



**UNIVERSITATEA DIN
BUCUREȘTI**

**FACULTATEA DE
MATEMATICĂ ȘI
INFORMATICĂ**



SPECIALIZAREA INFORMATICĂ

Lucrare de licență

DEZVOLTAREA UNEI APLICAȚII WEB PENTRU ÎNVĂȚAREA DE LIMBI STRĂINE

Absolvent

Scobiola Radu

Coordonator științific

Lect. Univ. Dr. Mihăilescu Marius Iulian

București, iunie 2023

Rezumat

La nivel global, aproximativ 40% din populația planetei nu știe să vorbească decât într-o singură limbă. În comparație, 43% din oameni știu să vorbească fluent în două limbi diferite, însă doar 17% sunt vorbitori fluenți în cel puțin trei limbi.

În momentul de față, aproximativ 1,2 miliarde de oameni studiază o limba străină, deci există un număr foarte mare de persoane care sunt în căutare de cât mai multe metode și resurse diferite pentru a le ajuta cu procesul de învățare.

Pentru aceasta vom realiza o aplicație web care va încerca să facă procesul de învățare cât mai ușor, dar și interactiv pentru a oferi o experiență cât mai plăcută fără a sacrifica calitatea cunoștințelor oferite. Aplicația va fi realizată în Angular pentru partea de front-end și va utiliza Firebase pentru partea de back-end.

Abstract

On a global scale, approximately 40% of the world's population knows how to speak just one language. By comparison, 43% of people know how to fluently speak two languages, but only 17% are fluent in at least three different languages.

Right now, approximately 1.2 billion people are studying a foreign language, so there is an exceptionally large number of people in the search for as many different methods and resources as possible to help them with the learning process.

For this reason, we will create a web application which will attempt to make the learning process as easy as possible, while also interactive to offer an experience as pleasant as possible without sacrificing the quality of the information offered. The application will be made in Angular for the front-end and for the back-end we will use Firebase.

Cuprins

Capitolul 1.....	5
Introducere.....	5
1.1 Motivație.....	5
1.2 Obiectivele proiectului	5
1.3 Descrierea aplicației și a funcționalităților principale.....	6
Capitolul 2.....	7
Metodologie	7
2.1 Studierea altor aplicații	7
2.1.1 Identificarea elementelor pozitive	7
2.1.2 Identificarea nevoilor utilizatorilor	9
2.2 Definirea specificațiilor aplicației	10
2.2.1 Tehnice	11
2.2.2 User experience	12
Capitolul 3.....	13
Aplicația web	13
3.1 Tehnologiile folosite	13
3.1.1 HTML	13
3.1.2 CSS	14
3.1.3 TypeScript.....	15
3.1.4 Angular.....	17
3.1.5 Firebase	19
3.2 Procesul de dezvoltare a aplicației	21
3.2.1 Front-end	22
3.2.2 Back-end.....	22

3.2.3 Legătura dintre front și back.....	25
Capitolul 4.....	29
Prezentarea aplicației	29
4.1 Generală.....	29
4.2 Pagina de logare și de creare cont.....	29
4.3 Pagina principală	30
4.4 Pagina de profil	30
Capitolul 5.....	32
Concluzii și perspective	32
5.1 Concluzii.....	32
5.2 Îmbunătățiri viitoare.....	33
Bibliografie	34

Capitolul 1

Introducere

1.1 Motivație

Învățarea de limbi străine a fost dintotdeauna un subiect care m-a pasionat foarte mult și mereu am căutat metode cât mai eficiente, dar și interesante totodată pentru a învăța cât mai ușor. Scopul principal al acestei lucrări este de a crea o aplicație care oferă lecții cât mai interactive și ușor de urmat, fără a renunța însă la calitatea acestora. De asemenea, în plus față de lecții, aplicația va oferi multe alte resurse folositoare pentru a învăța o limbă străină mai ușor.

1.2 Obiectivele proiectului

Obiectivul principal al acestei lucrări este de a crea o aplicație web care în primul rând are o interfață simplă și estetică, pentru a oferi o experiență cât mai plăcută posibil utilizatorului. De asemenea, aplicația va încerca să reprezinte o alternativă pentru oamenii care au încercat să învețe limbi străine folosind alte astfel de aplicații. Pentru aceasta, vor fi implementate metode sau resurse pentru învățare ce nu se regăsesc în alte aplicații de învățare a limbilor străine.

1.3 Descrierea aplicației și a funcționalităților principale

Aplicația va fi realizată în Angular și va avea următoarele pagini principale:

- **Pagina de register/login**, care va fi o pagină simplă pentru crearea unui cont/logarea într-un cont deja existent.
- **Pagina principală (dashboard-ul)**, unde utilizatorul își va putea edita contul și alege limba maternă și limba pe care își dorește să o învețe.
- **Pagina cu resursele de învățare**. Aici utilizatorul va putea alege din diferitele metode de învățare.
- **Pagina cu exercițiile**, unde utilizatorul va avea o listă cu diferite categorii de exerciții din care va putea alege, fiecare având un anumit nivel de dificultate și testând cunoștințe diferite.

Capitolul 2

Metodologie

2.1 Studiarea altor aplicații

Pentru a putea realiza o aplicație care să folosească metode moderne de învățare am ales să studiez cele mai populare două aplicații pentru învățarea limbilor străine: Duolingo și Babbel. Apoi am strâns mai multe recenzii ale utilizatorilor pentru a vedea care erau lucrurile principale pe care le apreciau la aceste aplicații și care erau lucrurile pe care le displăceau cel mai mult la ele. Astfel, cu ajutorul acestor informații a fost mult mai ușor de identificat ce anume ar trebui să implementeze aplicația și ce lucruri ar trebui să evite, sau să încerce să îmbunătățească.

Site-urile de review-uri folosite au fost Trustpilot [1] și GetApp [2], oferind mii de recenzii atât pozitive cât și negative pentru ambele aplicații. Dat fiind numărul mare de recenzii, am putut extrage principalele avantaje și dezavantaje ale fiecărei aplicații.

2.1.1 Identificarea elementelor pozitive

În primă fază am dorit să identific care sunt lucrurile pe care aceste două aplicații le fac cel mai bine și care sunt motivele principale pentru care au atât de mulți utilizatori. Știind aceste lucruri, va fi mult mai ușor să ne asigurăm că aplicația are cele mai importante funcții. Aspectele principale pozitive pentru aplicații au fost următoarele:

Duolingo:

- Lecțiile sunt scurte, interesante și captivante.
- Modul de structurare al lecțiilor este asemănător unui joc, ceea ce creează o experiență distractivă.

- Competiția săptămânală și structurarea acesteia într-un sistem de ligi diferite oferă o motivație în plus pentru a învăța.
- Versiunea gratuită a aplicației oferă multe, cea platită nu este absolut necesară pentru învățarea unei noi limbi.

Babbel:

- Explicații foarte amănunțite, informațiile sunt apoi fixate cu ajutorul exercițiilor.
- Multe noțiuni teoretice, mai asemănător cu un curs clasic.
- Funcție de revizuire pentru utilizator, permițându-i acestuia să parcurgă din nou un curs anterior și să dea iarăși testele în cazul în care își dorește să repete anumite noțiuni.
- Explică gramatica unei limbi.
- Oferă exemple de dialog întâlnite în viața de zi cu zi.

Astfel, aspectele principale ale aplicațiilor care duc la o experiență bună a utilizatorului au fost identificate ca fiind următoarele:

- Lecții scurte, dar care să includă noțiuni teoretice.
- Un echilibru între o lecție interactivă și distractivă și una informativă.
- Lecții ce învață mai ales cuvinte sau fraze care ar fi de cel mai mult folos într-un dialog. Oferirea de exemple de limbaj colocvial.
- Abilitatea unui utilizator de a revizui lecțiile și exercițiile trecute.

2.1.2 Identificarea nevoilor utilizatorilor

După identificarea aspectelor pozitive ale celor două aplicații, următorul pas a fost de a identifica principalele funcții care le displac utilizatorilor pentru a le evita în implementarea aplicației. De asemenea, au fost identificate și anumite funcții pe care utilizatorii și le-ar dori să existe. Principalele critici aduse aplicațiilor sunt următoarele:

Duolingo:

- Structura gramaticii nu este explicată din punct de vedere teoretic, ca și utilizator trebuie să o memorezi pe de rost.
- Aplicația nu este foarte bună pentru a învăța cum să porți un dialog într-o limbă străină.
- Multe cuvinte noi care apar direct în exerciții, fără a fi explicate înainte într-un modul teoretic.
- Lipsa de flexibilitate în privința ritmului de învățare. Utilizatorii nu pot alege să meargă direct la alte lecții dacă deja consideră că stăpânesc suficient de bine anumite concepte.

Babbel:

- Unele lecții sunt plictisitoare, ceea ce tinde să ducă la pierderea concentrării și a interesului utilizatorului.
- Lecțiile sunt scurte și apar între exerciții. Nu există o listă în care utilizatorul să poată vedea toate lecțiile la un loc pentru a învăța. Singurul mod în care utilizatorii pot învăța noțiuni teoretice este prin a alege să facă exerciții.
- Conținutul cursurilor diferă în funcție de limba nativă a vorbitorului, ceea ce duce la o lipsă de echilibru. Unele cursuri au mult mai multe informații pentru unele limbi native decât pentru altele.

- Nu are și o versiune gratuită a aplicației, doar un free trial pentru o săptămână.

În urma acestor recenzii, am stabilit care ar fi cele mai importante lucruri de evitat în crearea aplicației. De asemenea, am observat și care sunt funcțiile principale pe care utilizatorii și le-ar dori de la o astfel de aplicație, însă lipsesc. Acestea sunt:

- Prezența explicațiilor teoretice, pentru ca și procesul de învățare să aibă loc tot pe aplicație, nu doar cel de exersare.
- Punerea la dispoziție a tuturor resurselor de învățare pentru a fi descărcate sau accesate de către utilizator oricând are acesta nevoie de ele.
- Libertatea utilizatorului de a învăța în orice ordine dorește acesta și în propriul său ritm.
- Menținerea unei uniformități privind calitatea cursurilor indiferent de limba nativă a utilizatorului.

2.2 Definirea specificațiilor aplicației

În urma identificării funcționalităților care sunt cele mai apreciate și dorite de către utilizatori, și cele care ar trebui evitate, a urmat să stabilim care vor fi specificațiile aplicației.

Acestea sunt împărțite în două mari categorii. Prima dată, vom stabili care sunt specificațiile tehnice ale aplicației, care sunt funcționalitățile principale ale acesteia. Apoi vor fi stabilite și aspectele aplicației ce țin de user experience, de interfața pe care o are utilizatorul și de funcționalitățile implementate pe care le va avea acesta la dispoziție.

2.2.1 Tehnice

În primă fază, va fi nevoie de un sistem pentru logare și autentificare a utilizatorilor atunci când aceștia doresc să acceseze conținutul aplicației. Pentru aceasta vom realiza o bază de date în Firebase care va memora toate conturile create, fiecare cu propriul său ID, adresă de email, și parolă. De asemenea, vor fi reținute și alte date care ar putea fi de folos administratorilor, cum ar fi data la care fiecare cont a fost creat și când a fost ultima dată accesat.

După aceasta, va trebui implementat un dashboard, unde utilizatorul își va putea gestiona contul. Acesta va avea opțiunea de a-și schimba parola și poza de profil. De asemenea, aici el își va putea alege limba nativă, fapt ce va schimba limba peste tot în aplicație.

Pe lângă acestea, vor fi implementate sisteme de a randomiza exercițiile ce apar în cadrul unui test precum și opțiunea de a descărca materialele puse la dispoziție pentru învățare.

Progresul fiecărui utilizator va fi salvat și stocat de asemenea în baza de date pentru ca acesta să nu fie pierdut.

Pentru a gestiona interacțiunea cu baza de date Firebase vom folosi Firebase SDK și API-uri. Acestea ne vor permite să efectuăm operații de scriere și citire în baza de date, precum și să gestionăm autentificarea utilizatorilor.

De asemenea, vor fi implementate măsuri de criptare a datelor sensibile ale utilizatorilor pentru a asigura securitatea acestora. Vom utiliza practici recomandate de securitate, cum ar fi verificarea și validarea datelor de intrare pentru prevenirea atacurilor de tip SQL injection. [3] [4]

Nu în ultimul rând, performanța bună a aplicației va fi unul dintre obiectivele principale. Pentru a asigura acest lucru vom implementa tehnici de optimizare precum încărcarea tardivă a resurselor și minimizarea fișierelor. Aceste lucruri vor reduce timpul de încărcare al aplicației și îi vor îmbunătăți viteza de răspuns. [5]

2.2.2 User experience

Pentru a oferi o experiență cât mai plăcută utilizatorilor, vom acorda o atenție deosebită tuturor aspectelor ce țin de user experience (UX) în cadrul aplicației noastre. Iată aspectele ce le vom lua în considerare atunci când realizăm design-ul aplicației.

În primul rând, design-ul aplicației trebuie să fie intuitiv și coerent. Interfața aplicației va fi ușor de înțeles și de utilizat, astfel încât utilizatorii să poată naviga prin aplicație fără niciun fel de dificultăți. Vom folosi elemente de navigare familiare și des întâlnite pentru a asigura acest lucru și consistența aplicației.

Fluxul de utilizare al aplicației va fi unul simplu, ușor de urmărit de către utilizatori. Vom evita supraîncărcarea cu informații sau opțiunile inutile pentru a oferi a o experiență cât mai fluidă și ușor de urmărit.

Utilizatorii își vor putea urmări progresul în timp real pentru a le oferi o imagine de ansamblu asupra tuturor lucrurilor pe care le-au învățat. Acest lucru îi va motiva să continue să progreseze.

Vom implementa notificări și mesaje clare, atât pentru eventuale erori, cât și pentru performanța utilizatorului în cadrul unui test. Oferind feedback în timp real, utilizatorii vor înțelege mai bine interacțiunile lor cu aplicația.

Ne vom asigura că aplicația este complet responsivă și adaptată la toate tipurile diferite de dispozitive. Unul dintre cele mai importante aspecte este ca aplicația să arate bine și să funcționeze corect indiferent de dispozitivul de pe care este accesată, sau de dimensiunea ecranului acestuia.

Vom folosi o paletă de culori consistentă în cadrul întregii aplicații împreună cu fonturile și elementele grafice specifice. Astfel, nu doar că dăm o identitate a aplicației, dar vom forma o atmosferă plăcută și prietenoasă pentru un mediu de învățare, luând în considerare psihologia culorilor. De asemenea, menținerea unui design consistent va duce la familiarizarea în timp a utilizatorilor cu aplicația. Unele dintre cele mai mari nemulțumiri ale utilizatorilor apar atunci când o aplicație își schimbă complet design-ul cu care erau aceștia obișnuiți.

Capitolul 3

Aplicația web

3.1 Tehnologiile folosite

În realizarea aplicației au fost folosite mai multe tehnologii pentru a realiza diferite aspecte ale acesteia. Am implementat diferite limbaje de programare, framework-uri și platforme în comuniune unul cu celelalte pentru a realiza o aplicație completă, ce oferă o experiență plăcută. În continuare vom prezenta fiecare tehnologie în parte ce a fost folosită în cadrul lucrării.

3.1.1 HTML

HTML (HyperText Markup Language) este un limbaj de marcare utilizat pentru crearea și structurarea conținutului paginilor web. Este un element fundamental și esențial al dezvoltării web și reprezintă fundația pe care este construită o pagină web. Am folosit cea mai recentă versiune a limbajului și anume HTML5.

Principala funcție a HTML este de a defini structura paginii și de a-i organiza conținutul. În esență, HTML se folosește de etichete (sau tag-uri) pentru a delimita diferite secțiuni și elemente ale unei aplicații web. Acestea sunt mai apoi interpretate de către browser și afișate în mod corespunzător.

Un document HTML este realizat dintr-o serie de elemente, fiecare având o etichetă de început și una de încheiere. Există o multitudine de etichete diferite, fiecare având roluri diferite, pentru a defini tipuri diferite de conținut. Pe lângă etichetele clasice, cum ar fi `<title>`, `<div>`, `<p>`, există multe etichete diferite pentru funcții mai specifice. Dintre acestea amintim etichetele pentru imagini (``), link-uri (`<a>`),

liste (****, ****, ****), tabele (**<table>**, **<tr>**, **<td>**) și formulare și butoane (**<form>**, **<input>**, **<button>**).

Pe lângă etichetele predefinite, HTML oferă și opțiunea de a crea etichete personalizate prin intermediul atributelor **class** și **id**. Acestea permit dezvoltatorilor să atribuie stiluri personalizate prin intermediul CSS, sau funcționalități cu ajutorul JavaScript.

Prin utilizarea tuturor uneltelor puse la dispoziție de către HTML, dezvoltatorii web pot crea pagini web moderne și care funcționează pe o multitudine de motoare de căutare diferite. HTML reprezintă o fundație solidă pentru dezvoltarea de aplicații web, stând la baza construirii paginilor web.

3.1.2 CSS

CSS (Cascading Style Sheets) este un limbaj de dezvoltare web folosit pentru a stiliza aspectul conținutului HTML. Acesta funcționează în strânsă legătură cu HTML, permițând dezvoltatorilor să controleze modul în care elementele HTML sunt afișate în pagină.

Funcția sa principală este de a aplica stiluri și de a schimba design-ul elementelor HTML. CSS utilizează reguli și selecții pentru a aplica stiluri la o întreagă colecție de elemente similare în loc de a defini aspectul fiecărui element individual.

CSS folosește un model cascading, ceea ce înseamnă că se pot defini regulile în mai multe locuri. Acestea pot mai apoi să fie aplicate în funcție de specificitatea selectorului și de ordinea în care sunt definite. Astfel, utilizatorii pot controla aspectul elementelor într-un mod detaliat și să creeze diferite ierarhii de stiluri.

Un stil CSS este definit într-un bloc de reguli, care conține un selector și un set de declarații. Selectorul specifica elementul, sau elementele asupra cărora se aplică stilul. Declarațiile definesc diferite proprietăți, precum culoarea, dimensiunea, fontul, marginile și multe altele.

Există o gamă largă de valori și proprietăți ce pot fi editate de către dezvoltatori pentru a adapta aspectul elementelor HTML la nevoile lor. De asemenea, CSS are și suport concepte avansate, cum ar fi selectori de clasă, identificatori, pseudo-clase și pseudo-elemente. Toate aceste lucruri permit stilizarea selectivă a anumitor stări sau elemente.

Un alt aspect foarte des folosit și important al CSS este faptul că permite

dezvoltatorilor să aplice stiluri responsive, care sunt adaptate la diferite dimensiuni de ecrane și dispozitive (desktop, laptop, mobile etc.). Acest lucru se realizează prin utilizarea de media queries pentru definirea stilurilor diferite în funcție de mărimea ecranului și prin intermediul tehnicilor de layout flexibil sau grid.

CSS poate fi inclus într-un document HTML prin intermediul unui bloc `<style>` în antetul paginii, sau prin includerea unui fișier extern CSS prin utilizarea unui element `<link>`. Această separare a conținutului de către stiluri face gestionarea și întreținerea paginilor web mult mai facilă.

Asemeni lui HTML, CSS este un limbaj esențial pentru dezvoltarea de aplicații web deoarece cu ajutorul său este creat aspectul unei pagini. De la stilizări simple, până la design-uri complexe, interactive și responsive, CSS oferă o gamă largă de opțiuni pentru a transforma conținutul unei pagini web într-un aspect vizual plăcut și atractiv utilizatorilor.

3.1.3 TypeScript

TypeScript este un limbaj de programare open-source și gratuit care a fost dezvoltat de către Microsoft care extinde și îmbunătățește JavaScript adăugând diferite funcționalități. Acesta a fost lansat la 1 octombrie 2012 și a devenit foarte repede unul dintre cele mai populare limbaje pentru dezvoltarea de aplicații web datorită avantajelor sale față de JavaScript. El a fost creat pentru a elimina deficiențele lui JavaScript în dezvoltarea de aplicații pe scară largă atât în cadrul lui Microsoft, cât și pentru clienții lor externi.

Principalul său obiectiv este de a adăuga tipizare statică la JavaScript, ceea ce înseamnă că dezvoltatorii pot specifica tipurile de date pentru variabile, funcții și alte elemente din codul lor. Acest lucru aduce mai multe avantaje, cum ar fi:

- **Detectarea erorilor la compilare.** TypeScript verifică semantica codului înainte de a rula aplicația. Astfel, multe erori des întâlnite sunt detectate și semnalate în timpul procesului de dezvoltare, evitându-se apariția lor în timpul rulării programului.
- **Instrumente de dezvoltare și funcție de autocomplete îmbunătățite.** Editorii și mediile de dezvoltare care susțin TypeScript oferă funcționalități

avansate de autocompletare și sugestii de cod. Acest lucru crește nivelul de productivitate al programatorilor și reduce erorile de cod ce pot apărea pe parcursul dezvoltării.

- **Înțelegere mai bună a codului.** Tipurile de date și semnăturile funcțiilor ajută la o mai bună înțelegere a codului. Acest lucru duce la o eficiență mai mare a programatorilor deoarece mai puțin timp este pierdut descifrând codul.
- **Refactorizare și reutilizare ușoară.** TypeScript facilitează refactoring-ul și reutilizarea codului prin furnizarea de informații precise despre structura și tipurile de date ale acestuia.

La momentul compilării, codul de TypeScript este transformat în cod de JavaScript înainte de a fi rulat într-un mediu de dezvoltare sau într-un browser. Acest lucru înseamnă că dezvoltatorii pot utiliza sintaxa și funcțiile avansate ale lui TypeScript, iar rezultatul final va fi cod perfect funcțional de JavaScript. De asemenea, cum TypeScript este un super-set al lui JavaScript, acest lucru înseamnă că orice cod valid din punct de vedere sintactic de JavaScript va fi și cod valid de TypeScript.

Pe lângă tipizare, TypeScript mai adaugă în plus câteva funcționalități avansate, cum ar fi:

- **Clase și obiecte:** TypeScript permite dezvoltatorilor să folosească concepte de programare orientată pe obiecte, cum ar fi definirea de clase și crearea de obiecte. Aceste lucruri duc la o organizare mai bună a codului și de asemenea permite utilizarea de concepte precum încapsularea și moștenirea.
- **Module:** TypeScript oferă suport nativ pentru modulare, permițând împărțirea codului în module pentru a fi importat sau exportat între diferite fișiere sau proiecte. Aceasta ajută la organizarea și reutilizarea codului în aplicații mai mari.
- **Generice:** TypeScript oferă suport pentru generice, care permit dezvoltatorilor să scrie cod generic pentru a-l adapta la diferite tipuri de date. Acest lucru sporește flexibilitatea și reutilizabilitatea codului.

- **Interfețe:** TypeScript permite implementarea de interfețe. Acestea definesc sintaxa pe care ar trebui s-o respecte o entitate. Interfețele conțin declarațiile pentru proprietăți, metode și event-uri, care mai apoi trebuie definite de către clasa derivată.

TypeScript poate fi utilizat într-o varietate de scenarii de dezvoltare web, de la aplicații mici și site-uri web, până la aplicații complexe de tip enterprise. Este susținut de o comunitate activă, are o documentație bogată și fiind unul dintre cele mai populare limbaje de programare din lume este ușor de învățat și de adoptat de către dezvoltatori, indiferent de nivelul lor de cunoștințe.

3.1.4 Angular

Angular este un framework open-source folosit pentru crearea de aplicații web bazat pe TypeScript care a fost dezvoltat de către Google și este o rescriere completă a lui AngularJS, care fusese creat de aceeași echipă. [6]

Am ales să folosesc un framework pentru această lucrare deoarece în general acestea cresc eficiența și performanțele dezvoltatorilor de aplicații web. Un framework oferă o structură și salvează mult timp prin faptul că nu trebuie scris cod de la 0 de fiecare dată.

Angular se folosește de DOM (Document Object Model), care tratează un document HTML ca pe o structură de arbore în care fiecare nod reprezintă o parte din document. Angular folosește un DOM real, adică atunci când are loc o schimbare asupra unui element din arbore întreg arborele este reîncărcat, nu doar elementul schimbat.

Angular se bazează pe o arhitectură de framework MVC (model-view-controller) și are diagrame detaliate care ilustrează cum este realizat transferul de date dintre view și controller.

O aplicație Angular are următoarele blocuri pe care este construită:

- **Modul:** un mecanism de grupare a componentelor, directivelor și serviciilor care îndeplinesc funcții asemănătoare. Un modul trebuie să fie capabil de a fi combinat cu alte module pentru a crea o aplicație. Practic, un modul este un container pentru diferite părți ale unei aplicații.

- **Componentă:** toate aplicațiile Angular sunt construite din una sau mai multe componente. Componentele sunt clase TypeScript simple cu șabloane HTML și nume. Șablonul HTML al unei componente poate accesa date din clasa sau componenta sa corespunzătoare dacă are un nume. Componentele Angular pot avea stiluri CSS asociate cu ele și șabloanele HTML le pot accesa.
- **Șablon:** este un super-set HTML. Include toate funcționalitățile HTML, însă pe lângă acestea are de asemenea capacitatea de a introduce datele componentelor în HTML. Pe lângă aceasta, de asemenea generează elementele HTML DOM în mod dinamic.
- **Metadata:** folosit de către Angular pentru a informa framework-ul cum ar trebui să fie folosita o clasă.
- **Serviciu:** este folosit pentru a oferi funcționalități foarte specifice într-o aplicație. Are o singură funcție însă o realizează foarte bine, scopul său este de a reduce repetiția. În loc de a defini o funcție în cadrul unei componente, este mai practic să fie separat într-un serviciu ce poate fi folosit și pentru alte componente. Serviciile pot fi folosite pentru a controla logica de afaceri a aplicației.

Arhitectura Angular are atât avantaje cât și dezavantaje. În continuare le vom enumera pe acestea, începând mai întâi cu avantajele sale.

- Structura MVC (model-view-controller) permite separarea părților aplicației ce țin de funcționalitate de stratul UI, creând astfel un design modular. Controller-ul se ocupă de toate request-urile din partea aplicației, inclusiv de request-urile ce vin din view și lucrează împreună cu modelul pentru a crea datele necesare. View-ul folosește datele produse de controller și afișează imaginea finală.
- Componentele, serviciile, directivele și pipe-urile pot fi grupate în module în Angular, care apoi pot fi utilizate pentru a crea o aplicație. Această separare înseamnă că fiecare modul conține doar funcțiile care sunt definite și folosite în cadrul său. Acest lucru duce la un cod mai curat, ușor de înțeles și de

întreținut, indiferent de dimensiunea aplicației.

- Dependențele pot fi rezolvate utilizând design pattern-ul de Dependency Injection. Când aceste dependențe există, serviciile sunt grupate împreună pentru a ne asigura că își vor completa obiectivele. Dependency Injection practic separă un job în mai multe părți. În Angular de exemplu sunt create instanțe ale serviciilor care apoi sunt injectate în clase.

Deși Angular are numeroase avantaje și motive pentru care a fost ales pentru realizarea acestei lucrări, este de menționat că există și câteva dezavantaje. Acestea au trebuit să fie luate în considerare în procesul de creare al aplicației web. Dintre dezavantaje amintim următoarele.

- Are multe concepte unice sale, ceea ce înseamnă că simplele cunoștințe de HTML, CSS și JavaScript/TypeScript nu sunt suficiente, trebuie învățate și conceptele specifice lui Angular. Din acest motiv este un framework mai greu de utilizat inițial, până ce dezvoltatorul devine familiar cu el.
- Chiar și pentru aplicații de mărimi mai mici, trebuie instalate foarte multe lucruri diferite. Acest lucru rezultă deseori într-o aplicație voluminoasă, care ocupă foarte mult spațiu de stocare.
- Deși în general structura foarte bine definită a lui Angular reprezintă un avantaj, în anumite cazuri aceasta poate reprezenta un dezavantaj. Structura rigidă poate însemna mai puțină flexibilitate, ceea ce duce la o abordare mai rigidă a procesului de dezvoltare a aplicației.

3.1.5 Firebase

Firebase este o platformă de dezvoltare a aplicațiilor mobile și web care a fost dezvoltată de Google. Aceasta oferă un set larg de servicii și unelte pentru dezvoltarea, testarea, gestionarea și scalarea aplicațiilor.

Firebase oferă o bază de date în timp real, denumită Firebase Realtime Database. Aceasta este o bază de date NoSQL care permite stocarea și sincronizarea datelor în

timp real între clienți și diferite dispozitive. Schimbările efectuate în baza de date sunt imediat propagate către toți clienții conectați, oferind o experiență în timp real pentru toți utilizatorii.

Pentru autorizare și autentificare este pus la dispoziția utilizatorilor un sistem simplu de utilizat. Acesta permite utilizatorilor să se conecteze folosind un cont de Google, Facebook, Github, sau altele, sau să utilizeze autentificare prin email și parolă. Firebase gestionează securitatea și autentificarea utilizatorilor, oferind diferite opțiuni de configurare și control granular asupra accesului la resursele aplicației.

Firebase oferă de asemenea și un serviciu de stocare a fișierelor în cloud. Acest lucru permite încărcarea, descărcarea și gestionarea fișierelor (imagini, video, documente etc.) direct din aplicație.

Există de asemenea și funcția de analitice, unde utilizatorii pot vizualiza diferite statistici folosite despre aplicația lor. Dezvoltatorii pot vedea nu doar date generale, cum ar fi numărul de vizitatori într-o anumită perioadă de timp, sau totalul de încasări produse de aplicație, dar și date mai specifice, cum ar fi urmărirea acțiunilor și a interacțiunilor vizitatorilor site-ului. Pe lângă acestea, Firebase oferă și instrumente pentru monitorizarea performanței și a erorilor, fapt care ajută la identificarea și rezolvarea problemelor în timp util.

Firebase Cloud Messaging (FCM) oferă servicii de trimis notificări în timp real pentru aplicații mobile și web. Aceasta permite trimiterea de notificări personalizate utilizatorilor, indiferent dacă aplicația este în utilizare sau dacă rulează în fundal, indiferent dacă platforma folosită este Android, iOS, sau web.

Pentru a realiza comunicarea între aplicația realizată cu Angular și Firebase va fi folosit Firebase API, care este un set de biblioteci și metode oferite de Firebase pentru a interacționa cu serviciile sale.

Firebase oferă un set de Software Development Kits (SDK-uri) pentru diferite platforme sau framework-uri. Pentru Angular vom folosi Firebase SDK for JavaScript, care include module și funcții specifice pentru a ne ajuta să comunicăm cu serviciile Firebase.

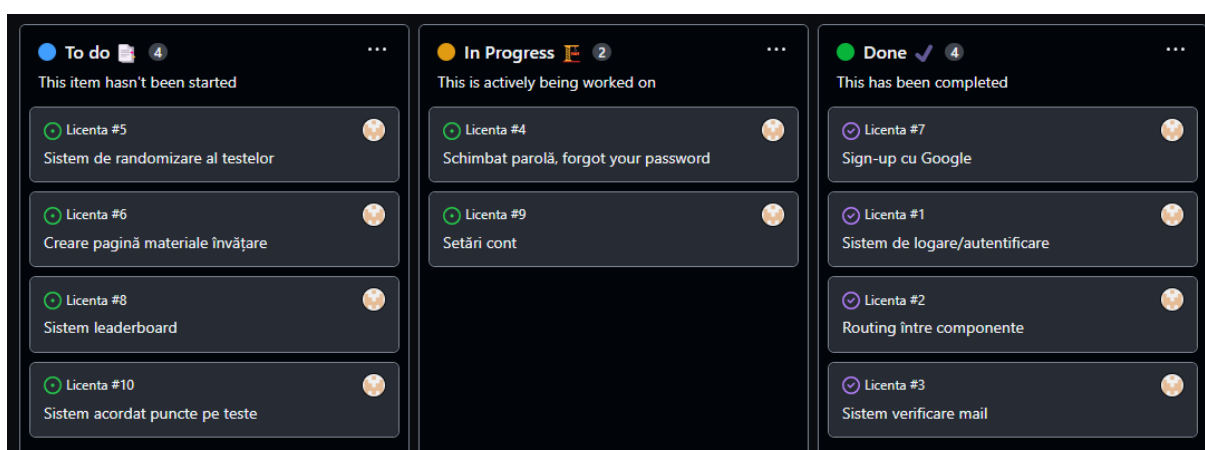
De asemenea, vom folosi biblioteca AngularFire [7] care a fost dezvoltată special pentru a integra Firebase într-o aplicație Angular. Această bibliotecă oferă componente, servicii și metode Angular pentru a interacționa cu serviciile Firebase, inclusiv serviciile pentru baza de date, autentificare, stocare și altele. AngularFire simplifică mult procesul de conectare a aplicației Angular la Firebase și gestionarea datelor în timp real.

API-ul Firebase ne permite să utilizăm diferite servicii Firebase direct în aplicația noastră Angular. De exemplu, putem utiliza serviciul **AngularFireDatabase** pentru a efectua operații de citire și scriere în baza de date, sau serviciul **AngularFireAuth** pentru a gestiona autentificarea utilizatorilor.

În plus, Firebase API ne pune la dispoziție și multe metode diferite. Pentru autentificare și autorizare avem metode precum **signInWithEmailAndPassword()** pentru autentificarea cu ajutorul adresei de email și parolei, sau **signInWithGoogle()** pentru autentificarea prin contul de Google. Pe lângă acestea există și metode pentru a scrie într-o bază de date, precum **setValue()** sau **update()**, iar observabilele Angular ne permit să ascultăm schimbările și să reacționăm adecvat. Astfel, putem crea o aplicație reactivă care răspunde la modificările datelor în timp real.

3.2 Procesul de dezvoltare a aplicației

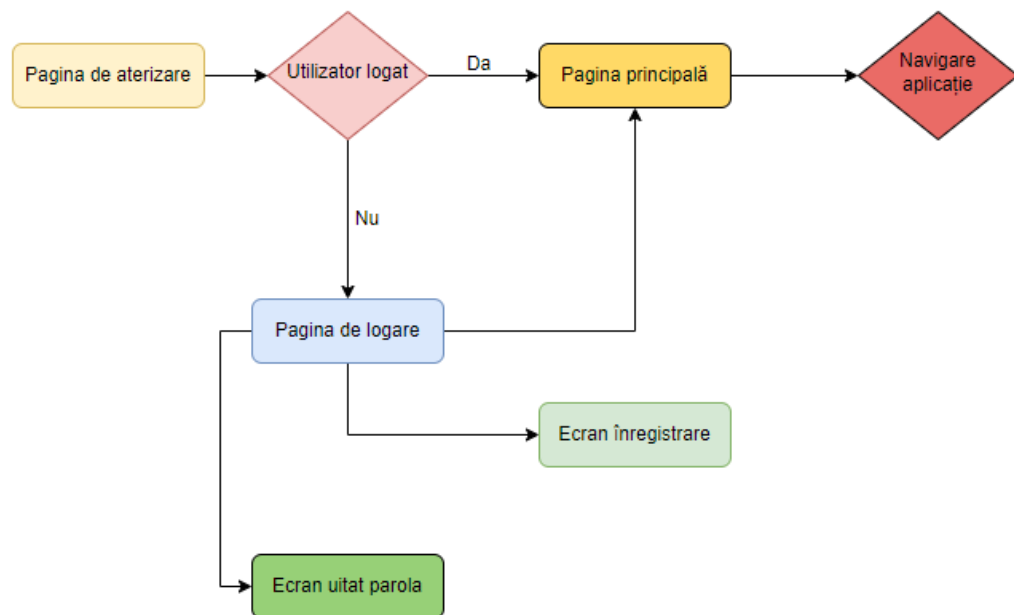
În procesul de dezvoltare am folosit resurse precum proiectele de tip **board** oferite de Github pentru a organiza într-un mod mai bun și mai eficient cerințele de implementare. Acestea au fost împărțite în trei categorii diferite, în funcție de statusul de dezvoltare: **To do**, **In progress** și **Done**. De asemenea, am deschis un issue nou atunci când o sarcină era în progres pentru a organiza și urmări mai bine starea în care se aflau funcționalitățile.



Pe parcursul dezvoltării oricând am identificat noi funcționalități care trebuiau adăugate, am actualizat în mod adecvat lista. Astfel am ținut în permanență evidența lucrurilor care trebuiau să fie implementate.

3.2.1 Front-end

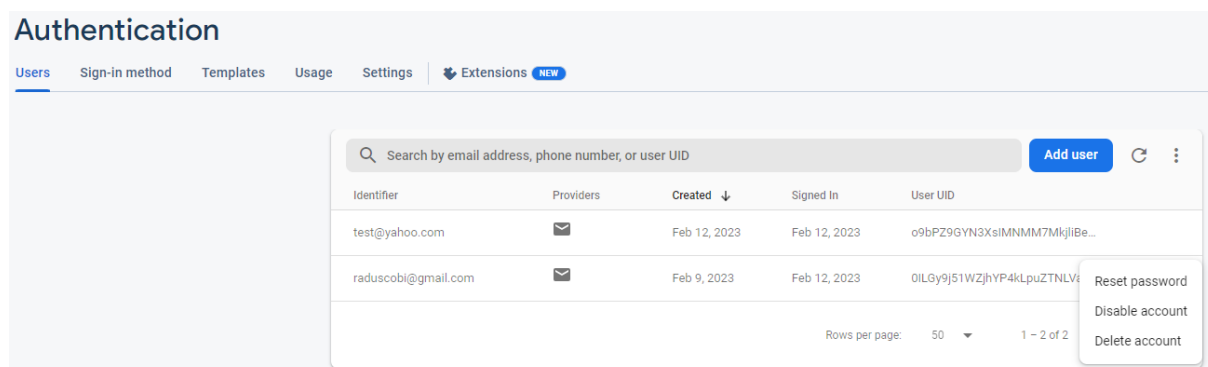
Navigarea aplicației de către client se va realiza conform următoarei diagrame:



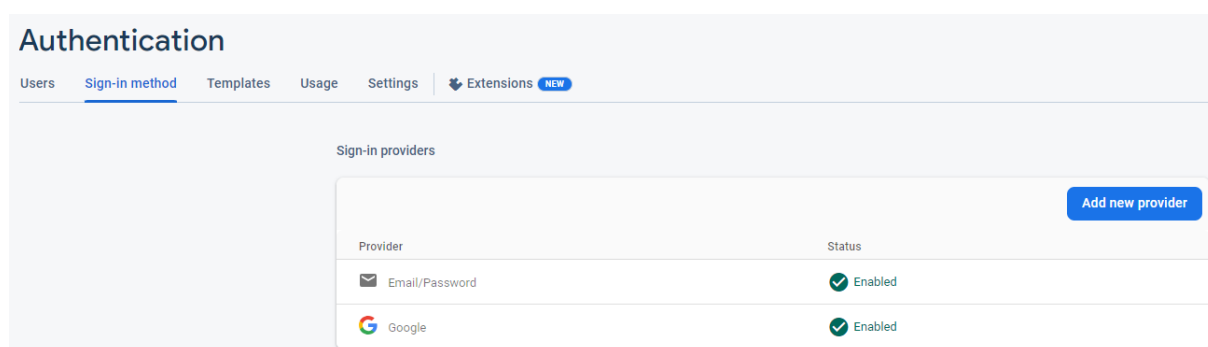
Navigarea este una intuitivă și am ales să evităm complicațiile în structura aplicației pentru a realiza acest lucru. De asemenea, după acțiunile precum crearea unui cont, logare, ne-am asigurat că utilizatorul este în permanență informat cu privire la potențiale erori, sau în cazul în care operația s-a efectuat cu succes.

3.2.2 Back-end

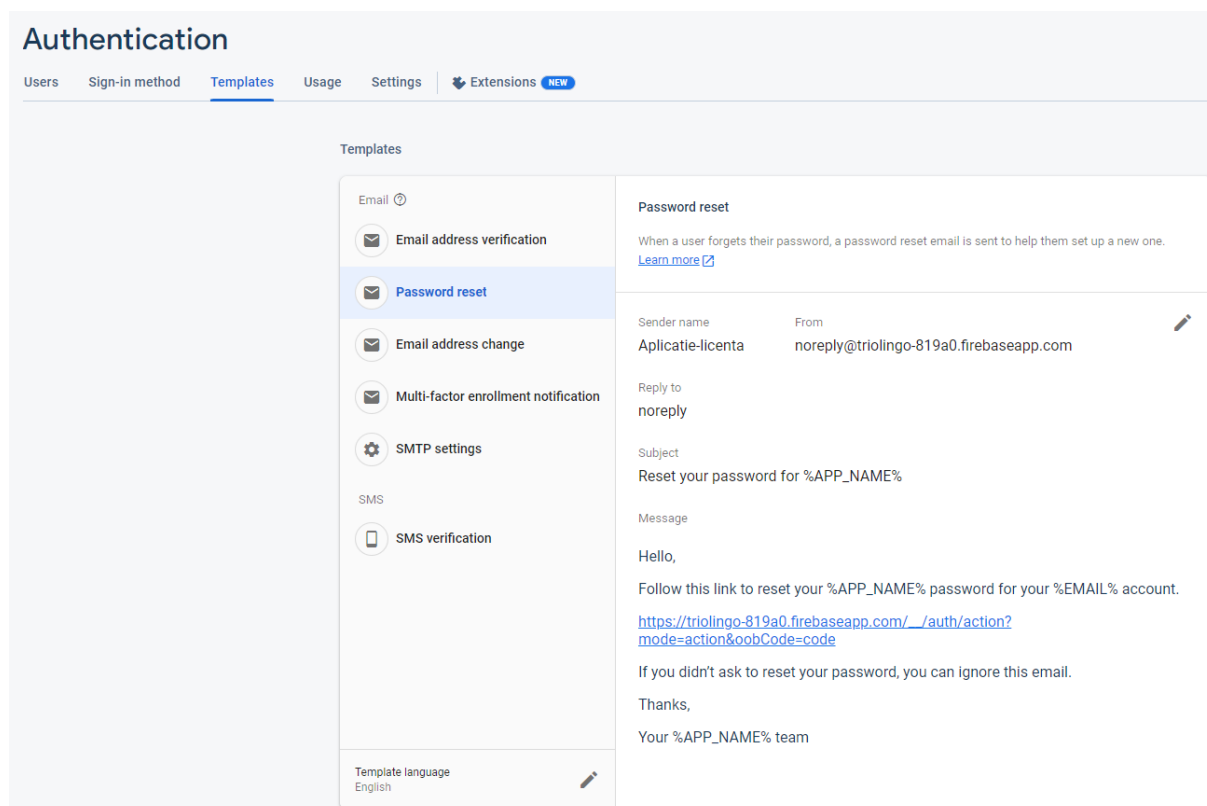
Partea de back-end a fost realizată cu ajutorul lui Firebase. În primul rând am realizat sistemul de autentificare, cu ajutorul căruia putem gestiona utilizatorii înregistrați și metodele prin care aceștia își pot crea un cont. Ca și admin, putem șterge sau dezactiva un cont. De asemenea, putem reseta parola unui utilizator din acest meniu. [8]



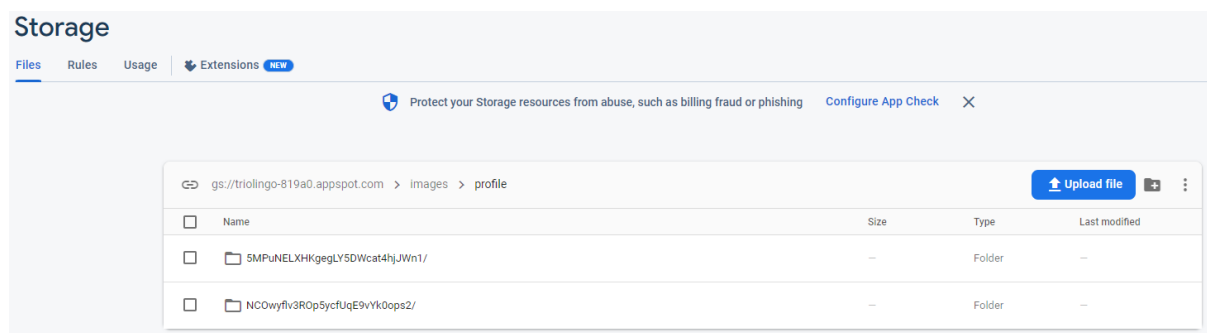
Pentru aplicație am ales ca metodele de logare să fie prin crearea unui cont nou cu email și parolă sau prin conectarea cu un cont de Google deja existent. Firebase ne pune la dispoziție un mod ușor de adăuga noi provideri pentru logare. [9]



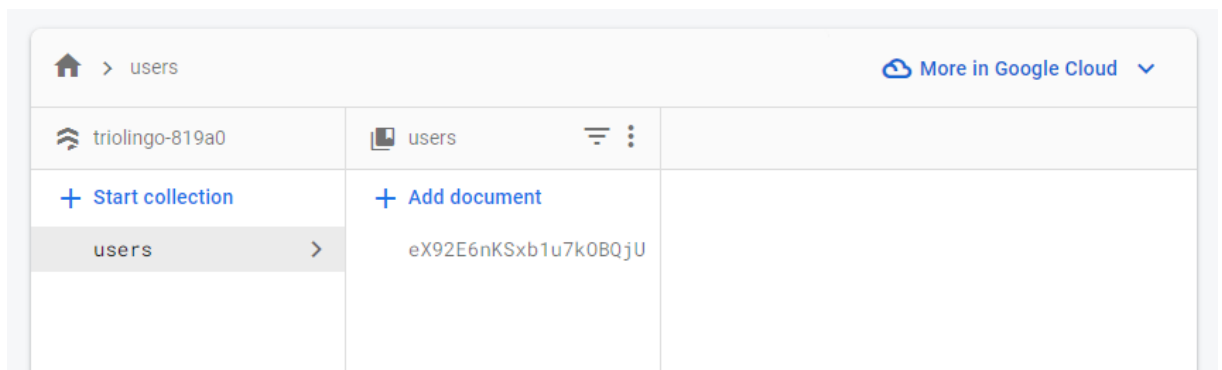
Pe lângă acestea, avem de asemenea șabloane pentru email-urile de verificare a contului și pentru resetarea parolei. Ca și în cazul metodelor de logare, aici ne sunt oferite multe unelte pentru personalizarea mesajului, a numelui expeditorului, a domeniului etc.



Pentru stocarea de fișiere ne vom folosi de spațiul de stocare oferit de către Firebase, care ne oferă posibilitatea de a încărca sau șterge fișiere și de a le asocia cu un utilizator anume cu ușurință. Planul fără cost de la Firebase ne oferă la dispoziția noastră 1GB spațiu de stocare, ceea ce este mai mult decât suficient pentru această lucrare. [10] [11]



Pentru stocarea altor date decât cele oferite de către Firebase pentru utilizatori, în cazul nostru date precum numărul de puncte, limba nativă, limba selectată pentru învățare, folosim baza de date NoSQL Cloud Firestore pusă la dispoziție de către Firebase. [12] [13] [14]



3.2.3 Legătura dintre front și back

Comunicarea dintre Angular și Firebase s-a realizat în principal pentru toate funcționalitățile care țin de gestionarea conturilor utilizatorilor. Asta implică atât setările fiecărui cont, cât și procesul de autentificare și de creare al contului.

Am implementat un serviciu pentru autentificare care a gestionat toate funcționalitățile care țineau de crearea unui cont, de logare și delogare și pentru trimiterea unui mail pentru confirmarea creării unui cont, schimbarea parolei etc.

Pentru crearea unui cont am creat funcția **signUp**, care primește ca și parametri numele, adresa de email și parola, iar apoi creează un cont cu acele date și îl adaugă în baza de date Firebase.

```
signUp(name: string, email: string, password: string) {  
  return from(  
    createUserWithEmailAndPassword(this.auth, email, password)  
  ).pipe(  
    switchMap(({ user }) => {  
      return from(updateProfile(user, { displayName: name })).pipe(  
        switchMap(() => sendEmailVerification(user)),  
        tap(() => {  
          this.logout();  
          this.router.navigate(['/login']);  
        })  
      )  
    })  
  );  
}
```

```
);  
}
```

Funcția **createUserWithEmailAndPassword** loghează un utilizator în mod automat după crearea cu succes a contului, însă noi nu vrem acest lucru. După crearea contului, utilizatorul este automat delogat deoarece trebuie confirmată adresa de email mai întâi.

Pentru logare avem două funcții, **loginWithGoogle** și **login**.

```
loginWithGoogle() {  
  const provider = new GoogleAuthProvider();  
  return from(signInWithPopup(this.auth, provider)).pipe(  
    switchMap((userCredential: UserCredential) => {  
      const user: User = userCredential.user;  
      return of(user);  
    }),  
    tap(() => {  
      this.router.navigate(['/home']);  
    })  
  );  
}
```

Aceasta se folosește de funcția **GoogleAuthProvider** oferită de către Firebase, iar după conectare utilizatorul este trimis direct către pagina principală din aplicație.

```
login(username: string, password: string) {  
  return from(signInWithEmailAndPassword(this.auth, username,  
password)).pipe(  
    switchMap((userCredential: UserCredential) => {  
      const user: User = userCredential.user;  
      if (user.emailVerified) {  
        return of(user);  
      } else {  
        throw new Error('Please verify your email address to log in.');      }  
    })  
  );  
}
```

Funcția **login** se folosește de funcția **signInWithEmailAndPassword** pentru a loga utilizatorul cu username, care în cazul nostru este adresa de email, și parolă. De asemenea, adăugăm o condiție în plus înainte de a loga utilizatorul, acesta fiind logat doar dacă și-a confirmat adresa de email. Altfel primește un mesaj în care este informat să o confirme.

Pe lângă acestea mai avem o funcție pentru trimiterea unui link de resetare al parolei, care va fi folosit atât în cazul în care utilizatorul și-a uitat parola, cât și atunci când acesta își dorește să o schimbe. Funcția se folosește de **sendPasswordResetEmail** oferit de Firebase. Funcția va trimite un email cu un link care odată accesat, oferă posibilitatea de a introduce o nouă parolă.

```
forgotPassword(email: string) {  
  return from(sendPasswordResetEmail(this.auth, email));  
}
```

Am implementat de asemenea și funcția pentru delogare, care se folosește de funcția **signOut**.

```
logout() {  
  return from(this.auth.signOut());  
}
```

Pentru schimbarea pozei de profil din pagina de setări am creat două funcții, **onUploadButtonClick** și **uploadImage**. Prima funcție o apelează pe cea de-a doua pentru ca încărcarea pozei să nu se întâmple automat atunci când utilizatorul se loghează.

```
onUploadButtonClick() {  
  this.user$.pipe(take(1)).subscribe((user) => {  
    if (user) this.uploadImage(user);  
  });  
}
```

A doua funcție afișează poza în pagină și apelează serviciul pentru încărcare al pozei și pe cel pentru actualizarea profilului utilizatorului, care comunica cu Firebase.

```
uploadImage(user: User) {  
  if (user && this.selectedFile) {  
    const imagePath =  
      `images/profile/${user.uid}/${this.selectedFile.name}`;  
    this.imageUploadService  
      .uploadImage(this.selectedFile, imagePath)  
      .pipe(  
        this.toast.observe({  
          loading: 'Uploading image...',  
          success: 'Image uploaded successfully.',  
          error: 'Failed to upload image.',  
        })),  
        concatMap((photoURL) =>
```

```

        this.authService.updateProfileData({ photoURL })
    )
)
.subscribe(() => {
    this.photoURL = this.photoURL;
    console.log('Profile image URL updated successfully.');
```

Serviciul pentru încărcarea pozei conține următoarea funcție, **uploadImage**, care va încărea poza în spațiul de stocare Firebase.

```

uploadImage(image: File, path: string) : Observable<string>{
    const storageRef = ref(this.storage, path);
    const uploadTask = from(uploadBytes(storageRef, image));
    return uploadTask.pipe(
        switchMap((result) => getDownloadURL(result.ref))
    );
}
```

Ultima funcție implementată pentru a asigura funcționalitatea încărcării unei poze de profil este cea din serviciul de autentificare, **updateProfileData**, care va actualiza profilul unui utilizator.

```

updateProfileData(profileData: Partial<UserInfo>): Observable<any>{
    const user = this.auth.currentUser;
    return of(user).pipe(
        concatMap(user => {
            if(!user) throw new Error('Not Authenticated');
```

Metoda se folosește de funcția **updateProfile**, oferită de Firebase, care va actualiza contul unui utilizator, **user**, cu datele oferite, stocate în **profileData**.

Capitolul 4

Prezentarea aplicației

4.1 Generală

Prioritatea aplicației a fost de a crea un mediu prietenos, ușor de navigat, așa că am priorizat o interfață atractivă, cu o paletă de culori caldă și plăcută vederii. De asemenea, am folosit mesaje informative, de tipul pop-up pentru a informa în permanență utilizatorul cu vedere la acțiunile pe care acesta le face.

4.2 Pagina de logare și de creare cont

Pentru a accesa și naviga conținutul aplicației, utilizatorul trebuie mai întâi să își creeze un cont, rutele fiind protejate împotriva utilizatorilor neautentificați. Aceștia pot fie să își creeze un cont folosind o adresă de mail și o parolă, însă contul va fi activat doar odată ce adresa este confirmată. Utilizatorul mai are și opțiunea de a crea un cont folosind un cont deja existent de Google. Tot aici, utilizatorii pot cere un link de resetare a parolei în cazul în care au uitat-o. Interfața este una intuitivă și ușor de navigat.

Login

Email address *

Password *

[Forgot Password?](#)

Login

Login with Google

New to Triolingo? [Sign up!](#)

Sign Up

Name *

Email address *

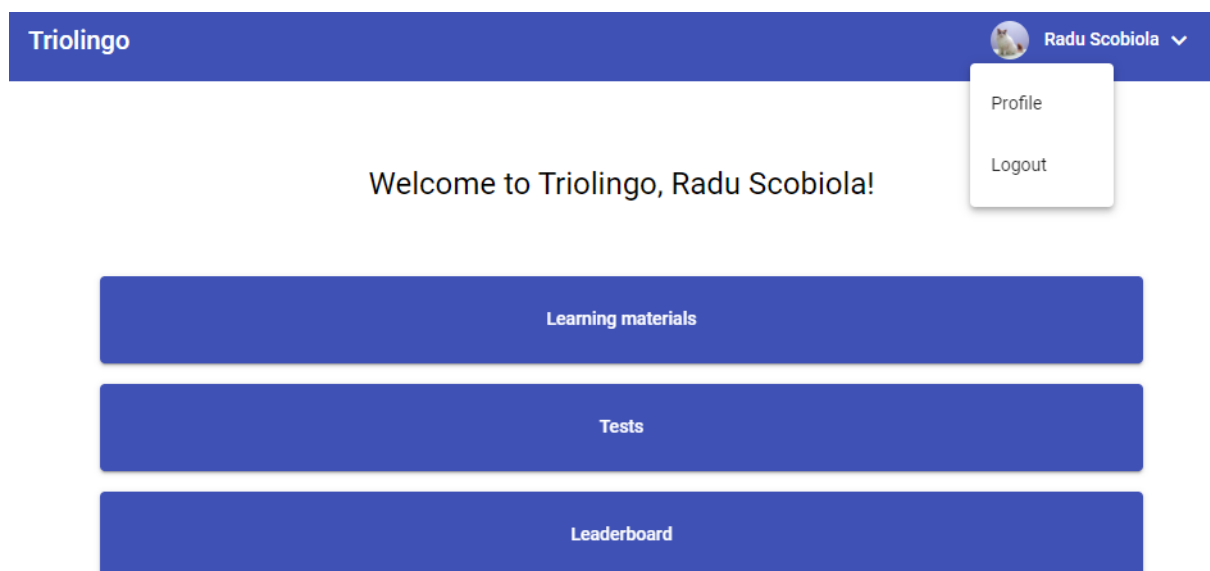
Password *

Confirm Password *

Sign up

4.3 Pagina principală



Pagina principală este cea din care se realizează toată navigarea utilizatorului în cadrul aplicației. Din această cauză am optat pentru un design cât mai minimalist și ușor de urmărit, pentru a nu cauza confuzii utilizatorului. De asemenea, în continuare folosim aceeași paletă de culori, pentru a crea un element de familiaritate. Utilizatorul își va vedea de asemenea poza sa de profil și va avea și un meniu de tipul dropdown pentru a-și vizita profilul sau a se deloga din aplicație.



4.4 Pagina de profil

În această pagină utilizatorul își poate schimba setările contului, sau îl poate personaliza cum dorește acesta. El va avea opțiunea de a își schimba parola, poza de profil, sau de a seta limba sa nativă, precum și pe cea pe care dorește să o învețe. Am folosit elemente de UI familiare și des întâlnite, pentru ca navigarea paginii să fie cât mai intuitivă. De asemenea, aici utilizatorul va putea vedea câte puncte a acumulat pe parcursul testelor.

User Profile



Upload

Change Password

Pick Native Language

English ▼

Save

Pick Language to Learn

▼

Save

Points accumulated: 200

După cum se poate observa, am păstrat o consistență privind paleta de culori și stilul general al UI-ului. De asemenea, în cazurile când am folosit iconițe, ne-am asigurat că folosim iconițe care sunt utilizate în mod convențional, pentru a ne asigura că utilizatorul știe ce reprezintă.

Capitolul 5

Concluzii și perspective

5.1 Concluzii

Prin această lucrare am expus viziunea, rolul și procesul de dezvoltare al unei aplicații de învățare a limbilor străine. Aceasta își propune oferirea unei alternative pentru cei ce își doresc să învețe alte limbi, încercând să îmbine elemente de la mai multe aplicații asemănătoare, fără a le repeta însă greșelile, ce pot produce nemulțumiri utilizatorilor, sau care pot face potențiali utilizatori să evite aplicația.

Deși eram familiar cu Angular, Typescript și Firebase, această lucrare a reprezentat totuși o mare provocare pentru mine, deoarece a fost prima oară când am fost nevoit să realizez de unul singur o aplicație completă. Chiar și cu multele servicii puse la dispoziția mea de către Firebase, tot a fost dificilă crearea comunicării dintre Angular și Firebase. De mare ajutor aici mi-au fost resursele puse la dispoziție de către comunitatea care se ocupă de dezvoltarea atât a lui Angular, cât și a lui Firebase, deoarece am putut găsi cu ușurință documentații ample despre diferite pachete, module, sau funcții. De asemenea, în urma acestui proiect, simt că am dobândit o înțelegere mult mai adâncă asupra acestor tehnologii.

În plus, pe plan personal, învățarea de limbi străine este un domeniu de interes pentru mine. Astfel, prin această lucrare am putut descoperi noi tehnici folosite pentru a ușura procesul de a învăța o limbă nouă, descoperind multe sfaturi de la experții din domeniu. Cred de asemenea că dacă aș continua dezvoltarea aplicației pentru a oferi mai multe resurse, teste și limbi, ar putea fi o alternativă adevărată pentru cei care își doresc o sursă în plus pentru învățare.

Concluzionând, această aplicație a reușit să îndeplinească scopurile pe care mi le-am propus atunci când am început să lucrez la ea. Consider nu doar că am livrat un

produs finit, care își realizează obiectivul, dar și că există oameni care ar putea beneficia în mod real de aplicația realizată.

5.2 Îmbunătățiri viitoare

Deși mă consider mulțumit de funcționalitățile pe care aplicația le are în momentul de față, consider că există câteva îmbunătățiri pe care mi le-aș dori să le implementez înainte de a face aplicația disponibilă pentru public. Acestea au fost identificate ca fiind următoarele:

- Oferirea mai multor resurse pentru învățare. Consider că în momentul de față numărul de resurse pus la dispoziție este unul suficient pentru un nivel de bază de cunoaștere, însă nu sunt suficiente resurse pentru cei care își doresc să dobândească cunoștințe mai avansate.
- Mai multe categorii de teste. Consider că ar putea fi dezvoltate mai multe categorii de cunoștințe din care utilizatorul să fie testat, pentru a îi evalua nivelul de cunoștințe într-un mod mai precis.
- Pe lângă tabela care arată punctele utilizatorilor ar putea fi realizate și alte competiții, activități pentru a motiva utilizatorii să învețe cât mai mult. Deși tabela este o sursă bună de motivație, sunt de părere că nu este în totalitate suficientă.

Prin cele de mai sus am dorit să subliniez faptul că personal consider că aplicația are un bun potențial pentru a atrage utilizatori și a-i ajuta în învățarea unor limbi străine, însă în același timp mai sunt aspecte ale sale care pot fi îmbunătățite. Cum am mai menționat, acestea ar trebui adresate înainte de lansarea aplicației către public.

Bibliografie

- [1] „Trustpilot,” [www.trustpilot.com](https://www.trustpilot.com/review/duolingo.com), <https://www.trustpilot.com/review/duolingo.com>. Accesat la 5 Iunie 2023.
- [2] „GetApp,” [www.getapp.com](https://www.getapp.com/all-software/a/babbel/reviews/), <https://www.getapp.com/all-software/a/babbel/reviews/>. Accesat la 5 iunie 2023.
- [3] Fabio Alessandro Locati - OpenStack Cloud Security-Packt Publishing.
- [4] Denton, J. (2018). Learning Openstack Networking - Third Edition. Packt.
- [5] Diego Kreutz, M. I. (2014, Oct 8). Software-Defined Networking: A Comprehensive Survey.
- [6] „Angular”, [www.angular.io](https://angular.io/docs), <https://angular.io/docs>. Accesat la 8 iunie 2023.
- [7] „Github”, [www.github.com](https://github.com/angular/angularfire), <https://github.com/angular/angularfire>. Accesat la 12 iunie 2023.
- [8] E. L. Axenova, V. S. (2017). Openstack Keystone identification service drop-in replacement.
- [9] „Firebase”, [firebase.google.com](https://firebase.google.com/docs/auth), <https://firebase.google.com/docs/auth>. Accesat la 13 iunie 2023.
- [10] „Firebase”, [firebase.google.com](https://firebase.google.com/docs/storage), <https://firebase.google.com/docs/storage>. Accesat 15 iunie 2023.
- [11] Markelov, A. (2016). "Openstack Dashboard" . In Certified Openstack Administrator Study Guide (pp. 87-89). Apress.
- [12] „Firebase”, [firebase.google.com](https://firebase.google.com/docs/firestore), <https://firebase.google.com/docs/firestore>. Accesat la 15 iunie 2023.
- [13] Kevin Jackson, C. B. (2018). Openstack Cloud Computing Cookbook - fourth edition. Packt.
- [14] Markelov, A. (2016). Openstack Compute. In Certified Openstack Administrator Study Guide (pp. 65-86). Apress.