

**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

Cuprins

Cuprins

[Cuprins 3](#_Toc167912831)

[1 Introducere 5](#_Toc167912832)

[1.1 Motivația alegerii proiectului 5](#_Toc167912833)

[1.2 Scopul și obiectivele proiectului 6](#_Toc167912834)

[1.3 Domeniul de aplicabilitate 6](#_Toc167912835)

[1.4 Stadiul actual al problemei abordate 7](#_Toc167912836)

[2 Arhitectura și organizarea aplicației 8](#_Toc167912837)

[2.1 Arhitectura generală a aplicației 8](#_Toc167912838)

[2.2 Tehnologii folosite 8](#_Toc167912839)

[2.2.1 Angular 8](#_Toc167912840)

[2.2.2 RxJS 10](#_Toc167912841)

[2.2.3 Typescript 10](#_Toc167912842)

[2.2.4 Angular Material 11](#_Toc167912843)

[2.2.5 Bootstrap 11](#_Toc167912844)

[2.2.6 Firebase 12](#_Toc167912845)

[2.2.7 TensorFlow & Keras 13](#_Toc167912846)

[2.3 Organizarea Proiectului 14](#_Toc167912847)

[2.3.1 Frontend 14](#_Toc167912848)

[2.3.2 Backend 16](#_Toc167912849)

[2.3.3 Inteligență artificială 18](#_Toc167912850)

[3 Implementarea Aplicației Web 19](#_Toc167912851)

[3.1 Elemente comune 19](#_Toc167912852)

[3.2 Angular Routing 19](#_Toc167912853)

[3.3 Pagina Principală (Home) 21](#_Toc167912854)

[3.4 Pagina de Medici (Medics) 23](#_Toc167912855)

[3.5 Pagina de Autentificare 26](#_Toc167912856)

[3.6 Pagina de Profil 30](#_Toc167912857)

[3.7 Pagina de Programări (Appointments) 32](#_Toc167912858)

[3.7.1 Secțiunea “Create an appointment” 32](#_Toc167912859)

[3.7.2 Secțiunea “View your appointments” 37](#_Toc167912860)

[3.8 Pagina de Program (Schedule) 39](#_Toc167912861)

[3.8.1 Secțiunea “Schedule Table” 39](#_Toc167912862)

[3.8.2 Secțiunea “Schedule Details” 42](#_Toc167912863)

[Bibliografie 44](#_Toc167912864)

# Introducere

## Motivația alegerii proiectului

Informatica medicală, precum și informatica sănătății sunt discipline care îmbină domeniul medical cu cel informatic, într-un efort de a îmbunătății ajutorul medical acordat pacienților. Aceste discipline se bazează pe expertiza ambelor discipline pentru a folosi tehnologia pentru grija pacienților, diagnosticare si 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 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 consultaților în sistemul național de bază medical. Informarea mai bună a pacienților, precum și îmbunătățirea modului prin care sunt gestionați 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-medic 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.

## 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.

Un alt obiectiv important este sprijinirea comunicării medic-pacient prin crearea unei platforme care să ofere o comunicare eficientă și securizată între membrii echipei medicale si pacienți, facilitând schimbul rapid de informații relevante.

Al patrulea 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.

## 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.

## 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 digitală.

# Arhitectura și organizarea aplicației

## 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.

## Tehnologii folosite

### Angular

Angular 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 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 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 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.

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”*. [3]

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.

### RxJS

RxJS (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.

### Typescript

TypeScript 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.” [4]. 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 [4].

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 aplicaților care devin mai mari și mai complexe.

### Angular Material

Angular Material [5] 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 [6] 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.

### Bootstrap

Bootstrap [7] 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.

### Firebase

Firebase [8] 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 aplicaților 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 aplicației.

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** [9]**:** 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:** 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.*

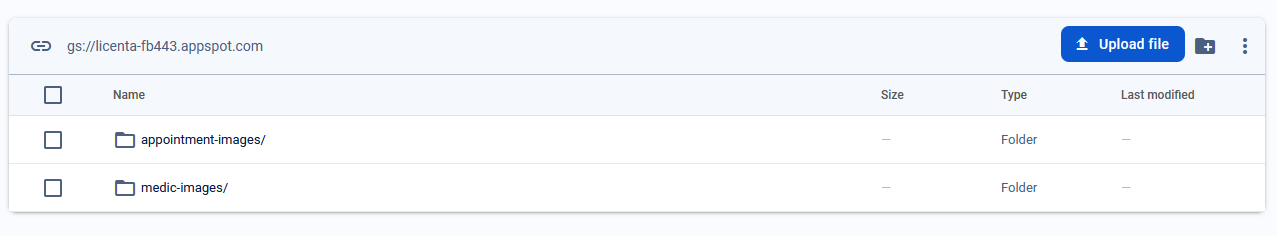


Figure 2‑1 Structura folderelor serviciului de stocare

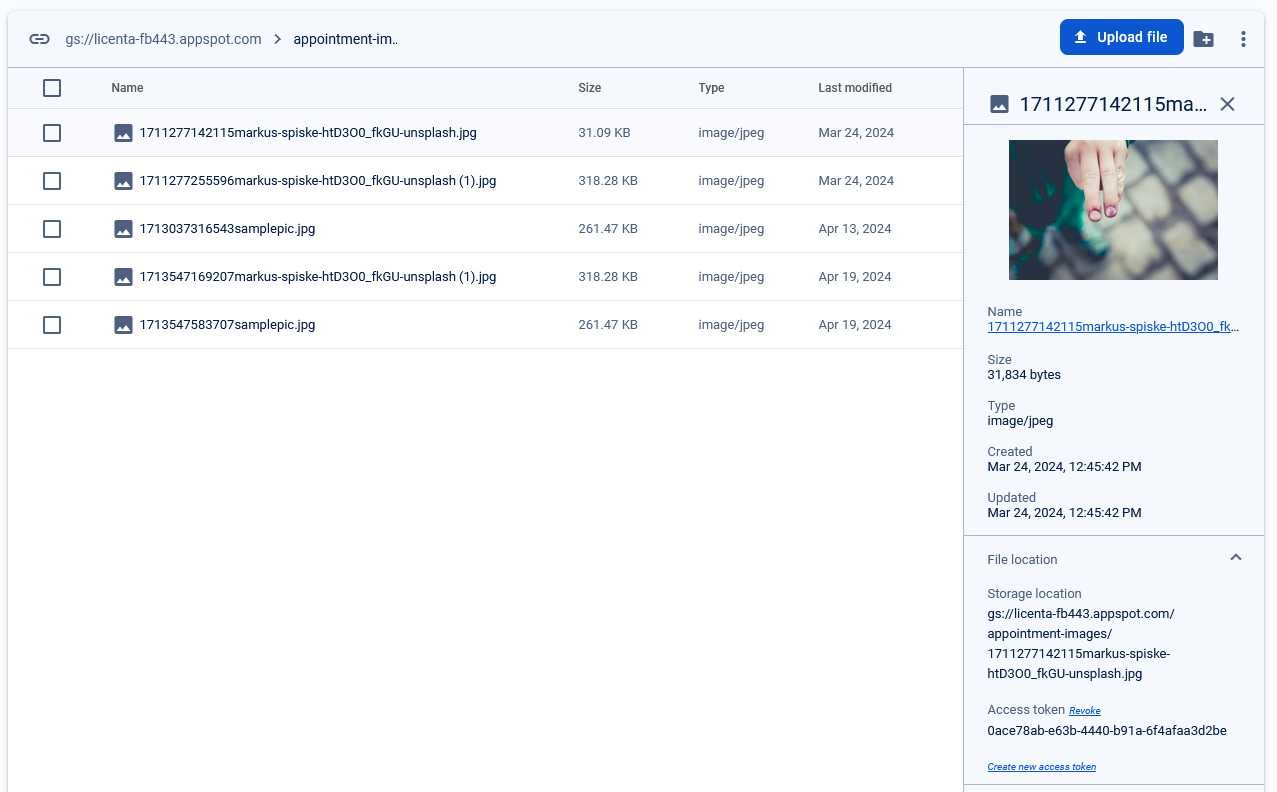


Figure 2‑2 Imaginile trimise de pacienți la crearea unei programări

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:** 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.

### TensorFlow & Keras

TensorFlow [10] 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 [11] 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.

## Organizarea Proiectului

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

### 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.

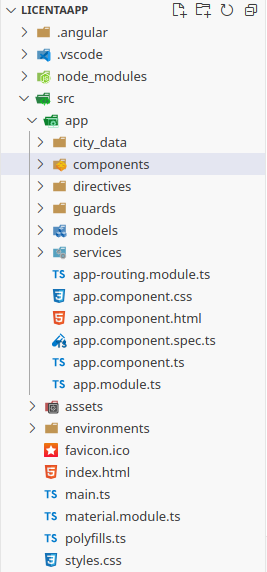


Figure 2‑3 Structura folderelor aplicației Angular

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 care protejează accesul la acestea de către utilizatorii neautorizați.

### Backend

Partea de backend este constituită din platforma Firebase. Gestionarea resurselor folosite poate fi făcută prin accesarea site-ului [console.firebase.google.com](https://console.firebase.google.com/) 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.

### 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. 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.

# Implementarea Aplicației Web

## 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” [5], 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.

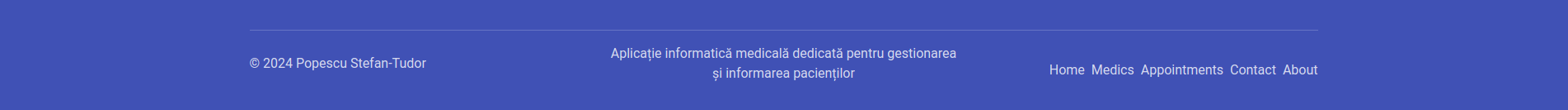


Figure 3‑2 Subsolul (Footer) aplicației web

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

## Angular Routing

Angular Routing [12] 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 între diferite vizualizări (Views) sau componente fără a reîncărca întreaga pagină, oferind o experiență de utilizare fluidă și interactivă.

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 parametrii dinamici, rute protejate prin gărzi (guards) și date suplimentare.



Figure 3‑3 Definirea rutelor folosind Angular Routing

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 introducere 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.

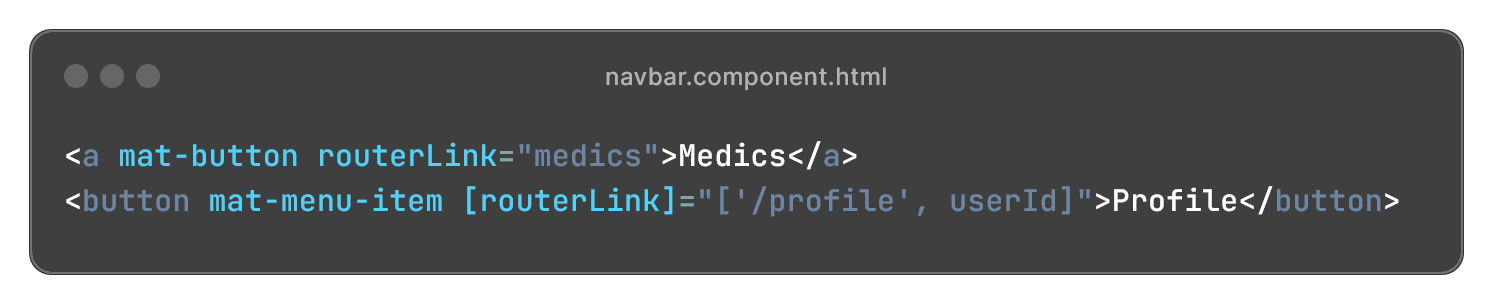


Figure 3‑4 Exemplu de utilizare a directivei RouterLink și a parametrilor de rută

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.

## 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 “Carusel” și o secțiune cu butoane “Call to Action”.

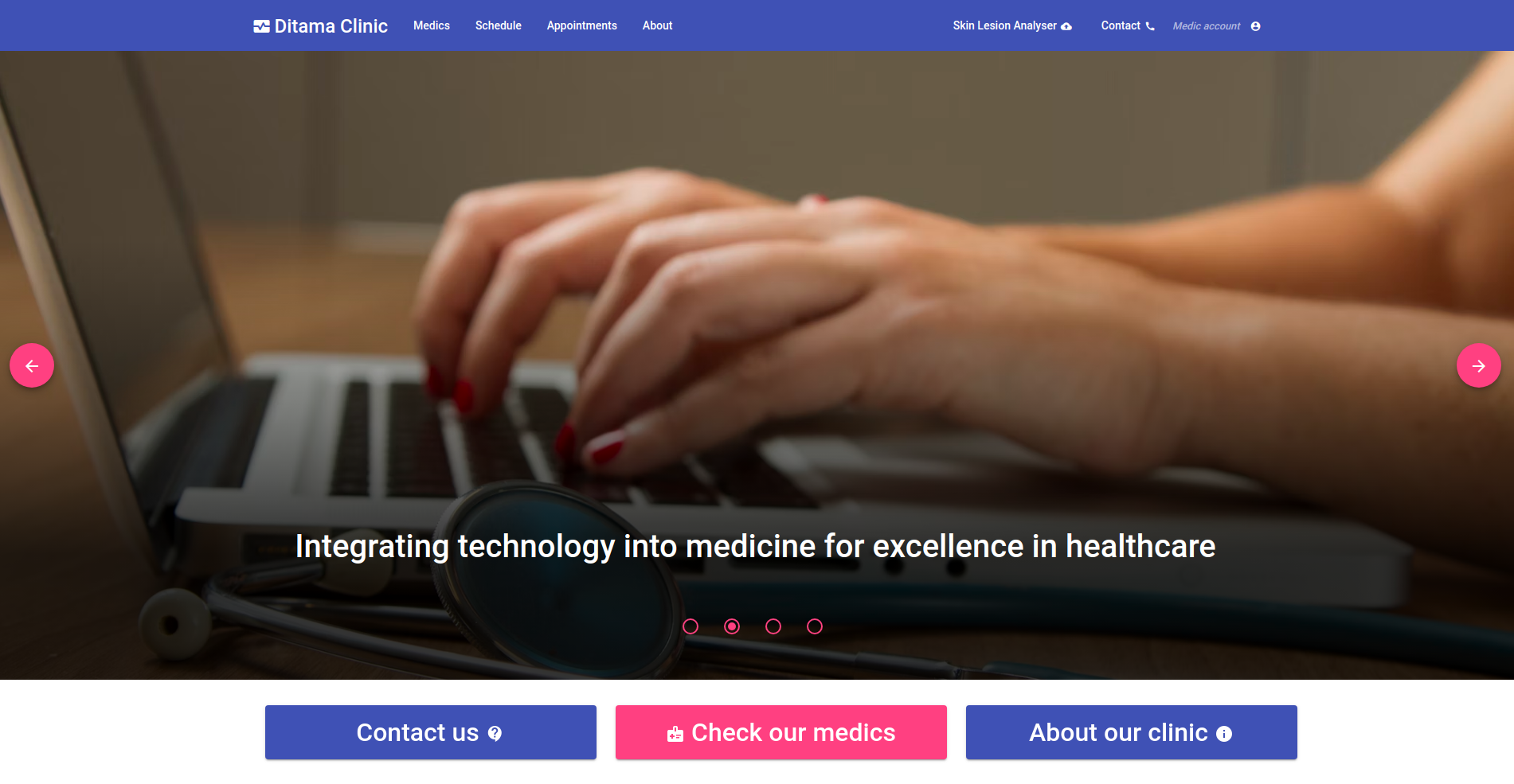


Figure 3‑5 Pagina principală (Home)

Secțiunea “Carusel” 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.

## 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 [5]. 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.

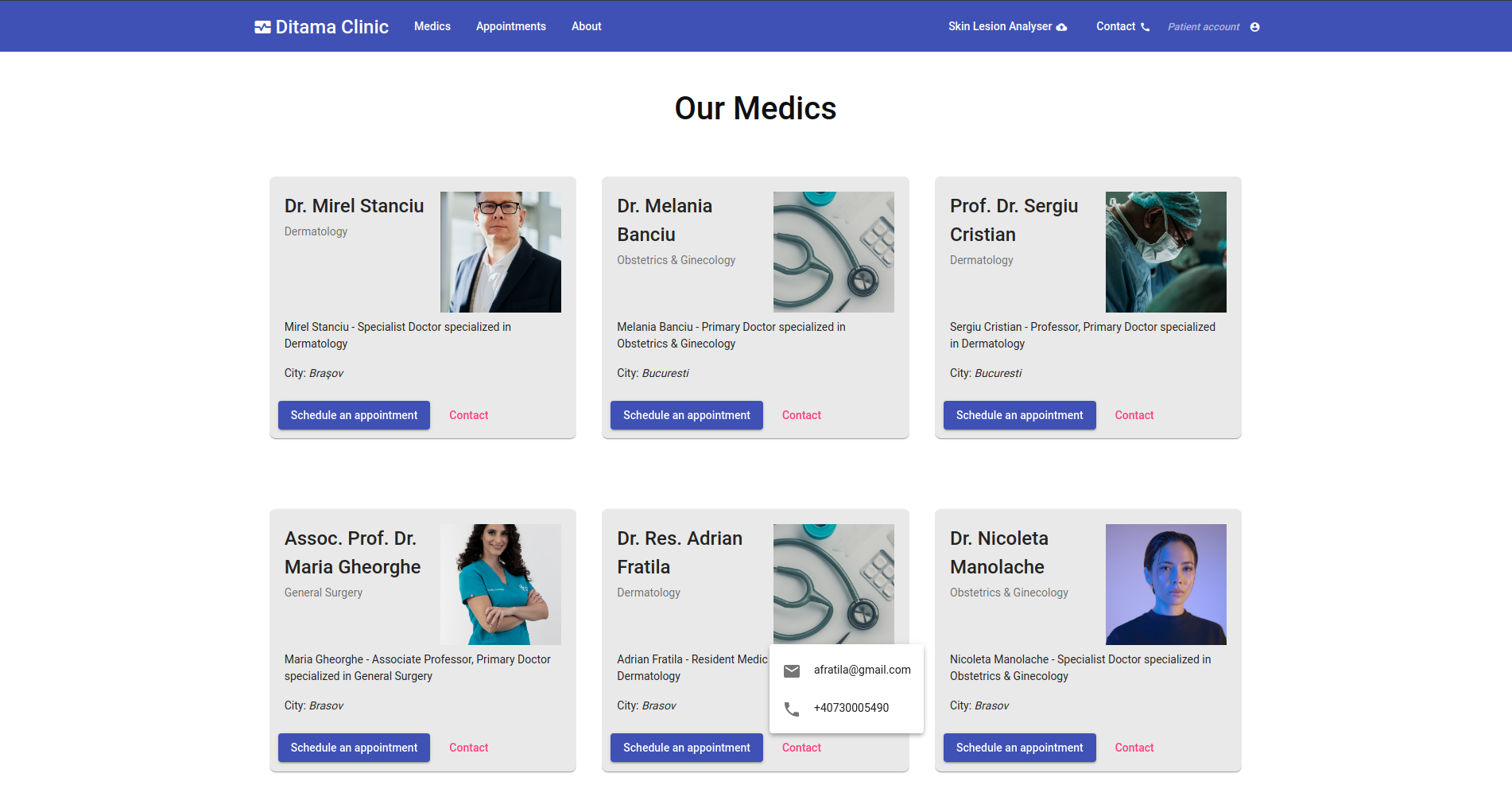


Figure 3‑6 Pagina de Medici

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 [5]), 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.

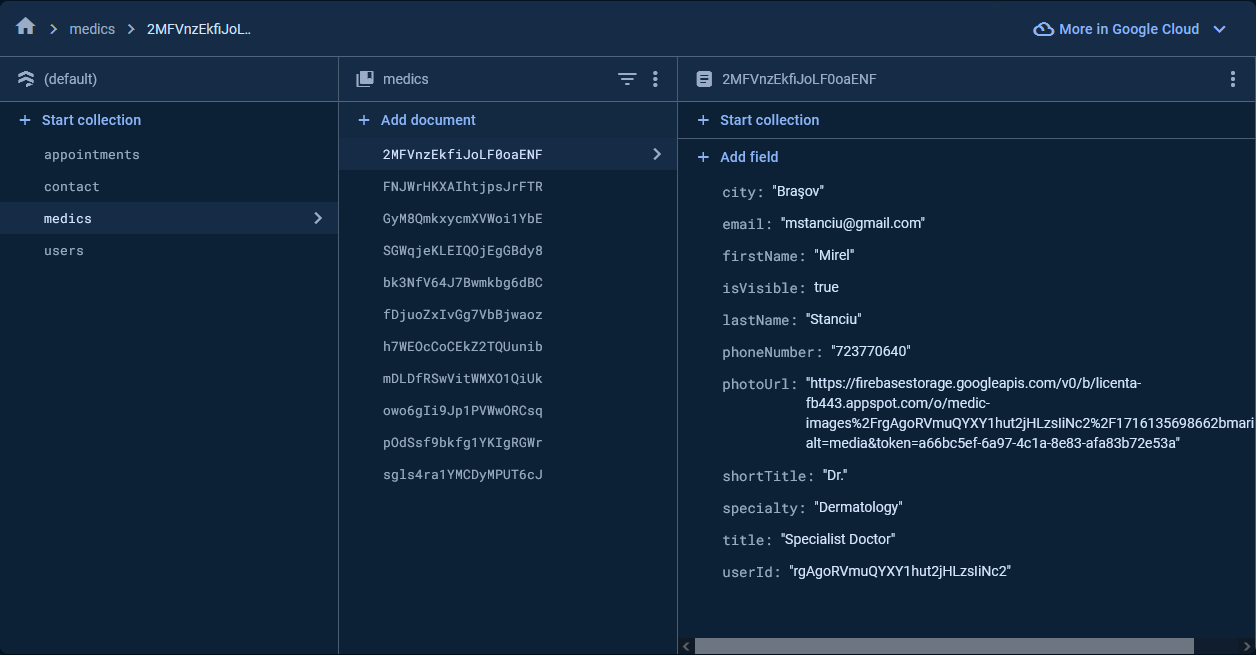


Figure 3‑7 Datele aferente unui medic stocate în baza de date Firestore

Pentru afișarea condiționată a componentelor se utilizează directiva “ngIf” [13]. 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” [13], 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. Această abordare simplifică codul vizual, facilitează reutilizarea acestuia și reduce cantitatea de cod necesară.



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.

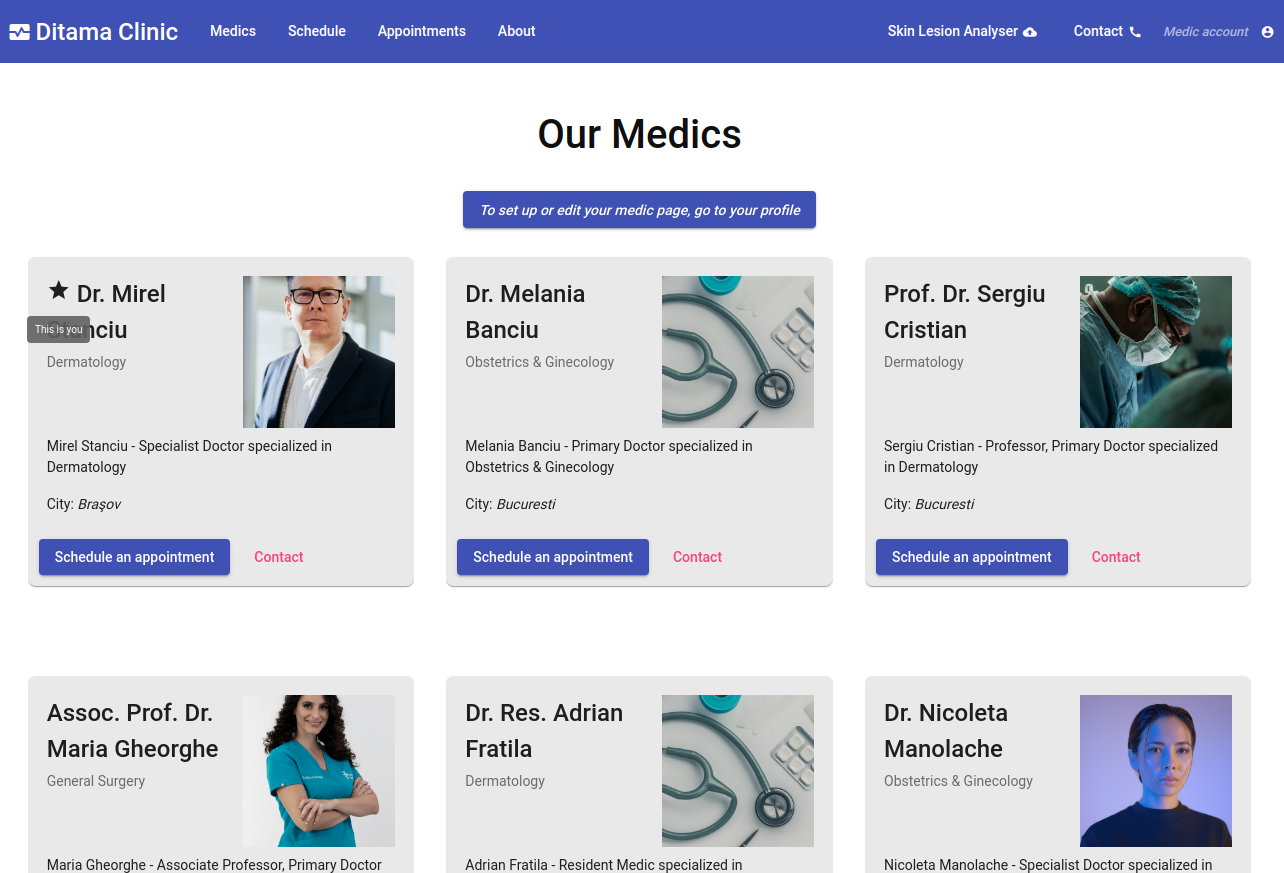


Figure 3‑9 Aspectul paginii de medici pentru un utilizator cu rolul de medic

## 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.

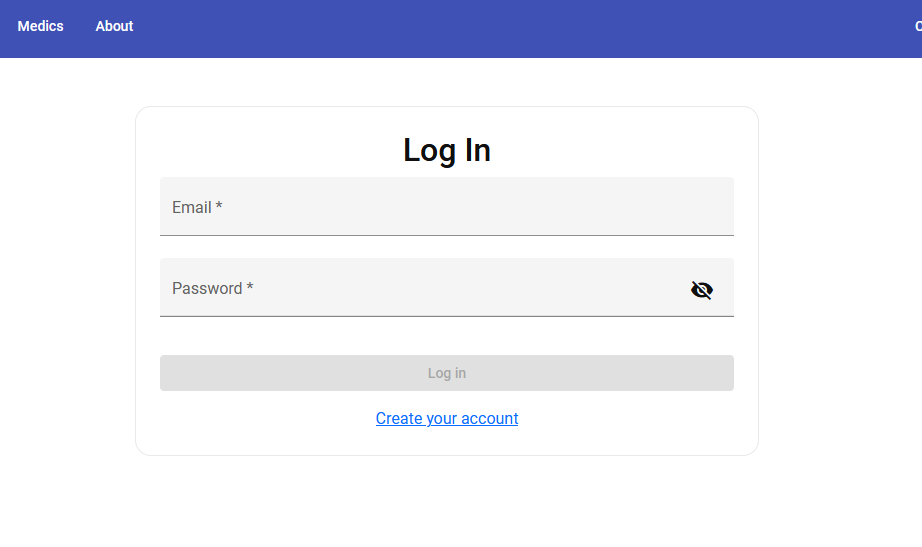


Figure 3‑10 Interfața paginii de autentificare

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 formularelor 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.



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 oferiți 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.

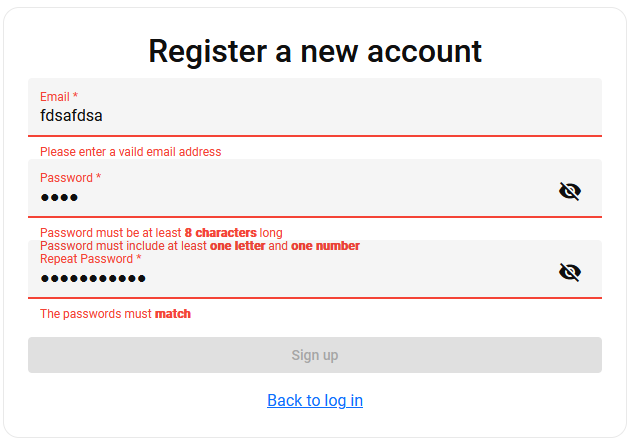


Figure 3‑12 Erori afișate utilizatorului în cazul introducerii de valori invalide

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.

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 [9]. 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” [14] și “Sign in with email / password” [14] 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).

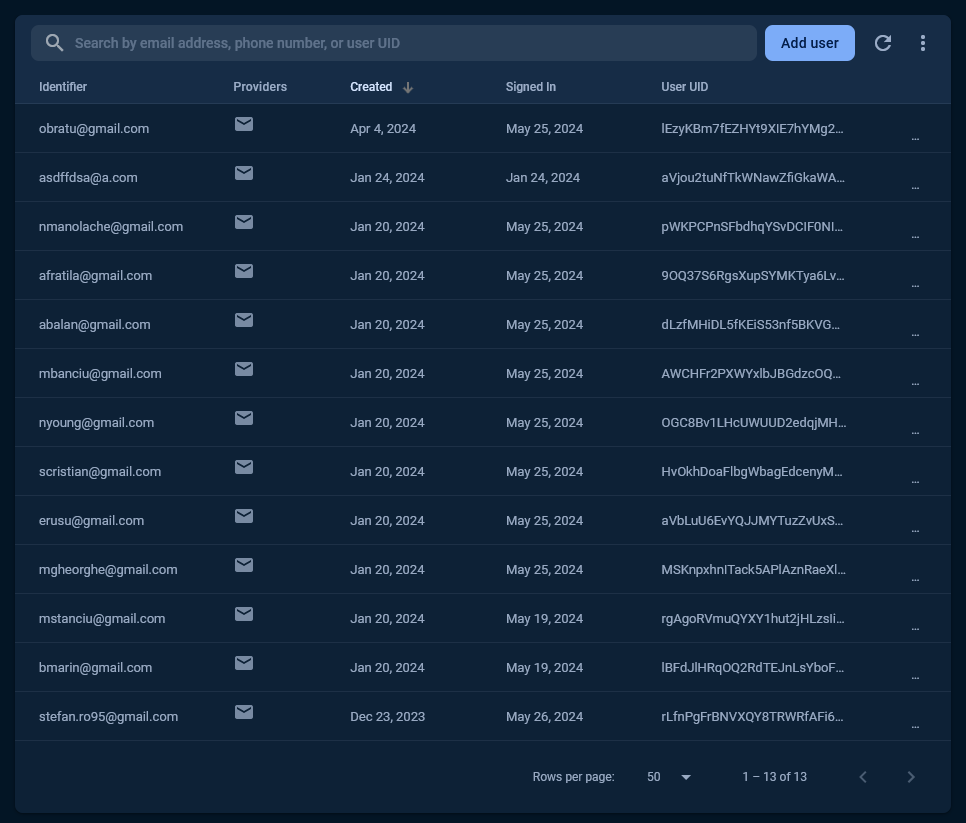


Figure 3‑13 Interfața de vizionare și gestionare a utilizatorilor autentificați din Firebase

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.

## Pagina de Profil

Pagina de profil poate fi accesata odata ce utilizatorul se autentifica in aplicatie. 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.

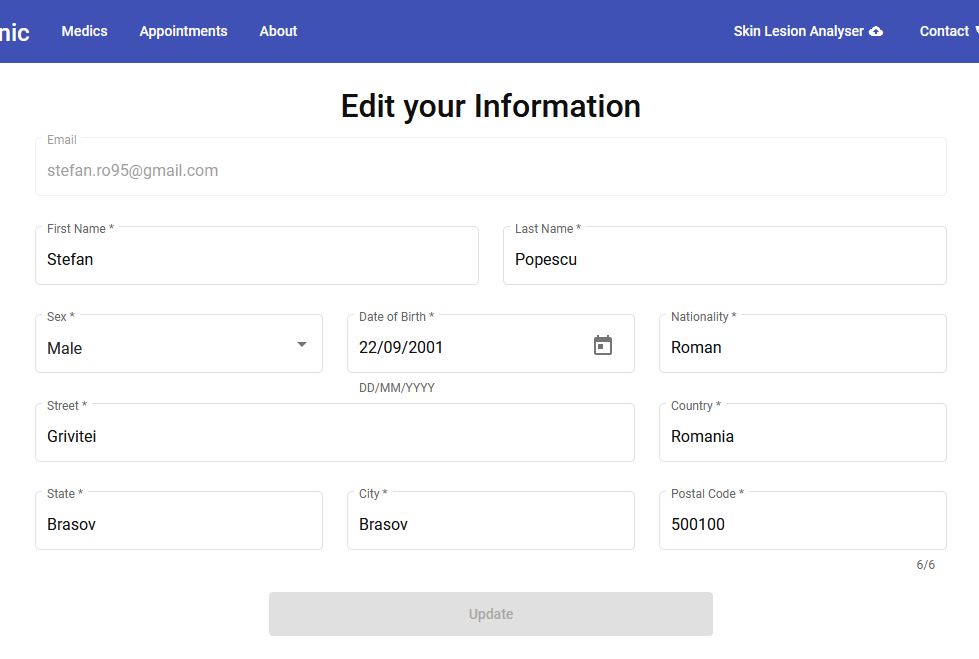


Figure 3‑14 Pagina de Profil

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

## 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.

### Secțiunea “Create an appointment”

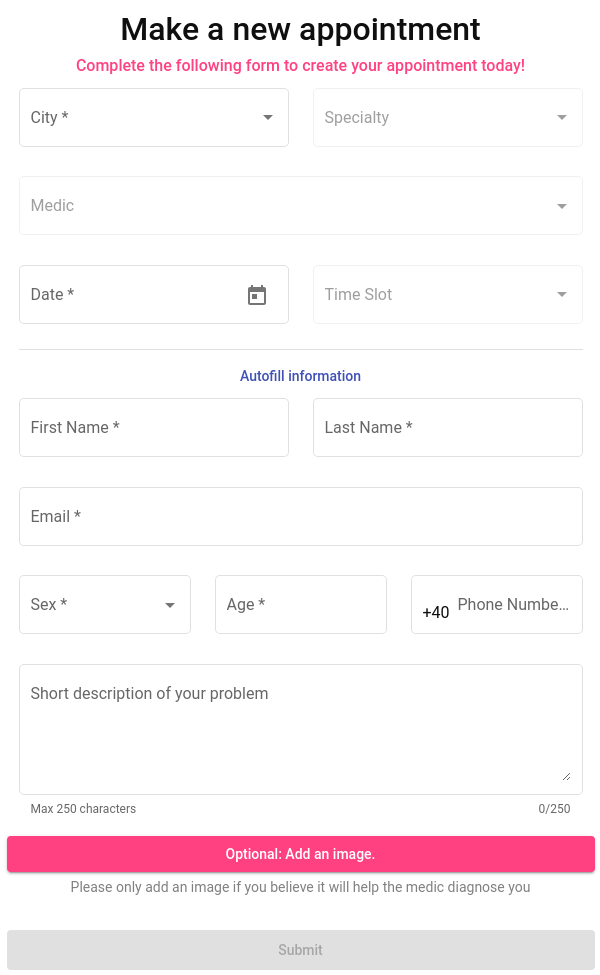


Figure 3‑16 Formularul din secțiunea "Create an Appointment"

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).

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.

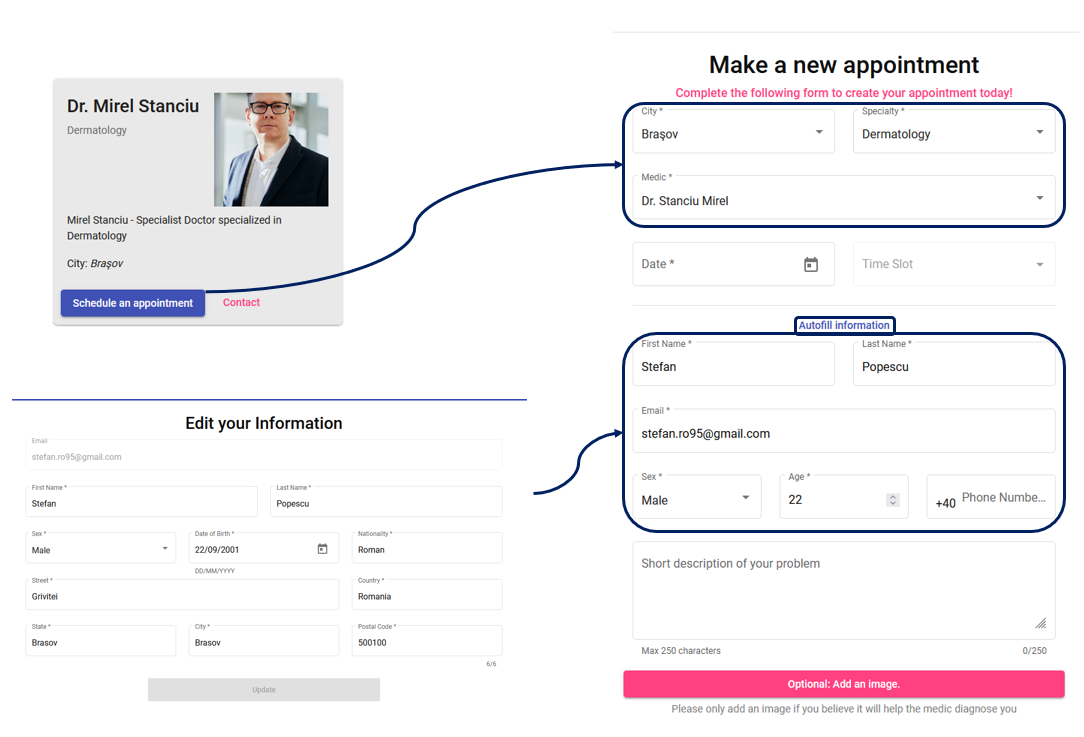


Figure 3‑17 Diagrama care exemplifică completarea automată a datelor unei programări

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.

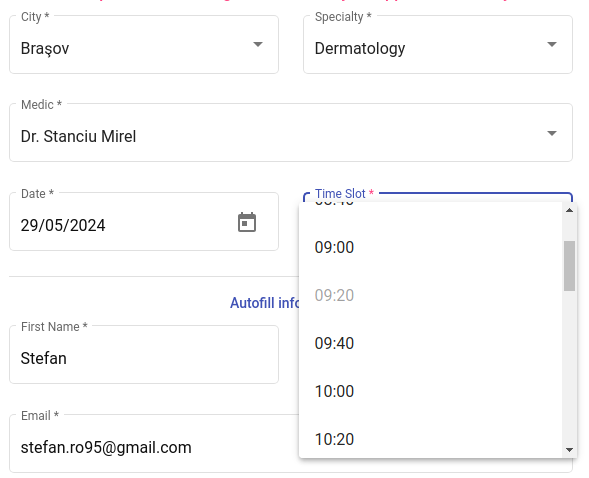


Figure 3‑18 Câmpurile de tip dropdown coordonate. Se poate observa că nu se poate selecta ora 09:20 deoarece medicul selectat are deja o consultație programată la acea oră

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 “FirebaseAppointment.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”, “FirebaseAppointment” folosește tipul “Timestamp”, specific

Firestore, pentru a stoca datele calendaristice. Această conversie este esențială deoarece Firestore stochează datele de tip calendaristic folosind tipul “Timestamp” [15]. Prin urmare, toate datele unei programări stocate în backend sunt de tip “FirebaseAppointment”. 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 “FirebaseAppointment” și vice-versa.

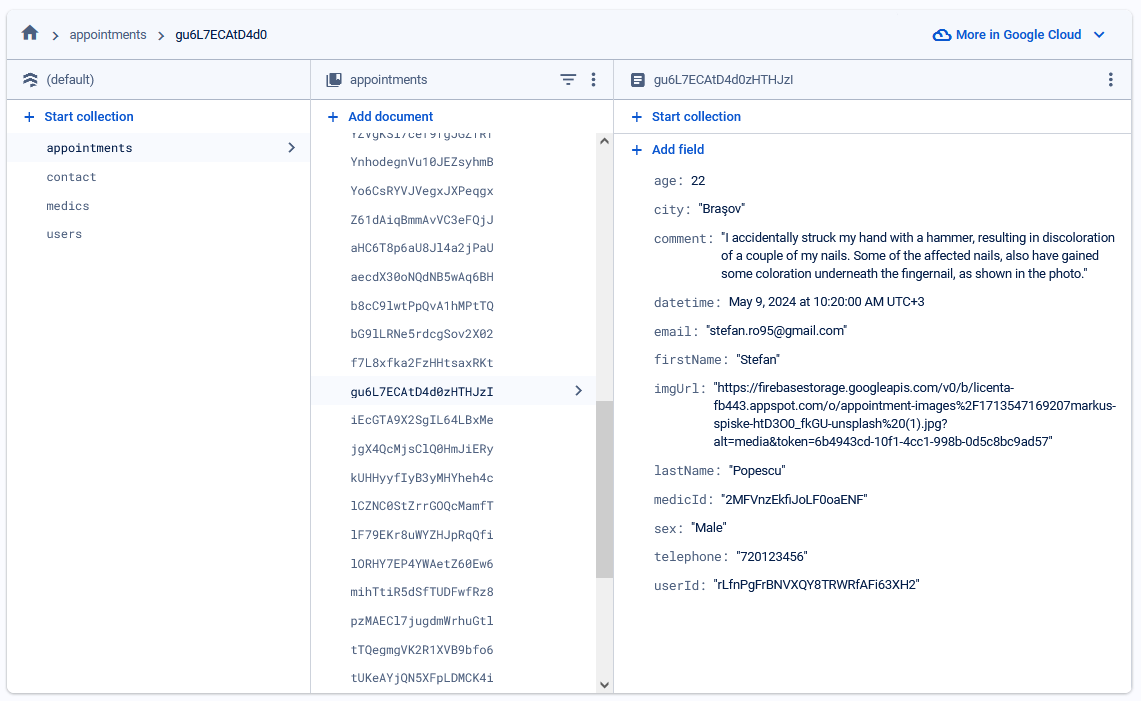


Figure 3‑19 Structura bazei de date Cloud Firestore pentru programări

După efectuarea conversiei, metoda “addAppointment()” apelează API-ul Firestore pentru a adăuga programarea în baza de date. 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.

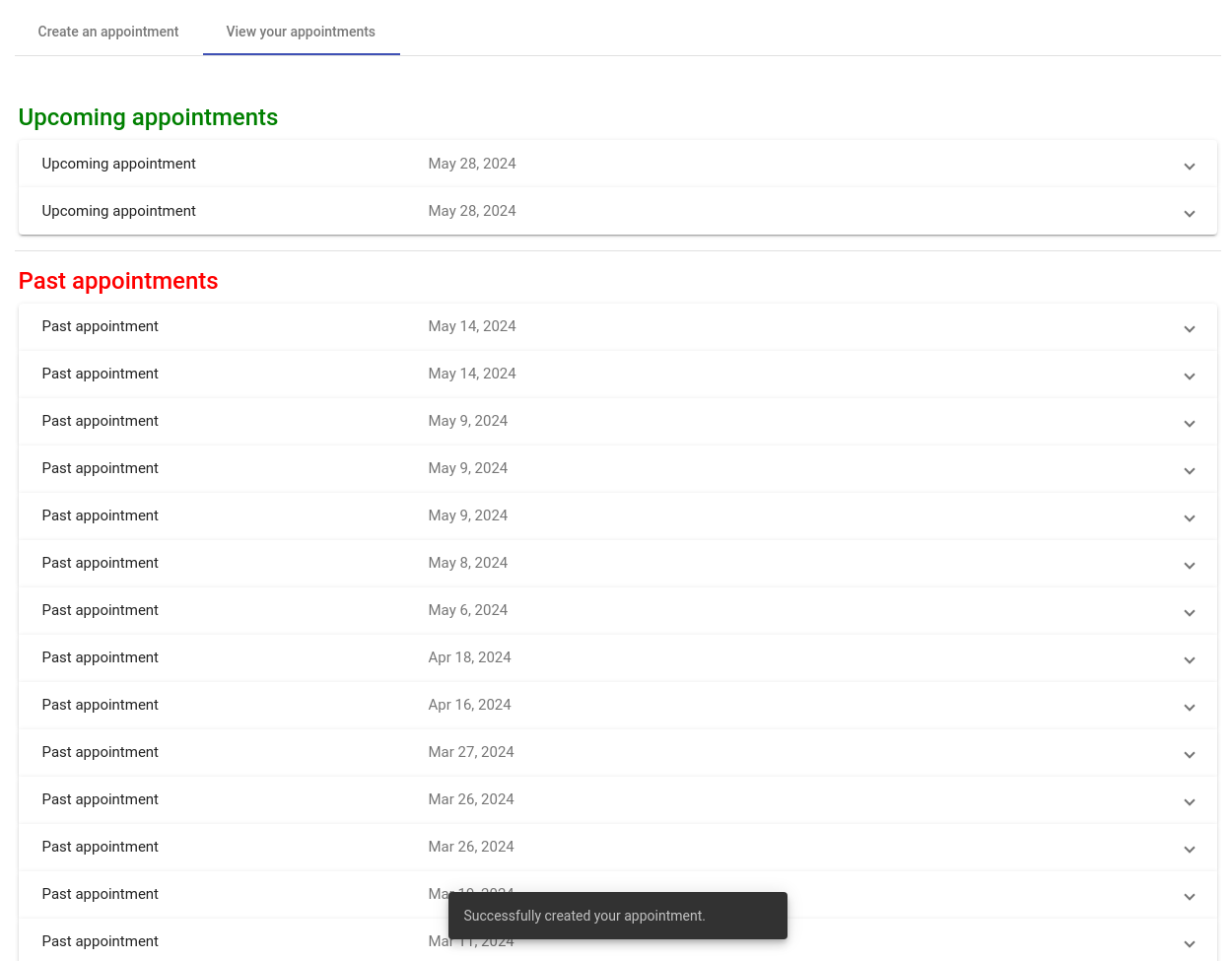


Figure 3‑20 Secțiunea "View your Appointments". În acest caz, utilizatorul a fost redirecționat automat după crearea unei programări (se poate observa un snackbar informativ)

### 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-pane”l, 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.

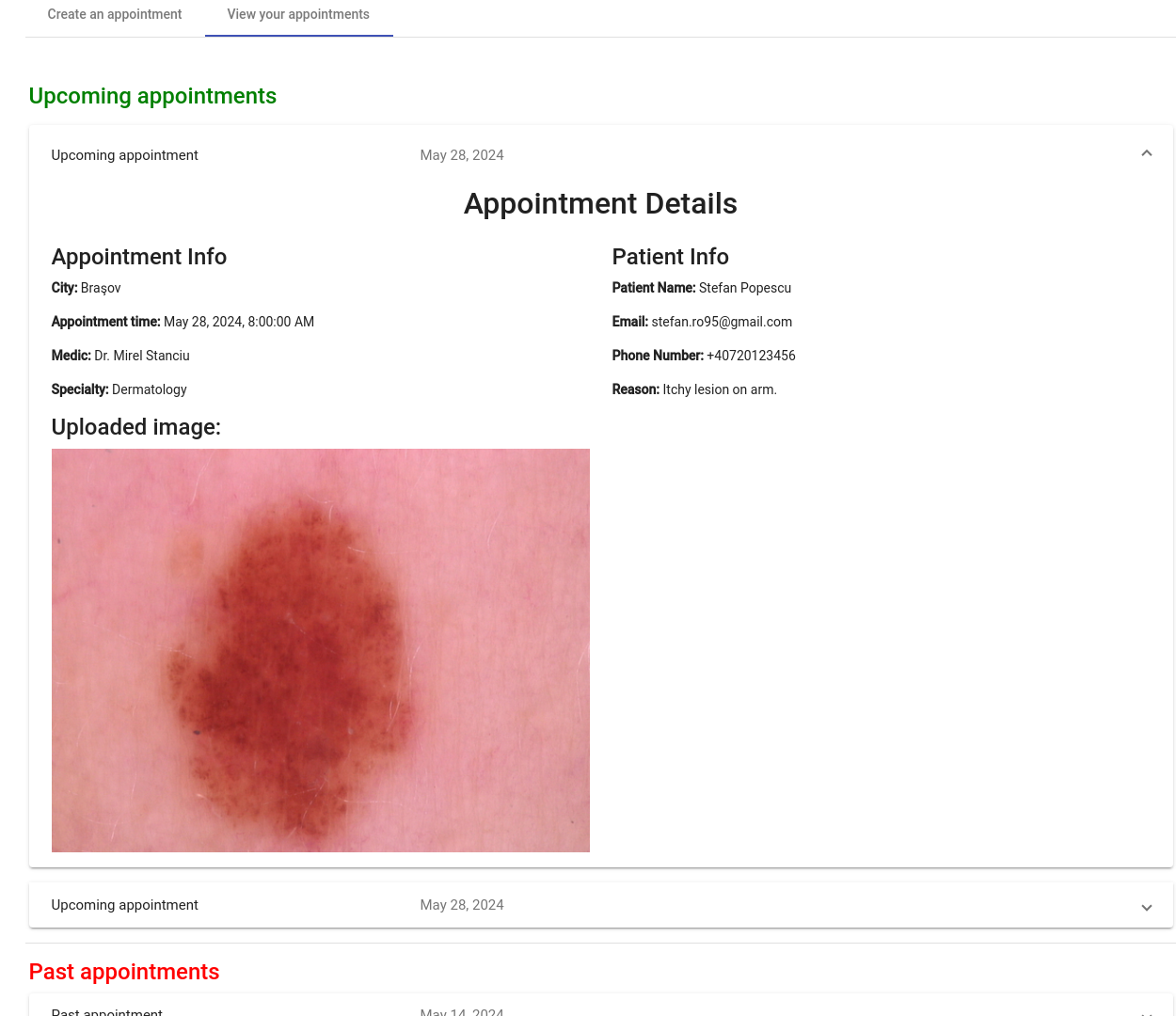


Figure 3‑21 "Deschiderea" unei programări din secțiunea "View your appointments" pentru a vedea detaliile acesteia

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.

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”.

## Pagina de Program (Schedule)

### Secțiunea “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 in 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 in cazul in care este necesar mai puțin timp pentru o programare).

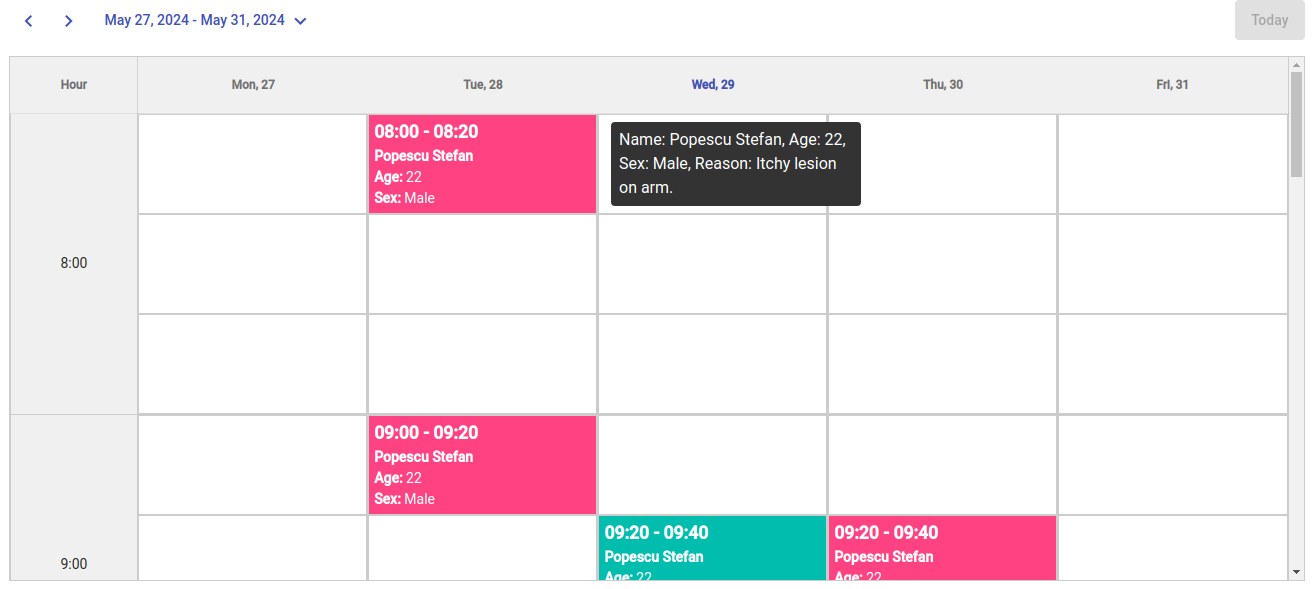


Figure 3‑22 Interfața paginii de program a medicului

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 [5] 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 saptămâna afişată în calendar diferă de săptămâna curentă, butonul devine activ. Când utilizatorul apasă pe acest buton, vizualizarea calendarului se actualizează pentru a afișa săptămâna curentă.

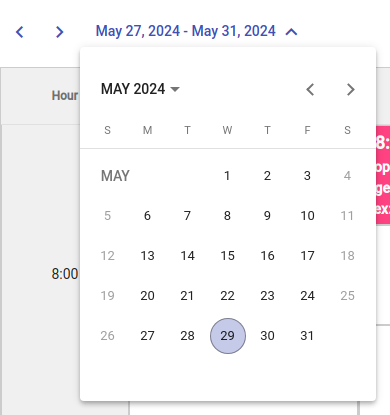


Figure 3‑23 Componenta de selectare a unei date calendaristice (Datepicker)

Pentru afişarea 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).

Astfel, 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 utilzatorul plasează cursorul deasupra unei casete de programare, va apărea un tooltip (creat utilizând componenta mat-tooltip [5] 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.

### Secțiunea “Schedule Details”

Dacă medicul apasă pe caseta unei programări, se deschide un dialog creat cu ajutorul componentei “mat-dialog” [5] 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.

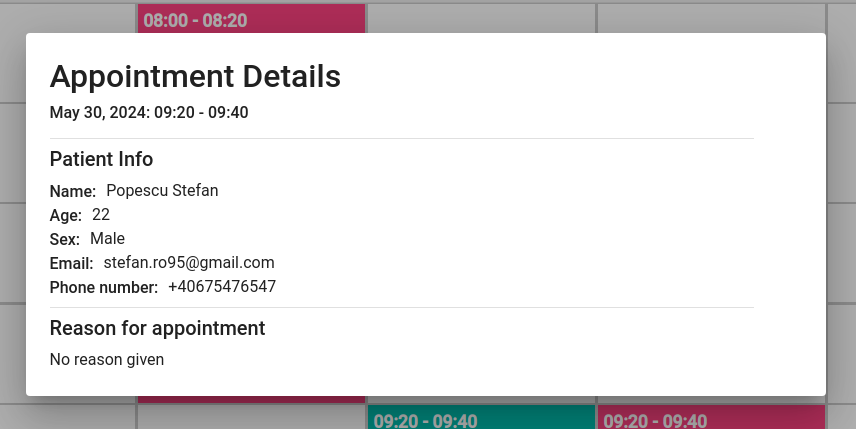


Figure 3‑24 Componenta "Dialog" care afișează detalii despre programare

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.

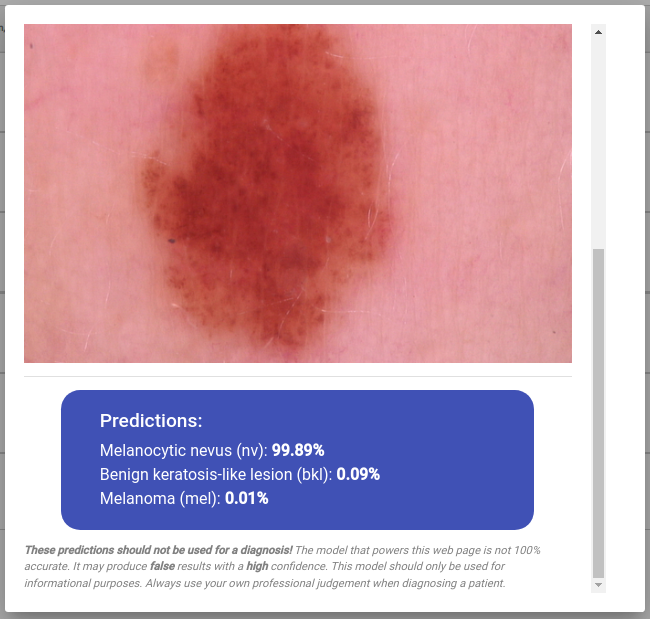


Figure 3‑26 Predicțiile modelului pentru imaginea data

# Bibliografie

|  |  |
| --- | --- |
| [1] | S. H. Edward, C. J. James and C. F. Michael, Biomedical Informatics: Computer Applications in Health Care and Biomedicine, Springer, 2021. |
| [2] | P. Zhao, I. Yoo, J. Lavoie, B. J. Lavoie and E. Simoes, "Web-Based Medical Appointment Systems: A Systematic Review," *Journal of Medical Internet Research,* vol. 19, no. 4, 2017. |
| [3] | B. K. Ragala, "Advantages and Disadvantages of Angular," 22 September 2023. [Online]. Available: https://www.knowledgehut.com/blog/web-development/advantages-and-disadvantages-of-angular. |
| [4] | G. A. M. T. M. Bierman, "Understanding TypeScript," in *ECOOP 2014–Object-Oriented Programming: 28th European Conference*, Uppsala, Springer Berlin Heidelberg, 2014, pp. 257-281. |
| [5] | Angular, "Angular Material UI component library," Angular, [Online]. Available: https://v14.material.angular.io/. |
| [6] | Google, "Material Design," Google, [Online]. Available: https://m3.material.io/. |
| [7] | Bootstrap, "Bootstrap - The most popular HTML, CSS, and JS library in the world.," Bootstrap, [Online]. Available: https://getbootstrap.com/. |
| [8] | Firebase, "Firebase | Google's Mobile and Web App Development Platform," [Online]. Available: https://firebase.google.com/. |
| [9] | Firebase, "Firebase Authentication," [Online]. Available: https://firebase.google.com/docs/auth/. |
| [10] | TensorFlow, [Online]. Available: https://www.tensorflow.org. |
| [11] | "Keras: Deep Learning for humans," [Online]. Available: https://keras.io. |
| [12] | Angular, "Angular - Angular Routing," Angular, [Online]. Available: https://angular.io/guide/routing-overview. |
| [13] | Angular, "Angular - Structural Directives," Angular, [Online]. Available: https://angular.io/guide/structural-directives. |
| [14] | Firebase, "Firebase Auth REST API," [Online]. Available: https://firebase.google.com/docs/reference/rest/auth. |
| [15] | Firebase, "Timestamp | Firebase," [Online]. Available: https://firebase.google.com/docs/reference/android/com/google/firebase/Timestamp. |
| [16] | "Learn Angular," December 2023. [Online]. Available: https://angular.dev/tutorials/learn-angular. |
| [17] | U. o. S. Florida, "What is Medical Informatics?," 28 February 2023. [Online]. Available: https://www.usfhealthonline.com/resources/health-informatics/what-is-medical-informatics/. |
| [18] | Angular, "Angular - Component Lifecycle," Angular, [Online]. Available: https://angular.io/guide/lifecycle-hooks. |