive:

- este o bază de date de tip open-source, ceea ce înseamnă că este gratuită și poate fi utilizată fără costuri de licență;
- este scalabilă, ceea ce înseamnă că poate gestiona creșterea volumului de utilizatori și a traficului de date; această caracteristică face ca baza de date aleasă să fie potrivită pentru aplicații cu cerințe ridicate de performanță;

- oferă mecanisme de securitate avansate, cum ar fi autentificarea și autorizarea, criptarea și auditarea datelor, ceea ce o face potrivită pentru aplicații care gestionează date cu caracter sensibil.

Deoarece avem de-a face, atât din punct de vedere al pacienților, cât și din punct de vedere al personalului medical, cu date cu caracter sensibil, acestea implică o importanță deosebită a securității în cadrul aplicației. Pentru a ne alinia la normele prevăzute de către Uniunea Europeană, conform GDPR [10] impuse din anul 2018 pentru îmbunătățirea securității informațiilor, acestea sunt criptate în cadrul aplicației de pe serverul de back-end și apoi salvate în baza de date. Mai multe detalii despre acest subiect vor fi tratate și prezentate în **Capitolul 4**.

Fiind vorba despre o aplicație care vine în ajutorul sistemului informatic al unei unități medicale, tabelele sunt absolut necesare. Acestea stochează informații despre pacienți, medici, specializările acestora, programările din trecut și viitor. Mai multe detalii despre modul de relaționare dintre tabele se regăsesc în Figura 2.

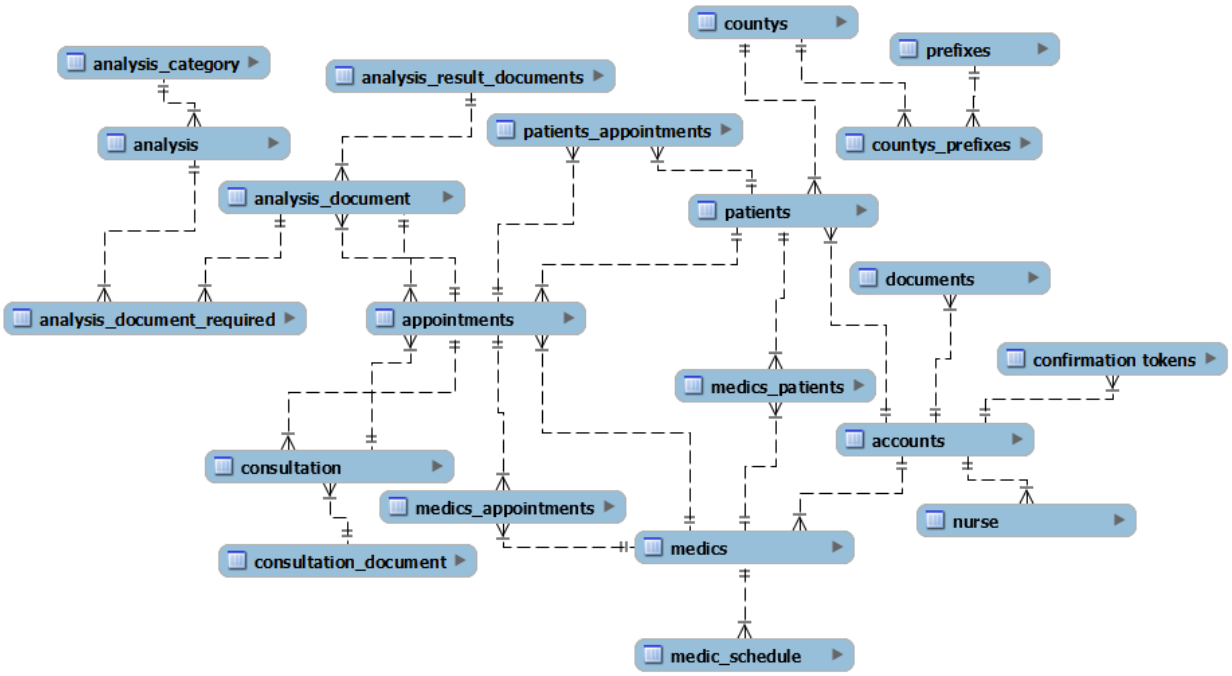

Figura 2. Relațiile dintre tabelele folosite de aplicația back-end

O tabelă de o mare importanță este cea care face referire la programări (*appointments*). O programare este punctul de legătură dintre mai multe componente ale aplicației. Fiecare dintre acestea are la rândul ei un tabel asociat în baza de date, spre exemplu tabelul *medics* ține evidența tuturor medicilor angajați ai spitalului care folosește aplicația și fiecare din aceștia poate

fi asociat unei înregistrări din tabelul *appointments*. Astfel, tabelul *appointments* conține câte o cheie străină pentru fiecare componentă care ia parte la programare. Aceste componente sunt:

- Medici;
- Pacienți;
- Documentul consultației;
- Documentul analizelor.

Toate aceste componente ale tabelului appointments se regăsesc în Figura 3, de mai jos:

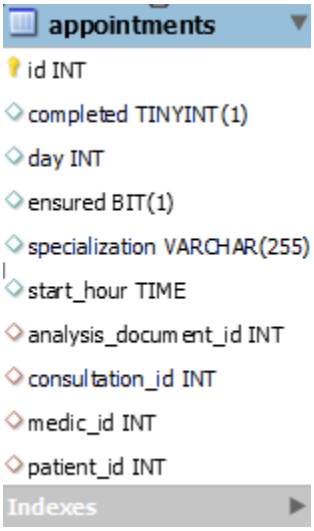

Figura 3. Structura tabelului *appointments*

În cadrul tabelului ilustrat în Figura 3 avem următoarele componente, după cum urmează:

- *id*: reprezintă identificatorul unic al programării;
- *completed*: reprezintă statutul programării (încheiată ⇔ 1, altfel ⇔ 0);
- *day* : reprezintă ziua din lună în care a avut loc programarea;
- *ensured*: reprezintă statutul de asigurat al pacientului (adevărat ⇔ asigurat, fals ⇔ altfel);
- *specialization*: reprezintă specializarea medicului;
- *start_hour:* reprezintă ora la care are loc programarea;
- *analysis_document_id*: câmpul este o cheie străină ce reprezintă identificatorul unic al documentului cu rezultatele analizelor cerute de medic;
- *consultation_id*: câmpul este o cheie străină, care face legătura cu tabelul *consultations,* ce reprezintă identificatorul unic al documentului rezultat în urma consultației; tabelul

consultations stochează documentul rezultat în urma consultației, împreună cu identificatorul programării asociate;

- *medic_id*: câmpul este o cheie străină, care face legătura cu tabelul medics, reprezintă identificatorul unic al medicului. Tabelul *medics* stochează toate informațiile necesare creării contului de tip medic;
- *patient_id*: campul este o cheie străină, care face legătură cu tabelul *patients,* reprezintă identificatorul unic la pacientului. Tabelul *patients* stochează toate informațiile necesare creării contului de tip pacient.

Această bază de date, cu toate tabelele și relațiile dintre ele, are aplicabilitate pentru un singur spital public. Inspirația cu privire la această alegere este Spitalul Clinic Județean de Urgență "Sf. Spiridon" din Iași[15], în cadrul căruia a fost confirmată (prin răspunsuri la întrebări adresate personal către unii angajați) necesitatea unei aplicații similare celei din lucrarea de față. Pentru o aplicație la nivel mai mare, arhitectura bazei de date va avea nevoie de îmbunătățiri, precum adăugarea de noi tabele și eficientizarea traficului de date.

3.2. Componenta back-end

O platformă web potrivită pentru un sistem informatic folosită într-un spital, trebuie să fie de încredere, robustă și ușor de îmbunătățit. Fiind vorba de o instituție publică atât de importantă, unde este vorba de vieți omenești, componenta de procesare și prelucrare a datelor, informațiilor și anume cea de server, unde se realizeaza toate instrucțiunile, nu poate să aibă momente în care să nu funcționeze, indiferent de durata acestora.

Deoarece sistemul actual cedează zilnic (conform surselor[16] și relatărilor cetățenilor), serverul platformei WellnessWave a fost implementat într-un limbaj de programare capabil să îndeplinească toate cerințele prezentate în paragraful anterior. Astfel, pentru a realiza această componentă, a fost ales limbajul de programare **Java**, care este un limbaj open-source, ce oferă beneficii printre care simplitate, robustețe, portabilitate[17].

Componenta de back-end se ocupă de prelucrarea informațiilor primite de la utilizator prin intermediul interfeței cu utilizatorul, dar și de criptarea și trimiterea informațiilor către baza de date. Astfel, această componentă comunică direct atât cu partea de front-end, cât și cu baza de date.

[15] https://www.spitalspiridon.ro/
[16] https://www.zf.ro/eveniment/cnas-despre-problemele-din-ultimele-zile-sistemul-informatic-din-21674294
[17] https://profs.info.uaic.ro/~acf/java/Cristian_Frasinaru-Curs_practic_de_Java.pdf

Un aspect foarte important de precizat în cadrul componentei de back-end este folosirea, pentru automatizarea procesului de înregistrare a unui utilizator, a tehnologiei de OCR-izare (detaliat în **Capitolul 5**) pentru completarea automată a datelor personale folosind ca date de intrare o imagine ce este reprezentată de cartea de identitate. Folosirea acestei tehnologii reprezintă un prim pas în cadrul aplicației către o digitalizare cât mai eficientă a proceselor birocratice prevăzute în spitalele românești, aceasta având o gamă largă de aplicații.

3.2.1. Back-end și Baza de date:

Comunicarea dintre baza de date și back-end se face prin intermediul API-ului Java Persistence[18] (JPA). Acest este un set de concepte ce gestionează persistența datelor în aplicațiile de tip Java, oferind o abstractizare a conceptelor ce țin de baza de date și definind concepte cheie, cum ar fi entități, manageri de entități, tranzacții și interogări.

Pentru a putea modela entitățile din viața reală ce iau parte la activitatea unui spital, au fost implementate clase ce descriu structura tabelului asociat din baza de date. Aceste clase mapează un tabel din baza de date astfel încât să poată fi gestionată de JPA. Pentru a fi posibil acest lucru, fiecare dintre aceste clase au fost adnotate cu notația de tipul *@Entity*[19] pentru a marca clasa ca fiind o entitate JPA.

Cele mai importante astfel de adnotări folosite în aceste clase sunt cele:

- *@OneToOne*[20], *@ManyToOne*[21], *@ManyToMany*[22], *@JoinColumn*[23] - ce descriu relațiile dintre tabele;
- *@Id*[24] - anunță JPA că respectivul câmp al clasei reprezintă cheia primară a tabelului;
- *@GeneratedValue* - genereaza automat cheia primară;
- *@Builder, @ToString, @AllArgsConstructor, @NoArgsConstructor, @Getter, @Setter Lombok*[25] - reprezintă o librărie Java ce generează automat funcționalități de tipul *Getter, Setter, Equals.*

[18] https://docs.oracle.com/javaee/6/tutorial/doc/bnbpz.html
[19] https://www.baeldung.com/jpa-entities
[20] https://www.baeldung.com/jpa-one-to-one
[21] https://vladmihalcea.com/manytoone-jpa-hibernate/
[22] https://www.baeldung.com/jpa-many-to-many
[23] https://www.baeldung.com/jpa-join-column
[24] https://www.baeldung.com/hibernate-identifiers
[25] https://projectlombok.org/features/

Pentru a sintetiza și organiza aplicația astfel încât codul să poată fi ușor de citit și ordonat, clasele cu funcționalități comune au fost grupate în pachete cu un nume descriptive. Astfel, toate clasele ce mapează tabele din baza de date către clasele Java au fost grupate în pachetul *Entity*.

Una dintre clasele ce fac parte din pachetul *Entity* este *Analysis* (Figura 4). Aceasta mapează tabelul *analysis* din baza de date, care este folosit pentru a stoca diferite informații despre una dintre analizele pe care spitalul le pune la dispoziție, informații precum numele acesteia și categoria din care face parte.

```
@Builder
@ToString
@AllArgsConstructor
@NoArgsConstructor
@Getter
@Setter
@Entity
@Table(name = "analysis")
public class Analysis {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Integer id;

    @OneToOne(cascade = CascadeType.ALL)
    @GeneratedValue(strategy = GenerationType.AUTO)
    @JoinColumn(name = "analysis_category_id",  referencedColumnName
= "id")
    private AnalysisCategory analysisCategory;

    private String name;
}
```

Figura 4. Exemplu de cod pentru clasa Analysis cu adnotare *@Entity*

Pe lângă clasele ce mapează tabele, sunt necesare și clase Java care se ocupă cu extragerea și salvarea datelor din baza de date. Aceste clase se regăsesc în pachetul *Repository*. Fiecare dintre aceste clase implementează interfața JpaRepository[26], care primește doi parametri:

- primul parametru: reprezintă clasa ce mapează tabelul din baza de date; un exemplu de clasă este cea menționată anterior din pachetul Entity (Figura 4);
- al doilea parametru: reprezintă tipul de dată ce alcătuiește cheia primară a tabelului dat ca prim parametru; în exemplul anterior aceasta este de tip Integer, cu numele "id".

Toate clasele din pachetul *Repository* sunt adnotate cu *@Repository*[27]. Adnotarea *@Repository* face parte din Framework-ul Spring și este utilizată pentru a marca o clasă ca fiind un repository. Un repository este o clasă care oferă acces la date și care este utilizată pentru a gestiona operațiile de citire și scriere în baza de date, ceea ce este exact necesarul părții de backend a platformei. Un exemplu de astfel de clasă este *AnalysisRepository* și se poate observa în Figura 5, clasă ce are ca scop operarea informațiilor din tabelul Analysis.

```
@Repository
public interface AnalysisRepository extends JpaRepository<Analysis,Integer>
{

    List<Analysis> getAllByAnalysisCategory_Name(String categoryName);

    Analysis getAnalysisByName(String name);

    List<Analysis> findAnalysesByNameContaining(String input);
}
```

Figura 5. Cod exemplu interfața AnalysisRepository cu adnotare *@Repository*

În această clasă putem observa adnotarea *@Repository* și parametrii pentru interfața JpaRepository: *Analysis*, care este clasa ce mapează tabelul cu același nume și tipul de date al cheii primare din acest tabel, în acest caz Integer.

În interiorul clasei, putem observa trei metode importante și anume:

- `List<Analysis> getAllByAnalysisCategory_Name(String categoryName)`

[26] https://docs.spring.io/spring-data/data-jpa/docs/current/api/org/springframework/data/jpa/repository/JpaRepository.html

[27] https://docs.spring.io/spring-data/jpa/docs/1.5.0.RELEASE/reference/html/jpa.repositories.html

metoda care va prelua din baza de date toate intrările (rândurile) din tabelul *Analysis* ce au numele categoriei egal cu parametrul metodei; rezultatul va fi mapat sub forma unei liste de obiecte de tip *Analysis*;

- `Analysis getAnalysisByName(String name)`
  metoda va extrage din baza de date analiza din tabelul *Analysis*, care are numele egal cu parametru metodei; acesta este mapat sub forma unui obiect de tip *Analysis*;
- `List<Analysis> findAnalysesByNameContaining(String input)`
  metoda va extrage din baza de date toate rândurile din tabelul *Analysis* ale căror nume includ primul parametru; acestea vor fi mapate sub forma unei liste de obiecte de tip *Analysis*.

3.2.2. Front-end și back-end

Comunicarea între front-end și back-end se realizează prin intermediul unui API (Application Programming Interface). API-ul este un set de reguli și protocoale care permit aplicațiilor să comunice între ele. În general, front-end-ul face cereri către API-ul back-end-ului, iar acesta din urmă răspunde cu datele solicitate. Aceste cereri și răspunsuri sunt transmise prin intermediul protocolul HTTP (HyperText Transfer Protocol).

Pentru această comunicare s-au implementat mai multe pachete ce au un rol esențial în această comunicare dintre cele două sisteme.

Primul pachet este "Controller", ce conține mai multe clase, fiecare dintre acestea conținând următoarele adnotări:

- *@RestController*[28]: este o adnotare utilizată în Java pentru a marca o clasă ca fiind un Rest Controller. Această anotare este utilizată în cadrul framework-urilor Java, cum ar fi Spring, pentru a defini endpoint-urile API-ului și pentru a gestiona cererile primite de la client. Mai precis, un endpoint este unul din capetele unui canal de comunicare dintre două sisteme (server back-end și aplicație front-end). În cazul de față un endpoint este reprezentat de o resursă de tip URL (ex: http://localhost:7070/api/login);

[28] https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/web/bind/annotation/RestController.html

- *@RequestMapping*[29] este o adnotare utilizată în Java pentru a defini endpoint-urile API-ului într-un controller REST. Această anotare este utilizată în cadrul framework-urilor Java pentru a mapa cererile primite de la client la metodele corespunzătoare din controller. Pentru resursele ce se regăsesc la endpoint-uri de tipul *http://.../api/...* adnotarea @RequestMapping este necesară pentru a putea identifica cu exactitate acțiunea (metoda sau codul de executat) așteptată în urma accesării resursei;
- *@RequiredArgsConstructor*[30]: o adnotare lombok care creează un constructor cu parametrii necesari pentru alcătuirea clasei.

Un exemplu de astfel de clasă poate fi observat în Figura 6, clasă ce are rolul de a prelua toate cererile către endpoint-urile de tipul **http://localhost:7070/api/analysis/…** unde *api/analysis* este parametrul adnotării *@RequestMapping.*

```
@RestController
@RequestMapping("api/analysis")
@RequiredArgsConstructor
public class Analysis {

    @Autowired
    private AnalysisService analysisService;

    @GetMapping("/all")
    public ResponseEntity<List<AnalysisResponseByCategory>>
getAllAnalysis() {
        List<AnalysisResponseByCategory> list =
analysisService.getAllByCategory();
        if(list == null){
            return new ResponseEntity<>(null,HttpStatus.BAD_REQUEST);
        }
        return ResponseEntity.ok(list);
    }
    @GetMapping("/search/{input}")
    public ResponseEntity<AnalysisResponse>
getAllAnalysisLike(@PathVariable String input) {

        AnalysisResponse response =
analysisService.getAllAnalysisLike(input);
        if(response == null){
```

[29] https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/web/bind/annotation/RequestMapping.html

[30] https://projectlombok.org/api/lombok/RequiredArgsConstructor

```
            return new ResponseEntity<>(null,HttpStatus.BAD_REQUEST);
        }
        return ResponseEntity.ok(response);
    }
}
```

Figura 6. Exemplu de cod pentru clasă cu adnotări *@RestControlle*r, *@RequestMapping, @RequiredArgConstructor*

Toate cererile care sunt plasate către endpoint-ul */api/analysis* vor fi preluate de către această clasă, datorată adnotării *@RequestMapping*. Clasa are un singur atribut de tip *AnalysisService*, care are rolul de a prelucra datele primite de la front-end. Clasa conține și două metode care așteaptă un request de tip GET către calea menționată din *RequestMapping*.

Toate cererile care sunt plasate către endpointurile API sunt preluate de clasele aferente pachetului *Controller*. Aceste clase, în schimb, nu sunt responsabile și de prelucrarea cererilor. Pentru a decupla logica de primire și prelucrare, pachetul *Service* a fost implementat pentru a îmbunătăți citirea codului și modularizarea proiectului.

Singura adnotare a claselor din pachetul *Service* este *@Service*[31], care este utilizată în Java pentru a marca o clasă ca fiind un serviciu în cadrul unei aplicații și în framework-uri pentru a defini o clasă ce conține logica de afacere (en. business logic).

O astfel de clasă poate fi observată în Figura 7, ce are rolul de a prelucra cererile legate de analizele medicale.

```
@Service
public class AnalysisService {

    @Autowired
    private AnalysisRepository analysisRepository;

    public List<AnalysisResponseByCategory> getAllByCategory() {
        List<String> categoryNames =
analysisCategoryService.getAllCategoryNames();
        List<AnalysisResponseByCategory> list = new ArrayList<>();
        for (String s: categoryNames ) {
            List<Analysis> analysisList =
analysisRepository.getAllByAnalysisCategory_Name(s);
            List<String> analysisNames = this.extractNames(analysisList);
            list.add(AnalysisResponseByCategory
```

[31] https://www.baeldung.com/spring-component-repository-service

```
                    .builder()
                    .categoryName(s)
                    .allAnalysisName(analysisNames)
                    .build());
        }
        return list;
    }
      .....
}
```

Figura 7. Exemplu de cod pentru o clasă cu adnotarea *@Service*

Această clasă este un atribut al controllerului *Analysis*. În interiorul *AnalysisService* este reprezentată metoda *getAllByCategory*, care este apelată din clasa *Analysis*. Metoda are rolul de preluarea a tuturor analizelor în funcție de categorie. Clasele de tip *Service* reprezintă puncte de legătură între controlere și baza de date, rolul lor principal fiind acela de a prelua, prelucra și de a răspunde cererilor.

O reprezentare a relațiilor dintre clasele din pachetul Entity se pot regăsi în Figura 8.

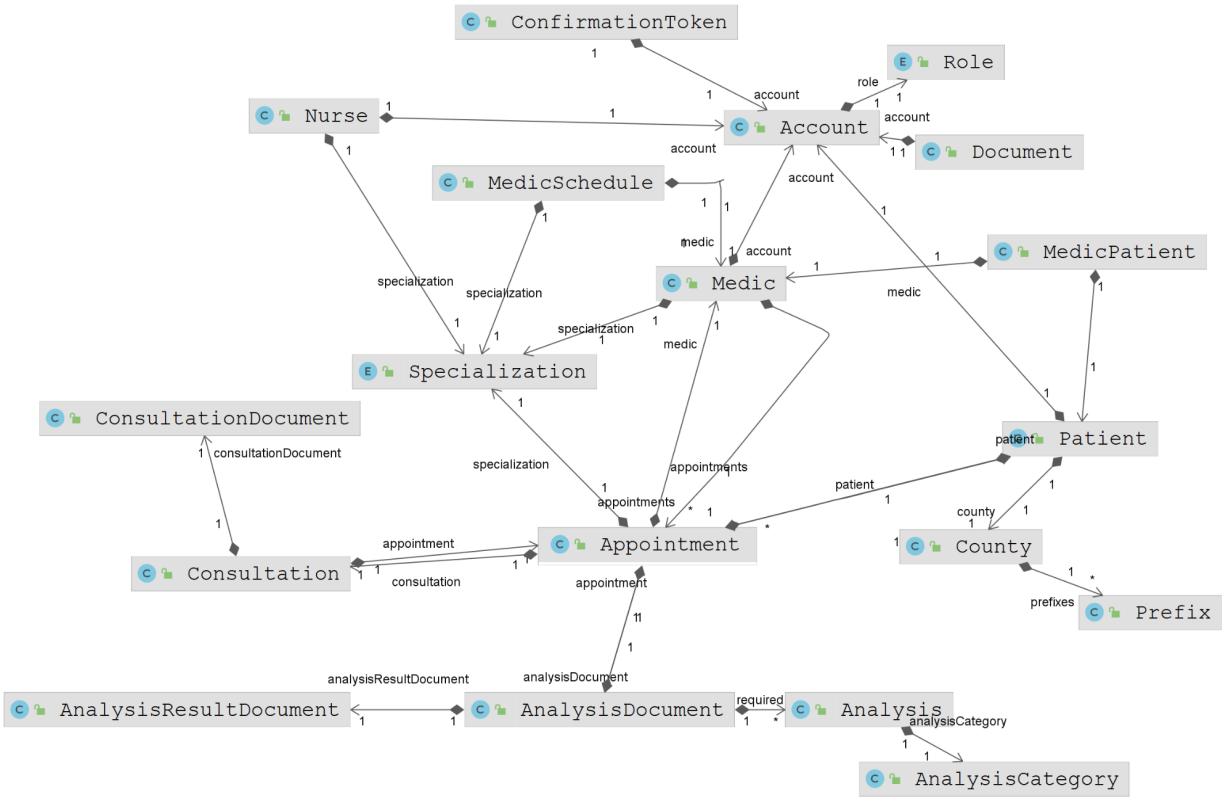

Figura 8. Diagrama de clase restrânsă

3.3. Componenta front-end

Aferent fiecărei aplicații de infrastructură, care are în structura sa arhitecturală o bază de date, ce stochează informații, o componentă de prelucrare și procesare de informații, reprezentată de de back-end, este stringent nevoie și de o interfață grafică destinată utilizatorilor (în cazul de față atât pacienții cât și personalul instituției sanitare). Această componentă este cea de front-end și reprezintă partea vizibilă și interactivă a unei aplicații web sau mobile, care permite utilizatorilor să interacționeze cu procesele din spate și anume procesarea și stocarea informațiilor. Aceasta include elemente precum interfața grafică, design, animații, butoane, meniuri și alte elemente care permit utilizatorilor să interacționeze cu aplicația.

Aceasta este esențială în dezvoltarea unei aplicații, deoarece este prima interacțiune a utilizatorului cu aplicația în sine, în momentul utilizării sistemului. O interfață bine proiectată și ușor de utilizat poate face diferența între o aplicație populară și una care nu are o uzanță atât de mare. De asemenea, un front-end bine proiectat poate îmbunătăți experiența utilizatorului și poate crește gradul de satisfacție al acestuia. De asemenea, prin intermediul aplicației front-end sunt preluate datele rezultate din urma interacțiunii cu utilizatorul, acestea urmând a fi transmise către serverul de back-end. În urma prelucrării acestor date, aplicația de front-end preia răspunsurile serverului și le expune utilizatorului într-o manieră inteligibilă, filtrată și intuitivă.

Implementarea front-end-ului s-a realizat utilizând combinația clasică de fișiere H.T.M.L (Hypertext mark-up language[32]), JavaScript[33] și C.S.S (Cascade Style Sheets[34]). Toate acestea sunt găzduite pe o mașină locală, utilizând serviciul Live Server[35], pus la dispoziție de către compania Microsoft prin intermediul utilitarului Visual SC - acesta fiind un mediu de dezvoltare și pentru componentele de tip front-end. Fiecare din aceste tipuri de fișiere servește un anumit scop și anume:

- fișierele de tip HTML sunt folosite pentru structura paginilor web. Acestea au extensia de tipul *.html*, iar limbajul este bazat pe standardele impuse de World Wide Web Consortium[36], ce conține recomandări pentru formatul documentelor distribuite și interschimbate online;

[32] https://en.wikipedia.org/wiki/HTML
[33] https://en.wikipedia.org/wiki/JavaScript
[34] https://www.w3.org/Style/CSS/Overview.en.html
[35] https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer
[36] https://www.w3.org/

- cele de tip JavaScript se ocupă de funcționalitățile butoanelor. Acesta este un limbaj de scripting utilizat pentru crearea de conținut dinamic al unei pagini web ce ajută la îmbunătățirea interacțiunii cu utilizatorul;
- cele de tipul CSS preiau responsabilitatea design-ului grafic. Acesta este un limbaj bazat pe reguli prin care se specifică stilurile care se vor aplica unui element sau grup de elemente particular.

Structura proiectului front-end este împărțită pe mai multe directoare, fiecare din ele conținând atât fișiere de tipurile menționate anterior, cât și fișiere cu rol ajutător sau stilistic, de exemplu poze pentru fundal (background) și fonturi pentru text. Împărțirea fișierelor în acest mod ajută la păstrarea structurii curate și ușor de urmărit. În cadrul implementării componentei de front-end am avut în vedere respectarea următoarei structuri:

- **Account**: conține fișierele **.html* aferente paginilor web destinate vizualizării conturilor de tip *patient* și *medic*;
- **Medics**: conține fișierele **.html* aferente acțiunilor specifice unui medic (de exemplu editarea consultației curente, vizualizarea pacienților înrolați, înregistrarea unui nou cont etc.);
- **Nurse**: conține fișierele **.html* aferente paginilor web specifice unei asistente (de exemplu: formularul de recoltare al analizelor, crearea unui cont, finalizarea analizelor);
- **Patients**: conține fișierele **.html* aferente paginilor web specifice unui pacient (crearea unui cont, a unei programări, vizualizarea medicilor la care este înrolat etc.);
- **Js**: conține fișierele **.js* organizate în funcție de tipul de cont al utilizatorului și funcționalitățile specifice;
- **Css**: conține fișierele **.css* care se ocupă cu stilizarea paginilor web.

Toate aceste mari componente descrise în cadrul acestui capitol, de back-end, de front-end și baza de date, pun bazele unei aplicații complexe și complete din punct de vedere arhitectural.

4. Politica de confidențialitate

Un subiect de interes care trebuie de asemenea tratat, este unul dintre rolurile principale ale bazei de date. O bază de date este esențială oricărei aplicații statice sau platforme web, având mai multe roluri, de la stocarea fișierelor, într-un mod ordonat și modular, care să faciliteze o accesare rapidă și eficientă, până la protejarea lor. Cu alte cuvinte, protecția și confidențialitate datelor cu caracter personal ale unui individ alcătuiesc un subiect sensibil și de mare interes în ultimul deceniu, întrucât orice utilizare necorespunzătoare a acestora poate pune în pericol viața.

În anul 2016 Uniunea Europeană a publicat setul de reguli aferent protejării datelor cu caracter personal. Fiecare țară membră a UE, trebuie să se supună acestui set de reguli, ca urmare, fiecare autoritate națională, fiind obligată să se muleze pe acest nou sistem. Astfel, în anul 2018 ANS PDCD[37], s-a aliniat la standardul GDPR impus de către Uniunea Europeană. Acest set de reguli prevede protecția datelor cu caracter personal și utilizarea acestora de către orice instituție publică și/sau privată de pe teritoriul țării în cauză.

Principalele aspecte din regulament sunt:

Consimțământ:

Consimțământul în cadrul GDPR [10] (Regulamentul General privind Protecția Datelor) este un act juridic unilateral al persoanei vizate, prin care aceasta își dă acordul prelucrării datelor sale cu caracter personal de către un operator. Conform GDPR, consimțământul trebuie să fie dat în mod liber, informat și neechivoc, iar operatorul trebuie să poată demonstra că a obținut consimțământul în mod valabil. În plus, obținerea consimțământului nu exonerează operatorul de îndeplinirea celorlalte obligații prevăzute de lege. Prevederile GDPR privind consimțământul se aplică nu doar datelor stocate în cazul contractelor, ci și datelor folosite pentru marketing și pentru cookie-uri.

Drepturile subiecților:

Drepturile subiecților sunt drepturile pe care le au persoanele vizate de prelucrarea datelor lor cu caracter personal conform GDPR. Aceste drepturi includ drept la informare, acces, rectificare, ștergere, restricționarea prelucrării, portabilitatea datelor, opoziție și dreptul de a nu fi supus unei decizii automate. Acestea sunt importante pentru a proteja drepturile și libertățile persoanelor vizate în ceea ce privește prelucrarea datelor lor cu caracter personal.

[37] https://www.dataprotection.ro/?page=noua%20_pagina_regulamentul_GDPR

Notificarea încălcării regulamentului:

Notificarea încălcării regulamentului este o obligație prevăzută de GDPR care se aplică operatorilor de date cu caracter personal. În cazul în care are loc o încălcare a securității datelor cu caracter personal, operatorul trebuie să notifice autoritatea de supraveghere a protecției datelor în termen de maximum 72 de ore de la descoperirea încălcării și, dacă este cazul, să notifice și persoanele vizate. Scopul notificării este de a asigura transparența și responsabilitatea în ceea ce privește prelucrarea datelor cu caracter personal și de a proteja drepturile și libertățile persoanelor vizate.

Chiar dacă orice server de baze de date oferă suport pentru protecția datelor, prin prisma implementărilor oferite, fiind vorba despre o aplicație web ce stochează și prelucrează datele sensibile ale cetățenilor, este necesar un grad foarte ridicat de protecție împotriva atacurilor cibernetice. Studiul intitulat "*Cyber security in the age of COVID-19: A timeline and analysis of cyber-crime and cyber-attacks during the pandemic*" [11] ne arată că prin prisma pandemiei de SarsCov2, atacurile cibernetice s-au întețit, fapt ce denotă mai mult sporirea gradului de protecție a datelor cu caracter personal.

O persoană rău intenționată poate ataca orice platformă cu minimul de efort depus, dacă aceasta nu este bine gândită, implementată și securizată din punct de vedere al siguranței cibernetice. Persoana în cauză care realizează atacul poate extrage informații cu caracter personal precum codul numeric personal, domiciliu, credențialele contului etc.

Codul numeric personal poate fi folosit în scopuri ilegale, de exemplu pentru de identitate. Credențialele contului pot fi folosite pentru autentificarea pe platformă, de unde atacatorul poate accesa documentele pacientului, care la rândul lor pot conține alte informații personale despre pacient. Când vine vorba de a face rău unei persoane, având acces la datele personale, atacatorii pot deveni foarte creativi, în majoritatea cazurilor recurgând la șantaj sau amenințări.

Datorită acestor temeri, datele cu caracter personal sunt criptate integral, pentru a spori securitatea aplicației. Astfel, chiar și în cazul cel mai nefavorabil în care o persoană reușește să obțină acces la baza de date, el nu poate să utilizeze datele în niciun fel deoarece acestea sunt criptate cu o cheie la care doar administratorii au acces.

Pentru a asigura securitatea datelor, s-a implementat clasa *AES*, clasă ce aparține pachetului *Utils* al aplicației de back-end. Această clasă conține metode pentru generarea unei chei secrete, criptarea și decriptarea șirurilor de intrare folosindu-se algoritmul *Advanced*

Encryption Standard [7] in modul *Cipher Block Chaining* (CBC) [12] cu padare de tip PKCS5[38]. O astfel de arhitectură poate fi observată în Figura 9.

Clasa folosește o derivare a cheii bazată pe parola (PBKDF2) cu o funcție hash SHA-256 [13] pentru a genera o cheie secretă. Această cheie este generată folosindu-se o parolă setată în prealabil (“mysecretpassword”) și un salt generat aleatoriu. Cheia secretă generată este apoi stocată într-un fișier numit "secret.key" aferent directorului de lucru curent.

Metoda “encrypt” primește un șir de caractere ca parametru, criptează folosindu-se cheia secretă și vectorul de inițializare (IV - initialization vector[39]) și returnează un șir de caractere în formatul Base64-encoded [14].

Metoda “decrypt” primește ca parametru un șir de caractere criptat în formatul Base64-encoded, și îl decriptează folosindu-se cheia secretă și vectorul de inițializare. Dacă decriptarea a avut loc cu succes, metoda returnează un șir de caractere ce reprezintă parametrul metodei decriptat.

Metoda "getSecretKey" citește cheia secretă din fișierul "secret.key" și o returnează ca obiect SecretKey.

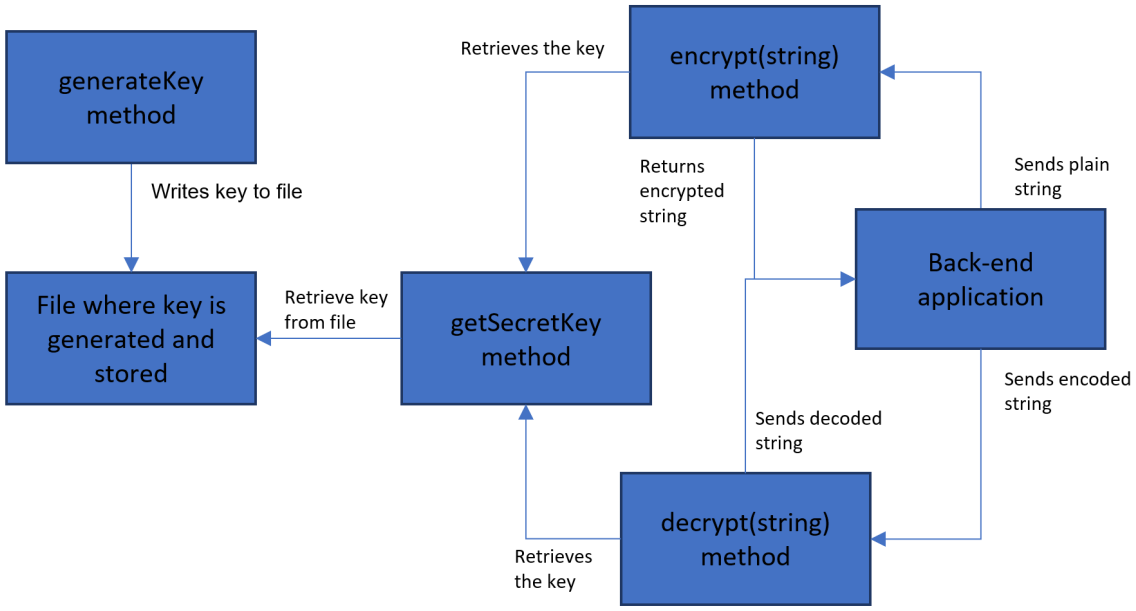

Figura 9. Descrierea modului de funcționare a criptării și decriptării

[38] https://www.rfc-editor.org/rfc/rfc2898
[39] https://www.techtarget.com/whatis/definition/initialization-vector-IV

5. Automatizarea sistemului

5.1. Necesitatea automatizării

Cum a fost prezentat anterior, pentru a eficientiza sistemul de înregistrare a pacienților, a fost implementat un serviciu prin care, trimițând o simplă poză a cărții de identitate românești, informațiile necesare creării contului sunt extrase, printr-un proces de recunoaștere optică a caracterelor (Optical Character Recognition - OCR[40]). Acest proces este în strânsă legătură cu înregistrarea electronică a datelor dintr-un document. Prin intermediul acesteia, se transferă operațiunea manuală de introducere sau completare de date într-o operațiune digitală, mai precisă, mai rapidă, mai ecologică și economică.

Situația actuală din spitalele de stat și din mai multe instituții publice, este una nefavorabilă din punct de vedere al înregistrării electronice a datelor. Sistemele actuale sunt în proces de digitalizare, însă o mare parte din acestea încă sunt strâns legate de sistemul vechi, bazat pe o stocare și organizare haotică a datelor provenite din multiple surse. Volumul abundent de date ce necesită procesarea manuală este un factor ce încetinește și îngreunează tragic modul de gestionare al informațiilor din orice instituție.

Migrarea către procedura de înregistrare electronică vine cu avantaje precum procesarea mai rapidă a datelor, precizia sporită, costuri reduse, transferul rapid de date și accesibilitatea sporită. În mod ideal tehnologia OCR își propune transformarea în format digital al conținutului oricărui tip de document scanat sau fotografiat. Lucrarea de față s-a axat pe un use case centrat pe această problemă și anume preluarea automată a datelor dintr-o imagine a cărții de identitate românești.

Implementarea acestei funcționalități ajută la înregistrarea unor noi utilizatori în aplicație, dat fiind că un spital are nevoie de toate datele personale ale pacienților și angajaților săi, date ce se regăsesc în cartea de identitate. Odată ce datele ajung în sistem, acestea devin accesibile de oriunde și ușor de completat automat de către aplicație (de exemplu la o consultație). Generându-se alte documente în format digital în urma accesării serviciilor spitalului (de exemplu buletinul de analize), stocarea lor pe server oferă posibilitatea utilizatorului să le poată vizualiza oricând, la nevoie, doar accesându-și contul prin intermediul aplicației.

[40] https://aws.amazon.com/what-is/ocr/

5.2. Tehnologia OCR

Optical Character Recognition (OCR) tradus în *Recunoașterea Optică a Caracterelor* face parte din domeniul inteligenței artificiale. *„Recunoaşterea optică a caracterelor (OCR) transformă imagini de text, cum ar fi documentele scanate, în caractere de text. Cunoscută şi sub numele de recunoaşterea textului, OCR face posibilă editarea şi reutilizarea textului conţinut de imaginile scanate. OCR utilizează o formă de inteligenţă artificială, cunoscută sub numele de recunoaşterea modelului, pentru identificarea individuală a caracterelor unui text dintr-o pagină, inclusiv semnele de punctuaţie, spaţiile şi sfârşitul de linie”* (Microsoft[41]).

Tehnologia OCR datează din jurul anilor 1900, sistemele timpurii începând în anii 1950-1970[42]. Acestea au fost inițial construite pentru cazuri particulare cum ar fi sortarea scrisorilor în funcție de codul poștal. Datorită creșterii utilizatorilor de computere personale începând cu anii 1990 și folosirea tehnologiei OCR a fost în continuă dezvoltare. Aceasta a început să fie folosită din ce în ce mai des în cadrul digitalizării cărților și a altor materiale printate făcând mai ușoară accesarea și căutarea de informații. La începutul anilor 2000 tehnologia OCR a introdus noi algoritmi și resurse hardware îmbunătățite, lucru ce a dus la o creștere exponențială a performanțelor [15]. Aici s-au pus bazele adoptării ulterioare a acestei tehnologii în mai multe industrii cum ar fi managementul documentelor și procesarea facturilor. De asemenea, și Google a introdus motorul de căutare Google Books[43] după ce a reușit să digitizeze zeci de milioane de cărți.

Recunoaștere digitală a caracterelor se poate realiza atât prin intermediul scanner-elor cât și prin intermediul lentilelor. Tehnologia OCR folosește tehnici optice și o parte de procesare digitală a documentelor, așadar vor fi prezentate pe scurt principiile de funcționare ale acesteia.

5.3. Cum funcţionează OCR

În spatele tehnologiei OCR stă o combinație de algoritmi de procesare de imagini [16], recunoașterea șabloanelor și tehnici de învățare automată, care extrag textul din imagini sau documente scanate. OCR se aplică astfel pe imagini și produce ca rezultat șiruri de caractere în urma procesării datelor extrase din datele de intrare. Rezultatul poate fi stocat în orice document text editabil (de exemplu, în fișiere *.txt, *.docs etc.).

[41] http://office.microsoft.com/en-us/help/about-optical-character-recognition-ocr-HP003081255.aspx
[42] https://www.veryfi.com/ocr-api-platform/history-of-ocr/
[43] https://www.google.com/intl/en/googlebooks/about/index.html

Procesul de recunoaștere optică poate fi privit într-o notă mai generală prin prisma următorilor pași care sunt urmăriți în implementare.

Pașii necesari în implementarea OCR conform cu literatura de specialitate [17] sunt:

- **achiziția imaginilor**: sistemul poate avea încorporat un subsistem de achiziție de date sub formă de imagini sau le poate primi sub formă de fișiere de tip imagine ca input;
- **preprocesarea**: pentru a obține cele mai bune rezultate posibile în urma analizării unei imagini, aceasta este supusă unor transformări care îi îmbunătățesc calitatea (de exemplu ajustări de contrast, reducerea zgomotelor etc.);
- **localizarea textului**: imaginea preprocesată este analizată (folosind tehnici precum detecția de contururi) pentru a fi localizate regiunile care conțin text;
- **segmentarea caracterelor**: dacă există regiuni care conțin text cu mai multe caractere acestea vor fi separate pentru a putea fi interpretate individual;
- **recunoașterea caracterelor**: caracterele sunt interpretate folosind șabloane de recunoaștere; acestea includ în mod normal antrenarea unui model de învățare automată (de exemplu o rețea neuronală sau o bază de date cu caractere etichetate); odată interpretate, caracterele obținute sunt mapate către reprezentarea lor textuală;
- **reconstrucția textului**: rezultatele interpretării caracterelor sunt reorganizate, combinate și aranjate în ordinea corectă pentru a reconstrui textul original, implicând modelarea limbajului, verificarea ortografiei etc;
- **afișarea și procesarea**: rezultatul final este textul reorganizat care poate fi procesat mai departe și formatat după nevoile programatorului, implicând indexarea textului, extracția de date etc.

În cadrul lucrării curente, pentru a pune în aplicare tehnici de OCR-izare în implementare, s-a folosit API-ul *Tesseract*[44]. Acesta pune la dispoziția oricui metodele de procesare OCR (fiind de tip open-source) având la bază un motor bazat pe OCR (engine). Acesta nu include partea de preprocesare a datelor de intrare însă oferă rezultate excepționale.

API-ul Tesseract [8] a fost ales in detrimentul altor implementari (ABBYY FineReader[45]) deoarece acesta prezintă avantaje precum: *recunoaștere precisă a textului* (Tesseract are o acuratețe bună în recunoașterea caracterelor, având o rată de succes ridicată), *sursă gratuită* (fiind un proiect cu sursă gratuită, accesul la codul sursă este permis oricui, ceea

[44] https://github.com/tesseract-ocr/tesseract
[45] https://pdf.abbyy.com/

ce oferă flexibilitate și control asupra implementării), *ușor de utilizat și integrat* (oferă API-uri pentru diferite limbaje de programare precum Java, Python, C++ etc.), *continuă îmbunătățire* (fiind de tip sursă gratuită, există o comunitate care aduce îmbunătățiri constante) [9].

Implementarea curentă pune în aplicare OCR-izarea pentru o imagine a unei cărți de identitate, extrăgând din aceasta mai multe date personale relevante în procesul de creare a unui nou cont. Utilizatorul încarcă în aplicație o poză cu cartea de identitate, iar aceasta trece prin următorii pași:

- imaginea este încărcată sub forma unui obiect de tip *Mat*[46] (vector multidimensional ce reprezintă pixelii și alte informații cu privire la imagine) prin intermediul funcției *Imgcodecs.imread()* din cadrul librărie OpenCV[47] (foarte importantă în procesare de imagini, fiind o librărie open-source dedicată învățării automate și algoritmilor specifici *computer vision*[48]);
- se calculează dimensiunile în care se va încadra cartea de identitate din poză în funcție de dimensiunile imaginii inițiale;
- se definește regiunea de interes (en.: Region of Interest - ROI) sub forma unui element de tip dreptunghi (clasa *Rect*[49] din pachetul OpenCV);
- se creează o nouă imagine în urma decupării imaginii inițiale sub forma regiunii de interes;
- se inițializează un nou element de tip Tesseract și se aplică funcția *doOCR* pe imaginea decupată;
- rezultatul se transmite spre analizare către metode adiacente spre a extrage din el datele de interes.

  Pentru mai multe detalii, a se consulta Anexa 2.

Detaliile de interes ce rezultă în urma analizării bazate pe OCR sunt: codul numeric personal, județul de domiciliu, vârsta, sexul, numele complet. Toate aceste date sunt necesare pentru înregistrarea unui cont nou în cadrul aplicației.

[46] https://www.tutorialspoint.com/explain-the-mat-class-in-java-opencv-librar
[47] https://opencv.org/
[48] https://www.ibm.com/topics/computer-vision
[49] https://docs.opencv.org/3.4/javadoc/org/opencv/core/Rect.html

Figura 9. Specimen carte de identitate

După cum se poate observa în Figura 9, datele personale ale cetățeanului se regăsesc atât explicit, în diverse secțiuni ale cărții de identitate, secțiuni care sunt precedate de tipul informației furnizate (seria și numărul cărții de identitate, numele, prenumele, locul nașterii etc.) cât și în zona albă situată în partea de jos a cărții de identitate, numită Zonă de citire optică. Pentru cărțile de identitate eliberate în România, această zonă este compusă din doua linii și sunt specificate în *Documentul ICAO 9303*[50], care conține toate datele documentului pentru a fi prelucrate automat.

Cartea de identitate este cel mai important document al unui cetățean român după certificatul de naștere. Aceasta se eliberează pentru prima dată la vârsta de 14 ani și conține date precum codul numeri personal, domiciliul, sexul și o poză a persoanei în cauză.

Ministerul Administrației Interne este cel care emite acest act. Datele acestea sunt încapsulate în cele două linii dedicate citirii optice, asemenea pașapoartelor. Zona dedicată citirii optice (supranumită și MRZ - *Machine readable zone* - Tabelul 1 și Tabelul 2) folosește caracterele de tipul OCR-B (create în 1968), într-un format cu spații mai generoase și aerisite,

[50] https://www.icao.int/publications/pages/publication.aspx?docnum=9303

având câte 36 de caractere pe fiecare linie. Nu sunt incluse caractere diacritice în această zonă pentru a nu cauza probleme în recunoaștere automată.

Tabelul 1. Detalii MRZ linia 1 - Forma și conținutul cărții de identitate simplă (sursă mai.gov.ro)

Grup caractere	Câmp	Valoare
1-2	Tip document	ID
3-5	Țara	ROU
5-36	Nume familie	NUME FAMILIE
	Separator nume familie și prenume	<<
	Prenume	PRENUME

Tabelul 2. Detalii MRZ linia 2 - Forma și conținutul cărții de identitate simplă (sursă mai.gov.ro)

Grup caractere	Câmp	Valoare
1-9	Serie și număr carte de identitate, se completează cu "<" până la 9 caractere	AR123456
10	Cod Validare pentru grupul 1-9	2
11-13	Naționalitatea	ROU
14-19	AA–LL-ZZ data nașterii	010101
20	Cod Validare pentru grupul 14-19	1
21	Sex	M
22-27	AA-LL-ZZ data expirării	210101
28	Cod Validare pentru grupul 22-27	5
29	Serie CNP	1
30-35	Parte de sfârșit a CNP-ului	010111
36	Cod Validare grupurile 1-10, 14-20, 22-35	7

Informațiile care pot fi extrase din aceasta zonă de citire optică sunt:

- **numele de familie** - acesta se regăsește după apariția caracterelor *IDROU*. Aceste caractere apar în orice carte de identitate românească, iar numele de familie este succesiunea de caractere până la apariția primului simbol <;
- **prenumele** - se vor regasii după simbolul < aflat după numele de familie, despărțite de simbolul <;
- **județul de domiciliu** - este reprezentat de primele două caractere din al doilea rând și reprezintă prefixul județului. Aceste prefixe aparțin unui singur județ, dar există și excepții și anume un județ poate avea mai multe prefixe. Un exemplu este pentru municipiul Iași, care are prefixele IZ, MZ, MX;
- **vârsta** - este reprezentată de primele două caractere de tipul număr, de lângă caracterele *ROU,* de pe randul al doilea și reprezintă anul nașterii. Având aceste numere, se poate calcula vârsta;
- **genul** - după cele 7 caractere aflate în imediata vecinătate a șirului de caractere *ROU* de pe randul al doilea, se găsește un caracter alfabetic, mai exact *M* sau *F*, reprezentând sexul deținătorului;
- **CNP** - având toate informațiile de mai sus, împreuna cu informațiile despre luna și ziua în care s-a născut utilizatorul (care se pot extrage tot din această zonă), se poate reconstrui CNP-ul persoanei.

Automatizarea procesului de înregistrare prin intermediul folosirii tehnologiei OCR are rezultate foarte bune, în general. În ceea ce privește automatizarea folosind tehnici de OCR-izare din cadrul aplicației ce face subiectul acestei lucrări, rezultatele obținute au fost satisfăcătoare, dar cu posibilitate de îmbunătățire și extindere spre aplicarea acestor algoritmi și pentru transformarea documentelor fizice în documente digitale. Aceste rezultate au fost obținute în urma unui proces de testare a 100 de eșantioane (imagini ale unor cărți de identitate) din care 89 au fost recunoscute în totalitate, iar pentru restul a fost nevoie de intervenție manuală a utilizatorului pentru a corecta anumite date. Astfel s-a obținut o acuratețe de 89%.

6. Rezultate

În urma implementării aplicației conform cu explicațiile din această lucrare, aceasta reușește să rezolve problemele spitalelor de stat din România precum: lipsa unui sistem informatic stabil, nerespectarea programărilor și imposibilitatea de a crea una online, digitalizarea proceselor birocratice regăsite în procesul de consultație al pacienților, de prelevare a analizelor, de transmitere a rezultatelor etc.

Aplicația WellnessWave oferă soluții pentru problemele menționate anterior astfel:

- **lipsa unui sistem informatic stabil** => aplicația aduce numeroase funcționalități și tehnologii de actualitate care asigură o bună funcționare a sistemului, astfel încât acesta se dovedește a fi stabil; implementarea funcționalităților se bazează pe limbajul Java care oferă multe oportunități de scalare, îmbunătățire și eficientizare; implementarea interfeței se bazează pe tehnologiile de bază și anume HTML, CSS și JavaScript, toate acestea formând un mediu propice pentru adăugarea de noi pagini web, noi moduri interactive de prezentarea a conținutului etc.;
- **nerespectarea programărilor și imposibilitatea de a crea una online** => deoarece programările nu sunt centralizate în vreun fel, modul de operare este haotic și duce la un disconfort accentuat atât pentru pacienți cât și pentru personalul medical; ca drept soluție, sistemul de programări online asigură un mod organizat și punctual de a trata pacienții, aceștia știind cu exactitate ora la care trebuie să se prezinte la cabinet, nu vor sta la cozi infernale, iar medicii își vor putea organiza ziua de lucru, fiind informați înainte câți pacienți vor avea de consultat respectiv tratat și care sunt aceștia;
- **digitalizarea proceselor birocratice legate de analize** => aplicația oferă facilitatea de a urmări recoltarea și procesarea analizelor, astfel pacientul și personalul medical nu mai trebuie să completeze și să "plimbe" numeroase formularea de la unii la alții, când totul se poate face automat la câteva click-uri distanță; totodată este posibilă și stocarea documentelor fiecărui pacient și accesare acesteia de către personalul medical, pacientul ne mai fiind nevoit să se plimbe cu dosarul conținând istoricul său medical, iar medicul care îl va aștepta la o eventuală consultație, îi va analiza înaintea programării ultimele analize și schema de tratament.

Utilizarea aplicației WellnessWave aduce, pe lângă soluțiile enumerate anterior, și o serie de avantaje care dovedesc, încă o dată, necesitatea unui astfel de sistem informatic din cadrul spitalelor de stat românești:

- un nivel mai rapid de procesare a datelor în cadrul unei consultații sau a efectuării analizelor;
- reducerea considerabilă a cantităților de maculatură utilizate în procesul birocratic;
- simplificarea atribuțiilor personalului medical, ducând la creșterea moralului acestora și implicit a calității muncii lor;
- permiterea transmiterii simple a documentelor între pacient și medic, între medic și asistenții care recoltează și manageriază analizele;
- eficientizarea prin înlăturarea timpului mort.

Concluzii și implementări viitoare

Aplicația WellnessWave a fost creată pentru a satisface nevoia unui sistem informatic eficient, stabil și rapid, nevoie regăsită în spitalele de stat din România. Ideea aplicației a pornit din analiza modului de lucru din Spitalul Clinic de Urgență "Sf. Spiridon" din Iași, prin prisma experienței personale și a schimburilor de opinii cu alți pacienți și diferiți angajați ai unității medicale.

Obiectivele pe care le-a urmărit acest proiect au fost atinse cu succes prin implementarea tuturor funcționalităților propuse. Cu toate acestea, unele din componentele aplicației prezintă posibilitatea de îmbunătățire, iar alte componente oferă posibilitatea de extindere sau scalabilitate în funcție de viitoarele nevoi ce vor apărea.

În cazul în care, în viitor, se va dori ca aplicația să fie folosită în fiecare spital de stat din România, aceasta va trebui să suporte mai multe modificări precum:

- crearea unor noi tabele în baza de date, care să cuprindă multitudinea de informații legate de fiecare spital și pacienții acestuia;
- crearea de clase noi în cadrul aplicației back-end pentru tabelele nou adăugate;
- regândirea modului de generare a programărilor;
- optimizarea traficului de date;
- refacerea arhitecturii pentru a putea satisface noile cerințe;
- modelarea aplicației front-end și a interfeței pentru a putea separa spitalele între ele.

Totodată, aceste modificări ar apărea în contextul centralizării tuturor spitalelor într-un singur sistem, însă este posibil ca fiecare spital să folosească individual aplicația WellnessWave fără probleme.

Un alt exemplu de facilitate care poate fi îmbunătățită în viitor este modul de utilizare a tehnologiei OCR. Aceasta ar implica posibilitatea transformării documentelor fizice în variante digitale pentru a avea acces facil la informațiile scante de pacienți într-un format editabil. Totodată, aplicarea în sine a algoritmilor poate fi îmbunătățită prin analizarea rezultatelor și identificarea potențialelor neajunsuri spre rectificarea lor.

Aplicația, pe lângă faptul că poate suporta îmbunătățiri ale variantei deja existente, este deschisă și spre adăugarea unor noi funcționalități precum: promovarea unor campanii de prevenție, generarea unor rapoarte statistice despre numeroase aspecte legate de sănătate publică, telemedicină etc.

În ceea ce privește promovarea campaniilor de prevenție, acestea sunt necesare pentru a aduce la cunoștința publicului larg riscurile și modurile în care aceștia se pot proteja împotriva factorilor de risc precum virusurile. Acest lucru s-a dovedit eficient în perioada pandemiei când statul român a recurs la mai multe moduri de diseminare a informațiilor cu privire la modul de protejarea împotriva SarsCov2. Printre acestea, s-a regăsit și platforma *fiipregatit.ro*[51]. Similar cu această abordare, se poate implementa și în cadrul WellnessWave un cumul de campanii atât pentru SarsCov2 cât și pentru alte afecțiuni, obiceiuri dăunătoare etc.

Cât despre generarea unor rapoarte statistice, implementarea unei astfel de facilități ar duce în primul rând la conștientizarea problemelor pe care le au cetățenii români, iar apoi la generarea unor măsuri care să ducă la rezolvarea sau prevenirea acestor probleme. De exemplu, dacă s-ar observa, pe baza unor astfel de rapoarte, o creștere a numărului de cazuri de gripă într-o anumită zonă, acest lucru va determina autoritățile responsabile să înainteze investigații și să genereze recomandări de purtare a măștii în spații închise (de exemplu) pentru a opri răspândirea unui astfel de virus. Astfel de lucruri pot salva societăți întregi de la supraaglomerarea spitalelor și implicit de la extenuarea personalului medical care este vital; practic se poate preveni situația care s-a observat în timpul pandemiei de SarsCov2, când spitalele din întreaga lume au fost depășite de situație.

În concluzie, modernizarea sistemelor informatice ale statului poate aduce foarte multe beneficii, însă adevărata provocare este punerea lor în aplicare astfel încât societatea să le accepte.

"It is not the strongest of the species that survives, nor the most intelligent, but the one most responsive to change." - Charles Darwin

[51] https://fiipregatit.ro/ghid/covid19-recomandari/

Bibliografie

[1] Paul, Ing & Bâru, Emanuel & Fetti, Alin & Lucia, Valeria & Nicula, & Voina-Tonea, Andrada. (2022). Studiu privind soluțiile digitale folosite în unitățile medicale private study on digital solutions used in private medical institutions;

[2] Cebotari (Moroi) Tatiana. Rolul investiţiilor în digitalizarea sistemului de ocrotire a sănătăţii în Republica Moldova, CZU: 336.531:614.2:004(478), JEL: H51, I15, Pag. 93-94,, Academia de Studii Economice din Moldova, Disponibil în IBN: 30 martie 2023;

[3] Gemma A.Williams, Nick Fahy, Dalhia Aissat, Marie-Camille Lenormand, Louisa Stüwe, Isabelle Zablit-Schmidt, Samuel Delafuys, Yann-Maël Le Douarin and Natasha Azzopardi Muscat. Covid-19 and the use of digital health tools: opportunity amid crisis that could transform health care delivery;

[4] Cristian Vlădescu, Silvia Gabriela Scîntee, Victor Olsavszky, Cristina Hernández-Quevedo and Anna Sagan. Romania: health system review;

[5] MANDA, C. C. (2023). Digitalizarea administrației publice din România – între nevoile si aspiratiile unei societăți moderne a secolului XXI. *Smart Cities International Conference (SCIC) Proceedings*, *9*, 41–61;

[6] Panait, Nicoleta & Rădoi, Mădălina. (2022). The degree of digitalization of public services in Romania. Management & Marketing. Challenges for the Knowledge Society. 14. 875;

[7] Abdullah, Ako. (2017). Advanced Encryption Standard (AES) Algorithm to Encrypt and Decrypt Data;

[8] R. Smith, "An Overview of the Tesseract OCR Engine," Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), Curitiba, Brazil, 2007, pp. 629-633, doi: 10.1109/ICDAR.2007.4376991 ;

[9] Smith, K.R., Grant, S. & Thomas, R.E. Testing the public's response to receiving severe flood warnings using simulated cell broadcast. Nat Hazards 112, 1611–1631 (2022). https://doi.org/10.1007/s11069-022-05241-x;

[10] Quinn, P. Research under the GDPR – a level playing field for public and private sector research?. Life Sci Soc Policy 17, 4 (2021);

[11] Harjinder Singh Lallie, Lynsay A. Shepherd, Jason R.C. Nurse, Arnau Erola, Gregory Epiphaniou, Carsten Maple, Xavier Bellekens. Cyber security in the age of COVID-19: A timeline and analysis of cyber-crime and cyber-attacks during the pandemic, Computers & Security, Volume 105, 2021, 102248, ISSN 0167-4048;

[12] Bellare, M., Kilian, J., Rogaway, P. (1994). The Security of Cipher Block Chaining. In: Desmedt, Y.G. (eds) Advances in Cryptology — CRYPTO ’94. CRYPTO 1994. Lecture Notes in Computer Science, vol 839. Springer, Berlin, Heidelberg;

[13] Gilbert, H., Handschuh, H. (2004). Security Analysis of SHA-256 and Sisters. In: Matsui, M., Zuccherato, R.J. (eds) Selected Areas in Cryptography. SAC 2003. Lecture Notes in Computer Science, vol 3006. Springer, Berlin, Heidelberg;

[14] Sumartono, Isnar & Siahaan, Andysah Putera Utama & Arpan, Arpan. (2016). Base64 Character Encoding and Decoding Modeling. International Journal of Recent Trends in Engineering & Research. 2. 63-68;

[15] Jamshed Memon, Maira Sami, Rizwan Ahmed Khan. (2020) Handwritten Optical Character Recognition (OCR): A Comprehensive Systematic Literature Review (SLR) in Computer Vision and Pattern Recognition - Cornell University;

[16] Zhang, Xun & Bai, Wanrong & Cui, Haoyang. (2023). Review of Optical Character Recognition for Power System Image Based on Artificial Intelligence Algorithm. Energy Engineering. 120. 665-679. 10.32604/ee.2023.020342.

Bibliografie web

[1] Ministerul Sănătății:
https://www.ms.ro/
[2] Platforma Ro-Alert:
https://ro-alert.ro/
[3] Societatea Română de Medicină Preventivă:
https://srmp.ro/
[4] Asociația Română de Sănătate Publică și Management Sanitar:
https://arspms.ro/
[5] Centru Pentru Politici și Servicii de Sănătate:
https://cpss.ro/ro/index.php
[6] Comisia Europeană:
https://digital-strategy.ec.europa.eu/ro/policies
[7] Platforma Națională Pentru Plătirea Taxelor și Impozitelor:
https://www.ghiseul.ro/ghiseul/public/
[8] Java:
https://www.java.com/en/
[9] MySql:
https://www.mysql.com/
[10] H.T.M.L.:
https://html.com/
[11] JavaScript:
https://javascript.info/
[12] Pagina Spitalului Clinic Județean de Urgență “Sf. Spiridon” din Iași:
https://www.spitalspiridon.ro/
[13] Curs Programare Avansata:
https://profs.info.uaic.ro/~acf/java/Cristian_Frasinaru-Curs_practic_de_Java.pdf
[14] Documentație Oracle JPA:
https://docs.oracle.com/javaee/6/tutorial/doc/bnbpz.html
[15] Documentație Baeldung JPA:
https://www.baeldung.com/learn-jpa-hibernate
[16] Documentație Project Lombok:
https://projectlombok.org
[17] Documentație Spring:
https://docs.spring.io
[18] Consorțiul WWW (World Wide Web W3C):
https://www.w3.org
[19] Live Server Visual Studio Code:

https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer
[20] Regulament G.D.P.R.:
https://www.dataprotection.ro/?page=noua%20_pagina_regulamentul_GDPR
[21] Documentatie P.K.C.S.:
https://www.rfc-editor.org/rfc/rfc2898
[22] Documentatie I.V.:
https://www.techtarget.com/whatis/definition/initialization-vector-IV
[23] Documentatie Amazon O.R.C.:
https://aws.amazon.com/what-is/ocr/
[24] Articol despre Istoria O.C.R.:
https://www.veryfi.com/ocr-api-platform/history-of-ocr/
[25] GitHub Tesseract:
https://github.com/tesseract-ocr/tesseract
[26] Abby FineReader:
https://pdf.abbyy.com/
[27] OpenCV:
https://opencv.org/
[28] Articol IBM despre Computer Vision:
https://www.ibm.com/topics/computer-vision
[29] Documentație Clasa Rect OpenCv
https://docs.opencv.org/3.4/javadoc/org/opencv/core/Rect.html
[30] Platforma Națională Pentru Situații de Urgență:
https://fiipregatit.ro/ghid/covid19-recomandari/

Anexa 1: Cod Criptare AES

Cod generare cheie + constante:

```
private static final int SALT_LENGTH = 16;
private static final int ITERATIONS = 100000;
private static final int KEY_LENGTH = 256;
private static final String SECRET_KEY_FILENAME = "secret.key";
private static final String ALGORITHM = "AES/CBC/PKCS5Padding";
private static final byte[] IV = "1234567890123456".getBytes();

    public static SecretKey generateKey()  {
        try {
            char[] password = "mysecretpassword".toCharArray();
            byte[] salt = new byte[SALT_LENGTH];
            SecureRandom secureRandom = new SecureRandom();
            secureRandom.nextBytes(salt);
            SecretKeyFactory factory =
SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256");
            KeySpec spec = new PBEKeySpec(password, salt, ITERATIONS, KEY_LENGTH);
            SecretKey secretKey = new
SecretKeySpec(factory.generateSecret(spec).getEncoded(),
                                                                "AES");
            Files.write(Paths.get(SECRET_KEY_FILENAME), secretKey.getEncoded());
            return secretKey;
        }
        catch (Exception e){
            e.printStackTrace();
        }
        return null;
    }
```

Dupa ce cheia a fost createa, apoi se pot utiliza metoda de **extragerea** cheii din fișier:

```
public static SecretKey getSecretKey() {
        try {
            byte[] keyBytes = Files.readAllBytes(Paths.get(SECRET_KEY_FILENAME));
            return new SecretKeySpec(keyBytes, "AES");
        }
        catch (Exception e){
            e.printStackTrace();
        }
        return null;
    }
```

După ce cheia a fost extrasă din fișier, se pot apela metodele de *encrypt* si *decrypt.*

Metoda encrypt:

```
public static String encrypt(String input)  {
        try {
            Cipher cipher = Cipher.getInstance(ALGORITHM);
            cipher.init(Cipher.ENCRYPT_MODE, getSecretKey(), new
IvParameterSpec(IV));
            byte[] encrypted =
cipher.doFinal(input.getBytes(StandardCharsets.UTF_8));
            byte[] combined = new byte[IV.length + encrypted.length];
            System.arraycopy(IV, 0, combined, 0, IV.length);
            System.arraycopy(encrypted, 0, combined, IV.length, encrypted.length);
            return Base64.getEncoder().encodeToString(combined);
        }
        catch (Exception e){
            e.printStackTrace();
        }
        return null;
    }
```

Metoda decrypt:

```
public static String decrypt(String input) {
        try {
            byte[] combined = Base64.getDecoder().decode(input);
            System.arraycopy(combined, 0, IV, 0, IV.length);
            byte[] encrypted = new byte[combined.length - IV.length];
            System.arraycopy(combined, IV.length, encrypted, 0, encrypted.length);
            Cipher cipher = Cipher.getInstance(ALGORITHM);
            cipher.init(Cipher.DECRYPT_MODE, getSecretKey(), new
IvParameterSpec(IV));
            byte[] decrypted = cipher.doFinal(encrypted);


            return new String(decrypted, StandardCharsets.UTF_8);
        }
        catch (Exception e){
            e.printStackTrace();
        }
        return null;
    }
```

Anexa 2: Utilizare OCR

Metoda *extractString* din clasa *OcrTools*, pachetul *Utils*. Această metodă primește ca parametru un document de tip imagine și returnează șirul de caractere rezultat aplicând tehnologia OCR.

```
public  String extractString(MultipartFile file) {
   try {
       Path tempFilePath = null;
       tempFilePath = Files.createTempFile("temp", file.getOriginalFilename());
       File tempFile = tempFilePath.toFile();
       Files.copy(file.getInputStream(), tempFilePath,
StandardCopyOption.REPLACE_EXISTING);

       // Read the image data from the temporary file into a Mat object
       Mat image = Imgcodecs.imread(tempFile.getAbsolutePath());

       // Calculate the dimensions of the rectangle to be extracted
       int width = (int) (105 * image.cols() / 210 * 1.6); // Multiply by scaling
factor of 1.1
       int height = (int) (75 * image.rows() / 297 * 1.6); // Multiply by scaling
factor of 1.1

       // Define the ROI (Region of Interest) as a rectangle
       int x = (image.cols() - width) / 2;
       int y = (image.rows() - height) / 2;
       Rect roi = new Rect(x, y, width, height);

       // Extract the ROI from the original image
       Mat cropped_image = new Mat(image, roi);

       // Display the cropped image
       Imgcodecs.imwrite("cropped_image.jpeg", cropped_image);

       File imageFile = new File("cropped_image.jpeg");
       ITesseract tesseract = new Tesseract();

       try {
           // Set the path to the tessdata directory containing the language data
files
           tesseract.setDatapath("C:/Program Files/Tesseract-OCR/tessdata");

           // Set the language to use for OCR (English in this example)
           tesseract.setLanguage("ron");
```

```
            // Perform OCR on the cropped image
            String result = tesseract.doOCR(imageFile);

            // Print the OCR result
            tempFile.delete();
            return result;
        } catch (TesseractException e) {
            e.printStackTrace();
        }
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
    return "Error";
}
```