

**Universitatea  
Transilvania  
din Brașov**

**FACULTATEA DE INGINERIE ELECTRICĂ  
ȘI ȘTIINȚA CALCULATOARELOR**

# PROIECT DE DIPLOMĂ

Conducător științific:

Șef lucr. Dr. Ing. POPA Luminița

Absolvent:

Popescu Ștefan-Tudor

**BRAȘOV, 2024**

Departamentul: Automatică și Tehnologia Informație

Programul de studii: Tehnologia Informației

*POPESCU Ștefan-Tudor*



# Aplicație informatică medicală dedicată pentru gestionarea și informarea pacienților







Conducător științific:

Șef lucr. Dr. Ing. *POPA Luminița*

Brașov, 2024

### FIȘA PROIECTULUI DE DIPLOMĂ

Universitatea Transilvania din Brașov	Proiect de diplomă nr. ....
Facultatea de Inginerie Electrică și Știința Calculatoarelor	
Departamentul de Automatică și Tehnologia Informației	Viza facultății
Programul de studii: Tehnologia Informației	Anul universitar: 2023 - 2024
Candidat: Popescu Ștefan-Tudor	Promoția: 2024
Conducător științific: Șef lucr. Dr. Ing. Luminița Popa	
<b>PROIECT DE DIPLOMĂ</b>	
Titlul lucrării: <i>Aplicație informatică medicală dedicată pentru gestionarea și informarea pacienților</i>	
Problemele principale tratate: 1. Documentarea și studierea tehnologiilor necesare pentru dezvoltarea proiectului. 2. Arhitectura și organizarea aplicației 3. Implementarea Aplicației Web. 4. Implementarea modelului de inteligență artificială 5. Concluzii și contribuții personale;	
Locul și durata practicii: ICDT L6, Laborator Sisteme Informatică – VIV5A, octombrie 2023-iunie 2024	
Bibliografie: [1] S. H. Edward, C. J. James și C. F. Michael, Biomedical Informatics: Computer Applications in Health Care and Biomedicine, Springer, 2021.; [2] P. Zhao, I. Yoo, J. Lavoie, B. J. Lavoie și E. Simoes, „Web-Based Medical Appointment Systems: A Systematic Review,” <i>Journal of Medical Internet Research</i> , vol. 19, nr. 4, 2017.; [3] Simonyan și A. Zisserman, „Very Deep Convolutional Networks For Large-Scale Image Recognition,” <i>arXiv</i> , 10 April 2015. [4] Szegedy, V. Vanhoucke, S. Ioffe și J. Shlens, „Rethinking the Inception Architecture for Computer Vision,” <i>arXiv</i> , 11 December 2015.	
Aspecte particulare: - Platforma este concepută pentru a facilita programarea rapidă și ușoară pentru pacienți și pentru a oferi o interfață funcțională și estetică pentru practicienii medicali. De asemenea, utilizează inteligența artificială pentru a oferi diagnostice prezumtive ale leziunilor cutanate, informând astfel publicul larg despre posibilele probleme de sănătate.	
Primit tema la data de: 03.10.2023	
Data predării lucrării: 14.06.2023	
Director departament, prof. dr. ing. Sorin-Aurel MORARU	Conducător științific, Șef lucr.dr.ing. Luminița Popa 
Candidat, Popescu Ștefan-Tudor	

PROIECT DE DIPLOMĂ – VIZE		
Data vizei	Capitole/ problemele analizate	Semnătura conducătorului științific
15.12.2023	Stadiul actual și documentare asupra tematicii	
8.03.2024	Stabilirea tehnologiilor și proiectarea arhitecturii aplicației	
26.04.2024	Implementarea aplicației Web	
20 .05.2024	Implementarea modelului de inteligență artificială	
12.06.2024	<b>Finalizare documentatie si Verificare procentaj similaritate Turnitin (total ≤15%și o singura sursa ≤5%)-3%</b>	
<b>APRECIEREA ȘI AVIZUL CONDUCĂTORULUI ȘTIINȚIFIC</b>		
<p><i>Tema lucrării de licență este de actualitate, are aplicabilitate practică și este tratată corespunzător. Sunt utilizate tehnologii de actualitate. În lucrare sunt evidențiate contribuțiile personale ale autorului. Resursele bibliografice sunt numeroase ,adecvate și actuale. Proiectul de licență respectă cerințele impuse prin temă precum și cele privind elaborarea și redactarea stabilite la nivelul facultății.Lucrarea îndeplinește condițiile științifice și metodologice pentru a fi susținută în fața comisiei de licență.</i></p>		
Data: ..... 12.06.2024.....	ADMIS pentru susținere/ RESPINS	Conducător științific, Șef lucr.dr.ing. Luminița Popa 
<b>AVIZUL DIRECTORULUI DE DEPARTAMENT</b>		
Data: .....	ADMIS pentru susținere/ RESPINS	Director departament, prof. dr. ing. Sorin-Aurel MORARU
<b>SUSȚINEREA PROIECTULUI DE DIPLOMĂ</b>		
Sesiunea: iunie 2024.		
Rezultatul susținerii	PROMOVAT cu media:	
	RESPINS <b>cu</b> refacerea lucrării	
	RESPINS <b>fără</b> refacerea lucrării	
Președinte de comisie, prof. dr. ing. Sorin-Aurel MORARU .....(semnatura)		

# CUPRINS

---

## CUPRINS

Cuprins.....	5
Rezumat .....	8
Abstract .....	8
Listă de Figuri, Tabele și Coduri Sursă .....	10
Lista de Acronime.....	13
1    Introducere.....	15
1.1 Motivația alegerii proiectului .....	15
1.2 Scopul și obiectivele proiectului.....	16
1.3 Domeniul de aplicabilitate .....	17
1.4 Stadiul actual al problemei abordate.....	17
2    Arhitectura și organizarea aplicației .....	18
2.1 Arhitectura generală a aplicației .....	18
2.2 Tehnologii folosite.....	18
2.2.1 Angular .....	18
2.2.2 RxJS .....	21
2.2.3 Typescript .....	21
2.2.4 Angular Material .....	22
2.2.5 Bootstrap .....	22
2.2.6 Firebase .....	23
2.2.7 TensorFlow & Keras .....	25
2.2.8 VGG19.....	25
2.2.9 ResNet50.....	25
2.2.10 InceptionV3.....	26
2.2.11 InceptionResNetV2.....	26
2.2.12 EfficientNet.....	26
2.3 Organizarea Proiectului .....	27
2.3.1 Frontend .....	27
2.3.2 Backend.....	29
2.3.3 Inteligență artificială .....	31

3	Implementarea Aplicației Web.....	32
3.1	Elemente comune.....	32
3.2	Angular Routing .....	32
3.3	Pagina Principală (Home).....	35
3.4	Pagina de Medici (Medics).....	36
3.5	Pagina de Autentificare.....	40
3.6	Pagina de Profil.....	44
3.7	Pagina de Programări (Appointments) .....	46
3.7.1	Secțiunea “Create an appointment” .....	46
3.7.2	Secțiunea “View your appointments” .....	53
3.8	Pagina de Program (Schedule).....	54
3.8.1	“Schedule Table” .....	55
3.8.2	“Schedule Details” .....	58
3.9	Pagina profilului de medic (Medic-Profile).....	60
3.10	Pagina despre (About) .....	63
3.11	Pagina de contact (Contact) .....	65
3.12	Pagina de analiză a leziunilor pielii (Skin Lesion Analyser).....	66
3.13	Pagina neautorizat (Not Authorized) .....	68
3.14	Gestionarea și tratarea erorilor.....	69
3.14.1	Securitate și Protejare Backend .....	70
4	Antrenarea modelului de inteligență artificială .....	72
4.1	Procurarea datelor .....	72
4.2	Prelucrarea datelor .....	72
4.2.1	Augmentarea datelor .....	73
4.3	Antrenarea modelelor .....	74
4.3.1	Pregătirea datelor pentru antrenare .....	74
4.3.2	Modelele antrenate.....	75
4.3.3	Antrenarea modelelor.....	75
4.4	Utilizarea metodelor callback .....	79
4.4.1	Scăderea ratei de învățare .....	79
4.4.2	Early stopping & Model checkpoint .....	80
4.5	Afișarea rezultatelor.....	81
4.6	Rezultate obținute și comparații între modele .....	84
5	Concluzii .....	88
6		

5.1 Concluzii Generale .....	88
5.2 Potențiale Îmbunătățiri și Direcții Viitoare de Cercetare și Dezvoltare .....	89
5.2.1 Potențiale Îmbunătățiri pentru Aplicația Web .....	89
5.2.2 Potențiale Îmbunătățiri pentru Componenta de Inteligență Artificială.....	90
Bibliografie .....	91

## REZUMAT

În domeniul sănătății, relația dintre pacienți și sistemul medical prezintă adesea provocări semnificative. Aceste dificultăți sunt accentuate de intervalele prelungite dintre debutul simptomelor și primirea îngrijirii medicale de specialitate, alături de ineficiențele administrative, ceea ce duce la frustrarea pacienților și deteriorarea stării lor de sănătate. În răspuns la aceste probleme, această lucrare prezintă dezvoltarea unei platforme informatice medicale moderne, menită să îmbunătățească gestionarea programărilor pacienților și să furnizeze informații relevante despre sănătate. Platforma este concepută pentru a facilita programarea rapidă și ușoară pentru pacienți și pentru a oferi o interfață funcțională și estetică pentru practicienii medicali. De asemenea, utilizează inteligența artificială pentru a oferi diagnostice prezumtive ale leziunilor cutanate, informând astfel publicul larg despre posibilele probleme de sănătate.

Proiectul se bazează pe o aplicație web dezvoltată folosind Angular și Firebase pentru backend, integrând perfect modele AI antrenate separat. Această aplicație unificată asigură utilizatorilor accesul la toate funcționalitățile într-un singur loc. Acest proiect nu doar abordează ineficiențele actuale ale sistemului de sănătate, ci și stabilește o fundație pentru avansările viitoare în informatica medicală și tehnologia de îngrijire a pacienților.

## ABSTRACT

In the realm of healthcare, the relationship between patients and the medical system often poses significant challenges. These difficulties are accentuated by the prolonged intervals between the onset of symptoms and the receipt of specialized medical care, coupled with administrative inefficiencies, leading to patient frustration and deteriorating health conditions. In response to these issues, this thesis presents the development of a modern medical information platform aimed at enhancing patient appointment management and providing pertinent health information. The platform is designed to facilitate quick and easy scheduling for patients and offer a functional and aesthetically pleasing interface for medical practitioners. Additionally, it employs artificial intelligence to deliver presumptive diagnoses of skin lesions, thereby informing the general public about potential health concerns.



The core of the project involves a web application developed using Angular and Firebase for the backend, seamlessly integrating separately trained AI models. This unified application ensures users have access to all functionalities in one place. This project not only addresses current inefficiencies in the healthcare system but also sets a foundation for future advancements in medical informatics and patient care technology.

## LISTĂ DE FIGURI, TABELE ȘI CODURI SURSĂ

### FIGURI

Figura 2-1. Structura folderelor serviciului de stocare.

Figura 2-2. Imaginile trimise de pacienți la crearea unei programări.

Figura 2-3. Structura folderelor aplicației Angular.

Figura 2-4. Structura de organizare a proiectului Firebase.

Figura 2-5. Utilizarea resurselor proiectului Firebase.

Figura 3-1. Antetul (Header) aplicației web pentru un utilizator cu rol de medic.

Figura 3-2. Subsolul (Footer) aplicației web.

Figura 3-5. Pagina principală (Home).

Figura 3-6. Pagina de Medici.

Figura 3-7. Structura unui document de tip medic în baza de date Firestore.

Figura 3-9. Aspectul paginii de medici pentru un utilizator cu rolul de medic.

Figura 3-10. Interfața paginii de autentificare.

Figura 3-12. Erori afișate utilizatorului în cazul introducerii de valori invalide.

Figura 3-13. Interfața de vizionare și gestionare a utilizatorilor autentificați din Firebase.

Figura 3-14. Pagina de Profil.

Figura 3-15. Mesajul de succes afișat folosind Snackbar.

Figura 3-16. Formularul din secțiunea "Create an Appointment".

Figura 3-17. Diagrama care exemplifică completarea automată a datelor unei programări.

Figura 3-18. Câmpurile de tip dropdown coordonate. Se poate observa că nu se poate selecta ora 09:20 deoarece medicul selectat are deja o consultație programată la acea oră.

Figura 3-19. Structura bazei de date Cloud Firestore pentru programări.

Figura 3-20. Secțiunea "View your Appointments". În acest caz, utilizatorul a fost redirecționat automat după crearea unei programări (se poate observa un snackbar informativ).

Figura 3-21. "Deschiderea" unei programări din secțiunea "View your appointments" pentru a vedea detaliile acesteia.

Figura 3-22. Interfața paginii de program a medicului.

Figura 3-23. Componenta de selectare a unei date calendaristice (Datepicker).

Figura 3-24. Componenta "Dialog" care afișează detalii despre programare.

Figura 3-25. Componenta dialog în cazul în care pacientul a trimis o poză la crearea programării.

Figura 3-26. Predicțiile modelului pentru imaginea data.

Figura 3-28. Pagina de profil a medicului.

Figura 3-29. Pagina "About".

Figura 3-30. Pagina "Skin Lesion Analyser", după ce utilizatorul a încărcat o imagine.

Figura 3-31. Componenta dialog cu informații detaliate despre fiecare tip de leziune.

Figura 3-32. Pagina "Not Authorized" pentru un utilizator autentificat, dar care nu deține permisiunile necesare pentru a accesa pagina dorită.

Figura 3-33. Mesaj de eroare afișat utilizatorului prin intermediul unui snackbar.

Figura 4-1. Exemplu de imagini augmentate folosind operațiile menționate anterior.

Figura 4-8. Exemplu de grafice folosite pentru a evalua modelele antrenate.

Figura 4-9. Exemple de valori pe care le-am afișat după antrenarea unui model.

Figura 4-10. Exemplu de matrice de confuzie pentru un model antrenat.

Figura 4-13. Valorile metricilor modelelor antrenate.

Figura 4-14. Graficele de antrenare a modelului ResNet50.

Figura 4-15. Matricea de confuzie a modelului ResNet50.

## TABELE

Figura 4-11. Valorile de eroare și acuratețe a modelelor antrenate.

Figura 4-12. Valorile de precizie, rechemare și scor F1 a modelelor antrenate.

## CODURI SURSĂ

Figura 3-3. Definirea rutelor folosind Angular Routing.

Figura 3-4. Exemplu de utilizare a directivei RouterLink și a parametrilor de rută.

Figura 3-8. Exemplu de utilizare a directivelor ngIf și ngFor.

Figura 3-11. Generarea Reactive Form pentru pagina de autentificare.

Figura 3-27. Codul folosit pentru a extrage numele orașelor.

Figura 4-2. Preprocesarea datelor pentru modelul EfficientNet.

Figura 4-3. Optimizarea performanței folosind tf.data.AUTOTUNE.

Figura 4-4. Definirea modelului de bază pentru modelul EfficientNetB0.

Figura 4-5. Definirea straturilor adăugate modelelor.

Figura 4-6. Definirea funcției de scădere a ratei de învățare.

Figura 4-7. Definirea metodelor callback.

## LISTA DE ACRONIME

SPA – Single Page Application

HTML – HyperText Markup Language

CSS - Cascading Style Sheets

API - Application Programming Interface

DOM - Document Object Model

RxJS - The Reactive Extensions for JavaScript

HTTP - Hypertext Transfer Protocol

UI - User Interface

BaaS - Backend as a Service

SQL - Structured Query Language

AI – Artificial Intelligence

CNN - Convolutional Neural Network

NPM - Node Package Manager

CLI - Command Line Interface

TS – Typescript

IDE - Integrated Development Environment

URL - Uniform Resource Locator (aka Universal Resource Locator)

CTA - Call to Action

JSON - JavaScript Object Notation

HAM – Human Against Machine

GPU - Graphics Processing Unit

RGB - Red-Green-Blue

TF – TensorFlow

ReLU - Rectified Linear Unit

LR – Learning Rate

# 1 INTRODUCERE

## 1.1 MOTIVAȚIA ALEGERII PROIECTULUI

Informatica medicală, precum și informatica sănătății sunt discipline care îmbină domeniul medical cu cel informatic, într-un efort de a îmbunătăți ajutorul medical acordat pacienților. Aceste discipline se bazează pe expertiza ambelor discipline pentru a folosi tehnologia pentru grija pacienților, diagnosticare și cercetare.

Evoluarea rapidă a acestor domenii se datorează în mare parte digitalizării și numărului monumental de date generate în ziua de azi. Volumul datelor legate de medicină și sănătate s-a înmulțit de 8 ori din 2013 până în 2018 și se estimează că va crește la o rată anuală compusă de 36% între 2018 și 2025 [1]. Astfel, importanța informaticii sănătății crește în mod constant pe măsură ce industria sănătății devine din ce în ce mai complexă și bazată pe date.

O soluție informatică în domeniul medical poate furniza suport extins pentru pacienți, incluzând categorii vulnerabile care pot fi afectate de dizabilități ce le restricționează capacitatea de a solicita și căuta asistență, sau care pot locui în regiuni cu acces limitat la servicii medicale specializate. De asemenea, implementarea unei astfel de soluții informatice poate optimiza eficiența lucrătorilor medicali prin gestionarea programărilor pacienților chiar și în afara unui program uzual de muncă.

Lumea vastă a îngrijirii sănătății poate fi greu de înțeles pentru pacienții fără o pregătire medicală care de multe ori aleg să nu meargă la controale decât dacă se ajunge într-o situație gravă. De asemenea, o mare parte din timpul lucrătorilor medicali este alocat gestionării pacienților, a programărilor acestora și pentru înscrierea consultațiilor în sistemul național de bază medical. Informarea mai bună a pacienților, precum și îmbunătățirea modului prin care sunt gestionati aceștia este un aspect important al sistemului de sănătate, și aceste lucruri pot beneficia cel mai mult de ajutor din partea domeniului informatic. Alte aplicații ale informaticii în medicină sunt:

- Crearea, menținerea sau facilitarea de noi modalități pentru a înregistra și păstra datele medicale electronice.
- Îmbunătățirea comunicării medic-medici și medic-pacient
- Stocarea, gestionarea și analiza datelor pentru cercetare.

Implementarea programărilor prin intermediul internetului, aduc beneficii cum ar fi rata redusă de neprezentare, scăderea muncii necesare a personalului medical, scăderea timpului de așteptare și o satisfacție îmbunătățită a pacienților [2].

Am decis să elaborez acest proiect pentru a simplifica și accelera întregul proces, oferind suport tuturor utilizatorilor. Observând provocările persistente și decalajele existente în infrastructura medicală din România, am dorit să contribui la crearea unei soluții inovatoare și necesare.

Proiectarea platformei ia în considerare nevoile specifice ale ambelor categorii de utilizatori, personal medical și pacienți, oferind o experiență intuitivă, ușor de utilizat și accesibilă chiar și pentru cei cu abilități digitale limitate. Prioritizarea clarității, simplității și eficienței este un considerent esențial în dezvoltarea acestei aplicații.

## 1.2 SCOPUL ȘI OBIECTIVELE PROIECTULUI

Scopul acestui proiect constă în dezvoltarea și implementarea unei aplicații medicale cu țelul de a îmbunătăți accesul, eficiența și calitatea serviciilor medicale, atât pentru personalul medical, cât și pentru pacienți.

Primul obiectiv al aplicației este dezvoltarea unei interfețe intuitive care să permită pacienților să acceseze informații de sănătate relevante, să efectueze autoevaluări și să inițieze comunicarea cu profesioniștii medicali într-un timp cât mai scurt.

Al doilea obiectiv constă în implementarea unui sistem de programare online care să faciliteze și să eficientizeze procesul de gestionare a programărilor pacienților pentru medici și personalul medical.

Al treilea obiectiv este reprezentat de folosirea inteligenței artificiale pentru informarea și diagnosticarea rapidă și corectă a pacienților.

Un ultim obiectiv constă în dezvoltarea unui sistem de colectare și analiză a datelor medicale pentru a susține luarea deciziilor bazate pe dovezi și pentru a ușura procesul de analize medicale.



### 1.3 DOMENIUL DE APLICABILITATE

Acest proiect vizează domeniul medical, fiind concepută pentru furnizorii de servicii medicale și destinată atât utilizării de către pacienți, cât și de către medici.

Publicul țintă primar este format din pacienți, medici, asistenți medicali, furnizori medicali, iar cel secundar este format din personal administrativ, pasionați de sănătate și furnizori de asigurări.

### 1.4 STADIUL ACTUAL AL PROBLEMEI ABORDATE

Infrastructura medicală din România se află într-un stadiu de tranziție, cu îmbunătățiri semnificative în ultimii ani, dar și cu provocări persistente. Pe plan informatic, infrastructura medicală din România se află într-o etapă de dezvoltare, cu progrese înregistrate în digitalizarea sistemelor medicale, dar și cu decalaje față de alte țări europene.

Platforma propusă vizează două categorii principale de utilizatori: pacienții și personalul medical. Pacienții reprezintă segmentul principal al publicului țintă, iar o parte semnificativă a pacienților sunt persoane vârstnice. Prin urmare, aceștia au nevoie de o interfață simplă și intuitivă, ușor de navigat și utilizat. Informațiile vor fi clare și concise, prezentate într-un limbaj accesibil, iar funcționalitățile ușor de înțeles și de utilizat, chiar și pentru cei cu abilități digitale limitate.

Alt segment semnificativ al utilizatorilor este reprezentat personalul medical, care utilizează platformele digitale pentru a accesa informații despre pacienți, a oferi consultații și a gestiona diverse aspecte ale practicii medicale. Deși pregătirea lor include și deprinderi tehnice și informatice, unii utilizatori pot avea cunoștințe limitate în domeniul tehnologiei. Din aceste motive, aceștia au nevoie de o interfață eficientă și organizată, care permite acces rapid la informațiile relevante și funcționalități specifice pentru gestionarea pacienților și a documentelor medicale și suport și instruire pentru a facilita adaptarea la platformele digitale.

## 2 ARHITECTURA ȘI ORGANIZAREA APLICAȚIEI

### 2.1 ARHITECTURA GENERALĂ A APLICAȚIEI

S-a optat pentru o aplicație web datorită facilității implementării, atât pentru personalul medical, cât și pentru utilizatorii obișnuiți. Această alegere elimină necesitatea utilizării unor componente hardware adiționale, cu costuri adiționale, contribuind la eficiența și accesibilitatea sporită a platformei.

Având în vedere amploarea sectorului medical, care implică un număr considerabil de utilizatori, și importanța crucială a domeniului, scalabilitatea aplicației devine un aspect fundamental. Am conceput și dezvoltat această soluție cu scalabilitatea ca principiu de bază, ceea ce se reflectă și în tehnologiile folosite, asigurând capacitatea sa de a se adapta și de a evolua odată cu nevoile în creștere ale sistemului medical.

Arhitectura aplicației integrate este dezvoltată utilizând Angular, Angular Material, Bootstrap și Firebase și combină eficient aspecte de design, interactivitate și gestionare a datelor. Această abordare permite crearea unei aplicații web moderne, scalabile și prietenoase cu utilizatorul.

### 2.2 TEHNOLOGII FOLOSITE

#### 2.2.1 Angular

Angular [3] este un framework frontend open-source dezvoltat de către Google, utilizat pentru a construi aplicații web single-page (SPA) moderne și interactive. Este construit pe TypeScript, un limbaj de programare bazat pe JavaScript care oferă caracteristici suplimentare precum tipuri statice și clase. Angular oferă o gamă largă de caracteristici care simplifică dezvoltarea SPA-urilor complexe.

Componentele [4] reprezintă elementele de bază ale arhitecturii unei aplicații Angular, jucând un rol central în construirea interfeței utilizatorului și în organizarea codului. O componentă în Angular este o entitate independentă, care combină HTML-ul, CSS-ul și logica JavaScript/TypeScript pentru a crea o unitate funcțională și reutilizabilă.

O caracteristică importantă a componentelor în Angular este capacitatea de a fi reutilizabile. Acestea pot fi încapsulate și utilizate în întreaga aplicație, sau pot fi exportate sub formă de module pentru a fi împărtășite între diferite aplicații.

Data binding-ul reprezintă o caracteristică centrală a componentelor, asigurând sincronizarea automată a datelor din modelul aplicației cu elementele vizuale din interfața utilizatorului. Aceasta oferă o metodă eficientă de gestionare a interacțiunii între datele aplicației și afișarea lor în interfața grafică.

Serviciile [5] în Angular reprezintă o parte esențială a arhitecturii framework-ului, facilitând separarea preocupărilor și promovând reutilizarea codului. Un serviciu în Angular este o clasă care furnizează funcționalități specifice și poate fi injectată în componente, alte servicii sau directive. Această abordare permite dezvoltatorilor să extragă logica de afaceri sau funcționalitățile partajate din componente, îmbunătățind astfel organizarea și modularitatea aplicației.

Serviciile sunt utilizate pentru o varietate de scopuri, cum ar fi comunicarea cu servere externe prin intermediul API-urilor, gestionarea stării aplicației, implementarea autentificării, și realizarea unor calcule complexe. Prin centralizarea acestor funcționalități într-un serviciu, codul devine mai ușor de întreținut, testat și refolosit.

Directivele [6] reprezintă unul dintre conceptele fundamentale în Angular, oferind un mod puternic și flexibil de a manipula DOM-ul și de a adăuga comportamente personalizate în aplicații. În esență, directivele sunt instrucțiuni care extind funcționalitatea HTML-ului standard, permițând dezvoltatorilor să creeze interfețe bogate și interactivitate în aplicațiile lor web.

Directivele de atribut sunt cele mai comune și sunt utilizate pentru a modifica comportamentul și aspectul elementelor HTML existente. Acestea sunt aplicate folosind un atribut specific în etichetele HTML și pot fi utilizate pentru a atașa comportamente personalizate, cum ar fi manipularea evenimentelor, ascunderea sau afișarea elementelor, sau aplicarea stilurilor specifice.

Pe de altă parte, directivele structurale sunt utilizate pentru a modifica structura DOM-ului, adăugând sau eliminând elemente HTML întregi. Cele mai cunoscute directive structurale în Angular sunt "ngIf", "ngFor" și "ngSwitch", care permit adăugarea și eliminarea dinamică a elementelor bazate pe anumite condiții sau pe date din modelul aplicației.

“Pipes” [7] în Angular reprezintă o caracteristică esențială pentru transformarea datelor în șabloane, oferind o modalitate simplă și eficientă de a efectua transformări complexe direct în codul HTML. Acestea permit dezvoltatorilor să modifice aspectul și formatul datelor afișate utilizând o sintaxă concisă și declarativă. Prin utilizarea țevilor, dezvoltatorii pot aplica formate specifice asupra datelor fără a necesita logică suplimentară în componentă, ceea ce contribuie la separarea clară a preocupărilor și la menținerea unui cod curat și ușor de întreținut.

Un aspect important al țevilor în Angular este capacitatea de a gestiona schimbările datelor în timp real. Țevile standard disponibile în Angular includ funcționalități comune precum “DatePipe” pentru formatarea datelor calendaristice, “CurrencyPipe” pentru afișarea valorilor monetare, “DecimalPipe” și “PercentPipe” pentru formatarea numerelor și procentajelor.

Angular prezintă mai multe avantaje: facilitează dezvoltarea rapidă prin numeroasele caracteristici oferite, permite scalarea aplicațiilor complexe pe măsură ce nevoile cresc, asigură testabilitate ridicată prin diverse instrumente de testare, și beneficiază de o comunitate activă de dezvoltatori care oferă suport și resurse.

Internaționalizarea este procesul prin care se asigură că o aplicație este proiectată și pregătită pentru a fi utilizată în regiuni cu diferite limbi. Angular se poate ocupa de majoritatea lucrurilor când vine vorba de mai multe limbi. Datele, numerele, orele și alte lucruri sunt ușor de rezolvat în funcție de locație. Având în vedere domeniul de interes global al acestei aplicații, extinderea acesteia către un public internațional va fi facilă datorită capacităților Angular de a gestiona cu ușurință multiple limbi și adaptări specifice locației folosind pachetul “@angular/localize”. [8]

Decizia adoptării tehnologiei Angular pentru această aplicație a fost ghidată de trei aspecte cheie. În primul rând, scalabilitatea sa robustă, susținută de o arhitectură modulară și tehnologii web moderne, asigură adaptarea ușoară la cerințele viitoare. De asemenea, decizia a fost influențată de posibilitatea extinderii aplicației la o policlinică sau la o rețea de policlinici sau spitale, care impun un creșterea traficului și a volumului de date, specifice unui sistem medical complex. Pe lângă aceste avantaje tehnice, experiența mea anterioară cu Angular a contribuit semnificativ la eficiența procesului de dezvoltare. Familiaritatea cu acest framework a redus timpul de învățare și a permis valorificarea rapidă a cunoștințelor existente, conducând la o implementare eficientă și de calitate.

### 2.2.2 RxJS

RxJS [9] (Reactive Extensions for JavaScript) este o bibliotecă JavaScript care implementează conceptele de programare reactivă și fluxuri de date asincrone. În contextul Angular, RxJS joacă un rol crucial în gestionarea fluxurilor de date și a evenimentelor asincrone, contribuind la dezvoltarea unei aplicații reactive și eficiente.

O caracteristică esențială a RxJS este reprezentată de observabile și operatori. Observabilele sunt surse de date care pot emite valori sau evenimente în timp, iar operatorii sunt funcții care permit transformarea, filtrarea și combinarea acestor valori sau evenimente. Această abordare funcționează în mod similar cu fluxurile de date din alte limbaje de programare, cum ar fi stream-urile din Java sau observabilele din Rx.NET.

În Angular, RxJS este utilizat pe scară largă pentru a gestiona evenimente asincrone, cum ar fi cererile HTTP către server, evenimentele din interfața utilizatorului sau orice altă operație care implică întârzieri sau operații asincrone. De exemplu, atunci când se face o cerere HTTP pentru a obține date dintr-un server, rezultatul este împachetat într-un obiect de tip Observable și poate fi transformat sau combinat folosind operatorii RxJS.

### 2.2.3 Typescript

TypeScript [10] reprezintă un limbaj de programare open-source, dezvoltat de Microsoft, care extinde JavaScript prin adăugarea unui sistem de tipuri static. În timp ce fiecare program JavaScript este un program TypeScript, TypeScript oferă un sistem de module, clase, interfețe și un sistem bogat de tipări graduale. După cum spun Bierman, Abadi și Torgersen în articolul lor "Understanding typescript": "În ciuda succesului său, JavaScript rămâne un limbaj slab pentru dezvoltarea și întreținerea aplicațiilor mari." [11]. Acest lucru poate fi observat și din adopția limbajului typescript de majoritatea framework-urilor de dezvoltare web moderne. TypeScript oferă o tranziție ușoară pentru programatorii JavaScript - idiomurile de programare JavaScript bine stabilite sunt acceptate fără nicio rescriere majoră sau adnotări [11].

Acesta este limbajul de bază folosit de framework-ul Angular pentru a programa funcționalitatea aplicației. Inițial, Angular a fost conceput pentru a fi utilizat cu JavaScript, însă TypeScript oferă mai multe avantaje, cum ar fi tipizarea statică, care permite detectarea mai rapidă a erorilor, ceea ce duce la îmbunătățirea productivității dezvoltatorilor. Acest lucru poate părea contraintuitiv principilor JavaScript, însă acest

sistem de tipuri puternic ajută la menținerea codului și la scalabilitatea proiectelor Angular, ceea ce permite dezvoltarea aplicațiilor care devin mai mari și mai complexe.

#### **2.2.4 Angular Material**

Angular Material [12] este o bibliotecă de componente UI și stiluri CSS predefinite, dezvoltată de echipa Angular de la Google. Această bibliotecă oferă un set complet de componente care formează o interfață utilizator modernă și cu o funcționalitate consistentă. Fiind creată de echipa din spatele framework-ului Angular, această bibliotecă este foarte ușor de integrat în aplicația noastră și oferă componente care respectă principiile și normele de rigoare stabilite de sistemul Material Design [13] creat de Google.

Angular Material oferă o gamă largă de componente. Câteva dintre acestea care au fost folosite în aplicația dezvoltată includ: butoane, carduri, meniuri, bare de navigare, panouri extensibile, diagrame, căsuțe de dialog, etc. Motivul pentru care am ales să folosesc această bibliotecă a fost consistența, facilitatea de utilizare și personalizarea posibilă a acestor componente.

#### **2.2.5 Bootstrap**

Bootstrap [14] este un framework de frontend open-source, utilizat pentru dezvoltarea rapidă a site-urilor web și a aplicațiilor web responsive. A fost inițial dezvoltat de către Twitter și a fost lansat în 2011, devenind rapid unul dintre cele mai populare framework-uri frontend. Bootstrap oferă un set bogat de componente predefinite, stiluri CSS și un sistem de grilă, care facilitează alinierea și aranjarea conținutului pe diferite dispozitive și dimensiuni ale ecranelor. Aceste componente pot fi personalizate și extinse în funcție de nevoile specifice ale proiectului.

Datorită compatibilității cu majoritatea browserelor moderne și a extensibilității sale, Bootstrap este adesea utilizat pentru a crea site-uri web atractive și funcționale, fără a fi necesar să se înceapă de la zero în ceea ce privește stilurile și aspectul interfeței de utilizator. Am ales folosirea acestei tehnologii deoarece facilitează un sistem de grilă puternic și receptiv, care permite organizarea ușoară în pagină și ajustarea automată în funcție de dimensiunea ecranului, asigurând o experiență consistentă pe diverse dispozitive și rezoluții. Alte motive pentru alegerea acestei tehnologii au fost popularitatea sa, ceea ce înseamnă ca sunt disponibile numeroase resurse online ajutătoare și familiaritatea mea cu această tehnologie.

### 2.2.6 Firebase

Firebase [15] este o platformă de dezvoltare a aplicațiilor web sau mobile oferită de Google. Ea furnizează o suită de instrumente și servicii cloud care ajută dezvoltatorii să creeze, să îmbunătățească și să gestioneze aplicații, fără a fi nevoie să se preocupe de infrastructura de backend. Deoarece majoritatea aplicațiilor web și mobile au nevoie de funcționalități asemănătoare precum salvarea datelor într-o bază de date, autentificare, stocarea imaginilor sau a fișierelor, au fost concepute așa zisele “Backend as a Service (BaaS)”. Aceste servicii, printre care se numără și Firebase, oferă o metodă consistentă de a gestiona datele din backend, ceea ce permite dezvoltatorilor să se concentreze pe dezvoltarea unui produs inovativ fără a irosi timp și resurse asupra creării unui backend specializat pentru aplicația lor. Platforma Firebase permite utilizarea gratuită până la anumite limite lunare. Deoarece aplicația nu este în momentul actual folosită de către foarte mulți utilizatori, aceste limite sunt mai mult decât suficiente. Am ales tehnologia Firebase deoarece conține un set întreg de servicii necesare în dezvoltarea acestei aplicații.

Principalele caracteristici și servicii oferite de Firebase includ: Autentificare, Bază de date în timp real (Realtime Database), Bază de date Cloud (Cloud Firestore), Stocare și Hosting web. În dezvoltarea acestei aplicații am utilizat următoarele servicii:

#### **Autentificare [16]:**

Firebase oferă servicii de autentificare securizate. Am folosit acest serviciu pentru a permite utilizatorilor crearea de conturi și autentificarea. La nivel tehnic, autentificarea în Firebase se bazează pe un sistem care primește și procesează cererile de autentificare ale utilizatorilor. Atunci când un utilizator încearcă să se autentifice, datele sale de autentificare sunt trimise către serviciul Firebase Authentication, care le verifică și generează un token de autentificare unic pentru utilizatorul respectiv. Acest token este apoi returnat către aplicație și utilizat pentru a accesa resursele protejate. De fiecare dată când este efectuată o cerere către server, token-ul de autentificare este trimis împreună cu cererea pentru a valida identitatea și drepturile de acces ale utilizatorului. Firebase Authentication gestionează, de asemenea, sesiunile de autentificare ale utilizatorilor, asigurând securitatea și fiabilitatea procesului de autentificare în aplicațiile Firebase.

## Stocare [17]:

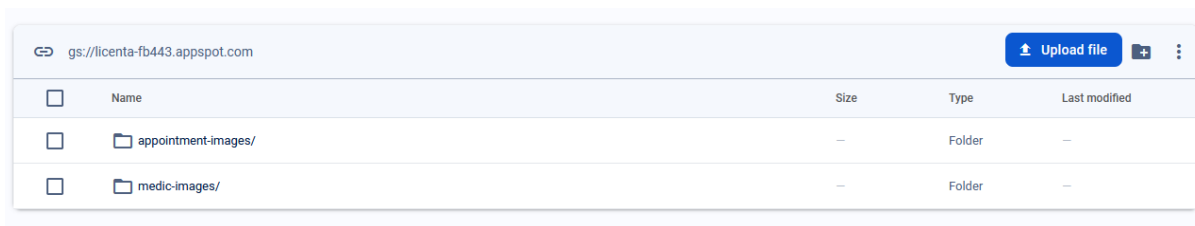


Figure 2-1 Structura folderelor serviciului de stocare

Acest serviciu a fost folosit pentru stocarea în cloud a fișierelor media (a imaginilor) folosite pentru programări sau profiluri de medic. Acestea au fost împărțite în două foldere: *appointment-images* și *medic-images*.

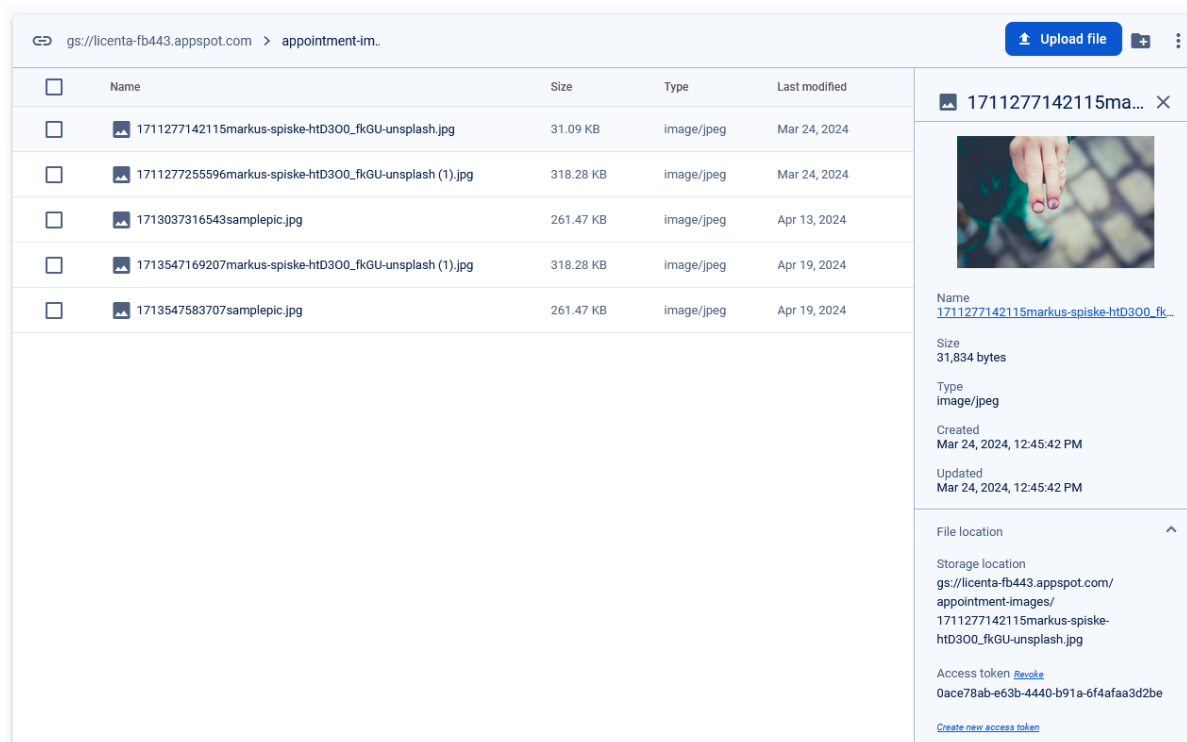


Figure 2-2 Imaginile trimise de pacienți la crearea unei programări

Fiecare dintre aceste imagini primește o locație de stocare unică atunci când este salvată de Firebase, locație care poate fi ulterior folosită pentru a identifica și accesa imaginea.



**Cloud Firestore [18]:** Este un serviciu de bază de date NoSQL, care oferă o structură scalabilă și flexibilă pentru stocarea și gestionarea datelor în cloud. Aceasta a fost folosită pentru a stoca datele utilizatorilor (tipuri de cont), formularele de contact trimise, profilurile de medici create, și programările făcute de către pacienți.

### 2.2.7 TensorFlow & Keras

TensorFlow [19] este o platformă open-source dezvoltată de Google, utilizată pentru construirea și antrenarea modelelor de învățare automată și de învățare profundă. A fost lansată inițial în 2015 și a devenit una dintre cele mai populare biblioteci de învățare automată din lume. Numele "TensorFlow" reflectă modul în care biblioteca gestionează operațiile matematice, în special operațiile cu tensori. Tensorii sunt obiecte matematice care generalizează conceptul de scalari, vectori și matrice. Ele pot avea dimensiuni multiple și sunt fundamentale în modelele de învățare automată și de învățare profundă, deoarece reprezintă datele și parametrii care sunt manipulați și transformați de aceste modele.

Keras [20] este o bibliotecă de rețele neuronale care este concepută pentru a fi o interfață simplă, ușor de înțeles și de utilizat pentru construirea și antrenarea modelelor de învățare automată. De-a lungul timpului, Keras a devenit tot mai integrat cu TensorFlow și este adesea utilizat împreună cu acesta, ca parte a TensorFlow Keras, care este acum modulul oficial de învățare profundă al TensorFlow.

### 2.2.8 VGG19

VGG19 [21] este un model de rețea neurală profundă, cunoscut pentru arhitectura sa complexă și performanțele remarcabile în domeniul viziunii artificiale. Arhitectura VGG19 este compusă din 19 straturi (de aici și denumirea "19" din VGG19), inclusiv straturi convoluționale și straturi complet conectate. Modelul folosește filtre convoluționale de dimensiuni mici (de obicei 3x3 pixeli) în toate straturile sale convoluționale, urmate de straturi de pooling pentru reducerea dimensiunii spațiale a imaginilor. Acest lucru permite modelului să captureze caracteristici de la niveluri de detaliu diferite ale imaginilor.

### 2.2.9 ResNet50

ResNet50 [22] este un model profund de rețele neuronale convoluționale (CNN) cunoscut pentru inovația adusă în abordarea problemelor de degradare a performanței în rețelele foarte adânci. Arhitectura ResNet50 este compusă dintr-o serie de blocuri

reziduale care permit antrenarea cu succes a rețelelor foarte profunde, până la 50 de straturi, fără a suferi degradarea performanței. Aceasta este realizată prin introducerea conexiunilor scurte sau skip connections, care permit fluxul informației nealterate printr-un strat la altul.

### **2.2.10 InceptionV3**

InceptionV3 [23] reprezintă o arhitectură avansată de rețele neurale convoluționale (CNN), dezvoltată de Google în cadrul proiectului său de cercetare asupra învățării profunde. Acest model se distinge prin utilizarea unor module de convoluție extrem de eficiente, denumite module Inception, care au fost concepute pentru a îmbunătăți performanța rețelelor în ceea ce privește extragerea caracteristicilor din imagini.

### **2.2.11 InceptionResNetV2**

InceptionResNetV2 [24] reprezintă o evoluție în arhitectura rețelelor neurale convoluționale (CNN), combinând elemente cheie din două dintre cele mai inovative modele dezvoltate anterior: Inception și ResNet. Principala caracteristică a InceptionResNetV2 este utilizarea modulelor Inception. În plus față de modulele Inception, InceptionResNetV2 adoptă și arhitectura de tip residual (ResNet), care a fost introdusă pentru a rezolva problema de degradare a performanței la rețelele neurale utilizând conexiuni de tip skip (sau conexiuni reziduale).

Prin combinarea modulelor Inception cu arhitectura ResNet, InceptionResNetV2 obține o rețea neurală care este puternică în capturarea detaliilor subtile și a relațiilor complexe între caracteristici în imagini. Această combinație permite modelului să obțină rezultate superioare în sarcini precum clasificarea obiectelor în imagini, recunoașterea de scene și analiza de conținut vizual.

### **2.2.12 EfficientNet**

EfficientNet [25] reprezintă o familie de arhitecturi de rețele neurale convoluționale (CNN) care au fost dezvoltate pentru a maximiza performanța și eficiența computațională în rezolvarea sarcinilor de învățare automată, în special în domeniul viziunii artificiale.

Structura de bază a EfficientNet este definită printr-un balans între adâncimea rețelei (numărul de straturi) și lățimea rețelei (numărul de canale în fiecare strat). În mod tradițional, antrenarea rețelelor CNN presupunea creșterea adâncimii și lățimii pentru a îmbunătăți performanța, însă acest lucru duce la o creștere semnificativă a costului computațional. EfficientNet propune o metodă de scalare uniformă a adâncimii, lățimii și rezoluției de intrare, pentru a atinge un echilibru optim între precizie și eficiență. Modelul EfficientNet utilizează un concept numit Compound Scaling, care ajustează simultan adâncimea (numărul de straturi), lățimea (numărul de canale) și rezoluția de intrare a imaginilor pentru a obține o arhitectură optimizată.

## 2.3 ORGANIZAREA PROIECTULUI

Aplicația web este alcătuită din 3 părți constitutive: Partea de frontend, Partea de backend și Partea de antrenare a inteligenței artificiale.

### 2.3.1 Frontend

Partea de frontend este formată din aplicația Angular împreună cu Angular Material și Bootstrap. Aceasta poate fi rulată de către utilizator prin intermediul unui browser. Angular este o platformă puternică pentru dezvoltarea de aplicații web single-page (SPA). Aplicația dezvoltată utilizează această arhitectură, ceea ce înseamnă că rulează într-o singură pagină web, iar conținutul este încărcat și actualizat dinamic, fără a fi necesară navigarea între pagini separate. Fișierele necesare aplicației sunt împărțite în diferite foldere pentru a putea fi gestionate și accesate ușor și intuitiv. Folderele .angular, .vscode sunt create automat și conțin informații și configurații specifice proiectului Angular, cum ar fi fișiere de configurare. Folderul "node\_modules" conține toate modulele și dependențele JavaScript instalate pentru proiectul Angular. Toate pachetele npm (Node Package Manager) adăugate aplicației prin comenzile "npm install x" sau "ng add x", cum ar fi Angular Material sau Bootstrap sunt descărcate și instalate în acest folder. De asemenea, acesta conține și pachetele instalate automat de către Angular CLI pentru a putea rula aplicația Angular.

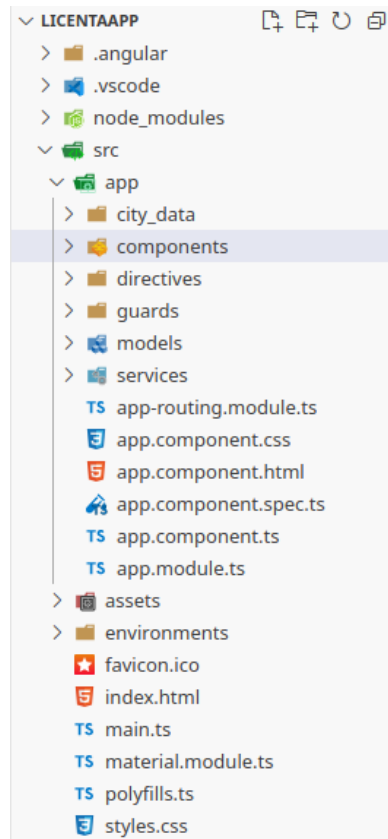


Figure 2-3 Structura folderelor aplicației Angular

Pilonul de bază al unei aplicații Angular, care conține toate fișierele și tot codul sursă a aplicației web, este folderul “src” (abreviere pentru “source”). Acesta este cel mai important folder dintr-o aplicație Angular și putem găsi în interiorul acestuia mai multe fișiere și foldere.

Subfolderul “Assets” este destinat stocării resurselor statice, cum ar fi imagini, fișiere CSS sau alte fișiere care trebuie incluse în aplicație. Acesta a fost folosit în proiectul dezvoltat pentru a stoca anumite imagini statice cum ar fi cele de pe paginile “about” și “skin lesion analyser”, și pentru a stoca modelul de inteligență artificială antrenat și convertit. Aceste resurse pot fi accesate în cod prin intermediul căilor relative.

În folderul “Environments” se găsesc fișiere de configurare specifică pentru medii diferite(cum ar fi dezvoltare, producție etc.). Aceste fișiere conțin variabile de configurare care pot fi diferite în funcție de mediul de rulare a aplicației.

Fișierele “main.ts” și “index.html” sunt punctul de intrare în aplicație, respectiv pagina HTML principală a aplicației. Primul dintre aceste fișiere este un fișier TypeScript (lucru care poate fi observat după extensia specifică “.ts”) care este transpus, compilat și

rulat de către Angular CLI pentru a porni aplicația. Acesta conține o funcție care încarcă modulul principal al aplicației și îl plasează în elementul DOM specificat. Fișierul "index.html" definește pagina principală a aplicației Angular și este responsabil pentru afișarea inițială a aplicației în browser. În mod implicit, acest fișier conține structura HTML de bază pentru orice pagină web (cum ar fi elementele "<html>", "<head>" și "<body>"). În interiorul acestuia se apelează componenta "<app-root>", care este componenta de bază în care este montată aplicația Angular. În acest fișier putem adăuga meta-etichete sau link-uri către fonturi sau alte resurse statice necesare pentru aplicație.

Folderul "app" este unul dintre cele mai importante foldere dintr-o aplicație Angular. Acesta conține codul sursă al aplicației și este locul unde putem găsi toate unitățile fundamentale Angular. Acest folder a fost împărțit în subfoldere cu denumire autoexplicative: "components", "directives", "guards", "models", "services". În aceste subfoldere se află toate fișierele create în decursul dezvoltării aplicației cum ar fi componentele, serviciile, modelele sau directivele, și au fost împărțite după tipul lor pentru a putea fi gestionate mai ușor. De asemenea putem găsi modulul "app.module", care este modulul de bază al aplicației, în care sunt declarate componentele, serviciile și alte resurse necesare folosite în proiect. Tot în acest folder se află și componenta rădăcina a aplicației "app.component", care are responsabilitatea de a afișa antetul, subsolul și de a inițializa sistemul de rutare Angular. Fișierul "app-routing.module" conține setările de configurare a rutelor în aplicația web, cum ar fi rutele, componentele aferente acestora și router guards, care protejează accesul la acestea de către utilizatorii neautorizați.

### 2.3.2 Backend

Partea de backend este constituită din platforma Firebase. Gestionarea resurselor folosite poate fi făcută prin accesarea consolei Firebase [26] și crearea unui proiect nou. După ce procesul de creare al proiectului a fost îndeplinit cu succes, îl putem accesa pe acesta de pe aceeași pagină web, unde putem gestiona instrumentele și serviciile pe care dorim să le folosim. Serviciile folosite în dezvoltarea acestei aplicații sunt: Authentication, Firestore Database și Storage. De pe pagina aferentă a fiecărui serviciu, se pot gestiona datele manual, precum și regulile de accesare a acelor servicii. Prin intermediul acestor reguli se poate proteja anumite date de la a fi accesate de către utilizatori neautentificați sau neautorizați.

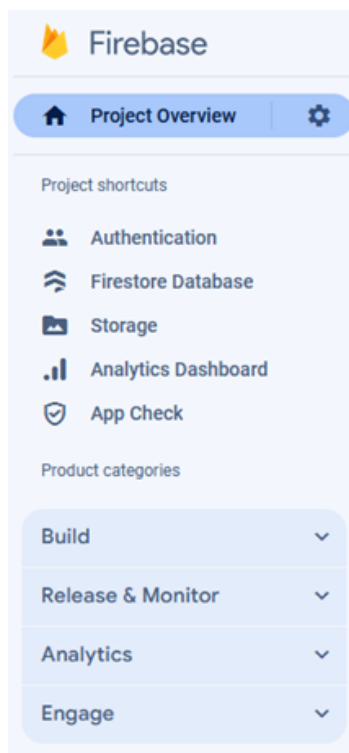


Figure 2-4 Structura de organizare a proiectului Firebase

De asemenea, în pagina principală a consolei Firebase se găsesc niște grafice care oferă o vedere de ansamblu asupra utilizării și performanței aplicațiilor și serviciilor Firebase. Prin acestea putem să vedem cota de folosire a tuturor serviciilor și să identificăm eventuale probleme.



Figure 2-5 Utilizarea resurselor proiectului Firebase

Baza de date folosită este "Firestore Database". Datele salvate sunt împărțite în colecții care conțin documente cu ID-uri unice, iar fiecare document conține o serie de perechi cheie-valoare. Acest lucru este asemănător unei baze de date convențională SQL, unde datele sunt împărțite în tabele și entități. Firestore este proiectat să fie extrem de scalabil și performant. Este capabil să gestioneze volume mari de date și să răspundă rapid la cererile clienților, indiferent de dimensiunea sau complexitatea aplicației. Una dintre caracteristicile distinctive ale Firestore este suportul pentru date în timp real.

Acest lucru înseamnă că modificările efectuate în baza de date sunt transmise în mod automat către toți clienții care sunt conectați, permițând actualizări instantanee a interfeței utilizator. De asemenea oferă un sistem robust de securitate și autorizare care se integrează perfect cu alte servicii Firebase.

### **2.3.3 Inteligență artificială**

Organizarea părții de inteligență artificială poate fi divizată în două subcategorii distincte: procesarea datelor și antrenarea modelului, reprezentând esențialul infrastructurii din spatele aplicației, și partea vizibilă utilizatorilor în cadrul aplicației web, unde aceștia interacționează cu modelul antrenat. Această separare clară a responsabilităților contribuie la o dezvoltare mai eficientă și la o gestionare mai simplă a întregului sistem.

Prelucrarea datelor și antrenarea modelului este realizată separat de aplicația web, utilizând IDE-ul PyCharm [27]. După antrenarea modelului, acesta este salvat și exportat. Acest model exportat este utilizat mai apoi în aplicația web, unde utilizatorii îl pot folosi pentru a analiza anumite imagini care conțin leziuni ale pielii.

## 3 IMPLEMENTAREA APLICAȚIEI WEB

### 3.1 ELEMENTE COMUNE

Primul lucru care poate fi observat la deschiderea aplicației, sunt antetul (Header) și subsolul (Footer). Aceste două secțiuni sunt prezente în fiecare pagină a aplicației și joacă un rol fundamental în navigarea aplicației de către utilizatori.

Antetul conține logo-ul și numele clinicii fictive asociate aplicației, meniul de navigare și informații despre contul utilizatorului autentificat. Acesta este creat printr-o componentă Angular Material numită "Toolbar" [12], ceea ce îi oferă o aparență plăcută care respectă principiile impuse prin Material Design.



Figure 3-1 Antetul (Header) aplicației web pentru un utilizator cu rol de medic

Subsolul este secțiunea plasată în partea de jos a fiecărei pagini, furnizând link-uri utile către cele mai importante pagini ale aplicației, precum și alte informații de interes, cum ar fi numele autorului și titlul proiectului.

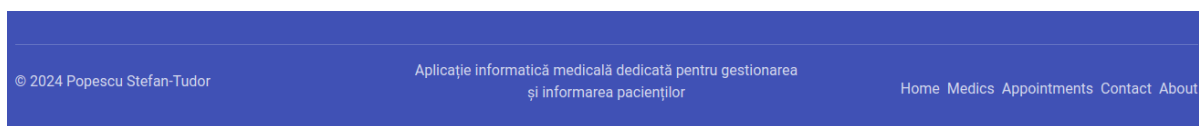


Figure 3-2 Subsolul (Footer) aplicației web

Aceste două elemente esențiale, antetul și subsolul, contribuie la coerența și ușurința utilizării aplicației, oferind utilizatorilor o experiență navigare consecventă și intuitivă.

### 3.2 ANGULAR ROUTING

Angular Routing [28] este un sistem sofisticat și flexibil de gestionare a navigației într-o aplicație Angular, esențial pentru dezvoltarea aplicațiilor *Single Page Application* (SPA). În esență, acest sistem permite crearea de aplicații web unde utilizatorii pot naviga



Aplicație informatică medicală dedicată pentru gestionarea și informarea pacienților

între diferite vizualizări (Views) sau componente fără a reîncărca întreaga pagină, oferind o experiență de utilizare fluidă și interactivă.

```
app-routing.module.ts

const routes: Routes = [
  { path: "", title: "Ditama Clinic - Home", component: HomeComponent },
  { path: "authenticate", title: "Authenticate", component: AuthComponent, canActivate:
[LoginAuthGuard] },
  {
    path: "medics",
    title: "Ditama Clinic - Medics",
    component: MedicsComponent,
  },
  {
    path: "medic-page/:userId",
    title: "Ditama Clinic - Set up your medic page",
    component: MedicProfileComponent,
    canActivate: [MedicPageGuard],
  },
  {
    path: "medic-page",
    redirectTo: "medic-page/ ",
  },
  {
    path: "appointments/:city/:specialty/:medicId",
    title: "Ditama Clinic - Appointments",
    component: AppointmentsComponent,
    canActivate: [AppointmentsGuard],
  },
  {
    path: "appointments",
    title: "Ditama Clinic - Appointments",
    component: AppointmentsComponent,
    canActivate: [AppointmentsGuard],
  },
  {
    path: "schedule",
    title: "Ditama Clinic - Medic Schedule",
    component: ScheduleComponent,
    canActivate: [ScheduleGuard],
  },
  {
    path: "profile/:userId",
    title: "Ditama Clinic - User Profile",
    component: UserProfileComponent,
    canActivate: [ProfileGuard],
  },
  { path: "profile", redirectTo: "profile/ " },
  { path: "contact", component: ContactComponent },
  { path: "about", component: AboutComponent },
  { path: "ai-photo-check", component: AIPhotoCheckComponent },
  { path: "not-authorized", title: "Diatma Clinic - Unauthorized Access", component:
NotAuthorizedComponent },
  { path: "**", redirectTo: "" },
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule],
})
export class AppRoutingModule {}
```


Figure 3-3 Definirea rutelor folosind Angular Routing

Implementarea Angular Routing începe cu definirea rutelor, care sunt configurate folosind 'RouterModule'. Acesta este un modul Angular specializat care trebuie importat în modulul principal al aplicației sau în modulele funcționale specifice. Configurația rutelor este realizată printr-o colecție de obiecte de rută, fiecare specificând un URL și componenta ce trebuie afișată atunci când acel URL este accesat. Aceste obiecte de rută pot include, de asemenea, alte proprietăți, cum ar fi parametri dinamici, rute protejate prin gărzi (guards) și date suplimentare.

Gărzile (guards), oferă o metodă de protecție a rutelor. Acestea permit controlul accesului la anumite rute, în funcție de condiții specifice, cum ar fi autentificarea utilizatorului. Ele pot fi utilizate pentru a preveni accesul neautorizat sau pentru a gestiona navigarea condiționată, asigurându-se că numai utilizatorii care îndeplinesc anumite criterii pot accesa anumite părți ale aplicației. Am folosit aceste gărzi pentru a proteja paginile "Appointments", "Schedule", "Profile" și "Medic page setup" de utilizatorii neautentificați și rutele "Schedule" și "Medic page setup" de utilizatorii care nu au rol de medic (utilizatori neautorizați). Astfel, când un utilizator încearcă să acceseze una din aceste pagini prin introducerea URL-ului corespunzător, acesta va fi redirecționat către pagina "Not Authorized", unde i se va cere să se autentifice cu alt cont pentru a obține accesul.

Navigarea între diferitele rute se realizează prin "RouterLink", care este o directivă utilizată în elementele de ancorare (link-uri) pentru a schimba URL-ul fără a reîncărca pagina. "RouterLink" asigură o tranziție lină și continuă între diferitele vizualizări (views), actualizând doar porțiunea relevantă a paginii.

Angular Routing permite, de asemenea, utilizarea parametrilor de rută pentru a gestiona navigarea dinamică. Parametrii de rută pot fi definiți și extrași din URL, permițând componentei să reacționeze în funcție de aceștia. De exemplu, în pagina de profil, un parametru folosit este "userId" care reprezintă un id unic pentru fiecare utilizator. Astfel, componenta va afișa detalii diferite pentru fiecare utilizator.



```

navbar.component.html

<a mat-button routerLink="medics">Medics</a>
<button mat-menu-item [routerLink]="['/profile', userId]">Profile</button>

```

Figure 3-4 Exemplu de utilizare a directivei RouterLink și a parametrilor de rută

În concluzie, Angular Routing reprezintă un pilon central al dezvoltării aplicațiilor moderne cu Angular, oferind o structură robustă și flexibilă pentru gestionarea navigației și a conținutului dinamic.

### 3.3 PAGINA PRINCIPALĂ (HOME)

Pagina “Home”, cunoscută și sub numele de pagina principală, reprezintă punctul central al unui site web. Este primul loc în care utilizatorii ajung atunci când accesează site-ul și joacă un rol crucial în crearea unei prime impresii puternice. Această pagină este concepută pentru a oferi o privire de ansamblu asupra conținutului și scopului site-ului, fiind structurată astfel încât să fie ușor de navigat și atrăgătoare vizual.

Pagina principală servește drept punct de plecare pentru explorarea ulterioară a site-ului, oferind legături către alte secțiuni importante și facilitând accesul rapid la informațiile esențiale. În aplicația dezvoltată, această pagină conține două secțiuni: o secțiune “Carousel” și o secțiune cu butoane “Call to Action”.

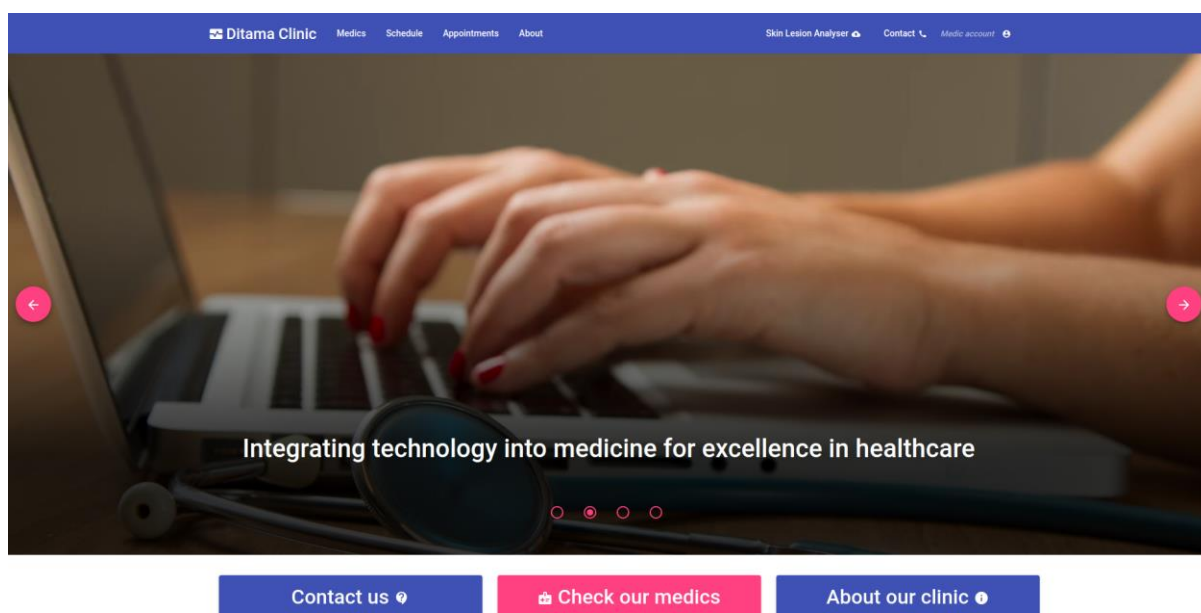


Figure 3-5 Pagina principală (Home)

Secțiunea “Carousel” conține un carusel de imagini, adică un element interactiv de design web utilizat pentru a prezenta o serie de imagini sau conținut multimedia într-un spațiu restrâns. Funcționează prin rotirea automată sau manuală a imaginilor, permițând utilizatorilor să vadă mai multe imagini într-o singură locație, fără a derula sau a naviga pe alte pagini.

Un aspect crucial al acestui carusel este navigabilitatea. Utilizatorii pot naviga între imaginile anterioare și următoare folosind butoanele de navigare, care sunt reprezentate de săgețile din stânga și din dreapta caruselului. De asemenea, au posibilitatea să acceseze direct orice imagine din carusel utilizând indicatorii de paginare, cunoscuți și sub numele de "bullets". În plus, caruselul se derulează automat, prezentând imaginile în mod automat, la un interval de timp prestabilit de 5 secunde.

Această secțiune este încapsulată într-o componentă carusel. Componenta a fost proiectată pentru a facilita adăugarea simplă și ușoară a altor elemente în carusel. Atunci când un nou element este adăugat, un indicator de paginare este creat automat pentru acesta, iar funcționalitatea rămâne stabilă și fără probleme. Acest carusel are rolul de a îmbunătăți experiența utilizatorului, a atrage atenția și de a crește interactivitatea și angajamentul pe site.

În cea de-a doua secțiune, se găsesc 3 butoane Call to Action (CTA). Butoanele CTA sunt elemente interactive esențiale, proiectate pentru a captiva atenția utilizatorilor și a-i îndemna să efectueze acțiuni specifice. În acest context, ele sunt utilizate pentru a ghida utilizatorii către navigarea către una dintre paginile prezentate.

Aspectul vizual al butoanelor Call to Action este crucial. Acestea sunt proiectate pentru a se evidenția vizual, prin culori contrastante, dimensiuni mari și mesaje incitante. Toate aceste caracteristici atrag imediat atenția utilizatorilor și le indică clar acțiunile disponibile.

În concluzie, pagina principală a fost concepută cu un design modern și o interfață prietenoasă pentru utilizator, menită să capteze atenția. Aceasta încurajează utilizatorii să exploreze mai departe site-ul și să navigheze către celelalte pagini ale aplicației.

### **3.4 PAGINA DE MEDICI (MEDICS)**

Pe pagina "Medici", utilizatorii pot găsi o listă cuprinzătoare a tuturor medicilor care fac parte din clinica virtuală. Fiecare medic este prezentat printr-o "carte" (Card) care conține informații precum numele, specializarea, titlul și orașul. De asemenea, sunt incluse butoane care facilitează contactul cu medicul respectiv sau programarea unei consultații. Aceste cărți sunt organizate într-o grilă cu trei elemente pe rând, pentru a oferi utilizatorilor o modalitate clară și concisă de a vizualiza medicii disponibili și pentru a facilita interacțiunea cu aceștia.

Componentele “cărți” și “grilă” sunt elemente predefinite din Angular Material: Card și Grid List [12]. Acestea prezintă un design plăcut și accesibil, conform principiilor Material Design, și sunt integrate cu ușurință în aplicații Angular. Alinierea grilei în pagină este realizată prin utilizarea sistemului de grilă Bootstrap. Grila este centrată și are margini laterale pentru a crea un aspect echilibrat și profesional.

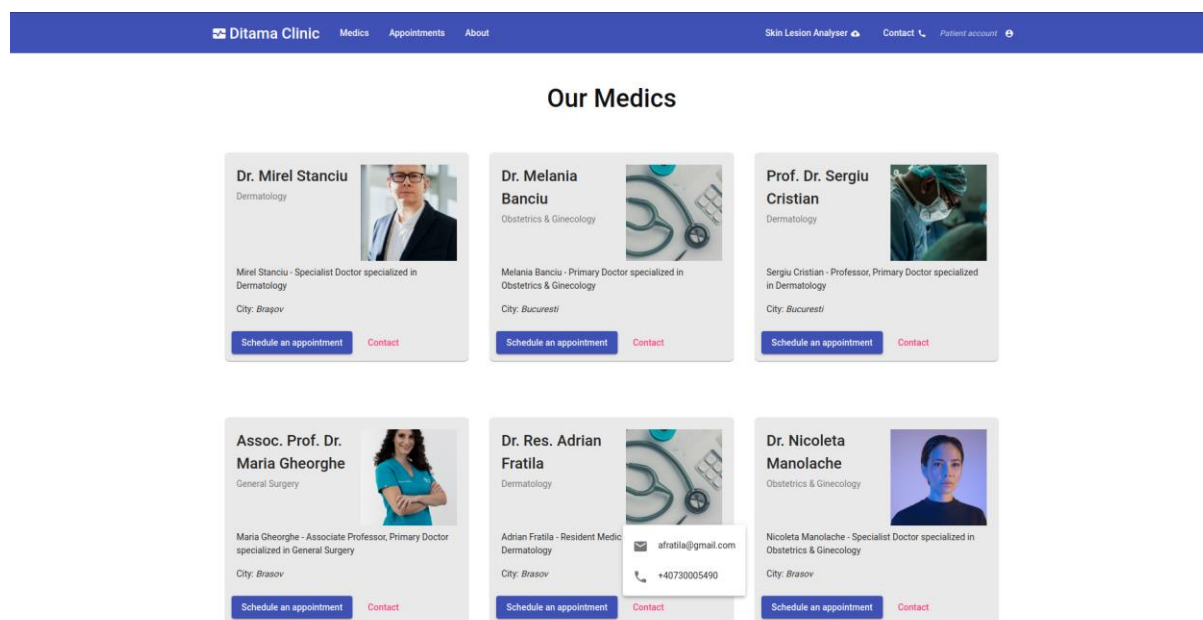


Figure 3-6 Pagina de Medici

În momentul inițializării componentei, prin intermediul metodei “ngOnInit”, se creează un apel către API-ul Firebase pentru a obține lista completă a medicilor și detaliile acestora din baza de date. Metoda “ngOnInit” este unul dintre “hook-urile” de viață a componentelor Angular, care oferă posibilitatea executării anumitor funcții în momente specifice ale ciclului lor de viață.

Datele referitoare la medici sunt stocate în colecția “medics” a bazei de date Firestore din Firebase. Pentru a accesa aceste date, se utilizează serviciul “medic.service”, care gestionează funcționalitățile asociate preluării listei de medici din Firestore. Acest serviciu realizează apeluri către API-ul Firebase și furnizează funcții care returnează un obiect de tip Observable conținând datele. Odată ce apelul către API este finalizat, observable-ul se încheie și returnează lista de medici înapoi la componentă.

În timpul procesului de încărcare a datelor, pentru a indica procesul de încărcare, se afișează o componentă de tip bara de progres (Progress bar [12]), componentă care face parte din Angular Material. Aceasta furnizează utilizatorilor un indiciu vizual că se efectuează operațiuni în fundal. După finalizarea cu succes a apelului către API, grila de

“cărți” este afișată, fiecare carte corespunzând unui medic din lista preluată. În cazul în care apar erori sau probleme la preluarea listei de medici (de ex: erori de conexiune la server), se afișează un mesaj de eroare care informează utilizatorii că a apărut o eroare, urmat de mai multe detalii despre eroarea respectivă.

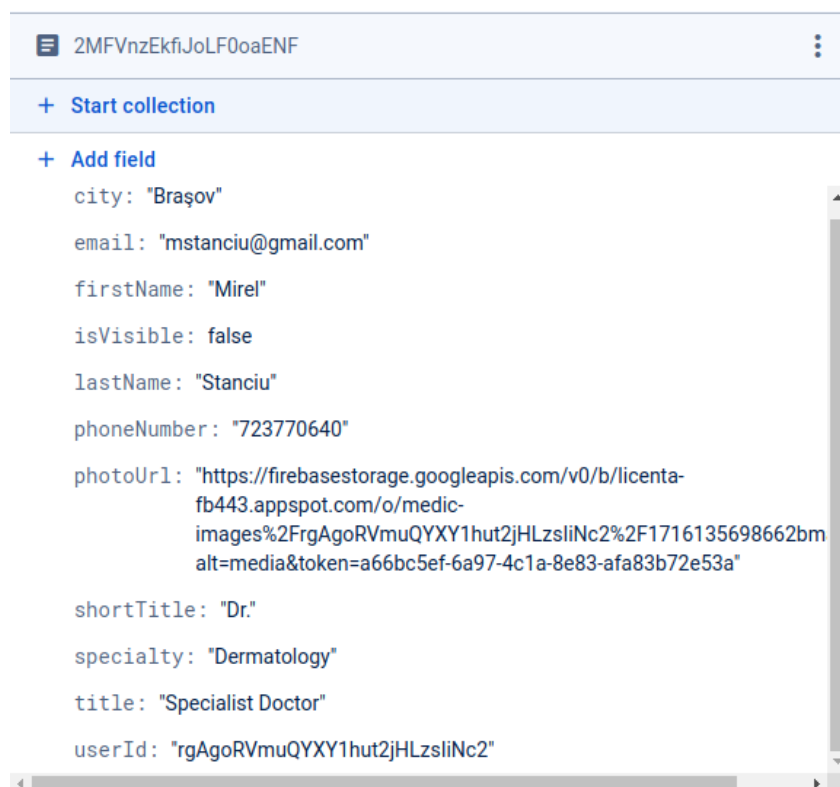


Figure 3-7 Structura unui document de tip medic în baza de date Firestore

Pentru afișarea condiționată a componentelor se utilizează directiva “ngIf” [29]. Această directivă nativă Angular permite încărcarea condiționată a anumitor elemente din HTML în funcție de evaluarea unei expresii booleene. Când expresia este adevărată, elementele sunt adăugate în DOM, iar când este falsă, ele sunt eliminate. Această abordare îmbunătățește performanța aplicației, reducând numărul de elemente DOM încărcate și simplifică gestionarea condițiilor complexe în șablonul HTML.

Pentru afișarea “cărților” se folosește directiva “ngFor” [29], care iterează prin lista de medici și generează câte o “carte” pentru fiecare dintre aceștia. “ngFor” este o directivă structurală în Angular care permite iterarea eficientă prin colecții, generând elemente dinamice pe baza datelor din array-uri sau obiecte iterabile. Această directivă oferă, de asemenea, acces la informații suplimentare precum indexul curent, primul și ultimul element, ceea ce permite o flexibilitate crescută în manipularea și afișarea datelor.

Aplicație informatică medicală dedicată pentru gestionarea și informarea pacienților

Această abordare simplifică codul vizual, facilitează reutilizarea acestuia și reduce cantitatea de cod necesară.

```
medics.component.html

<mat-grid-list *ngIf="!isLoading && !error" cols="3" gutterSize="2rem">
  <mat-grid-tile *ngFor="let medic of medics">
    <mat-card class="medic-card">...</mat-card>
  </mat-grid-tile>
</mat-grid-list>
```

Figure 3-8 Exemplu de utilizare a directivelor ngIf și ngFor

Pentru utilizatorii cu rol de medic, pagina prezintă două modificări. Prima este un buton situat la începutul paginii, pe care medicii îl pot apăsa pentru a crea sau edita profilul lor personal, astfel încât să apară în lista de medici. A doua modificare constă într-o stea afișată pe cartea medicului, dacă acesta și-a creat și publicat profilul, pentru a-l informa că aceea este cartea sa.

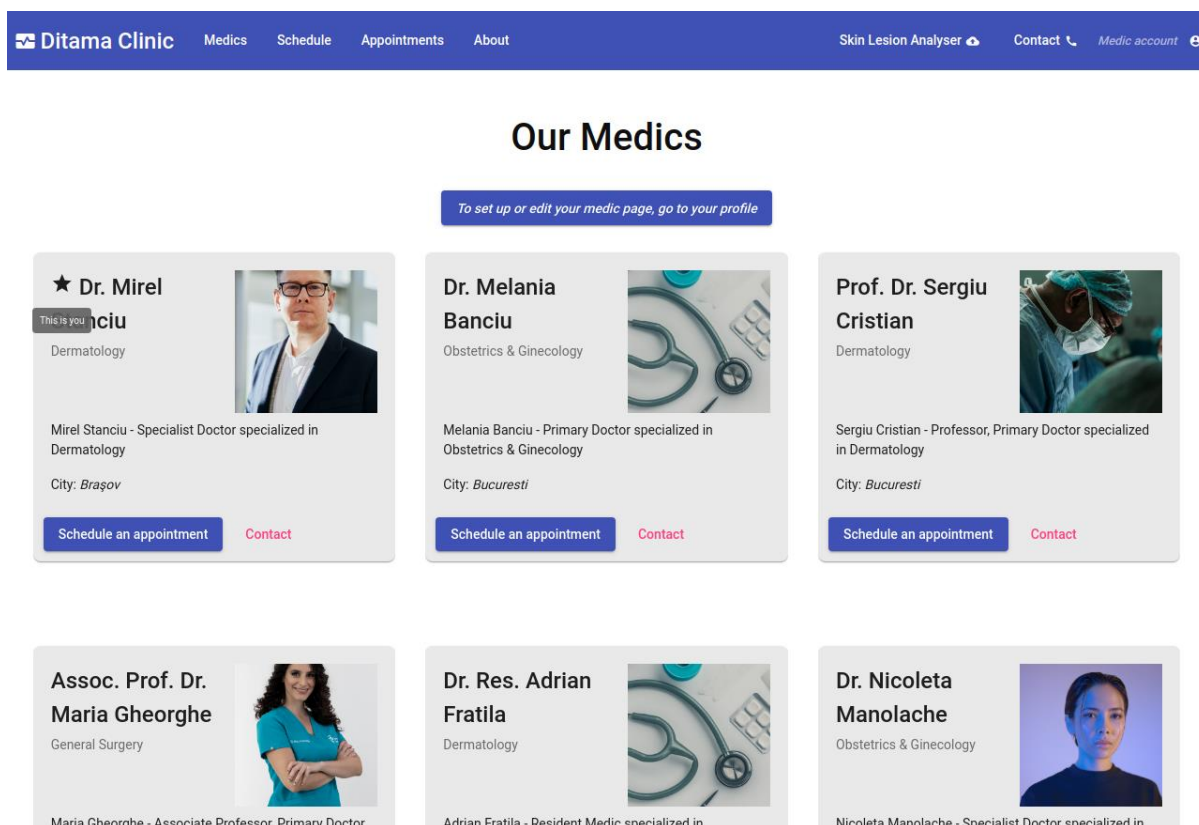


Figure 3-9 Aspectul paginii de medici pentru un utilizator cu rolul de medic

### 3.5 PAGINA DE AUTENTIFICARE

O pagină de autentificare este o interfață esențială pentru majoritatea aplicațiilor web și mobile, unde utilizatorii introduc acreditările lor pentru a accesa conținut sau funcționalități protejate. Prin această pagină se poate asigura că doar persoanele autorizate pot accesa anumite resurse.

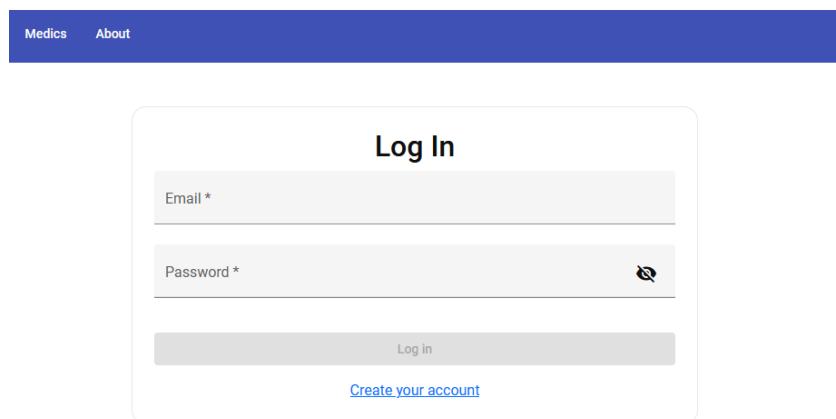
The image shows a web interface for a login page. At the top, there is a dark blue navigation bar with the text 'Medics' and 'About' in white. Below this, centered on the page, is a white rounded rectangle containing the login form. The form has a title 'Log In' in bold black text. It features two input fields: 'Email \*' and 'Password \*'. The password field has a small eye icon to its right. Below the input fields is a grey 'Log in' button. At the bottom of the form, there is a blue link that says 'Create your account'.

Figure 3-10 Interfața paginii de autentificare

În aplicația dezvoltată, această pagină include un formular care permite utilizatorilor să se înregistreze sau să se autentifice. Utilizatorii pot comuta între modul de înregistrare și modul de autentificare prin apăsarea unui buton. În modul de autentificare, formularul conține două câmpuri: email și parolă. În modul de înregistrare, formularul include trei câmpuri: email, parolă și confirmare parolă.

Formularul este creat folosind Angular Reactive Forms. Angular Reactive Forms reprezintă o abordare puternică și flexibilă pentru gestionarea formulelor în aplicațiile Angular. Spre deosebire de Template-driven Forms, care se bazează pe directivele Angular și manipularea directă a DOM-ului, Reactive Forms utilizează o programare declarativă și o structură de date bazată pe principiile reactive programming.

În Angular, Reactive Forms sunt construite utilizând obiectele "FormGroup" și "FormControl". FormGroup este o colecție de FormControl-uri care reprezintă un formular sau o secțiune a unui formular, în timp ce FormControl reprezintă un singur element de formular. Aceste obiecte permit dezvoltatorilor să definească și să gestioneze structura unui formular, inclusiv valorile implicite, stările și validările.



```
constructor(private fb: FormBuilder, private authService: AuthService, private router: Router)
{
  this.authForm = this.fb.group({
    email: ['', [Validators.required, Validators.email]],
    password: ['', [
      Validators.required,
      Validators.minLength(8),
      Validators.pattern(/^(?=.*[a-zA-Z])(?=.*\d).+$/)]
    ],
    repeatPass: ['', this.isLoginMode ? [] : [
      Validators.required,
      this.matchPasswordValidator()
    ]
  ]
  });
}
```

Figure 3-11 Generarea Reactive Form pentru pagina de autentificare

Un aspect esențial al Reactive Forms este capacitatea de a aplica validatori sincroni și asincroni. Validatorii sincroni, cum ar fi "Validators.required" sau "Validators.minLength", sunt funcții care rulează imediat ce valorile de intrare se schimbă. Un câmp dintr-un Reactive Form este considerat valid doar dacă toți validatorii săi sunt satisfăcuți, iar întregul formular este valid doar dacă toate câmpurile sale sunt valide. Pentru câmpul "repeatPass", am creat un validator personalizat "matchPasswordValidator()", care verifică dacă valoarea din acest câmp este identică cu valoarea din câmpul "password". Am utilizat acest validator, precum și alți validatori oferți de Angular pentru a asigura că formularul nu poate fi trimis cu valori invalide. În cazul în care un câmp conține o valoare invalidă, un mesaj de eroare va fi afișat utilizatorului folosind componenta "mat-error" din Angular Material. Aceasta abordare ajută la menținerea integrității datelor și oferă un feedback clar utilizatorilor pentru corectarea eventualelor erori.

Butonul de Sign up/Login este activat doar atunci când formularul este valid și dezactivat atunci când formularul este invalid. Acest buton utilizează flag-ul "isLoginMode" pentru a schimba textul afișat și acțiunea pe care o realizează. În modul de autentificare, butonul inițiază procesul de login, iar în modul de înregistrare, declanșează procesul de creare a unui nou utilizator.

**Register a new account**

Email \*  
fdsafdsa

Please enter a valid email address

Password \*  
••••

Password must be at least **8 characters** long  
Password must include at least **one letter** and **one number**

Repeat Password \*  
••••••••••

The passwords must **match**

Sign up

[Back to log in](#)

Figure 3-12 Erori afișate utilizatorului în cazul introducerii de valori invalide

Funcționalitatea esențială legată de autentificare și gestionarea utilizatorilor în aplicație este concentrată în cadrul serviciului “auth.service”. Acesta reprezintă un element de bază al sistemului, fiind responsabil pentru implementarea operațiilor cruciale asociate procesului de autentificare și administrare a sesiunilor utilizatorilor.

Prin intermediul acestui serviciu, se pot accesa funcții esențiale, cum ar fi “login()” și “signup()”, care facilitează procesele de autentificare și înregistrare, respectiv. Funcția “logout()” este responsabilă pentru deconectarea utilizatorului și curățarea sesiunii.

Administrarea autentificării și a utilizatorilor autentificați este gestionată folosind Firebase Authentication [16]. Un avantaj major al Firebase Authentication este securitatea integrată și gestionarea automată a autentificării.

În cadrul acestui proiect, am integrat funcțiile “Sign up with email / password” [30] și “Sign in with email / password” [30] oferite de Firebase Authentication. Aceste funcții permit utilizatorilor să se înregistreze sau să se autentifice folosind adresa lor de email și o parolă. În cazul unei erori în procesul de autentificare se va afișa o eroare pentru utilizatori. Dacă utilizatorul se înregistrează sau se autentifică cu succes, aceste funcții returnează un token de autentificare și informații suplimentare, precum cât timp este valabil acest token (exprimat în număr de secunde până la expirare).

<div> <input type="text" value="Search by email address, phone number, or user UID"/> <div> Add user </div> </div>				
Identifier	Providers	Created ↓	Signed In	User UID
obratu@gmail.com		Apr 4, 2024	May 25, 2024	lEzyKBm7fEZHYt9XIE7hYMG2...
asdffdsa@a.com		Jan 24, 2024	Jan 24, 2024	aVjou2tuNfTkWNawZfGkaWA...
nmanolache@gmail.com		Jan 20, 2024	May 25, 2024	pWKPCPnSFbdhqYSvDCIFONI...
afratila@gmail.com		Jan 20, 2024	May 25, 2024	9OQ37S6RgsXupSYMKTya6Lv...
abalan@gmail.com		Jan 20, 2024	May 30, 2024	dLzfMHIDL5fKEIS53nf5BKVg...
mbanciu@gmail.com		Jan 20, 2024	May 30, 2024	AWCHFr2PXWYxlbJBGdzcOQ...
nyoung@gmail.com		Jan 20, 2024	May 25, 2024	OGC8Bv1LHcUWUUD2edqjMH...
scristian@gmail.com		Jan 20, 2024	May 25, 2024	HvOkhDoaFlbgWbagEdcenyM...
erusu@gmail.com		Jan 20, 2024	May 25, 2024	aVbLuU6EvYQJJMYTuzZvUxS...
mgheorghe@gmail.com		Jan 20, 2024	May 25, 2024	MSKnpxhniTack5APIAznRaeXl...
mstanciu@gmail.com		Jan 20, 2024	Jun 2, 2024	rgAgoRVmuQYXY1hut2jHLzsl...
bmarin@gmail.com		Jan 20, 2024	May 30, 2024	lBFdJlHRqOQ2RdTEJnLsYboF...
stefan.ro95@gmail.com		Dec 23, 2023	Jun 2, 2024	rLfnPgFrBNVXQY8TRWRfAFi6...

Rows per page: 50
1 – 13 of 13

Figure 3-13 Interfața de vizionare și gestionare a utilizatorilor autentificați din Firebase

Pentru a gestiona acest token și informația despre expirarea acestuia, am salvat aceste date în memoria locală a browser-ului (localStorage). Pentru a facilita procesul de autentificare și deconectare automată, am implementat două funcții suplimentare: `autoLogin()` și `autoLogout()`. Funcția `autoLogin()` este apelată automat la deschiderea aplicației și verifică dacă token-ul de autentificare salvat în localStorage este încă valid (adică nu a expirat). În caz afirmativ, utilizatorul este autentificat automat. Pe de altă parte, funcția `autoLogout()` este responsabilă pentru deconectarea automată a utilizatorului atunci când token-ul expiră. Aceasta asigură că utilizatorul este gestionat corespunzător și în condiții de securitate, evitând utilizarea unui token expirat sau nefuncțional.

Prin intermediul funcției `getUser()` din serviciul `auth.service`, putem verifica starea utilizatorului curent autentificat în aplicație. Această funcție returnează valoarea `null` dacă utilizatorul nu este autentificat sau furnizează informații despre utilizator, cum

ar fi token-ul, în cazul în care acesta este autentificat. Această funcționalitate este utilizată în diverse pagini ale aplicației pentru a adapta interfața utilizatorului în funcție de starea sa de autentificare.

Conceptul de roluri este esențial în structura și funcționarea aplicației web. La crearea unui cont nou, utilizatorul primește automat rolul de pacient (Patient). Rolul de medic (Medic) poate fi atribuit unui utilizator doar de către administratorul backend-ului, oferindu-i acces la funcționalități și pagini specifice medicilor. Aceste roluri sunt fundamentale pentru gestionarea permisiunilor în cadrul aplicației, determinând accesul utilizatorilor la anumite pagini și acțiuni pe care le pot efectua. Astfel, rolurile asigură că utilizatorii interacționează cu aplicația în conformitate cu drepturile și responsabilitățile lor specifice, menținând securitatea și integritatea sistemului.

După autentificare, în bara de navigare a aplicației, apar modificări vizibile pentru utilizator. Se adaugă un buton nou care îl direcționează către pagina de programări (Appointments), un alt buton care îl conduce către pagina de verificare a programului (Schedule), în cazul în care utilizatorul conectat este medic, și un buton către pagina de analiză a leziunilor pielii (Skin Lesion Analyser). De asemenea, este afișat un text care indică rolul utilizatorului (Medic sau Patient). În locul butonului de login, apare un buton cu o iconiță, care la click deschide un meniu. Acest meniu include numele utilizatorului autentificat, un buton pentru pagina de profil (Profile) și un buton de logout. Meniul este creat folosind componenta "mat-menu" din Angular Materials. Astfel, meniul este construit în conformitate cu principiile design-ului Material și oferă o experiență coerentă și plăcută utilizatorului. Prin aceste modificări, interfața este adaptată în mod dinamic pentru a reflecta starea de autentificare a utilizatorului și pentru a oferi acces rapid și convenabil la funcționalitățile relevante.

### 3.6 PAGINA DE PROFIL

Pagina de profil poate fi accesată odată ce utilizatorul se autentifică în aplicație. Aceasta include un formular reactiv cu 11 câmpuri: Email, First Name, Last Name, Sex, Date of Birth, Nationality, Street, Country, State, City și Postal Code. Câmpul de email este completat automat și nu poate fi modificat, afișând adresa de email utilizată la înregistrarea contului. Restul câmpurilor sunt editabile și toate trebuie completate pentru a permite trimiterea formularului. Deși completarea acestor câmpuri nu este obligatorie,

informațiile colectate în această pagină pot fi utilizate ulterior în alte secțiuni ale aplicației. Fiecare câmp din formular prezintă un mesaj de eroare care apare atunci când sunt introduse informații invalide sau când câmpul este lăsat necompletat. Aceste mesaje de eroare asigură că utilizatorii sunt conștienți de necesitatea de a furniza informații corecte și complete, facilitând astfel validarea corectă a datelor înainte de trimiterea formularului.

The screenshot shows a web interface for editing user information. At the top, there is a navigation bar with links: 'nic', 'Medics', 'Appointments', 'About', 'Skin Lesion Analyser', and 'Contact'. The main heading is 'Edit your Information'. The form contains the following fields and values:

- Email: stefan.ro95@gmail.com
- First Name \*: Stefan
- Last Name \*: Popescu
- Sex \*: Male (dropdown menu)
- Date of Birth \*: 22/09/2001 (with a calendar icon and format DD/MM/YYYY)
- Nationality \*: Roman
- Street \*: Grivitei
- Country \*: Romania
- State \*: Brasov
- City \*: Brasov
- Postal Code \*: 500100

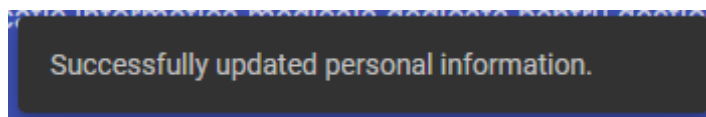
At the bottom right of the form, there is a '6/6' indicator. Below the form is a grey 'Update' button.

Figure 3-14 Pagina de Profil

Informațiile completate în acest formular, precum și alte detalii despre utilizator, cum ar fi rolul său, sunt stocate în colecția "users" din Firestore. Actualizarea acestor informații este gestionată prin intermediul serviciului "user.service.ts". Deoarece Firestore nu acceptă direct tipul "Date" și utilizează în schimb "Timestamp", a fost necesară crearea unei funcții în "user.service" care să convertească "Timestamp" în "Date" atunci când datele de profil ale utilizatorului sunt preluate din backend. La deschiderea paginii de profil, aplicația verifică dacă există deja date pentru utilizatorul curent în backend și, dacă aceste date există, ele sunt completate automat în câmpurile corespunzătoare, asigurând astfel o experiență fluidă și intuitivă pentru utilizator.

După completarea și trimiterea formularului, utilizatorul va fi notificat printr-un snackbar din Angular Material. Snackbar-ul este o componentă de notificare temporară care apare în partea de jos a ecranului, informând utilizatorul despre rezultatul unei acțiuni. Dacă operațiunea de trimitere a formularului se încheie cu succes, snackbar-ul va afișa un mesaj confirmând că acțiunea a fost finalizată cu succes. În cazul în care apare o eroare, snackbar-ul va afișa un mesaj de eroare corespunzător, informând utilizatorul despre problema întâmpinată. Această metodă de feedback vizual și rapid îmbunătățește

experiența utilizatorului, asigurându-se că acesta este întotdeauna conștient de starea acțiunilor sale.



*Figure 3-15 Mesajul de succes afișat folosind Snackbar*

### 3.7 PAGINA DE PROGRAMĂRI (APPOINTMENTS)

Pagina de programări reprezintă punctul de plecare pentru una dintre cele mai esențiale funcționalități ale aplicației: programarea la medic. Accesul la această pagină este restricționat folosind "Angular Guards" doar la utilizatorii înregistrați, asigurându-se astfel securitatea și relevanța informațiilor afișate. Structura paginii este organizată în două secțiuni principale: "Create an Appointment" și "View Your Appointments". Această diviziune funcțională a fost realizată utilizând Angular Material Tabs, care oferă o interfață intuitivă și ușor de navigat.

#### 3.7.1 Secțiunea "Create an appointment"

Secțiunea "Create an appointment" din cadrul paginii de programări permite utilizatorilor să stabilească întâlniri cu medicii înscriși în aplicație. Această secțiune este constituită dintr-un formular reactiv (Angular Reactive Form) care include diverse câmpuri esențiale pentru o programare eficientă: medicul ales, data și ora programării, informațiile personale ale pacientului și, opțional, o scurtă descriere și o fotografie reprezentativă a afecțiunii (în cazul în care afecțiunea este una vizibilă).

### Make a new appointment

Complete the following form to create your appointment today!

City \*

Specialty

Medic

Date \*

Time Slot

Autofill information

First Name \*

Last Name \*

Email \*

Sex \*

Age \*

+40 Phone Numbe...

Short description of your problem

Max 250 characters0/250

Optional: Add an image.

Please only add an image if you believe it will help the medic diagnose you

Submit

Figure 3-16 Formularul din secțiunea "Create an Appointment"

Formularul poate fi completat manual de către utilizator sau poate beneficia de funcționalități de completare automată. De exemplu, din pagina dedicată medicilor menționată anterior, utilizatorii pot accesa butonul "Schedule an appointment" situat în cartea fiecărui medic. Prin apăsarea acestui buton, utilizatorul este redirecționat către pagina de programări, specific în secțiunea "Create an appointment", cu informațiile relevante despre medic (orașul, specialitatea și numele medicului) completate automat.

În plus, datele personale ale pacientului pot fi completate automat în formular prin intermediul butonului "Autofill information". Acest buton este activ (poate fi apăsat) numai dacă utilizatorul a completat și a salvat anterior informațiile personale în pagina de profil. Astfel, dacă un utilizator a introdus detaliile sale în profil, aceste date vor fi preluate și populate automat în câmpurile corespunzătoare din formularul de programare, economisind timp și asigurând acuratețea informațiilor. Pentru a calcula vârsta pacienților pe baza datei lor de naștere, am creat o funcție personalizată denumită "calculateAge()".

Această funcție ia în considerare data nașterii și data curentă, calculând vârsta exactă a pacientului. Această informație este apoi utilizată și afișată în diverse secțiuni ale aplicației, asigurând acuratețea datelor demografice ale pacientului.

The diagram illustrates the automatic completion of appointment data across three forms:

- Dr. Mirel Stanciu Profile:**
  - Specialty: Dermatology
  - City: Brașov
  - Buttons: Schedule an appointment, Contact
- Edit your Information Form:**
  - Email: stefan.ro95@gmail.com
  - First Name: Stefan
  - Last Name: Popescu
  - Sex: Male
  - Date of Birth: 22/09/2001
  - Nationality: Roman
  - Street: Grivitei
  - Country: Romania
  - State: Brasov
  - City: Brasov
  - Postal Code: 500100
  - Update button
- Make a new appointment Form:**
  - City: Brașov (selected from dropdown)
  - Specialty: Dermatology (selected from dropdown)
  - Medic: Dr. Stanciu Mirel (selected from dropdown)
  - Date: (calendar icon)
  - Time Slot: (dropdown)
  - Autofill information section:
    - First Name: Stefan
    - Last Name: Popescu
    - Email: stefan.ro95@gmail.com
    - Sex: Male
    - Age: 22
    - Phone Number: +40 Phone Numbe...
  - Short description of your problem (text area, Max 250 characters)
  - Optional: Add an image button
  - Please only add an image if you believe it will help the medic diagnose you

Arrows indicate the flow of data: from the user's profile information to the appointment form, and from the doctor's profile to the appointment form.

Figure 3-17 Diagrama care exemplifică completarea automată a datelor unei programări

Această funcționalitate de completare automată este crucială pentru a asigura o experiență de utilizare fluentă și eficientă. Utilizatorii beneficiază de un proces simplificat și de o reducere a erorilor care pot apărea în cazul completării manuale. De asemenea, integrarea cu pagina de profil și cu pagina de medici demonstrează un nivel ridicat de interconectare între diversele secțiuni ale aplicației, îmbunătățind astfel coerența și navigabilitatea.

În secțiunea "Create an appointment", câmpurile pentru oraș, specializare, medic, dată și oră sunt de tip dropdown, concepute pentru a oferi utilizatorilor o selecție dintr-o listă predefinită de opțiuni. La încărcarea inițială a paginii, aplicația preia datele tuturor medicilor din baza de date printr-un apel la API-ul Firebase și colectează toate orașele în



care aceștia își desfășoară activitatea. Dropdown-ul pentru oraș este populat cu aceste locații distincte, iar utilizatorii pot selecta una din aceste locații.

Odată ce utilizatorul selectează un oraș, aplicația efectuează o nouă interogare în baza de date pentru a obține lista medicilor din orașul respectiv, împreună cu specializările acestora. Dropdown-ul pentru specializare este apoi actualizat cu toate specializările distincte. După selectarea unei specializări, se face un alt apel API pentru a prelua toate documentele din baza de date care corespund medicilor din orașul selectat și cu specializarea aleasă. Acest proces filtrează medicii relevanți, care sunt ulterior disponibili pentru selecție în dropdown-ul dedicat medicilor.

În mod similar, câmpurile pentru dată și timp funcționează într-o manieră coordonată pentru a oferi o experiență fluidă și intuitivă utilizatorului. După ce utilizatorul selectează un medic, aplicația interoghează baza de date pentru a determina disponibilitatea calendaristică a acestuia. Acest proces asigură că numai datele în care medicul este disponibil sunt prezentate în dropdown-ul de selecție a datei.

Odată ce o dată specifică este aleasă, aplicația verifică programul medicului pentru acea zi pentru a stabili orele și minutele exacte în care acesta este disponibil pentru programări. Dropdown-ul de timp este populat cu intervale de timp prestabilite, de la ora 08:00 până la 16:00, cu o frecvență de 20 de minute (de exemplu, 08:00, 08:20, 08:40, etc.), iar dacă medicul are deja o programare de la o anumită oră, aceasta va fi dezactivată și indisponibilă (nu va putea fi selectată). Această abordare permite utilizatorilor să selecteze o fereastră temporală convenabilă pentru consultația lor.

Aplicația este concepută astfel încât să fie extrem de flexibilă în privința intervalelor de timp între programări. Deși intervalul implicit este de 20 de minute, această valoare poate fi ajustată cu ușurință pentru a se potrivi mai bine necesităților specifice ale consultațiilor, fie că este necesar un timp mai scurt sau mai lung. Această flexibilitate este implementată în logica aplicației, asigurându-se că sistemul poate funcționa eficient indiferent de intervalul ales.

Figure 3-18 Câmpurile de tip dropdown coordonate. Se poate observa că nu se poate selecta ora 09:20 deoarece medicul selectat are deja o consultație programată la acea oră

Această metodologie asigură că utilizatorul are acces doar la date valide și actualizate în fiecare dintre aceste câmpuri. În plus, interacțiunea dintre aceste dropdown-uri este configurată pentru a menține integritatea datelor. Dacă utilizatorul schimbă specializarea, câmpul pentru medic va fi resetat automat. Similar, dacă orașul selectat este modificat, atât dropdown-urile pentru specializare, cât și pentru medic vor fi resetate, eliminând datele curente și solicitând o nouă selecție. Această structură nu doar că optimizează experiența utilizatorului, dar și previne erorile prin asigurarea că selecțiile sunt întotdeauna coerente și valide.

Odată ce toate câmpurile necesare au fost completate, utilizatorul poate trimite formularul de programare. Datele colectate din acest formular sunt organizate într-un obiect de tip "Appointment", definit în fișierul "appointment.ts" din Angular. Pe lângă informațiile de bază ale programării, se adaugă și ID-ul utilizatorului care a creat programarea, pentru a facilita identificarea ulterioară. Acest ID este obținut de la API în momentul deschiderii paginii de programări (Appointments).

Obiectul "Appointment" astfel creat este trimis la serviciul "AppointmentService", și mai precis, la funcția "addAppointment()". Această funcție are rolul de a converti obiectul "Appointment" într-un obiect de tip "FirebaseAppointment", definit în

"FirestoreAppointment.ts". Diferența principală între cele două tipuri de obiecte este reprezentată de tipul de dată folosit pentru variabila "datetime". În timp ce "Appointment" utilizează tipul "Date", "FirestoreAppointment" folosește tipul "Timestamp", specific Firestore, pentru a stoca datele calendaristice. Această conversie este esențială deoarece Firestore stochează datele de tip calendaristic folosind tipul "Timestamp" [31]. Prin urmare, toate datele unei programări stocate în backend sunt de tip "FirestoreAppointment". La recuperarea acestor date printr-un apel API, ele sunt convertite înapoi la tipul "Appointment" pentru a fi utilizate în aplicație. De fiecare dată când un obiect de tip "Appointment" este adăugat în baza de date, el este mai întâi transformat într-un "FirestoreAppointment" și vice-versa.

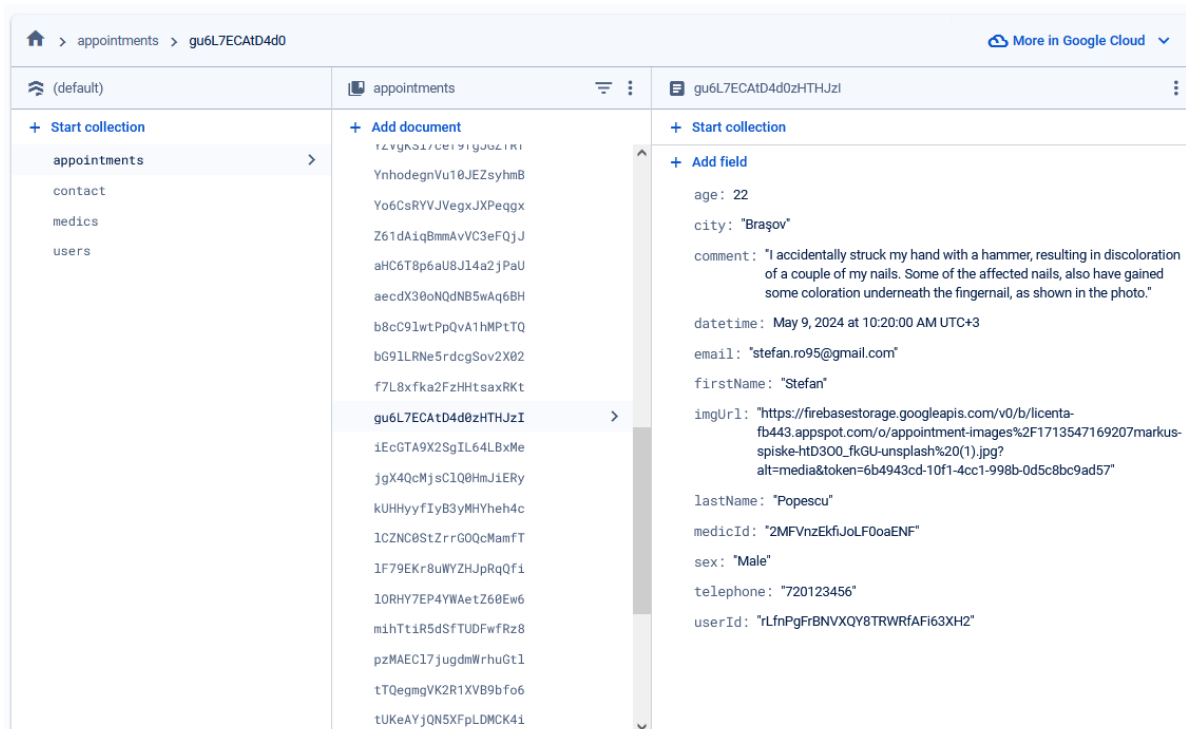


Figure 3-19 Structura bazei de date Cloud Firestore pentru programări

În cazul în care un utilizator de tip pacient adaugă o imagine la crearea unei programări, procesul de trimitere a formularului include un pas suplimentar pentru gestionarea acestei imagini. Înainte de încărcarea datelor programării în baza de date Firestore, aplicația utilizează serviciul Firebase Storage pentru a stoca imaginea. Imaginea este încărcată în acest serviciu în folderul specific "appointment-images/". După ce imaginea a fost încărcată cu succes, aplicația preia URL-ul generat pentru imagine și îl

adaugă în obiectul de tip "Appointment". Numai după finalizarea acestor pași, programarea este încărcată în baza de date Firestore în colecția appointments.

În cazul în care utilizatorul pacient nu a introdus o imagine, se sare peste pașii referitori la încărcarea imaginii și se trece direct la încărcarea obiectului "Appointment" în baza de date Firestore. Acest proces asigură că toate informațiile relevante, inclusiv imagini, sunt gestionate corespunzător și sunt disponibile pentru medic la vizualizarea detaliilor programării. Utilizarea serviciului Firebase Storage asigură o gestionare eficientă a fișierelor, oferind un mediu securizat și scalabil pentru stocarea și accesarea imaginilor medicale.

The screenshot displays the 'View your appointments' interface. At the top, there are two tabs: 'Create an appointment' and 'View your appointments', with the latter being active. Below the tabs, the section is titled 'Upcoming appointments' in green. It contains a table with two rows, each showing 'Upcoming appointment' and 'May 28, 2024'. Below this, the section is titled 'Past appointments' in red. It contains a table with 15 rows, each showing 'Past appointment' and a date ranging from 'May 14, 2024' down to 'Mar 11, 2024'. A dark toast message 'Successfully created your appointment.' is displayed over the bottom of the 'Past appointments' table.

Upcoming appointments	
Upcoming appointment	May 28, 2024
Upcoming appointment	May 28, 2024

Past appointments	
Past appointment	May 14, 2024
Past appointment	May 14, 2024
Past appointment	May 9, 2024
Past appointment	May 9, 2024
Past appointment	May 9, 2024
Past appointment	May 8, 2024
Past appointment	May 6, 2024
Past appointment	Apr 18, 2024
Past appointment	Apr 16, 2024
Past appointment	Mar 27, 2024
Past appointment	Mar 26, 2024
Past appointment	Mar 26, 2024
Past appointment	Mar 16, 2024
Past appointment	Mar 11, 2024

Figure 3-20 Secțiunea "View your Appointments". În acest caz, utilizatorul a fost redirecționat automat după crearea unei programări (se poate observa un snackbar informativ)

Odată ce apelul API pentru crearea programării este finalizat, utilizatorul primește o notificare prin intermediul unui snackbar, oferit de Angular Material. Acest snackbar afișează un mesaj care informează utilizatorul despre rezultatul operațiunii – fie că

programarea a fost creată cu succes, fie că a apărut o eroare. În cazul în care programarea a fost creată cu succes, adică apelul API s-a încheiat fără probleme, utilizatorul este redirecționat automat către secțiunea "View your Appointments". Această abordare integrată de notificare și redirecționare nu doar îmbunătățește experiența utilizatorului, dar și asigură o gestionare transparentă și eficientă a programărilor. Prin implementarea acestor funcționalități, aplicația devine mai user-friendly și mai robustă, facilitând interacțiuni mai fluide și eficiente pentru utilizatori.

### 3.7.2 Secțiunea "View your appointments"

Secțiunea "View your appointments" este esențială pentru gestionarea și vizualizarea programărilor și constă în două liste distincte: "Upcoming appointments" și "Past appointments". Programările din lista "Upcoming appointments" sunt cele care au o dată de programare în viitor, în timp ce lista "Past appointments" conține programările ale căror date de programare sunt în trecut. Programările din lista programărilor viitoare sunt afișate în ordine crescătoare, în funcție de data programării. Astfel, programarea care are loc cel mai curând este afișată prima în listă, iar celelalte urmează în ordine cronologică. În schimb, în lista programărilor trecute, programările sunt afișate în ordine descrescătoare, cu cele mai recente programări afișate primele.

Pentru afișarea acestor liste am utilizat componentele "mat-accordion" și "mat-expansion-panel" din Angular Material. Aceste componente permit organizarea compactă a informațiilor și îmbunătățesc experiența utilizatorului prin interacțiuni intuitive. Fiecare programare este afișată într-un "expansion-panel", care prezintă inițial doar detaliile esențiale. Atunci când utilizatorul dorește să afle mai multe informații despre o anumită programare, poate face clic pe aceasta pentru a extinde panel-ul și a vizualiza toate detaliile completate în momentul programării.

Această abordare are mai multe avantaje. În primul rând, permite utilizatorului să navigheze cu ușurință prin programările sale, oferind o vizualizare clară și ordonată. În al doilea rând, utilizarea "accordion" și "expansion-panel" asigură o utilizare eficientă a spațiului pe ecran, oferind posibilitatea de a afișa detalii suplimentare doar la cerere. Acest lucru este esențial pentru a menține o interfață curată și neaglomerată, îmbunătățind astfel experiența generală a utilizatorului. În plus, informațiile afișate în "expansion-panel" includ toate datele completate în momentul creării programării, cum ar fi numele medicului, data și ora programării, detalii personale ale pacientului și orice descriere sau fotografie adăugată. Această funcționalitate nu doar că facilitează accesul la

informații complete și relevante, dar și permite utilizatorilor să gestioneze și să verifice cu ușurință detaliile programărilor lor.

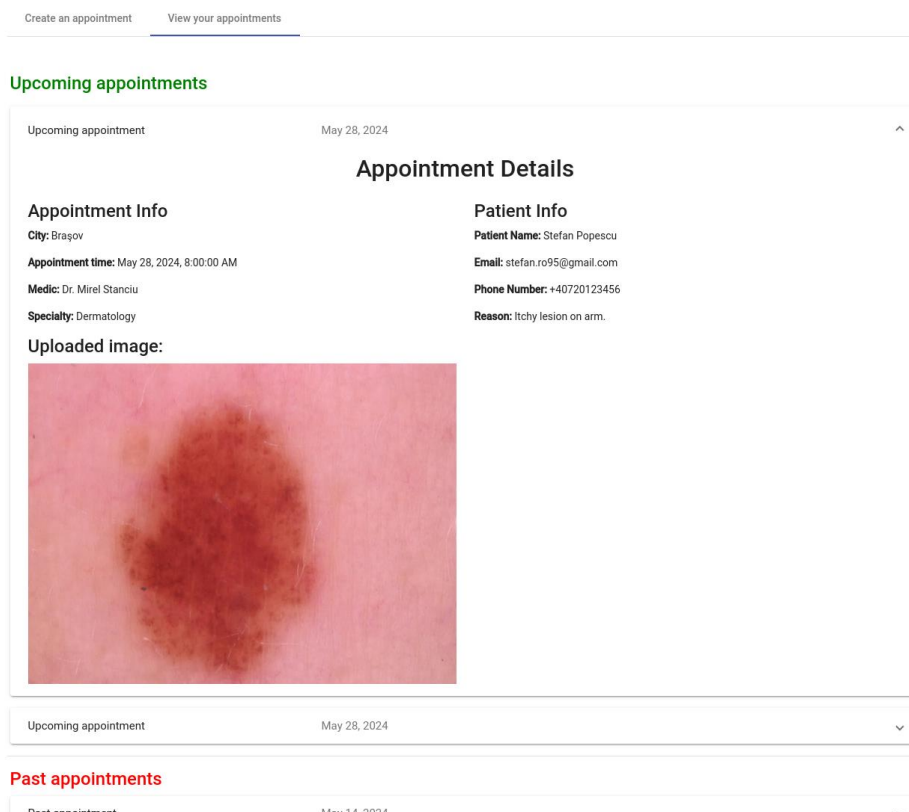


Figure 3-21 "Deschiderea" unei programări din secțiunea "View your appointments" pentru a vedea detaliile acesteia

Aceste două liste de programări sunt obținute prin intermediul unui apel API către Firestore, de unde sunt preluate documentele relevante pentru utilizatorul curent, identificat prin câmpul "userId" care corespunde "userId"-ului utilizatorului autentificat. Procesul de obținere a datelor implică un singur apel API, care aduce toate programările asociate utilizatorului respectiv. Odată ce datele sunt recepționate, acestea sunt prelucrate pentru a fi distribuite corespunzător în cele două liste: "Upcoming appointments" și "Past appointments". Această prelucrare implică compararea datei și orei fiecărei programări cu data și ora curentă. Programările care au o dată de programare în viitor sunt adăugate în lista "Upcoming appointments", în timp ce programările cu o dată de programare în trecut sunt adăugate în lista "Past appointments".

### 3.8 PAGINA DE PROGRAM (SCHEDULE)

În cazul în care un utilizator de tip medic accesează pagina "Schedule" fără a-și fi adăugat propriul profil în secțiunea dedicată medicilor, în locul conținutului uzual, va

apărea un mesaj informativ. Acest mesaj îi comunică utilizatorului că trebuie să își configureze profilul de medic în secțiunea de profil. Sub acest text informativ, se află un buton care redirecționează utilizatorul către pagina de profil, facilitând astfel completarea informațiilor necesare.

În cazul în care un utilizator de tip medic și-a creat anterior pagina de medic, la accesarea paginii "Schedule", va apărea un tabel care prezintă programul medicului, denumit "Schedule Table". Această abordare asigură o experiență coerentă și intuitivă pentru utilizatori, prevenind confuziile și asigurând că toți medicii au profilurile corect configurate înainte de a accesa și utiliza funcționalitățile avansate ale paginii "Schedule".

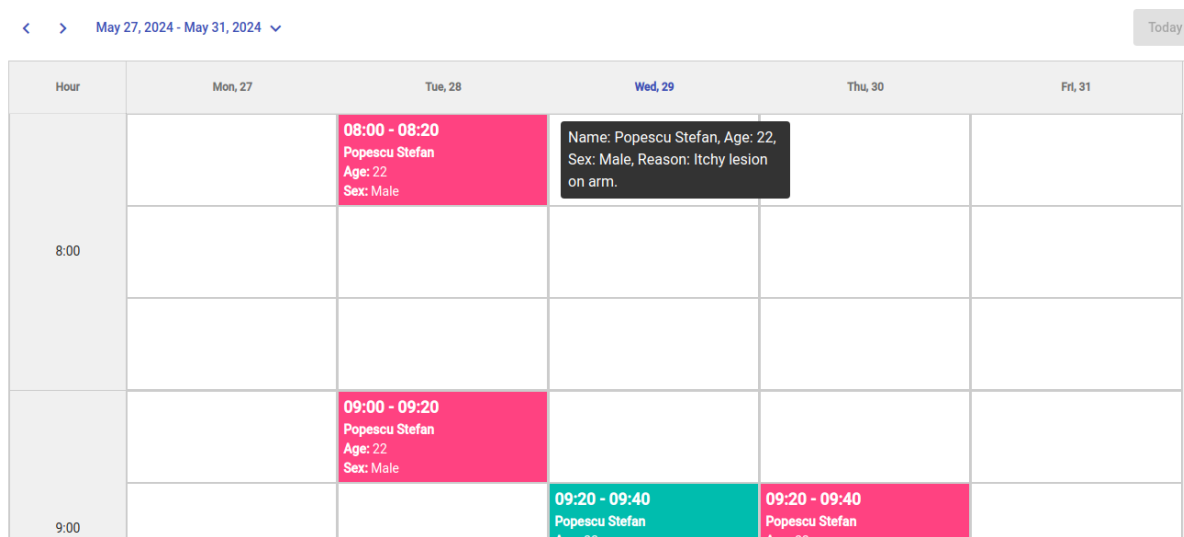
### 3.8.1 "Schedule Table"

Pagina de program (Schedule) din aplicație este dedicată utilizatorilor cu rolul de medic, oferindu-le o vizualizare clară și organizată a programărilor făcute la ei. Utilizatorii care nu au rolul de medic, nu pot accesa această pagină, iar butonul către aceasta este ascuns din meniul de navigare. Această pagină este construită sub forma unui calendar virtual, reprezentat printr-un tabel dinamic utilizând componenta "mat-table" din Angular Material.

Fiecare coloană a tabelului reprezintă o zi a săptămânii, etichetată cu ziua și data (de exemplu, "Mon, 27"). Dacă ziua curentă este inclusă în tabel, titlul coloanei respective va avea o culoare distinctă pentru a evidenția mai ușor ziua curentă. Fiecare rând corespunde unei ore din intervalul orar 8:00 - 15:00. Pentru a permite programări la intervale de 20 de minute, fiecare rând orar este subdivizat în trei rânduri mai mici. Această structură este flexibilă și poate fi ajustată pentru a permite alte intervale de timp, precum 15 sau 30 de minute, în funcție de necesitățile specifice ale programărilor medicale. Tabelul a fost proiectat încât să poată fi setată orice valoare în loc de 20 de minute (de exemplu 30 de minute pentru o programare în cazul în care este necesar mai mult timp sau 15 minute în cazul în care este necesar mai puțin timp pentru o programare).

Am implementat o funcționalitate în care primul rând și prima coloană ale tabelului sunt făcute "sticky", folosind proprietățile CSS disponibile în Angular Material. Acest lucru permite utilizatorilor să urmărească cu ușurință data și ora curentă din tabel atunci când derulează tabelul în jos, îmbunătățind astfel navigarea și vizualizarea informațiilor.

Pentru a evita și rezolva problemele cauzate de fusurile orare, am setat timpul tuturor datelor care nu necesită specificarea componentelor de timp la valorile (0,0,0,0) sau (23,59,59,999). Astfel, ne asigurăm că datele sunt coerente și nu apar discrepanțe din cauza diferențelor de fus orar. Datele sunt formatate utilizând "DatePipe" din Angular pentru a asigura o afișare uniformă și clară.



Hour	Mon, 27	Tue, 28	Wed, 29	Thu, 30	Fri, 31
8:00		08:00 - 08:20 Popescu Stefan Age: 22 Sex: Male	Name: Popescu Stefan, Age: 22, Sex: Male, Reason: Itchy lesion on arm.		
9:00		09:00 - 09:20 Popescu Stefan Age: 22 Sex: Male	09:20 - 09:40 Popescu Stefan Age: 22	09:20 - 09:40 Popescu Stefan Age: 22	

Figure 3-22 Interfața paginii de program a medicului

Deasupra tabelului, sunt plasate patru butoane care permit navigarea și personalizarea vizualizării calendarului: două săgeți pentru navigarea între săptămâni, un buton etichetat cu intervalul de zile selectate (de exemplu, "May 27, 2024 - May 31, 2024") care deschide un dropdown (o componentă de tip datepicker [12] din Angular Material) pentru selectarea manuală a unei date, și un buton "Today".

La prima deschidere a paginii, tabelul este configurat să afișeze zilele lucrătoare din săptămâna curentă, de luni până vineri. Navigarea între săptămâni se face prin butoanele săgeți: săgeata înapoi mută vizualizarea cu o săptămână în urmă, iar săgeata înainte avansează vizualizarea cu o săptămână. Selectarea unei date manuale din butonul datepicker actualizează calendarul pentru a afișa zilele lucrătoare din săptămâna ce conține data selectată. În cadrul datepicker-ului, zilele nelucrătoare (sâmbăta și duminica) sunt dezactivate (nu pot fi selectate), aceasta fiind realizată prin aplicarea unui filtru specific pe componenta datepicker. Butonul "Today" este dezactivat în mod implicit. Totuși, dacă săptămâna afișată în calendar diferă de săptămâna curentă, butonul devine



activ. Când utilizatorul apasă pe acest buton, vizualizarea calendarului se actualizează pentru a afișa săptămâna curentă.

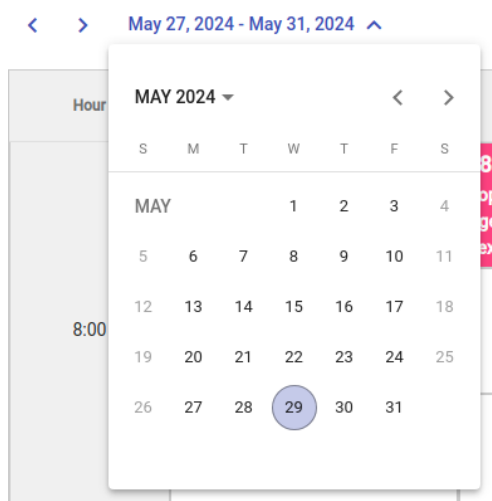


Figure 3-23 Componenta de selectare a unei date calendaristice (Datepicker)

Până la încărcarea completă a datelor, utilizatorilor li se afișează o componentă "Progress bar" din Angular Material, care semnalează că datele sunt în curs de încărcare. Aceasta oferă un feedback vizual imediat, reducând incertitudinea și îmbunătățind experiența utilizatorului.

Pentru afișarea programărilor, la prima accesare a paginii și de fiecare dată când utilizatorul schimbă săptămâna, se efectuează un apel API către baza de date Firestore pentru a prelua programările utilizatorului curent (medic) din săptămâna selectată. După ce programările sunt preluate, aplicația parcurge toate zilele, orele și intervalele de timp (de exemplu, intervale de 20 de minute) din săptămâna respectivă. Dacă în lista programărilor aduse de API se găsește o programare pentru ziua, ora și minutul respectiv, se introduce o casetă în tabel care afișează ora programării (de exemplu, 08:00 - 08:20), numele pacientului, vârsta pacientului și sexul pacientului. Dacă nu se găsește o programare la ziua, ora și minutul respectiv, caseta din tabel rămâne goală (o casetă albă fără text).

Medicul poate vedea clar în ce zi, la ce oră și pentru ce interval de timp are fiecare programare, precum și perioadele de timp liber. Singura valoare prezentă în obiectele de tip programare este o dată de tip Date care conține data (ziua), ora și minutul de început al programării. Ora de sfârșit a programării este calculată utilizând o funcție specifică, dezvoltată pentru acest scop. Această funcție determină ora de sfârșit a programării pe baza intervalului de timp alocat fiecărei programări (de exemplu, 20 de minute). Design-ul

flexibil al tabelului permite ajustarea ușoară a intervalelor de timp pentru programări, pentru a se adapta nevoilor specifice ale fiecărui medic și tipului de consultație oferit.

Dacă utilizatorul plasează cursorul deasupra unei casete de programare, va apărea un tooltip (creat utilizând componenta mat-tooltip [12] din Angular Material) care afișează informații suplimentare despre programare, inclusiv motivul specificat de pacient în momentul creării programării. Astfel, medicul poate vedea rapid motivul pentru care pacientul s-a programat, facilitând o gestionare mai eficientă a întâlnirilor.

### 3.8.2 "Schedule Details"

Dacă medicul apasă pe caseta unei programări, se deschide un dialog creat cu ajutorul componentei "mat-dialog" [12] din Angular Material. Acest dialog oferă toate detaliile relevante ale programării, inclusiv data și ora, precum și informații despre pacient, cum ar fi numele, vârsta, sexul, emailul, numărul de telefon și motivul programării, dacă acesta a fost specificat.

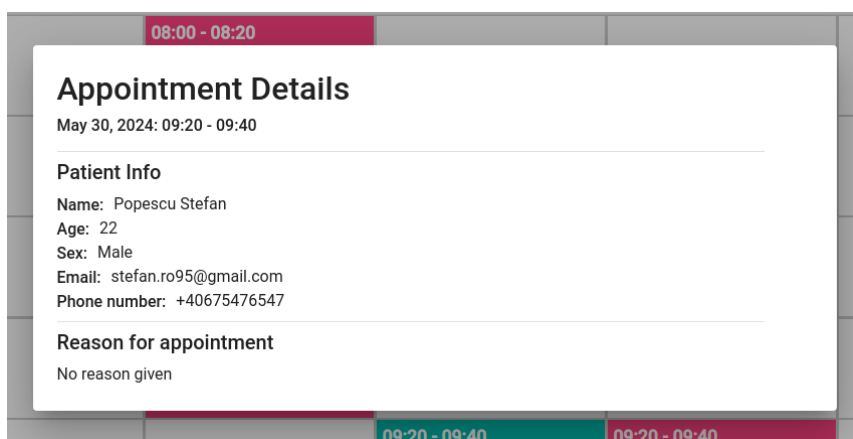
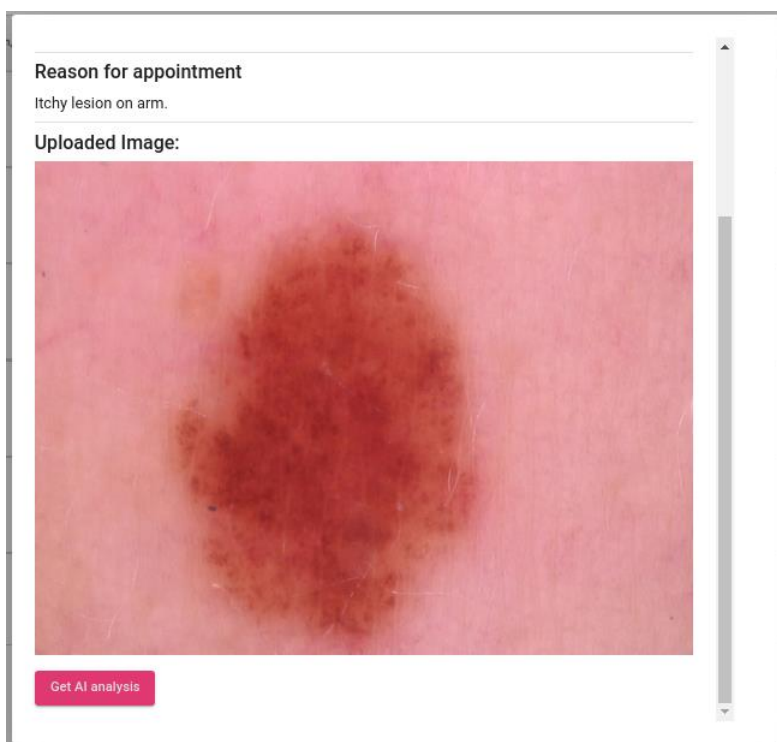


Figure 3-24 Componenta "Dialog" care afișează detalii despre programare

În cazul în care pacientul a încărcat și o fotografie la momentul programării, dialogul va include o secțiune suplimentară intitulată "Uploaded Image", unde este afișată fotografia respectivă, alături de un buton intitulat "Get AI Analysis".



*Figure 3-25 Componenta dialog în cazul în care pacientul a trimis o poză la crearea programării*

Dacă medicul apasă acest buton, imaginea este transmisă serviciului "lesionClassificationService", care utilizează un model de clasificare imagistică antrenat anterior și integrat în aplicație. Modelul analizează imaginea și returnează predicții sub forma unui procentaj pentru fiecare categorie de leziuni. De exemplu, modelul poate indica faptul că imaginea este 95% probabil un Nevus melanocitic (nv), 3% probabil un Melanom (mel), și așa mai departe.

Primele trei categorii de leziuni în ordine descrescătoare a procentajelor sunt afișate medicului, împreună cu procentajele corespunzătoare pentru fiecare categorie. Sub aceste predicții, este prezentat un mesaj de tip "disclaimer" care subliniază faptul că aceste predicții au doar rol informativ și nu ar trebui utilizate pentru diagnostice medicale definitive. Acest mesaj este esențial pentru a clarifica natura ajutătoare a modelului AI și pentru a preveni utilizarea necorespunzătoare a predicțiilor în luarea deciziilor clinice.

În ansamblu, această funcționalitate adaugă o dimensiune avansată de analiză și asistență pentru medici, permițându-le să obțină informații suplimentare despre leziuni cutanate într-un mod rapid și eficient, integrat direct în fluxul de lucru al programărilor.



Figure 3-26 Predicțiile modelului pentru imaginea data

### 3.9 PAGINA PROFILULUI DE MEDIC (MEDIC-PROFILE)

Pagina de profil a medicului reprezintă o componentă esențială a aplicației, oferind medicilor posibilitatea de a-și crea și edita profilul profesional. Accesul la această pagină este restricționat doar utilizatorilor care dețin rolul de medic și care și-au completat în prealabil profilul personal din pagina "Profile". Odată ce profilul personal este completat cu succes, utilizatorul poate accesa pagina de profil a medicului prin intermediul unui buton situat în partea superioară a paginii de profil, care devine activ după completarea profilului personal.

Pagina de profil a medicului conține un formular reactiv dezvoltat în Angular, permițând medicilor să introducă și să actualizeze informațiile profesionale esențiale. Aceste informații includ specialitatea, titulatura, orașul unde profesează sau unde își desfășoară activitatea de cabinet, un email profesional și un număr de telefon. Formularul este construit pentru a facilita o experiență de utilizare intuitivă și eficientă, asigurându-se că toate câmpurile necesare sunt ușor accesibile și completabile.

Pentru câmpul de selectare a orașului, s-a utilizat un dropdown care conține o listă cu cele mai mari 40 de orașe din România. Pentru a obține aceasta listă, am descărcat un fișier JSON oferit de Simplemaps [32], care conține informații detaliate despre toate orașele din România, inclusiv coordonate geografice și populație. Pentru a simplifica și eficientiza selecția orașelor, am creat un script Python care a extras doar numele orașelor din acest fișier. S-a decis utilizarea doar a celor mai mari 40 de orașe pentru a evita o listă prea lungă și dificil de navigat. De asemenea, orașele au fost sortate alfabetic, facilitând astfel găsirea rapidă a orașului dorit de utilizator.

```
extract-city-names.py

import json

# Define the input filename
input_filename = "ro.json"

# Read data from the JSON file
try:
    with open(input_filename, 'r', encoding='utf-8') as infile:
        data = json.load(infile)
except FileNotFoundError:
    print(f"Error: Input file '{input_filename}' not found.")
    exit()

# Extract city names
city_names = [city["city"] for city in data]

# Print the city names as an array
print(f"City names: {city_names}")
```

Figure 3-27 Codul folosit pentru a extrage numele orașelor

Pagina de profil a medicului nu se limitează doar la completarea informațiilor profesionale de bază, ci permite și personalizarea profilului prin încărcarea unei fotografii reprezentative. Interfața acestei funcționalități este concepută pentru a fi prietenoasă și intuitivă, oferind o experiență de utilizare optimizată. În acest sens, există un chenar destinat fotografiei, care este implementat cu poziționare relativă (CSS: "position: relative"). Peste acest chenar, se suprapune un buton Angular Material de tip "mat-fab", plasat cu poziționare absolută (CSS: "position: absolute"), pentru a permite utilizatorului să încarce o nouă imagine.

## Edit your medic page



### Dr. Stanciu Mirel

Specialist Doctor in Dermatology

#### Medic information

<div>Last Name</div> <div>Stanciu</div>	<div>First Name</div> <div>Mirel</div>
<div>Specialty *</div> <div>Dermatology</div>	
<div>Title *</div> <div>Specialist Doctor</div>	<div>Short Title *</div> <div>Dr.</div>

#### Doctor's office information

City \*

Brașov

Work Email \*

mstanciu@gmail.com

Phone Number \*

+40 723770640

Your page is currently visible

Update	Hide your page from the medic list
--------	------------------------------------

Figure 3-28 Pagina de profil a medicului

Lângă fotografia de profil, se află două div-uri care afișează numele medicului, titulatura și specialitatea, formate exact așa cum vor apărea în pagina de medici. Aceste informații se actualizează în timp real pe măsură ce utilizatorul completează câmpurile din formular, utilizând mecanismul de two-way data binding specific Angular. Aceasta înseamnă că orice modificare făcută în formular este imediat reflectată în secțiunea de previzualizare, permițând utilizatorului să vadă cum va arăta profilul său înainte de a-l salva și de a-l face vizibil în aplicație. Astfel, se asigură că modificările sunt sincronizate în timp real între modelul de date și interfața utilizatorului. Aceasta ajută la prevenirea erorilor și la asigurarea că datele introduse sunt corecte și complete înainte de a fi salvate.

După ce utilizatorul a completat formularul de profil, butonul de actualizare devine activ. La trimiterea formularului, se urmează un proces similar cu cel de la crearea unei programări. În primul rând, imaginea încărcată este stocată în Firebase Storage, în folderul "medic-images". În cadrul acestui folder, se creează un subfolder cu numele corespunzător ID-ului unic al utilizatorului medic, dacă acesta nu există deja. Imaginea este stocată în acest subfolder și, ulterior, se obține link-ul către imagine, care este adăugat într-un obiect de tip "Medic" ce conține datele medicului.

Acest obiect "Medic" cuprinde datele completate în formular, link-ul către imaginea încărcată, precum și informațiile preluate din profilul personal completate anterior în pagina de profil. Odată completat, obiectul este trimis către baza de date Cloud Firestore, unde este stocat în colecția "medics". În cazul în care medicul nu a introdus o fotografie personală, în Storage există o imagine implicită ("default-medical-img") utilizată pentru a reprezenta medicul.

După ce profilul a fost încărcat cu succes, utilizatorul este informat printr-un mesaj de tip snackbar. În plus, butonul "Make your page visible", situat lângă butonul de actualizare, devine activ. Acest buton permite medicului să-și seteze vizibilitatea profilului în pagina de medici. Astfel, medicul poate decide să-și ascundă sau să-și afișeze profilul oricând dorește, printr-o simplă apăsare de buton. După schimbarea cu succes a vizibilității profilului, utilizatorul primește o notificare printr-un snackbar, care conține, pe lângă textul informativ, și un buton „undo”. Dacă utilizatorul apasă acest buton, schimbarea vizibilității profilului este anulată, restabilind starea anterioară.

Acest sistem asigură flexibilitatea și controlul total al medicului asupra vizibilității și actualizării profilului său, integrând atât funcționalitatea de stocare a imaginilor, cât și gestionarea eficientă a datelor prin intermediul Firebase Storage și Cloud Firestore. Designul interfeței este gândit pentru a oferi o experiență de utilizare fluidă și intuitivă, iar actualizările în timp real permit medicilor să vizualizeze și să ajusteze profilul lor înainte de a-l face public, prevenind astfel eventualele erori. Toate aceste măsuri contribuie la crearea unui profil profesional și personalizat, ușor de gestionat și actualizat.

### 3.10 PAGINA DESPRE (ABOUT)

Pagina "Despre" ("About") este concepută ca o pagină de prezentare a clinicii și este împărțită în trei secțiuni distincte, fiecare având un rol specific în comunicarea valorilor și obiectivelor clinicii.

Prima secțiune, denumită "About", este dedicată introducerii generale a clinicii. Aceasta conține un scurt mesaj care evidențiază esența filosofiei clinicii. Acest mesaj este suprapus peste o imagine de fundal care completează vizual textul, conferind paginii un aspect atractiv și profesional. În partea inferioară a acestei secțiuni se află o iconiță reprezentând o săgeată orientată în jos. Apăsând pe această săgeată, utilizatorul este direcționat automat către secțiunea următoare printr-o acțiune de scroll automat, facilitând astfel navigarea. Alternativ, utilizatorul poate accesa următoarea secțiune și printr-un scroll manual.

Cea de-a doua secțiune, intitulată "Our Vision", este structurată într-un chenar distinct și are rolul de a comunica viziunea clinicii. Această secțiune subliniază angajamentul clinicii față de excelența în îngrijirea pacienților și obiectivele pe termen lung ale instituției. Textul este prezentat într-un format clar și concis, fiind încadrat într-un design care asigură lizibilitate și atrage atenția cititorului.

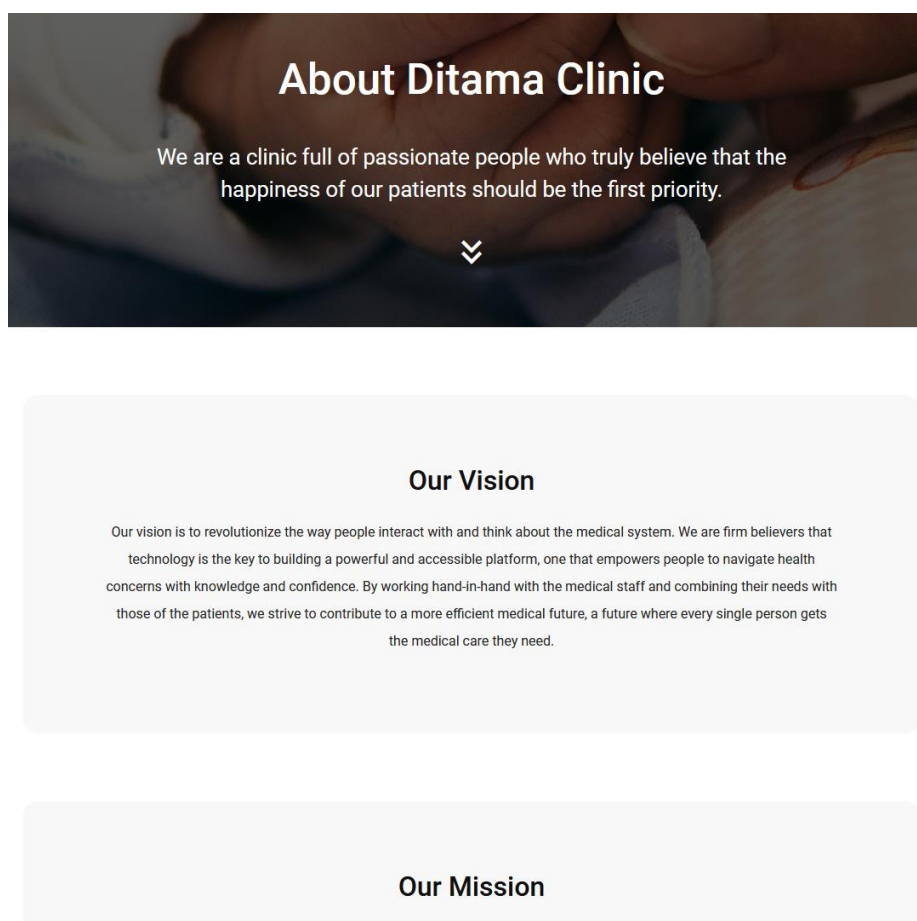


Figure 3-29 Pagina "About"



Ultima secțiune a paginii este dedicată misiunii clinicii și poartă denumirea "Our Mission." Similar secțiunii anterioare, această parte este prezentată într-un chenar distinct, păstrând coerența vizuală a paginii. Secțiunea "Our Mission" explică în detaliu scopurile și principiile fundamentale care ghidează activitatea clinicii.

Întregul design al paginii "Despre" este realizat pentru a oferi o experiență de navigare intuitivă și plăcută, utilizând elemente vizuale și de interfață care reflectă identitatea clinicii. Prin combinarea unui mesaj puternic și clar cu un design estetic, pagina "Despre" reușește să comunice eficient valorile și angajamentele clinicii, invitând utilizatorii să descopere mai multe despre instituție și filosofia sa.

### 3.11 PAGINA DE CONTACT (CONTACT)

Pagina de contact este structurată în două secțiuni distincte, fiecare având un scop specific.

În partea stângă se află un formular reactiv, destinat utilizatorilor care doresc să raporteze probleme sau să trimită mesaje către clinică. Acest formular include câmpurile: prenume (first name), nume de familie (last name), email, număr de telefon (phone number) și mesaj (message). Câmpul pentru mesaj este un textarea care permite utilizatorului să descrie în detaliu problema întâmpinată sau să transmită orice alt mesaj relevant. În plus, formularul conține checkbox-uri care permit utilizatorului să selecteze tipul de problemă sau mesaj pe care dorește să-l transmită, facilitând astfel categorisirea și gestionarea mai eficientă a raportărilor. După completarea formularului, utilizatorul poate trimite informațiile prin intermediul unui buton dedicat. La trimitere, datele sunt stocate în baza de date Firestore, asigurându-se astfel că toate rapoartele și mesajele sunt centralizate și pot fi accesate de personalul clinicii pentru a lua măsurile necesare.

Partea dreaptă a paginii este destinată informațiilor de contact ale clinicii. Această secțiune informativă include detalii esențiale precum adresa de email a clinicii, numărul de telefon și locația fizică. Aceste informații sunt prezentate într-un format accesibil și vizibil, facilitând utilizatorilor găsirea rapidă a datelor de contact necesare pentru a comunica direct cu clinica.

Prin combinarea acestor două secțiuni, pagina de contact oferă atât o modalitate eficientă pentru utilizatori de a raporta probleme și a trimite mesaje, cât și informațiile de contact necesare pentru o comunicare directă și imediată cu clinica. Pagina de contact

joacă un rol crucial în menținerea unei bune comunicări între clinică și utilizatorii săi, contribuind la îmbunătățirea continuă a serviciilor oferite.

### 3.12 PAGINA DE ANALIZĂ A LEZIUNILOR PIELII (SKIN LESION ANALYSER)

Pagina "Skin Lesion Analyser" este dedicată utilizatorilor autentificați și oferă posibilitatea de a încărca și analiza imagini ale leziunilor cutanate folosind un model de clasificare antrenat anterior. Utilizatorii pot introduce o imagine fie printr-un buton de upload, fie printr-o funcționalitate de drag and drop, care a fost implementată printr-o directivă custom. Această directivă facilitează o experiență de utilizare intuitivă și eficientă. Deasupra zonei de drag and drop se află un buton care deschide un dialog cu instrucțiuni detaliate privind utilizarea acestei funcționalități.

După ce utilizatorul încarcă o imagine, aceasta înlocuiește chenarul de drag and drop, iar modelul antrenat procesează imaginea pentru a oferi o analiză detaliată. Sub imaginea încărcată apare un buton care permite utilizatorului să încarce o altă poză dacă dorește. Similar cu secțiunea "Appointment Details" din pagina "Schedule", rezultatele analizei includ procentajele de clasificare pentru fiecare clasă posibilă a leziunii. Primele trei clase, ordonate descrescător după procentaj, sunt afișate utilizatorului.

## Do you have a spot, bump or lesion on your skin?

Upload a photo of it and our cutting edge AI will attempt to tell you what it is!

[How to use this feature ?](#)



 Upload a different file

### Predictions:

Melanocytic nevus (nv): **99.89%** →≡

Benign keratosis-like lesion (bkl): **0.09%** →≡

Melanoma (mel): **0.01%** →≡

#### Not intended for medical purposes

This feature is not intended to diagnose any medical condition and should not be relied on for any medical purposes. It is intended to provide information that can help you manage your well-being. The model that powers this web page is not 100% accurate. It may produce false results with a **high** confidence. This model should only be used for informational purposes. If you have any concerns about your health, or if a skin lesion is bothersome or changes in any way, **PLEASE CONTACT YOUR MEDICAL PROVIDER!**

Figure 3-30 Pagina "Skin Lesion Analyser", după ce utilizatorul a încărcat o imagine

În dreptul fiecărei clase, se află un buton adițional, care deschide un dialog cu informații detaliate despre tipul specific de leziune. Pentru a diferenția severitatea leziunilor, apare un text asociat fiecărei clase care este colorat: galben pentru leziunile mai puțin periculoase, cu recomandarea de a consulta un medic dacă leziunea provoacă disconfort sau suferă modificări, și roșu pentru leziunile mai grave, unde se insistă asupra necesității de a vizita un medic de urgență. Sub acest text colorat, apare o scurtă descriere a acelui tip de leziune.

## Melanoma

### SETTING UP A CONSULT IS HIGHLY RECOMMENDED!

Melanoma is a type of skin cancer that originates in melanocytes, the cells responsible for producing melanin, the pigment that gives skin its color.

Melanoma is a potentially serious form of skin cancer that requires prompt diagnosis and treatment for the best outcome.

Figure 3-31 Componenta dialog cu informații detaliate despre fiecare tip de leziune

Sub chenarul cu rezultatele se află un text de tip disclaimer, care subliniază că scopul paginii este pur informativ, având rolul de a educa utilizatorii și de a promova conștientizarea privind leziunile cutanate. Se menționează explicit că rezultatele oferite de modelul antrenat pot fi eronate și că acestea nu trebuie folosite ca substitut pentru diagnosticul profesional oferit de un medic.

Pagina "Skin Lesion Analyser" oferă o funcționalitate avansată de analiză a leziunilor cutanate și educă utilizatorii asupra importanței evaluării medicale corecte și încurajează consultarea medicilor pentru probleme de sănătate.

### 3.13 PAGINA NEAUTORIZAT (NOT AUTHORIZED)

Când un utilizator încearcă să acceseze o pagină pentru care nu are permisiuni, acesta este redirecționat automat către pagina "Not Authorized". Acest comportament se poate produce fie din cauza faptului că utilizatorul nu este autentificat, fie pentru că, deși autentificat, nu deține rolul necesar pentru a accesa pagina respectivă.

Pagina "Not Authorized" oferă un mesaj clar și specific care informează utilizatorul că nu are permisiuni pentru a accesa pagina dorită și îl încurajează să se conecteze cu un cont valid. În cazul utilizatorilor autentificați, mesajul se adaptează pentru a sugera conectarea cu un cont care are permisiuni elevate.



#### You are not authorized to access this page

It seems like your current account doesn't have permission to access this page.

Please sign in with a different account and try again.

If you believe this is a mistake, please contact your administrator.

[Logout](#)

[Return to Main Page](#)

*Figure 3-32 Pagina "Not Authorized" pentru un utilizator autentificat, dar care nu deține permisiunile necesare pentru a accesa pagina dorită*

Sub textul informativ, pagina afișează un buton care redirecționează utilizatorul către pagina principală ("Home"). Dacă utilizatorul nu este autentificat, acesta este

singurul buton prezent. În schimb, pentru utilizatorii autentificați, există și un buton suplimentar "Logout". Apăsarea acestuia va deloga utilizatorul și îl va redirecționa către pagina principală.

Această funcționalitate este esențială pentru a asigura securitatea și integritatea aplicației, garantând că numai utilizatorii cu drepturi adecvate pot accesa anumite pagini sau funcționalități. Redirecționarea și mesajele informative sunt implementate pentru a oferi o experiență de utilizare clară și eficientă, minimizând confuzia și oferind utilizatorilor direcții clare despre cum să procedeze în continuare. Această funcționalitate este esențială pentru gestionarea accesului și pentru protejarea resurselor aplicației de acces neautorizat.

### 3.14 GESTIONAREA ȘI TRATAREA ERORILOR

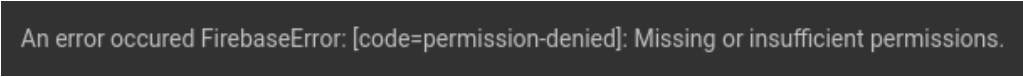
În dezvoltarea de aplicații web moderne, gestionarea erorilor (error handling) reprezintă o parte crucială pentru asigurarea robusteții și fiabilității sistemului. În cadrul platformei Angular, un framework de dezvoltare front-end, gestionarea erorilor este implementată folosind diverse tehnici și practici care facilitează detectarea, raportarea și tratarea erorilor într-un mod eficient.

În Angular, erorile pot apărea în diverse situații, precum erori de rețea în timpul interacțiunii cu API-uri, erori de validare la nivelul formularului, erori de cod în timpul procesului de execuție sau erori legate de configurarea aplicației. O abordare standard în gestionarea acestor erori este utilizarea operatorilor din RxJS pentru a captura și gestiona erorile în mod centralizat.

Am integrat gestionarea erorilor folosind "catchError" pentru a monitoriza și trata eșecurile apelurilor către API-uri. În cazul în care o eroare este detectată, aceasta este capturată în fluxul de date observabil și este tratată în mod corespunzător pentru a asigura o experiență de utilizare fluentă și informativă. Această abordare m-a ajutat să detectez și să gestionez eficient erorile care pot apărea în timpul comunicării cu serverul backend.

În astfel de situații, am transmis mai departe eroarea receptată de operatorul "catchError" pentru a afișa un mesaj de eroare relevant utilizatorului în interfața aplicației, astfel încât să fie conștient de problema apărută și să poată lua măsurile necesare sau să încerce din nou mai târziu.

Prin această metodă de tratare a erorilor, utilizatorul este informat despre erori pentru a avea cunoștință dacă aplicația întâmpină probleme și pentru a evita orice confuzii sau frustrări neintenționate. Această abordare asigură transparența în comunicarea stării aplicației și îi permite utilizatorului să înțeleagă clar natura problemelor care pot să apară în timpul utilizării, contribuind astfel la o experiență de utilizare mai fluidă și mai plăcută.



```
An error occured FirebaseError: [code=permission-denied]: Missing or insufficient permissions.
```

*Figure 3-33 Mesaj de eroare afișat utilizatorului prin intermediul unui snackbar*

### 3.14.1 Securitate și Protejare Backend

Securitatea backendului în cadrul unei aplicații web reprezintă o componentă crucială pentru protejarea datelor și funcționalităților sistemului împotriva accesului neautorizat, manipulării sau a altor amenințări cibernetice.

Un aspect esențial al securității backendului este autentificarea și autorizarea adecvată a utilizatorilor. Autentificarea confirmă identitatea utilizatorilor și le permite accesul la funcționalități în funcție de rolurile și permisiunile lor. Autorizarea, pe de altă parte, controlează ce acțiuni pot fi efectuate de către utilizatori după autentificare. Implementarea unui sistem solid de autentificare și autorizare implică utilizarea unor metode de criptare puternice pentru stocarea datelor de autentificare, precum și definirea și gestionarea corectă a rolurilor și permisiunilor.

În cadrul aplicației dezvoltate, autorizarea utilizatorilor este gestionată prin intermediul rolurilor, care determină accesul și permisiunile fiecărui utilizator în funcție de funcționalitatea sa. Rolurile sunt definite și atribuite în mod specific pentru diferite categorii de utilizatori, cum ar fi medicii și pacienții. Fiecare rol conferă anumite privilegii și restricții, contribuind astfel la crearea unei ierarhii clare a accesului la funcționalități critice ale aplicației.

Firebase gestionează automat sesiunile utilizatorilor autentificați și asigură că autentificarea și autorizarea sunt transparente și securizate. Utilizatorii autentificați primesc un token JWT (JSON Web Token) care este verificat în mod automat la fiecare cerere către Firebase, asigurând astfel că utilizatorii au acces doar la resursele pentru care au permisiuni.

Toate comunicările între aplicațiile client și serviciile Firebase sunt criptate prin HTTPS, ceea ce protejează datele în tranzit împotriva interceptării sau modificării de către atacatori.

Am implementat reguli Firebase pentru a gestiona accesul la baza de date Firestore și la Firebase Storage, restricționând permisiile de citire și scriere numai pentru utilizatorii care necesită această funcționalitate și care au rolurile corespunzătoare. De exemplu, doar utilizatorii care au creat o programare și medicul la care a fost asignată acea programare au permisiunea să acceseze datele respective din baza de date Firestore. Această abordare este esențială pentru securitatea și integritatea datelor, asigurând că doar persoanele autorizate pot accesa și manipula informațiile sensibile stocate în aplicație.

## 4 ANTRENAREA MODELULUI DE INTELIGENȚĂ ARTIFICIALĂ

Antrenarea unui model de inteligență artificială implică parcurgerea unor etape principale esențiale: procurarea datelor, procesarea datelor și antrenarea propriu-zisă a modelului.

### 4.1 PROCURAREA DATELOR

Setul de date HAM10000 [33] (Human Against Machine with 10000 training images) este unul dintre cele mai utilizate și cunoscute seturi de date pentru cercetarea în domeniul dermatologiei și al inteligenței artificiale aplicate în diagnosticarea leziunilor cutanate. Acesta a fost creat pentru a facilita dezvoltarea și evaluarea algoritmilor de învățare automată capabili să detecteze și să clasifice diferite tipuri de leziuni ale pielii.

HAM10000 conține imagini dermatoscopice, fiecare dintre acestea fiind asociată cu una dintre cele șapte categorii de afecțiuni cutanate. Aceste categorii includ diverse tipuri de leziuni benigne și maligne, precum melanomul, nevul melanocitic, keratoza actinică și keratoza seboreică, printre altele. Fiecare imagine este etichetată de către experți dermatologi, asigurând astfel un nivel ridicat de fiabilitate și consistență a datelor.

Am ales să utilizez setul de date HAM10000 datorită numărului său considerabil de imagini și a caracterului său cuprinzător (cele 7 clase acoperă peste 95% din toate leziunile cutanate examinate zilnic de medici dermatologi [34]). De asemenea, în contextul actual, inteligența artificială aplicată în medicină reprezintă un subiect de mare interes, având potențialul de a îmbunătăți semnificativ diagnosticul și tratamentul afecțiunilor cutanate.

### 4.2 PRELUCRAREA DATELOR

Inițial, imaginile din setul de date HAM10000 au fost distribuite în două foldere: "ham10000\_images\_part\_1" și "ham10000\_images\_part\_2". Primul pas în prelucrarea datelor a fost reorganizarea acestora în funcție de tipul de leziune. Am creat un script Python, care parcurge toate imaginile și le sortează în foldere separate, fiecare având denumirea specifică a tipului de leziune, cum ar fi "akiec", "bcc", și așa mai departe. Această reorganizare a facilitat accesul și manipularea ulterioară a imaginilor în procesul de antrenare a modelului de clasificare.

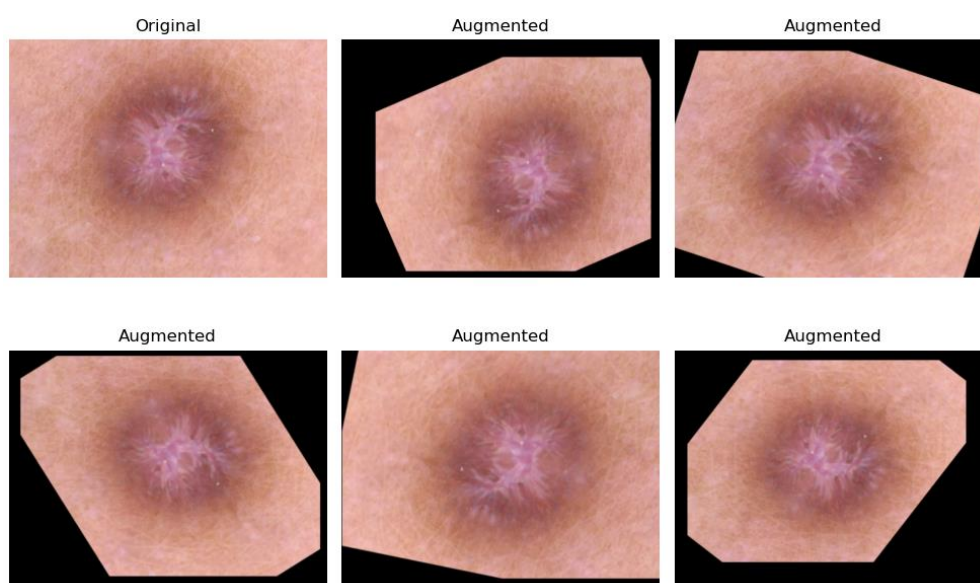


#### 4.2.1 Augmentarea datelor

În stadiul inițial, setul de date HAM10000 era foarte prost balansat din punct de vedere al distribuției claselor. Cea mai mare clasă cuprindea 6705 imagini, în timp ce cea mai mică avea doar 115 imagini. Pentru a rezolva această problemă, am utilizat tehnici de augmentare a datelor, precum over-sampling și under-sampling, implementate prin intermediul unui script Python. Am decis să aduc toate clasele la un număr uniform de 1000 de imagini.

Pentru clasele care aveau deja peste 1000 de imagini, am aplicat under-sampling, alegând aleatoriu 1000 de imagini din totalul existent. Pentru clasele care aveau inițial mai puțin de 1000 de imagini, am recurs la over-sampling. Acest proces a implicat generarea de imagini suplimentare similare cu cele existente, dar nu identice, pentru a mări setul de date.

Pentru a genera aceste imagini noi, am aplicat diverse tehnici de augmentare, cum ar fi rotirea cu unghiuri cuprinse între 0 și 180 de grade, flip vertical și orizontal, translatarea pixelilor pe orizontală sau verticală și scalarea în intervalul 0.9 până la 1.1. Aceste metode au permis crearea unui set de date mai echilibrat, esențial pentru antrenarea corectă și eficientă a modelului de clasificare a leziunilor cutanate. Această augmentare a contribuit semnificativ la îmbunătățirea performanței modelului, asigurând o mai bună reprezentare a tuturor claselor și reducând riscul de overfitting pe clasele mai mari.



*Figure 4-1 Exemplu de imagini augmentate folosind operațiile menționate anterior*

## 4.3 ANTRENAREA MODELELOR

### 4.3.1 Pregătirea datelor pentru antrenare

Pentru a identifica cel mai bun model de clasificare a leziunilor cutanate, am antrenat și comparat performanțele mai multor modele. Pentru a facilita acest proces, am utilizat Google Colab [35], o platformă bazată pe cloud care oferă un mediu de dezvoltare integrat pentru Python. Google Colab este extrem de util pentru proiectele de învățare automată datorită suportului său pentru GPU-uri, compatibilității cu diverse librării Python și accesului la resursele de calcul avansate.

În cadrul Google Colab, am folosit o unitate de procesare grafică (GPU) "T4 GPU", care oferă o putere de calcul semnificativă necesară pentru antrenarea rețelelor neuronale complexe. Utilizarea Google Colab a rezolvat problemele legate de compatibilitatea între librăriile Python, precum și limitările resurselor de calcul disponibile local.

Setul de date prelucrat în pașii anteriori, constând din imaginile echilibrate prin tehnicile de augmentare, a fost încărcat pe Google Drive [36]. Am montat acest Google Drive în Google Colab pentru a facilita accesul la setul de date mare, care nu putea fi încărcat complet în memoria locală a mediului Colab. Această metodă de montare a Google Drive a permis manipularea eficientă a setului de date și a asigurat accesul continuu la date în timpul sesiunilor de antrenare.

Următorul pas în pregătirea setului de date pentru antrenarea modelelor a fost să selectez un batch size adecvat și dimensiunile imaginilor. Am optat pentru un batch size de 32 și dimensiuni ale imaginilor de 224x224x3 (224x224 pixeli, cu 3 canale de culoare RGB). Aceste dimensiuni sunt standard în multe rețele neuronale convoluționale pre-antrenate și permit un echilibru optim între detaliile vizuale păstrate și performanța computațională.

Pentru a gestiona setul de date și a-l împărți în mod corespunzător, am utilizat funcția "tf.keras.utils.image\_dataset\_from\_directory" din biblioteca TensorFlow. Această funcție a facilitat separarea setului de date în date de antrenare și date de validare. Setul de date a fost împărțit inițial în proporție de 80% pentru antrenare și 20% pentru validare. Din setul de date de validare, am alocat ulterior o parte pentru testare, rezultând o

compoziție finală de 80% date pentru antrenare, 13% date pentru validare și 7% date pentru testare.

#### 4.3.2 Modelele antrenate

Pentru a compara performanțele diferitelor modele de clasificare, am antrenat cinci dintre cele mai performante și populare arhitecturi disponibile: VGG19, ResNet50, InceptionV3, EfficientNet și InceptionResNetV2. Aceste modele sunt bine cunoscute pentru performanțele lor superioare în diverse competiții de clasificare a imaginilor și sunt accesibile prin intermediul bibliotecii Keras, ceea ce le face ideale pentru implementare rapidă și eficientă.

#### 4.3.3 Antrenarea modelelor

Pentru fiecare dintre aceste modele, am urmat un proces similar de pregătire și antrenare. În primul rând, am importat funcția "preprocess\_input" specifică fiecărui model din biblioteca Keras și modelul pre-antrenat. Funcția "preprocess\_input" este esențială deoarece adaptează datele de intrare la formatul și scala necesare fiecărui model pre-antrenat.

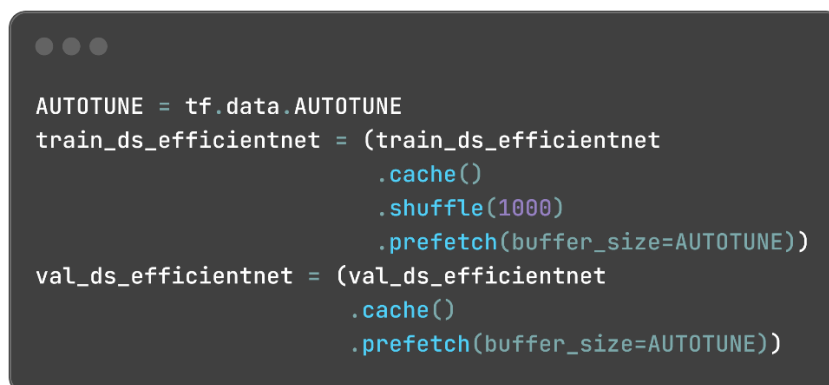
```
train_ds_efficientnet = train_ds.map(
    lambda x, y: (
        efficientnet_preprocess(x),
        tf.one_hot(y, depth=num_classes)
    )
)
val_ds_efficientnet = val_ds.map(
    lambda x, y: (
        efficientnet_preprocess(x),
        tf.one_hot(y, depth=num_classes)
    )
)
test_ds_efficientnet = test_ds.map(
    lambda x, y: (
        efficientnet_preprocess(x),
        tf.one_hot(y, depth=num_classes)
    )
)
```

Figure 4-2 Preprocesarea datelor pentru modelul EfficientNet

Pentru a optimiza performanța setului de date, am utilizat tehnica AUTOTUNE [37], care permite TensorFlow să ajusteze dinamic operațiile de încărcare a datelor în funcție de resursele disponibile și de cerințele specificului hardware. Această abordare maximizează eficiența operațiilor de citire și prelucrare a datelor, reducând timpul de încărcare și accelerând procesul de antrenare a modelului.

În cadrul acestui proces, am aplicat metoda "cache()" pentru a memora seturile de date de antrenament și validare în memorie, ceea ce permite accesul rapid la date în timpul repetării multiple a acestora în timpul antrenării. De asemenea, am utilizat operația "shuffle(1000)" pentru a amesteca datele din setul de date de antrenament, asigurând diversitatea și aleatorizarea exemplelor folosite în fiecare epocă de antrenament.

În plus, am folosit metoda "prefetch(buffer\_size=AUTOTUNE)" pentru a încărca în avans un număr optim de exemple de antrenament în memoria tampon, pregătind datele pentru prelucrare înainte ca modelul să le solicite. Această strategie maximizează utilizarea resurselor hardware disponibile și reduce timpul de așteptare între operațiile de încărcare și prelucrare a datelor.



```
AUTOTUNE = tf.data.AUTOTUNE
train_ds_efficientnet = (train_ds_efficientnet
                        .cache()
                        .shuffle(1000)
                        .prefetch(buffer_size=AUTOTUNE))
val_ds_efficientnet = (val_ds_efficientnet
                     .cache()
                     .prefetch(buffer_size=AUTOTUNE))
```

Figure 4-3 Optimizarea performanței folosind `tf.data.AUTOTUNE`

Următorul pas în procesul de antrenare a implicat definirea modelelor de bază pentru fiecare dintre modelele selectate. Aceste modele au fost configurate pentru a primi imagini de dimensiuni specifice, specificate prin intermediul parametrului "input\_shape". Pentru a crea o arhitectură personalizată, am utilizat parametrul "include\_top = false", care exclude stratul superior al modelului pre-antrenat, permițându-mi să adaug propriul strat de ieșire adaptat problemei abordate.

De asemenea, am specificat "weights=imagenet" pentru a utiliza ponderile pre-antrenate ale modelului pe setul de date ImageNet. Această abordare mi-a oferit avantajul de a beneficia de cunoștințele preexistente ale rețelei neuronale în ceea ce privește extragerea caracteristicilor generale din imagini. Prin urmare, am putut să încep antrenamentul de la un punct mai avansat și să obțin rezultate mai bune cu un efort mai mic de antrenare.

```
base_model = EfficientNetB0(  
    input_shape = (img_height, img_width, 3),  
    include_top = False,  
    weights = 'imagenet',  
)
```

*Figure 4-4 Definirea modelului de bază pentru modelul EfficientNetB0*

Următorul pas în procesul de antrenare a constat în adăugarea unui strat de clasificare personalizat la modelul de bază. Pentru a face acest lucru, am început prin a îngheța straturile din modelul de bază, ceea ce înseamnă că am blocat actualizarea ponderilor acestora în timpul antrenării. Această etapă este esențială pentru a păstra cunoștințele preexistente acumulate de modelele de bază.

Ulterior, am experimentat cu dezghețarea unui procentaj din straturile superioare ale modelelor, pentru a îmbunătății performanța pe setul de date abordat. Această strategie de dezghețare a permis modelelor să învețe și să se adapteze mai bine la specificitățile setului de date, sporind astfel performanța lor în sarcina complexă de clasificare a leziunilor cutanate.

De asemenea, am adăugat un set de straturi personalizate care să fie aplicate pe ieșirea modelului de bază.

```

from tensorflow.keras.regularizers import l2

x = layers.Flatten()(base_model.output)

x = layers.Dense(1024, activation='relu', kernel_regularizer=l2(0.01))(x)

x = layers.Dropout(0.5)(x)

x = layers.Dense(num_classes, activation='softmax')(x)

```

Figure 4-5 Definirea straturilor adăugate modelelor

În primul rând, am folosit stratul "Flatten" pentru a converti ieșirea modelului într-o singură dimensiune, astfel încât să putem conecta rezultatul la un strat dens. Apoi, am adăugat un strat dens complet conectat cu 512 neuroni ascunși și funcția de activare "ReLU". Acest strat ajută la extragerea și învățarea de caracteristici mai abstracte din datele de intrare.

Pentru a evita overfitting-ul, am aplicat un strat de dropout. Dropout-ul este o tehnică de regularizare care temporar dezactivează o parte din neuroni în timpul antrenării, reducând astfel dependențele între aceștia și prevenind overfitting-ul. De asemenea, am aplicat regularizarea L2 pentru a controla overfitting-ul și pentru a îmbunătăți generalizarea modelului.

În final, am adăugat un strat cu 7 neuroni și funcția de activare "softmax" pentru a produce probabilități pentru fiecare clasă de leziune a pielii. Acest strat generează distribuția de probabilitate pentru fiecare clasă, permițând modelului să realizeze clasificarea corectă a imaginilor de piele în clasele corespunzătoare.

Am adăugat setul final de straturi create la modelul de bază și l-am compilat. Pentru funcția loss, am folosit "categorical\_crossentropy", care este o alegere comună pentru problemele de clasificare multi-clasă. Pentru metrice, am inclus "accuracy", "precision", "recall", "f1-score" și "categorical\_crossentropy". Aceste metrice oferă o evaluare cuprinzătoare a performanței modelului din diverse perspective, permițându-mi să analizez și să înțeleg mai bine modul în care modelul clasifică datele.

În cele din urmă, am rulat procesul de antrenare a modelului utilizând metoda "model.fit". Am specificat ca parametri setul de date de antrenare, setul de date de testare, numărul de epoci (100) și metodele de callback.

#### **4.4 UTILIZAREA METODELOR CALLBACK**

În procesul de antrenare al rețelelor neuronale, utilizarea callback-urilor este esențială pentru a gestiona și optimiza antrenarea modelului. Callback-urile sunt instrumente care permit execuția unor funcții specifice la anumite puncte pe parcursul antrenării, cum ar fi la sfârșitul unei epoci sau după procesarea unui lot de date. Ele sunt folosite pentru a îmbunătăți performanța modelului, a preveni overfitting-ul și a optimiza timpul de antrenare.

##### **4.4.1 Scăderea ratei de învățare**

Una din principalele probleme apărute a fost găsirea unei rate de învățare optime. Dacă rata de învățare este prea mare, atunci modelul poate să "sară" peste punctul minim, și să nu ajungă niciodată la acesta. Pe de altă parte, dacă rata de învățare este prea mică, atunci modelul învață foarte încet. Pentru a optimiza procesul de învățare a modelelor am folosit tehnica scăderii ratei de învățare.

Scăderea ratei de învățare este o tehnică esențială în antrenarea rețelelor neuronale, menită să îmbunătățească performanța și stabilitatea modelului [38]. În cadrul procesului de antrenare, inițial, este utilizată o rată de învățare mai mare pentru a permite modelului să facă ajustări semnificative ale ponderilor. Aceste ajustări rapide sunt cruciale pentru a ieși din regiuni de optim local și a converge către o soluție mai bună. Cu toate acestea, pe măsură ce modelul se apropie de o minimizare optimă a funcției de pierdere, este necesară o rată de învățare mai mică pentru a rafina ajustările și a preveni oscilațiile excesive în jurul valorilor optime.

Pentru implementarea scăderii ratei de învățare, am utilizat o funcție de programare a ratei de învățare ("LearningRateScheduler") din cadrul bibliotecii TensorFlow. Am definit o funcție de programare a ratei de învățare care reduce rata de învățare cu un factor de 0.1 la fiecare 30 de epoci. Această strategie permite modelului să beneficieze de o rată de învățare mai mare în primele epoci, unde este nevoie de ajustări semnificative, și să utilizeze o rată de învățare mai mică în epocile ulterioare, unde ajustările fine sunt esențiale pentru convergență.

```

epochs_to_wait = 30

def lr_schedule(epoch, lr):
    if epoch % epochs_to_wait == 0 and epoch > 0:
        return lr * 0.1
    else:
        return lr

lr_scheduler = LearningRateScheduler(lr_schedule)

```

Figure 4-6 Definirea funcției de scădere a ratei de învățare

Prin utilizarea acestui cod, am reușit să controlez dinamica procesului de antrenare și să evit potențialele probleme de overfitting sau stagnare. Reducerea graduală a ratei de învățare ajută la stabilizarea procesului de optimizare și îmbunătățirea performanței finale a modelului. Mai mult, acest tip de programare a ratei de învățare a fost demonstrat că îmbunătățește capacitatea de generalizare a modelului, permițându-i să se adapteze mai bine la noi date nemaîntâlnite.

#### 4.4.2 Early stopping & Model checkpoint

O provocare majoră în antrenarea rețelelor neuronale este determinarea duratei optime de antrenament. Dacă antrenamentul este prea scurt, modelul nu va învăța suficient din setul de date de antrenament și de testare, ducând la underfitting. Dacă antrenamentul este prea lung, modelul va învăța prea bine detaliile setului de date de antrenament, ceea ce va duce la overfitting și la performanțe slabe pe setul de date de testare. Metoda “Early Stopping” [39] este o metodă eficientă de rezolvare a acestei probleme care constă în antrenarea modelului până când performanța pe setul de date de validare începe să scadă.

Pentru a implementa acest algoritm, am utilizat un callback numit “EarlyStopping” oferit de Keras. Acesta monitorizează o metrică specifică, în acest caz, pierderea pe setul de validare (“val\_loss”), și oprește antrenarea dacă performanța modelului nu se îmbunătățește după un anumit număr de epoci (stabilit prin parametru “patience”). Am setat parametrul “patience” la 10, ceea ce înseamnă că antrenarea se va opri dacă



valoarea erorii pe setul de date de validare ("val\_loss") nu înregistrează nicio îmbunătățire timp de 10 epoci consecutive. Mai mult, prin "restore\_best\_weights=True", modelul revine la ponderile cele mai bune obținute pe parcursul antrenării, asigurând astfel utilizarea celei mai eficiente versiuni.

Un alt callback important pe care l-am utilizat este "ModelCheckpoint". Acesta permite salvarea automată a modelului în timpul antrenării. Am configurat "ModelCheckpoint" să salveze doar cele mai bune versiuni ale modelului, pe baza celei mai mici valori a erorii de validare ("val\_loss"). Astfel, chiar dacă modelul continuă să se antreneze și să experimenteze fluctuații în performanță, se va salva întotdeauna cea mai performantă variantă a modelului, care generalizează cel mai bine pe date noi, nemaivăzute.

```
callbacks = [  
    ModelCheckpoint('best_model.h5', save_best_only=True, monitor='val_loss', mode='min'),  
    EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True),  
    lr_scheduler  
]
```

Figure 4-7 Definirea metodelor callback

Aceste callback-uri combinate au permis o gestionare eficientă a procesului de antrenare, prevenind probleme de overfitting și asigurând că modelul antrenat este cel mai performant posibil. Ele contribuie la o antrenare mai robustă și mai eficientă, economisind timp și resurse computaționale prin oprirea timpurie a antrenării atunci când nu mai sunt așteptate îmbunătățiri semnificative.

## 4.5 AFIȘAREA REZULTATELOR

După finalizarea antrenării modelului, am extras informații esențiale care să mă ajute în observarea și gestionarea erorilor din antrenare și compararea modelelor între ele.

În primul rând, am generat și afișat grafice relevante pentru fiecare model, reprezentând metricile utilizate în funcție de epoci. Aceste grafice au inclus eroarea, acuratețea, precizia, rechemarea și scorul F1 al modelului în fiecare epocă, atât pentru setul de antrenare cât și pentru cel de validare. Astfel, am putut identifica eventualele probleme de overfitting sau underfitting.

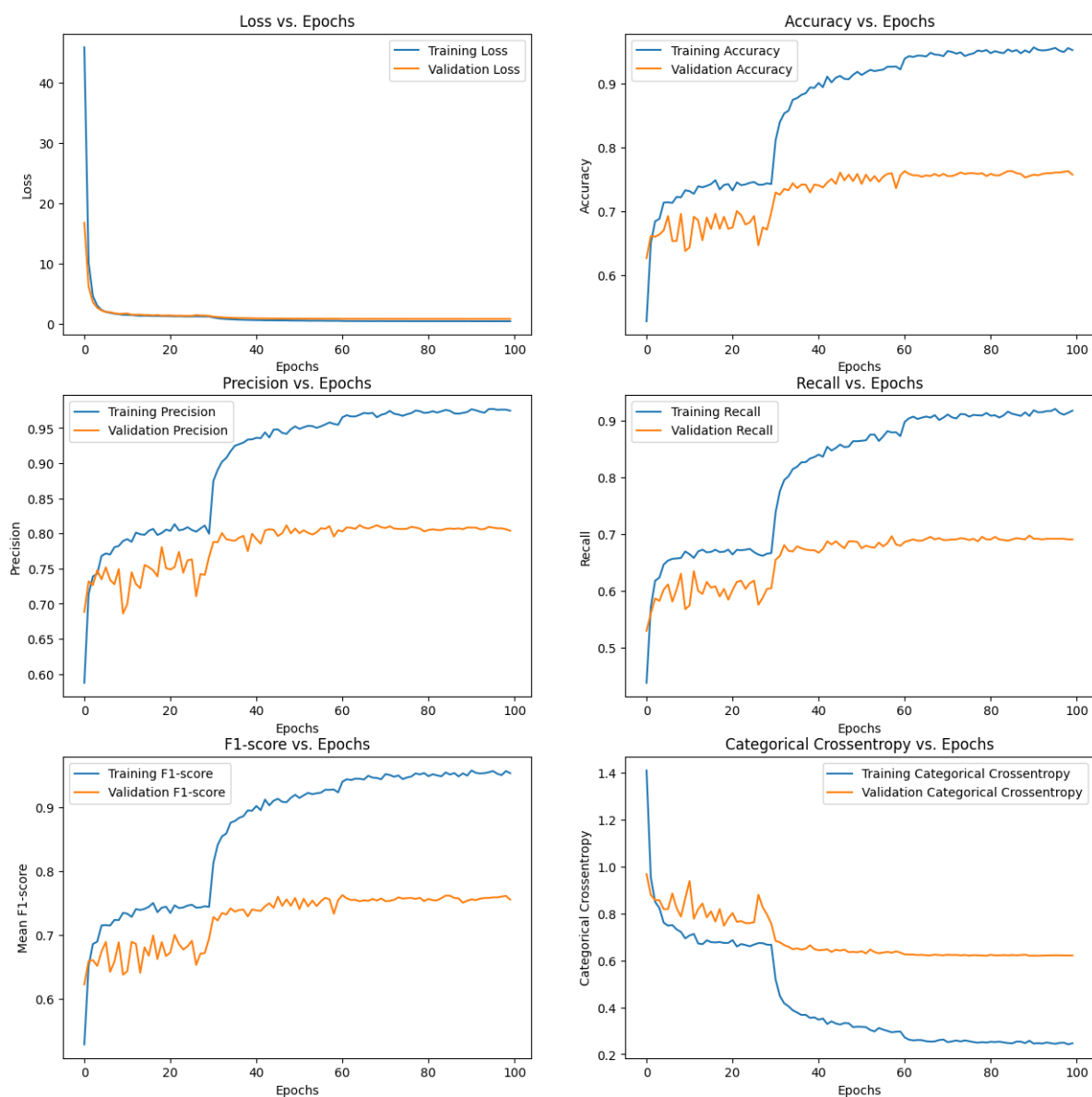


Figure 4-8 Exemplu de grafice folosite pentru a evalua modelele antrenate

În al doilea rând, am evaluat fiecare model folosind setul de date de testare creat anterior, pentru a observa abilitatea modelului de a generaliza pe date nemaîntâlnite. Am afișat valorile numerice ale fiecărei metrice la finalul antrenării pentru seturile de date de antrenare, validare și testare.

```
16/16 [=====] - 7s 177ms/step
Training Loss: 0.4388
Training Accuracy: 0.9527
Training Precision: 0.9748
Training Recall: 0.9173
Training Mean F1-score: 0.9529
Training Categorical Crossentropy: 0.2471
-----
Validation Loss: 0.8131
Validation Accuracy: 0.7578
Validation Precision: 0.8039
Validation Recall: 0.6908
Validation Mean F1-score: 0.7550
Validation Categorical Crossentropy: 0.6214
-----
Test Loss: 0.8255
Test Accuracy: 0.7421
Test Precision: 0.7821
Test Recall: 0.6766
Test F1 Score: 0.7381
Test Categorical Crossentropy: 0.6333
```

Figure 4-9 Exemple de valori pe care le-am afișat după antrenarea unui model

În cele din urmă, am creat și afișat matricea de confuzie pentru fiecare model. Matricea de confuzie este un instrument esențial în evaluarea performanței unui model de clasificare, deoarece oferă o imagine detaliată a predicțiilor corecte și incorecte. Fiecare element din matrice reprezintă numărul de instanțe în care o clasă specifică a fost prezisă corect sau incorect.

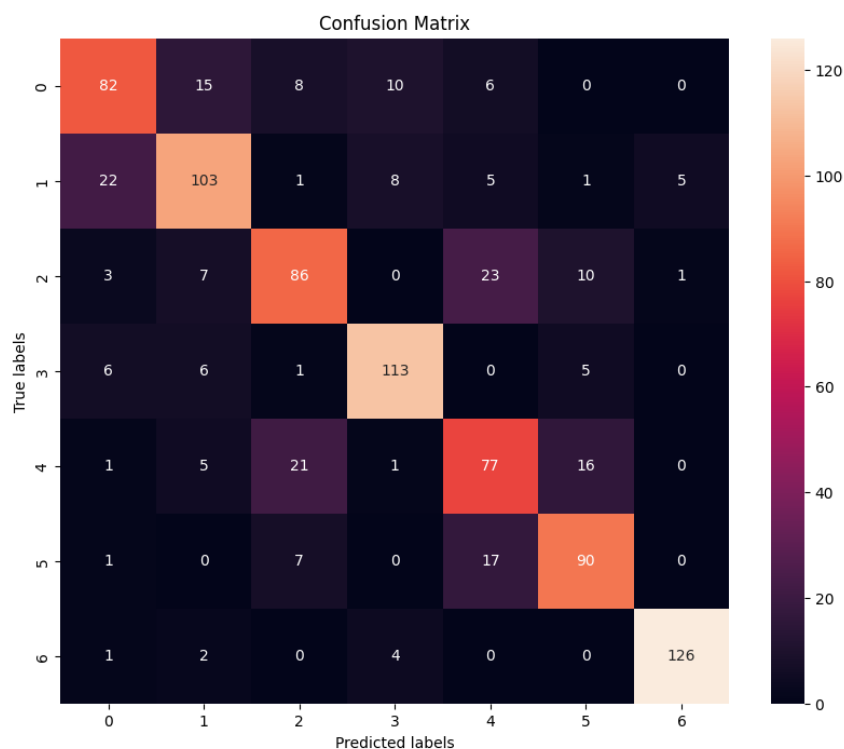


Figure 4-10 Exemplu de matrice de confuzie pentru un model antrenat

Prin analizarea matricei de confuzie, am putut identifica dacă modelul este la fel de performant pe toate clasele sau dacă există anumite clase unde are dificultăți. Această analiză detaliată a contribuțiilor fiecărei clase în performanța generală a modelului mi-a permis să fac ajustări și îmbunătățiri specifice pentru a optimiza rezultatele.

#### 4.6 REZULTATE OBTINUTE ȘI COMPARAȚII ÎNTRE MODELE

Pentru a compara modelele între ele, am analizat valorile metricilor obținute de fiecare model pe seturile de date de antrenare, validare și testare.

Model	Loss			Accuracy		
	train	val	test	train	val	test
VGG19	0.1166	1.2139	1.1152	0.9984	0.7288	0.7341
ResNet50	0.0057	0.5421	0.6874	1	0.8638	0.8591
Inceptionv3	0.3472	1.0602	1.0613	0.9811	0.7009	0.6806
InceptionResNetV2	0.32	0.8977	0.9615	0.9818	0.7511	0.7163
EfficientNet	0.2625	0.8227	0.8361	0.9916	0.7701	0.756

Figure 4-11 Valorile de eroare și acuratețe a modelelor antrenate

În contextul antrenării modelelor de învățare automată, două dintre cele mai esențiale metrici pentru evaluarea performanței sunt eroarea (loss) și acuratețea (accuracy). Acestea oferă informații valoroase despre modul în care modelul se comportă pe întregul set de date și sunt fundamentale pentru antrenarea și ajustarea modelelor. Otuși, acuratețea nu este întotdeauna suficientă pentru a evalua performanța unui model, mai ales în cazul clasificării pe mai multe clase. În astfel de cazuri, metricile de evaluare suplimentare, precum precizia (precision), rechemarea (recall) și scorul F1 (f1-score), devin esențiale.

Model	Precision			Recall			F1-Score		
	train	val	test	train	val	test	train	val	test
VGG19	0.9984	0.741	0.7428	0.9984	0.7154	0.7163	0.9984	0.7261	0.7297
ResNet50	1	0.8741	0.8689	1	0.8527	0.8413	1	0.8604	0.8552
Inceptionv3	0.9874	0.7286	0.7081	0.9684	0.6562	0.621	0.9812	0.6962	0.6751
InceptionResNetV2	0.9874	0.7785	0.7434	0.9679	0.7176	0.6726	0.9819	0.749	0.7131
EfficientNet	0.9948	0.7883	0.7834	0.9879	0.74	0.7321	0.9916	0.7667	0.7505

Figure 4-12 Valorile de precizie, rechemare și scor F1 a modelelor antrenate

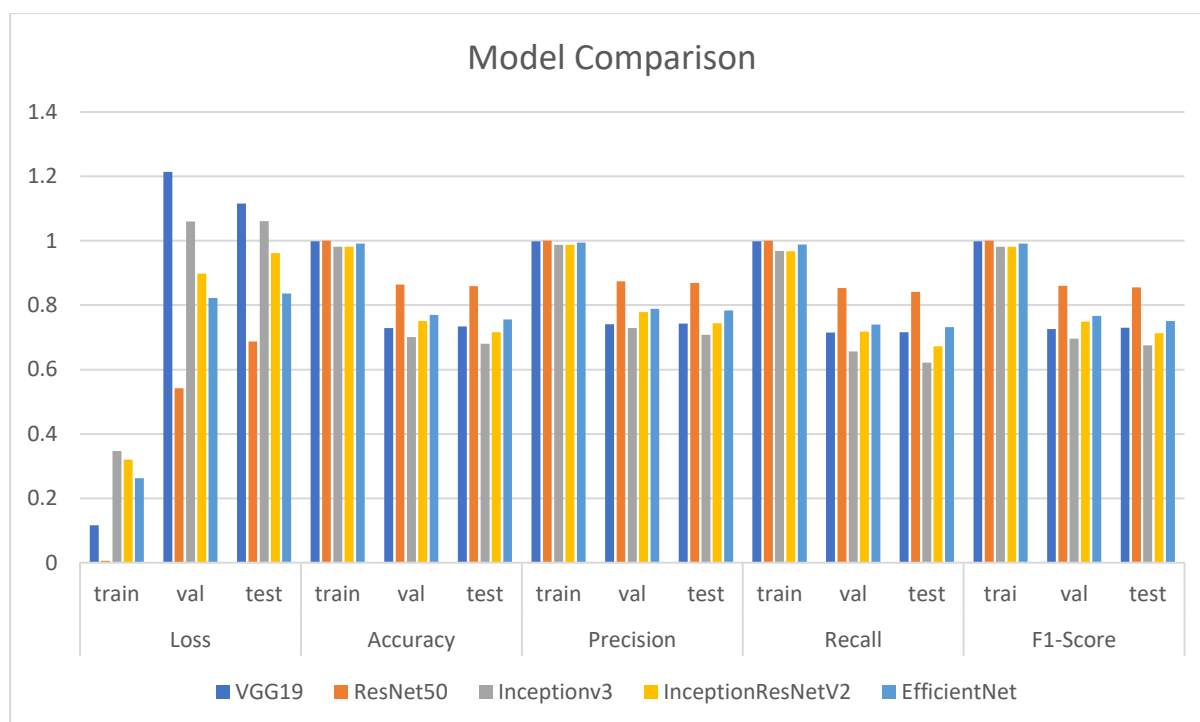


Figure 4-13 Valorile metricilor modelelor antrenate

Din analiza rezultatelor obținute, am putut trage câteva concluzii semnificative. În primul rând, performanța modelelor pe seturile de date de validare și de testare este satisfăcătoare și consistentă. Nu am observat o discrepanță mare între rezultatele obținute pe aceste două seturi de date, ceea ce indică o antrenare corectă a modelelor și sugerează că acestea nu au suferit de overfitting pe setul de date de validare.

Am identificat modelul ResNet50 ca fiind cel mai performant în cadrul studiului meu. Acesta a demonstrat o superioritate evidentă față de celelalte modele în toate metricile importante evaluate pe seturile de date de validare și de testare. Această performanță superioară poate fi atribuită înțelegerii sale profunde a complexității datelor și problemelor abordate în setul de date HAM10000.

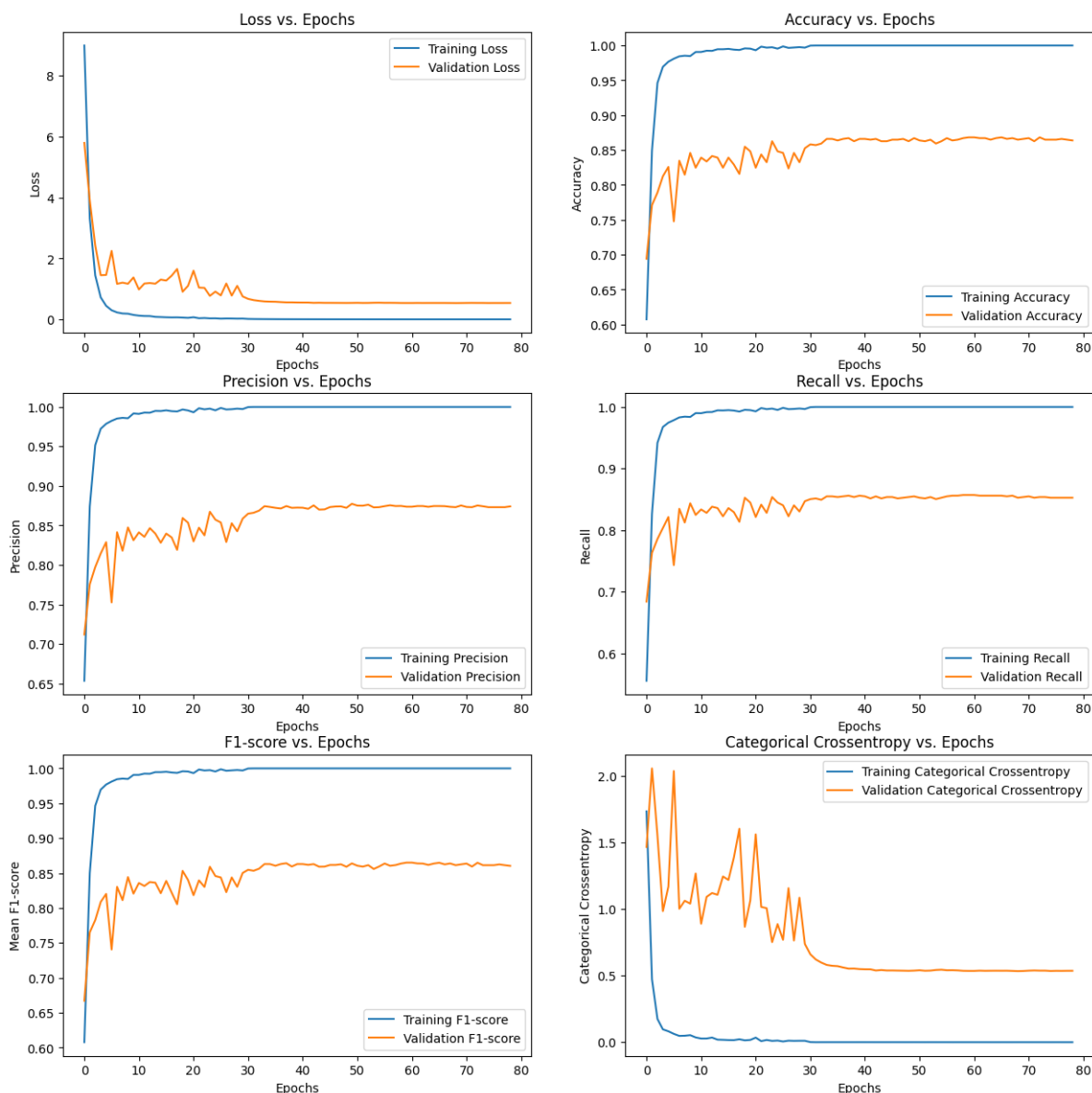


Figure 4-14 Graficele de antrenare a modelului ResNet50

Modelul ResNet50 a obținut cel mai mare avantaj din dezghețarea ultimelor straturi ale rețelei. Cele mai bune rezultate au fost obținute prin dezghețarea a 40% din straturile superioare ale modelului pentru a optimiza performanța acestuia în comparație cu antrenarea în condițiile în care toate straturile erau blocate.

ResNet50 a obținut rezultate consistente și competitive pe toate metricile, sugerând că este bine adaptat pentru cerințele specifice ale clasificării de leziuni cutanate în cadrul domeniului medical. În urma evaluării, am constatat că acest model a fost cel mai satisfăcător și, prin urmare, l-am integrat și implementat în aplicația web dezvoltată.

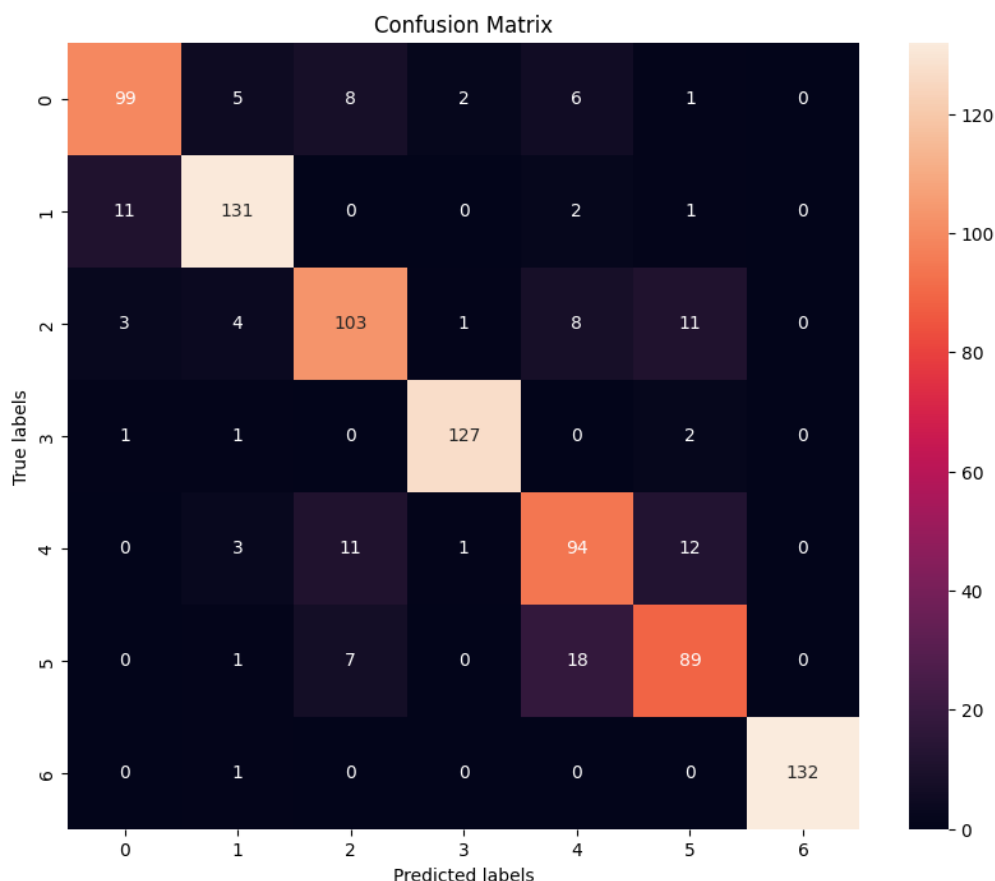


Figure 4-15 Matricea de confuzie a modelului ResNet50

Din analiza detaliată a matricei de confuzie, am observat că modelul prezintă o performanță bună în clasificarea tuturor claselor. Nu au fost identificate probleme majore în ceea ce privește clasificarea specifică a anumitor clase. Acest lucru sugerează că modelul este echilibrat și capabil să distingă eficient între diversele tipuri de leziuni cutanate, reflectând o capacitate generală de învățare și adaptare adecvată pe setul de date.

În concluzie, rezultatele obținute indică că modelele au fost antrenate în mod adecvat, fără a prezenta overfitting sever. Cu toate acestea, pentru a reduce și mai mult overfitting-ul, ar putea fi benefică explorarea unor tehnici suplimentare pentru regularizare sau optimizare a modelelor în viitoarele iterații ale proiectului.

## 5 CONCLUZII

### 5.1 CONCLUZII GENERALE

Relația cu sistemul medical poate fi una dificilă pentru mulți pacienți. De multe ori, aceștia se confruntă cu dificultăți în interacțiunea cu serviciile medicale, deoarece poate dura mult timp de la apariția simptomelor până la primirea ajutorului medical de specialitate. Aceste întârzieri și complicații administrative pot conduce la frustrare și pot agrava starea de sănătate a pacienților. De asemenea, mulți pacienți ajung foarte greu la medic sau chiar renunță să mai meargă la medic din cauza problemelor cu sistemul.

În acest context, am decis să elaborez acest proiect pentru crearea unei platforme informatice medicale moderne, care să fie performantă și ușor de utilizat, integrând tehnologii avansate, cum ar fi inteligența artificială. Această platformă își propune să gestioneze programările pacienților și să le furnizeze informații relevante despre starea lor de sănătate. Obiectivul principal al aplicației este de a facilita programarea rapidă și eficientă la medic pentru pacienți și de a oferi medicilor un mod simplu, funcțional și estetic de gestionare a consultațiilor.

Proiectul constă într-o aplicație web construită folosind Angular, având Firebase ca backend. Integrarea modelelor de inteligență artificială, antrenate separat, în aceeași aplicație permite utilizatorilor să acceseze toate funcționalitățile dintr-un singur loc. Astfel, am reușit să creez o platformă unificată și eficientă, care îmbină tehnologii avansate și oferă o experiență completă utilizatorilor.

Un alt scop principal al aplicației este de a informa publicul larg în legătură cu leziunile pielii, folosind inteligența artificială pentru a oferi diagnostice prezumtive, cu scop informativ, ale acestor leziuni. Am ales să utilizez setul de date HAM10000 datorită numărului mare de imagini pe care le conține, a cuprinderii sale extinse și a relevanței sale pentru domeniul aplicației web medicale, având în vedere interesul crescut pentru aplicarea inteligenței artificiale în medicină.

În procesul de dezvoltare, am urmat pași riguroși de prelucrare și echilibrare a datelor, antrenând și comparând performanța mai multor modele de rețele neuronale convoluționale, inclusiv VGG19, ResNet50, InceptionV3, EfficientNet și InceptionResNetV2. Dintre acestea, modelul ResNet50 s-a dovedit a fi cel mai



performant, fiind capabil să depășească celelalte modele în toate metricile importante pe seturile de validare și testare. Utilizarea tehnicilor de augmentare a datelor și a strategiilor de regularizare a contribuit la îmbunătățirea generalizării modelului.

În concluzie, platforma dezvoltată reprezintă o potențială soluție pentru modernizarea și eficientizarea sistemului medical, facilitând accesul pacienților la servicii medicale și oferind medicilor un instrument puternic de gestionare a consultațiilor. Integrarea inteligenței artificiale pentru diagnosticarea prezumtivă a leziunilor pielii aduce un plus de valoare, contribuind la informarea și educarea publicului larg. Astfel, proiectul nu doar îmbunătățește experiența utilizatorilor, dar și demonstrează potențialul tehnologiilor avansate în transformarea domeniului medical.

## **5.2 POTENȚIALE ÎMBUNĂTĂȚIRI ȘI DIRECȚII VIITOARE DE CERCETARE ȘI DEZVOLTARE**

### **5.2.1 Potențiale Îmbunătățiri pentru Aplicația Web**

O îmbunătățire potențială ar fi adăugarea mai multor pagini informative care să ofere utilizatorilor acces la resurse educaționale relevante despre sănătate și prevenirea bolilor. Aceste pagini ar putea include ghiduri de sănătate, articole medicale actualizate și răspunsuri la întrebările frecvente ale pacienților.

Funcționalitatea aplicației poate fi extinsă prin permiterea utilizatorilor, atât pacienți cât și medici, de a edita sau șterge programările după ce acestea au fost create. Această funcționalitate ar aduce un plus de flexibilitate și ar îmbunătăți gestionarea programărilor, adaptându-se la schimbările neprevăzute care pot apărea.

De asemenea, integrarea unei funcționalități prin care medicii să poată accepta sau refuza anumite programări ar contribui la o mai bună organizare a timpului și a resurselor medicale. Aceasta ar permite medicilor să își gestioneze mai eficient programul și să prioritizeze cazurile urgente sau mai complexe.

Implementarea unui sistem de notificări și alerte ar putea, de asemenea, îmbunătăți semnificativ experiența utilizatorilor. Notificările pentru programări viitoare, reamintiri pentru consultații și alerte personalizate bazate pe istoricul medical al pacientului ar aduce un plus de valoare aplicației.

### 5.2.2 Potențiale Îmbunătățiri pentru Componenta de Inteligență Artificială

Pentru a îmbunătăți performanțele modelelor de inteligență artificială, există mai multe metode care pot fi explorate. Una dintre aceste metode este ajustarea numărului de straturi înghețate în timpul antrenării. Testarea cu diverse configurații ale straturilor înghețate ar putea dezvălui combinații mai eficiente care să conducă la o acuratețe mai mare a modelelor.

O altă direcție de cercetare ar fi testarea unor noi arhitecturi de modele, diferite de cele utilizate până în prezent. De exemplu, se pot investiga modele mai noi sau mai complexe, care ar putea oferi performanțe superioare în clasificarea leziunilor pielii.

Optimizarea ratei de învățare este, de asemenea, crucială pentru îmbunătățirea performanțelor modelului. Scăderea ratei de învățare on plateau, adică ajustarea ratei de învățare în funcție de stagnarea performanței modelului pe setul de date de validare, poate conduce la o convergență mai eficientă și la o generalizare mai bună a modelului.

## BIBLIOGRAFIE

- [1] S. H. Edward, C. J. James și C. F. Michael, *Biomedical Informatics: Computer Applications in Health Care and Biomedicine*, Springer, 2021.
- [2] P. Zhao, I. Yoo, J. Lavoie, B. J. Lavoie și E. Simoes, „Web-Based Medical Appointment Systems: A Systematic Review,” *Journal of Medical Internet Research*, vol. 19, nr. 4, 2017.
- [3] Angular, „What is Angular? - Angular,” [Interactiv]. Available: <https://angular.dev/overview>.
- [4] Angular, „Anatomy of components - Angular,” [Interactiv]. Available: <https://angular.dev/guide/components>.
- [5] Angular, „Creating an injectable service - Angular,” [Interactiv]. Available: <https://angular.dev/guide/di/creating-injectable-service>.
- [6] Angular, „Directives - Overview - Angular,” [Interactiv]. Available: <https://angular.dev/guide/directives>.
- [7] Angular, „Pipes - Overview - Angular,” [Interactiv]. Available: <https://angular.dev/guide/pipes>.
- [8] B. K. Ragala, „Advantages and Disadvantages of Angular,” 22 September 2023. [Interactiv]. Available: <https://www.knowledgehut.com/blog/web-development/advantages-and-disadvantages-of-angular>.
- [9] RxJS, „RxJS,” [Interactiv]. Available: <https://rxjs.dev/>.
- [10] TypeScript, „TypeScript: JavaScript With Syntax For Types,” [Interactiv]. Available: <https://www.typescriptlang.org/>.
- [11] G. A. M. T. M. Bierman, „Understanding TypeScript,” în *ECOOP 2014–Object-Oriented Programming: 28th European Conference*, Uppsala, Springer Berlin Heidelberg, 2014, pp. 257–281.
- [12] Angular, „Angular Material UI component library,” Angular, [Interactiv]. Available:

<https://v14.material.angular.io/>.

- [13] Google, „Material Design,” Google, [Interactiv]. Available: <https://m3.material.io/>.
- [14] Bootstrap, „Bootstrap - The most popular HTML, CSS, and JS library in the world.,” Bootstrap, [Interactiv]. Available: <https://getbootstrap.com/>.
- [15] Firebase, „Firebase | Google's Mobile and Web App Development Platform,” [Interactiv]. Available: <https://firebase.google.com/>.
- [16] Firebase, „Firebase Authentication,” [Interactiv]. Available: <https://firebase.google.com/docs/auth/>.
- [17] Firebase, „Cloud Storage for Firebase,” [Interactiv]. Available: <https://firebase.google.com/docs/storage/>.
- [18] Firebase, „Firestore | Firebase,” [Interactiv]. Available: <https://firebase.google.com/docs/firestore/>.
- [19] TensorFlow, [Interactiv]. Available: <https://www.tensorflow.org>.
- [20] „Keras: Deep Learning for humans,” [Interactiv]. Available: <https://keras.io>.
- [21] K. Simonyan și A. Zisserman, „Very Deep Convolutional Networks For Large-Scale Image Recognition,” *arXiv*, 10 April 2015.
- [22] K. He, X. Zhang, S. Ren și J. Sun, „Deep Residual Learning for Image Recognition,” *arXiv*, 10 December 2015.
- [23] C. Szegedy, V. Vanhoucke, S. Ioffe și J. Shlens, „Rethinking the Inception Architecture for Computer Vision,” *arXiv*, 11 December 2015.
- [24] C. Szegedy, S. Ioffe, V. Vanhoucke și A. Alemi, „Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning,” *arXiv*, 23 August 2016.
- [25] M. Tan și Q. V. Le, „EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks,” *arXiv*, 11 September 2020.
- [26] Firebase, „Firebase console,” [Interactiv]. Available:

<https://console.firebase.google.com/>.

- [27] JetBrains, „PyCharm: the Python IDE for data science and web development,” [Interactiv]. Available: <https://www.jetbrains.com/pycharm/>.
- [28] Angular, „Angular - Angular Routing,” Angular, [Interactiv]. Available: <https://angular.io/guide/routing-overview>.
- [29] Angular, „Angular - Structural Directives,” Angular, [Interactiv]. Available: <https://angular.io/guide/structural-directives>.
- [30] Firebase, „Firebase Auth REST API,” [Interactiv]. Available: <https://firebase.google.com/docs/reference/rest/auth>.
- [31] Firebase, „Timestamp | Firebase,” [Interactiv]. Available: <https://firebase.google.com/docs/reference/android/com/google/firebase/Timestamp>.
- [32] Simplemaps, „Romania Cities Database | Simplemaps.com,” [Interactiv]. Available: <https://simplemaps.com/data/ro-cities>.
- [33] Harvard, „The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions - ViDIR Dataverse,” [Interactiv]. Available: <https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/DBW86T>.
- [34] P. Tschandl, C. Rosendahl și H. Kittler, „The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions,” *Scientific Data*, vol. V, nr. 1, p. 9, 2018.
- [35] Google, „colab.google,” [Interactiv]. Available: <https://colab.google/>.
- [36] Google, „Personal Cloud Storage & File Sharing Platform - Google,” [Interactiv]. Available: <https://www.google.com/drive/>.
- [37] Tensorflow, „Better performance with the tf.data API | TensorFlow Core,” [Interactiv]. Available: [https://www.tensorflow.org/guide/data\\_performance](https://www.tensorflow.org/guide/data_performance).
- [38] K. You, M. Long, J. Wang și M. I. Jordan, „How Does Learning Rate Decay Help

Modern Neural Networks,” 26 September 2019.

- [39] J. Brownlee, „A Gentle Introduction to Early Stopping to Avoid Overtraining Neural Networks,” 6 August 2019. [Interactiv]. Available: <https://machinelearningmastery.com/early-stopping-to-avoid-overtraining-neural-network-models/>.
- [40] „Learn Angular,” December 2023. [Interactiv]. Available: <https://angular.dev/tutorials/learn-angular>.
- [41] U. o. S. Florida, „What is Medical Informatics?,” 28 February 2023. [Interactiv]. Available: <https://www.usfhealthonline.com/resources/health-informatics/what-is-medical-informatics/>.
- [42] Angular, „Angular - Component Lifecycle,” Angular, [Interactiv]. Available: <https://angular.io/guide/lifecycle-hooks>.

**DECLARAȚIE PRIVIND ORIGINALITATEA  
LUCRĂRII DE LICENȚĂ / PROIECTULUI DE DIPLOMĂ / DISERTAȚIEI**

UNIVERSITATEA TRANSILVANIA DIN BRAȘOV

FACULTATEA INGINERIE ELECTRICĂ ȘI ȘTIINȚA CALCULATOARELOR

PROGRAMUL DE STUDII

TEHNOLOGIA INFORMATIEI

NUMELE ȘI

PRENUMELE POPESCU STEFAN-TUDOR

PROMOȚIA 2024

SESIUNEA Iunie - Iulie 2024

TEMA LUCRĂRII / PROIECTULUI / DISERTAȚIEI

APLICAȚIE INFORMATICĂ MEDICALĂ DEDICATĂ PENTRU

GESTIONAREA ȘI INFORMAREA PACIENȚILOR

CONDUCĂTOR ȘTIINȚIFIC

ȘEF LUCR. DR. ING. POPA LUMINIȚA

Declar pe propria răspundere că lucrarea de față este rezultatul muncii proprii, pe baza cercetărilor proprii și pe baza informațiilor obținute din surse care au fost citate și indicate conform normelor etice, în textul lucrării/proiectului, în note și în bibliografie.

Declar că nu s-a folosit în mod tacit sau ilegal munca altora și că nici o parte din teză/proiect nu încalcă drepturile de proprietate intelectuală ale altcuiva, persoană fizică sau juridică.

Declar că lucrarea/ proiectul nu a mai fost prezentat(ă) sub această formă vreunei instituții de învățământ superior în vederea obținerii unui grad sau titlu științific ori didactic.

În cazul constatării ulterioare a unor declarații false, voi suporta rigorile legii.

Data: 12.06.2024

Absolvent

POPESCU STEFAN-TUDOR

(nume, prenume, semnătură)