|  |  |  |
| --- | --- | --- |
|  | UNIVERSITATEA DIN CRAIOVA  FACULTATEA DE AUTOMATICĂ, CALCULATOARE ȘI ELECTRONICĂ  DEPARTAMENTUL DE CALCULATOARE ȘI TEHNOLOGIA INFORMAȚIEI |  |

PROIECT DE DIPLOMĂ

*Drăghici Bogdan*

COORDONATOR ȘTIINȚIFIC

*Şef lucr. dr. ing. Cătălin-Constantin Cerbulescu*

*IULIE 2022*

CRAIOVA

|  |  |  |
| --- | --- | --- |
|  | UNIVERSITATEA DIN CRAIOVA  FACULTATEA DE AUTOMATICĂ, CALCULATOARE ȘI ELECTRONICĂ  DEPARTAMENTUL DE CALCULATOARE ȘI TEHNOLOGIA INFORMAȚIEI |  |

*PLATFORMĂ WEB PENTRU GESTIONAREA ARBORELUI GENEALOGIC*

*Drăghici Bogdan*

COORDONATOR ȘTIINȚIFIC

*Şef lucr. dr. Ing. Cătălin-Constantin Cerbulescu*

*IULIE 2022*

CRAIOVA

*„Învățătura este o comoară care își urmează stăpânul pretutindeni.”*

Proverb popular

**DECLARAȚIE DE ORIGINALITATE**

Subsemnatul *DRĂGHICI BOGDAN*, student la specializarea *CALCULATOARE ROMÂNĂ* din cadrul Facultății de Automatică, Calculatoare și Electronică a Universității din Craiova, certific prin prezenta că am luat la cunoştinţă de cele prezentate mai jos şi că îmi asum, în acest context, originalitatea proiectului meu de licenţă:

* cu titlul *PLATFORMĂ WEB PENTRU GESTIONAREA ARBORELUI GENEALOGIC*,
* coordonată de  *Şef lucr. dr. Ing. Cătălin-Constantin Cerbulescu*,
* prezentată în sesiunea *IULIE 2022*.

La elaborarea proiectului de licenţă, se consideră plagiat una dintre următoarele acţiuni:

* reproducerea exactă a cuvintelor unui alt autor, dintr-o altă lucrare, în limba română sau prin traducere dintr-o altă limbă, dacă se omit ghilimele şi referinţa precisă,
* redarea cu alte cuvinte, reformularea prin cuvinte proprii sau rezumarea ideilor din alte lucrări, dacă nu se indică sursa bibliografică,
* prezentarea unor date experimentale obţinute sau a unor aplicaţii realizate de alţi autori fără menţionarea corectă a acestor surse,
* însuşirea totală sau parţială a unei lucrări în care regulile de mai sus sunt respectate, dar care are alt autor.

Pentru evitarea acestor situaţii neplăcute se recomandă:

* plasarea între ghilimele a citatelor directe şi indicarea referinţei într-o listă corespunzătoare la sfărşitul lucrării,
* indicarea în text a reformulării unei idei, opinii sau teorii şi corespunzător în lista de referinţe a sursei originale de la care s-a făcut preluarea,
* precizarea sursei de la care s-au preluat date experimentale, descrieri tehnice, figuri, imagini, statistici, tabele et caetera,
* precizarea referinţelor poate fi omisă dacă se folosesc informaţii sau teorii arhicunoscute, a căror paternitate este unanim cunoscută și acceptată.

Data, Semnătura candidatului,

|  |  |  |
| --- | --- | --- |
|  | UNIVERSITATEA DIN CRAIOVA  Facultatea de Automatică, Calculatoare şi Electronică  Departamentul de Calculatoare și Tehnologia Informației | Aprobat la data de  …………………  Şef de departament,  Prof. dr. ing.  Marius BREZOVAN |

**PROIECTUL DE DIPLOMĂ**

|  |  |
| --- | --- |
| Numele și prenumele studentului/-ei: | Drăghici Bogdan |
| Enunțul temei: | Platformă web pentru gestionarea arborelui genealogic |
| Datele de pornire: | Realizarea a două soluţii pentru implementarea aplicaţiei de gestiune a arborelui genealogic: o parte pentru modificarea şi stocarea datelor despre persoanele înrudite cu utilizatorii, şi o parte utilizată pentru vizualizarea sub formă de arbore şi interacţiunea cu utilizatorul. |
| Conținutul proiectului: | Primul capitol reprezentat de introducere conţine motivaţia realizării acestui proiect, cât şi scopul acestuia cu o descriere generală a aplicaţiei.  Capitolul al doilea conţine atât fundamentele teoretice şi tehnice folosite pentru dezvoltarea aplicaţiei.  Al treilea capitol include descrierea detaliată a proiectului pentru gestionarea arborelui genealogic, intitulat GeneTree, şi informaţii despre design şi implementarea acestuia.  Capitolul 4 este reprezentat de concluziile obţinute, care oferă informaţii despre realizări, limitări şi viitoare direcţii de dezvoltare ale aplicaţiei.  Ultimul capitol, capitolul 5, conţine informaţii bibliografice şi referinţe web folosite în procesul de dezvoltare al aplicaţiei. |
| Material grafic obligatoriu: | Diagrame, grafice, poze, documentaţie, cod sursă, prezentare Power Point |
| Consultații: | *Lunare* |
| Conducătorul științific  (titlul, nume și prenume, semnătura): | Șef lucrări dr. ing. Cătălin-Constantin CERBULESCU |
| Data eliberării temei: | 15.10.2021 |
| Termenul estimat de predare a proiectului: | 28.06.2022 |
| Data predării proiectului de către student și semnătura acestuia: |  |

|  |  |  |
| --- | --- | --- |
|  | UNIVERSITATEA DIN CRAIOVA  Facultatea de Automatică, Calculatoare şi Electronică  Departamentul de Calculatoare și Tehnologia Informației |  |

**REFERATUL CONDUCĂTORULUI ȘTIINȚIFIC**

|  |  |
| --- | --- |
| Numele și prenumele candidatului/-ei: | Drăghici Bogdan |
| Specializarea: | Calculatoare română |
| Titlul proiectului: | Platformă web pentru gestionarea arborelui genealogic |
| Locația în care s-a realizat practica de documentare (se bifează una sau mai multe din opțiunile din dreapta): | În facultate □ |
| În producție □ |
| În cercetare □ |
| Altă locație: [*se detaliază*] |

În urma analizei lucrării candidatului au fost constatate următoarele:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nivelul documentării | | Insuficient  □ | Satisfăcător □ | Bine  □ | Foarte bine  □ |
| Tipul proiectului | | Cercetare  □ | Proiectare  □ | Realizare practică □ | Altul  [*se detaliază*] |
| Aparatul matematic utilizat | | Simplu  □ | Mediu  □ | Complex □ | Absent  □ |
| Utilitate | | Contract de cercetare □ | Cercetare internă □ | Utilare  □ | Altul  [*se detaliază*] |
| Redactarea lucrării | | Insuficient  □ | Satisfăcător □ | Bine  □ | Foarte bine  □ |
| Partea grafică, desene | | Insuficientă  □ | Satisfăcătoare □ | Bună  □ | Foarte bună  □ |
| Realizarea practică | Contribuția autorului | Insuficientă  □ | Satisfăcătoare □ | Mare  □ | Foarte mare  □ |
| Complexitatea  temei | Simplă  □ | Medie  □ | Mare  □ | Complexă  □ |
| Analiza cerințelor | Insuficient  □ | Satisfăcător □ | Bine  □ | Foarte bine  □ |
| Arhitectura | Simplă  □ | Medie  □ | Mare  □ | Complexă  □ |
| Întocmirea specificațiilor funcționale | Insuficientă  □ | Satisfăcătoare □ | Bună  □ | Foarte bună  □ |
| Implementarea | Insuficientă  □ | Satisfăcătoare □ | Bună  □ | Foarte bună  □ |
| Testarea | Insuficientă  □ | Satisfăcătoare □ | Bună  □ | Foarte bună  □ |
| Funcționarea | Da  □ | Parțială  □ | Nu  □ | |
| Rezultate experimentale | | Experiment propriu  □ | | Preluare din bibliografie  □ | |
| Bibliografie | | Cărți | Reviste | Articole | Referințe web |
| Comentarii  și  observații | |  | | | |

În concluzie, se propune:

|  |  |
| --- | --- |
| ADMITEREA PROIECTULUI  □ | RESPINGEREA PROIECTULUI  □ |

Data, Semnătura conducătorului științific,

**REZUMATUL PROIECTULUI**

Proiectul de gestionare a arborelui genealogic, a cărei implementare a fost ulterior intitulată GeneTree, are ca scop să ofere utilizatorilor positibilitatea să îşi creeze propriul arbore genealogic prin adăugarea, editarea sau chiar ştergerea de informaţii despre persoanele înrudite.

Pentru a putea reprezenta structura arborelui genealogic, sunt stocate şi informaţii despre legăturile dintre persoane, care pot fi de 2 feluri: de tip părinte-copil, şi de tip soţ-soţie.

Fiecare utilizator al platformei are atribuit câte un singur arbore genealogic, pe care doar acesta îl poate modifica şi care iniţial conţine informaţii doar despre utilizatorul în cauză.

Toate informaţiile care sunt legate de utilizator şi de arborele său genealogic sunt stocate într-un mediu securizat, într-o bază de date, care poate fi accesată numai pe baza unei autentificări a utilizatorului.

Profilul utilizatorului va putea fi văzut de orice alt utilizator al aplicaţiei, cu excepţia a informaţiilor precum numărul de telefon şi e-mail-ul, care poate fi accesat doar în anumite situaţii. Accesul la informaţiile din arborele genealogic al unui utilizator şi vizualizarea sub formă arborescentă va putea fi realizat numai de către utilizatorul care are atribuit arborele genealogic respectiv, cât şi de către utilizatorii pe care acesta i-a recunoscut ca fiind rudele sale.

Motivaţia pentru realizarea acestei teme de licenţă este reprezentată atât de dorinţa de a păstra drept amintire cât mai multe informaţii despre rude, mai ales strămoşi, cât şi cu scopul de a facilita descoperirea de rude noi despre care putea afla cu greu.

Pentru realizarea proiectului sunt 2 soluţii: o soluţie pentru partea de backend şi o soluţie pentru partea de frontend.

Pentru partea de backend a fost utilizat limbajul de programare C#, cu tehnologia ASP .NET 5, folosind abordarea code-first pentru realizarea structurii bazei de date de tip SQL. Serverul folosit pentru baza de date este de tip SQL, şi anume MySQL. Pentru managerierea bazei de date s-a utilizat mediul integrat pentru gestiunea infrastructurii SQL Server denumit Microsoft SQL Server Management Studio (SSMS).

Pentru partea de frontend au fost utilizate limbajele TypeScript, JavaScript, HTML şi SCSS, care au fost utilizate în cadrul platformei de dezvoltare Angular, asupra căreia a fost adăugată şi platforma Ionic, având ca scop dezvoltarea în direcţia obţinerii unei aplicaţii receptive, multiplatformă, care să poată fi accesată atât de pe web dintr-un motor de navigare, cât şi printr-o aplicaţie de telefon.

***Termenii cheie***: Arbore genealogic, persoană, utilizator, rude, C#, ASP .NET, MySQL, SSMS, TypeScript, JavaScript, HTML, SCSS, multiplatformă, autentificare, .

**MULȚUMIRI**

Doresc a mulţumi conducătorului ştiinţific Șef lucrări dr. ing. Cătălin-Constantin Cerbulescu, pentru tot ajutorul acordat în dezvoltarea acestui proiect, cât şi pentru buna înţelegere pe care am avut-o alături de acesta. Totodată doresc să le mulţumesc tuturor celor care mi-au oferit părerile lor constructive despre acest proiect şi care mă sprijină în vederea dezvoltării lui ulterioare.

**CUPRINSUL**

[1 Introducere 1](#_Toc106753057)

[1.1 Scopul 1](#_Toc106753058)

[1.2 Motivația 1](#_Toc106753059)

[2 Fundamente teoretice şi tehnice 1](#_Toc106753060)

[2.1 Tehnologii utilizate 1](#_Toc106753061)

[2.1.1 Limbajul C# 1](#_Toc106753062)

[2.1.2 Platforma de dezvoltare .NET 2](#_Toc106753063)

[2.1.3 Entity Framework Core 2](#_Toc106753064)

[2.1.4 Limbajul de interogare structurat – MySQL 2](#_Toc106753065)

[2.1.5 Platforma Angular 3](#_Toc106753066)

[2.1.6 Cadrul Ionic 3](#_Toc106753067)

[2.1.7 Limbajul TypeScript 4](#_Toc106753068)

[2.1.8 Limbajul JavaScript 4](#_Toc106753069)

[2.1.9 HTML (HyperText Markup Language) 4](#_Toc106753070)

[2.1.10 SASS/SCSS (Syntactically Awesome Style Sheets) 4](#_Toc106753071)

[2.1.11 Sistem de versionare Git 5](#_Toc106753072)

[2.2 Şabloane folosite 5](#_Toc106753073)

[2.2.1 Repository 5](#_Toc106753074)

[2.2.2 Unit of Work 5](#_Toc106753075)

[2.2.3 Dependency Injection 6](#_Toc106753076)

[2.2.4 Singleton 6](#_Toc106753077)

[2.3 Pachete NuGet folosite 6](#_Toc106753078)

[2.3.1 AutoMapper 6](#_Toc106753079)

[2.3.2 AspNetCore 7](#_Toc106753080)

[2.3.3 EntityFrameworkCore 7](#_Toc106753081)

[2.3.4 Swashbuckle 8](#_Toc106753082)

[2.4 Pachete NPM folosite 8](#_Toc106753083)

[2.4.1 Auth0/angular-jwt 8](#_Toc106753084)

[2.4.2 Balkangraph/familytree.js 9](#_Toc106753085)

[2.4.3 Ionic/angular 10](#_Toc106753086)

[2.4.4 Ionic/storage 10](#_Toc106753087)

[2.4.5 Ionic2-calendar 11](#_Toc106753088)

[2.4.6 Leaflet 11](#_Toc106753089)

[2.4.7 Ngx-image-compress 12](#_Toc106753090)

[2.4.8 Ngx-translate 12](#_Toc106753091)

[2.4.9 Rxjs 13](#_Toc106753092)

[2.4.10 Swiper 14](#_Toc106753093)

[2.5 Medii de dezvoltare 14](#_Toc106753094)

[2.5.1 SQL Server Management Studio (SSMS) 14](#_Toc106753095)

[2.5.2 Visual Studio 2019 14](#_Toc106753096)

[2.5.3 Visual Studio Code 15](#_Toc106753097)

[3 Sistemul de gestiune al arborelui genealogic 15](#_Toc106753098)

[3.1 Cerinţe 15](#_Toc106753099)

[3.2 Cazuri de utilizare 16](#_Toc106753100)

[3.3 Structura bazei de date 17](#_Toc106753101)

[3.4 Arhitectura soluţiei backend 19](#_Toc106753102)

[3.4.1 Proiectul API 19](#_Toc106753103)

[3.4.2 Proiectul Business 21](#_Toc106753104)

[3.4.3 Proiectul Data 23](#_Toc106753105)

[3.4.4 Proiectul Domain 26](#_Toc106753106)

[3.5 Arhitectura soluţiei frontend 27](#_Toc106753107)

[3.5.1 Modulul shared 29](#_Toc106753108)

[3.5.2 Modulul core 30](#_Toc106753109)

[3.5.3 Modulul person 35](#_Toc106753110)

[3.5.4 Modulul user 35](#_Toc106753111)

[3.5.5 Modulul genealogy 35](#_Toc106753112)

[3.5.6 Modulul relative 36](#_Toc106753113)

[3.5.7 Modulul notifications 36](#_Toc106753114)

[3.5.8 Modulul settings 36](#_Toc106753115)

[3.5.9 Modulul support 36](#_Toc106753116)

[3.5.10 Modulul intro 36](#_Toc106753117)

[4 Funcţionalităţi ale aplicaţiei 36](#_Toc106753118)

[4.1 Funcţionalităţi Persoană 36](#_Toc106753119)

[4.1.1 Vizualizare detalii persoană 36](#_Toc106753120)

[4.1.2 Adăugare persoană 37](#_Toc106753121)

[4.1.3 Editare persoană 37](#_Toc106753122)

[4.1.4 Ştergere persoană 38](#_Toc106753123)

[4.1.5 Adăugare şi schimbare poză persoană 38](#_Toc106753124)

[4.2 Funcţionalităţi Legături persoane 39](#_Toc106753125)

[4.2.1 Vizualizare listă căsătorii 39](#_Toc106753126)

[4.2.2 Adăugare căsătorie 39](#_Toc106753127)

[4.2.3 Modificare căsătorie 40](#_Toc106753128)

[4.2.4 Ştergere căsătorie 41](#_Toc106753129)

[4.2.5 Vizualizare listă copii 41](#_Toc106753130)

[4.2.6 Vizualizare listă părinţi 41](#_Toc106753131)

[4.2.7 Adăugare părinte sau copil 42](#_Toc106753132)

[4.2.8 Ştergere părinte sau copil 44](#_Toc106753133)

[4.3 Funcţionalităţi Arbore 44](#_Toc106753134)

[4.3.1 Vizualizare listă persoane din arbore 44](#_Toc106753135)

[4.3.2 Vizualizare arbore genealogic 45](#_Toc106753136)

[4.3.3 Vizualizare evenimente din arbore 47](#_Toc106753137)

[4.4 Funcţionalităţi Utilizator 48](#_Toc106753138)

[4.4.1 Înregistrare 48](#_Toc106753139)

[4.4.2 Autentificare 49](#_Toc106753140)

[4.4.3 Schimbare parolă 49](#_Toc106753141)

[4.4.4 Deconectare 50](#_Toc106753142)

[4.4.5 Vizualizare profil utilizator 50](#_Toc106753143)

[4.4.6 Editare profil utilizator 50](#_Toc106753144)

[4.5 Funcţionalităţi Ocupaţie 51](#_Toc106753145)

[4.5.1 Vizualizare listă ocupaţie 51](#_Toc106753146)

[4.5.2 Adăugare ocupaţie 51](#_Toc106753147)

[4.5.3 Editare ocupaţie 51](#_Toc106753148)

[4.5.4 Ştergere ocupaţie 52](#_Toc106753149)

[4.6 Funcţionalităţi Educaţie 52](#_Toc106753150)

[4.6.1 Vizualizare listă educaţii 52](#_Toc106753151)

[4.6.2 Adăugare educaţie 52](#_Toc106753152)

[4.6.3 Editare educaţie 53](#_Toc106753153)

[4.6.4 Ştergere educaţie 53](#_Toc106753154)

[4.7 Funcţionalităţi Rude 53](#_Toc106753155)

[4.7.1 Căutare utilizatori 53](#_Toc106753156)

[4.7.2 Cerere rudenie 54](#_Toc106753157)

[4.7.3 Răspuns cerere de rudenie 55](#_Toc106753158)

[4.7.4 Vizualizare listă de rude 56](#_Toc106753159)

[4.7.5 Localizare rude 56](#_Toc106753160)

[4.8 Aparenţă şi accesibilitate 57](#_Toc106753161)

[4.8.1 Editare setări preferenţiale 57](#_Toc106753162)

[4.8.2 Editare aspect şi accesibilitate 57](#_Toc106753163)

[4.8.3 Selectare limbă 58](#_Toc106753164)

[4.8.4 Notificări 58](#_Toc106753165)

[4.8.5 Suport prin email 59](#_Toc106753166)

[4.8.6 Pagina de introducere 59](#_Toc106753167)

[5 Concluzii 60](#_Toc106753168)

[6 Bibliografie 61](#_Toc106753169)

[7 Referințe web 62](#_Toc106753170)

[A. Codul sursă 63](#_Toc106753171)

[B. Site-ul web al proiectului 63](#_Toc106753172)

[C. CD / DVD 63](#_Toc106753173)

[Index 64](#_Toc106753174)

**LISTA FIGURILOR**

[Figura 1. Actualizarea întregului tabel 63](#_Toc106752496)

**LISTA TABELELOR**

[Tabelul 1. Nume de utilizatori și valorile rezumat ale parolelor acestora 5](#_Toc309893145)

# Introducere

## Scopul

Proiectul acesta a fost conceput cu scopul de a permite oricărei persoane să îşi poată stoca şi gestiona şi vizualiza grafic, sub formă de arbore genealogic, informaţiile despre rudele sale, cât şi posibilitatea de a găsi rude noi în aplicaţie, care în urma unei cereri să le poată vedea arborele lor genealogic, cu ţelul de a extinde cunoştintele despre persoanele cu care este înrudită.

## Motivația

Motivaţia pentru realizarea acestei teme a venit în urma dorinţei personale de a afla şi a păstra cât mai multe despre rude, mai ales despre strămoşi, drept amintire. Există multe alte persoane care doresc să afle informaţii despre originile lor, strămoşi şi rude, precum şi modul în care acestea pot lua legătura cu ele. Consider că acest proiect va ajuta un un cerc relativ larg de persoane să obţină informaţii despre rudele lor şi înrădăcineze conexiuni cu altele noi.

# Fundamente teoretice şi tehnice

## Tehnologii utilizate

### Limbajul C#

C# este un limbaj de programare orientat pe obiecte, orientat pe componente, care oferă construcții de limbaj pentru a sprijini direct aceste concepte, făcându-l un limbaj natural în care să creeze și să utilizeze componente software. Mai multe caracteristici C# ajută la crearea de aplicații robuste și durabile. Colectarea gunoiului recuperează automat memoria ocupată de obiecte neutilizate inaccesibile. Tipurile nullabile protejează împotriva variabilelor care nu se referă la obiectele alocate. Gestionarea excepțiilor oferă o abordare structurată și extensibilă pentru detectarea și recuperarea erorilor. Expresiile Lambda acceptă tehnici de programare funcțională. Sintaxa Language Integrated Query (LINQ) creează un model comun pentru lucrul cu date din orice sursă. Suportul de limbaj pentru operațiuni asincrone oferă sintaxă pentru construirea de sisteme distribuite. Are de asemenea un sistem de tip unificat, în care toate tipurile C#, inclusiv tipurile primitive, cum ar fi int și double, moștenesc de la un singur tip de obiect rădăcină şi au în comun un set de operațiuni comune. Valorile de orice tip pot fi stocate, transportate și operate într-o manieră consecventă. În plus, acceptă atât tipurile de referință definite de utilizator, cât și tipurile de valori, permite alocarea dinamică a obiectelor și stocarea în linie a structurilor ușoare.

### Platforma de dezvoltare .NET

.NET este o platformă de dezvoltare gratuită, open-source, având licențe MIT și Apache 2, pentru construirea multor tipuri de aplicații. Partajați funcționalitatea între diferite aplicații și tipuri de aplicații folosind biblioteci de clase.

Cu .NET, codul și fișierele de proiect arată și se simt la fel, indiferent de tipul de aplicație pe care este construit. Permite acces în timpul de rulare, la API și capacități de limbă cu fiecare aplicație.

Se pot crea aplicații .NET pentru multe sisteme de operare, precum: Windows, macOS, Linux, Android, iOS, tvOS, watchOS. Arhitecturile de procesoare acceptate includ: x64, x86, ARM32, ARM64. .NET permite să utilizarea de capabilități specifice platformei, cum ar fi API-urile sistemului de operare.

.NET vine în diferite variante, cunoscute mai formal ca implementări. .NET 5+ (inclusiv .NET Core) este cea mai recentă implementare și rulează pe orice platformă. .NET Framework este implementarea originală a .NET și rulează numai pe Windows.

Fiecare implementare include un runtime și o bibliotecă de clase. Poate include, de asemenea, cadre de aplicație și instrumente de dezvoltare.

### Entity Framework Core

Entity Framework (EF) Core este o versiune ușoară, extensibilă, open source și multiplatformă a tehnologiei populare de acces la date Entity Framework. EF Core poate servi ca un mapper obiect-relațional (O/RM), care permite dezvoltatorilor .NET să lucreze cu o bază de date folosind obiecte .NET, elimină necesitatea majorității codului de acces la date care de obicei trebuie scris şi acceptă multe motoare de baze de date.

Cu EF Core, accesul la date se realizează folosind un model. Un model este alcătuit din clase de entități și un obiect context care reprezintă o sesiune cu baza de date. Obiectul context permite interogarea și salvarea datelor.

Pentru realizarea acestui proiect s-a codat manual un model pentru a se potrivi cu baza de date, care a fost utilizat pentru a crea o bază de date din model folosind migrări care permit evoluția acesteia pe măsură ce modelul se schimbă.

### Limbajul de interogare structurat – MySQL

MySQL este un sistem open-source, sub licenţa generală GNU, de gestionare a bazelor de date relaționale. Numele său este o combinație între „My”, numele fiicei co-fondatorului Michael Widenius și „SQL”, abrevierea pentru Structured Query Language. Reprezintă o bază de date relațională care organizează datele în unul sau mai multe tabele de date în care datele pot fi legate între ele prin relaţii, precum chei externe, care ajută la structurarea datelor. SQL este un limbaj folosit de programatori pentru a crea, modifica și extrage date din baza de date relațională, precum și pentru a controla accesul utilizatorilor la baza de date. MySQL funcționează cu un sistem de operare pentru a implementa o bază de date relațională în sistemul de stocare al unui computer, gestionează utilizatorii, permite accesul la rețea și facilitează testarea integrității bazei de date și crearea de copii de siguranță.

### Platforma Angular

Angular este o platformă de dezvoltare, construită pe TypeScript. Ca platformă, Angular include un cadru bazat pe componente pentru construirea de aplicații web scalabile, o colecție de biblioteci bine integrate care acoperă o mare varietate de caracteristici, inclusiv rutare, gestionare a formularelor, comunicare client-server și multe altele şi o suită de instrumente pentru dezvoltatori care ajută la dezvoltarea, construirea, testarea și actualizarea codului.

Angular e o platformă care se poate scala de la proiecte cu un singur dezvoltator la aplicații la nivel de întreprindere. Angular transformă șabloanele în cod foarte optimizat pentru mașinile virtuale JavaScript de astăzi, oferind toate beneficiile codului scris de mână cu productivitatea unui cadru, facilitând actualizarea cât mai simplă posibilă. Este special conceput pentru vizualizarea aplicațiilor pe Node.js®, .NET, PHP și alte servere pentru randare aproape instantanee doar în HTML și CSS.

Aplicațiile dezvoltate utilizând Angular se încarcă rapid şi oferă divizarea automată a codului, astfel încât utilizatorii să încarce doar codul necesar pentru a reda vizualizarea pe care o solicită. Creează vizualizări ale interfeței de utilizare cu sintaxă de șablon simplă și puternică. Oferă completare inteligentă a codului, erori instantanee și alte feedback în editorii și IDE-urile populare.

### Cadrul Ionic

Ionic este un set de instrumente UI open source pentru construirea de aplicații mobile și desktop performante, de înaltă calitate, folosind tehnologii web standardizate (HTML, CSS și JavaScript), şi API-uri web moderne, cum ar fi Custom Elements și Shadow DOM.

Ionic se concentrează pe interacțiunea UX și UI din frontend a unei aplicații - controale UI, interacțiuni, gesturi, animații. Se integrează cu alte biblioteci sau cadre, cum ar fi Angular, React sau Vue. Alternativ, poate fi folosit de sine stătător, fără nici un cadru frontend.

Ionic este singura stivă de aplicații mobile care le permite dezvoltatorilor web să creeze aplicații pentru toate magazinele majore de aplicații și web-ul mobil dintr-o singură bază de cod. Și cu stilul adaptiv, aplicațiile Ionic arată și se simt natural pe fiecare dispozitiv. Ionic folosește Capacitor (sau Cordova) pentru a se implementa nativ sau rulează în browser ca o aplicație web progresivă.

Ionic este conceput pentru a funcționa și a se comporta grozav pe toate dispozitivele și platformele mobile actuale, cu cele mai bune practici, cum ar fi tranziții eficiente accelerate de hardware și gesturi optimizate pentru atingere. Cu componente gata făcute, tipografie și o temă de bază superbă (dar extensibilă) care se adaptează fiecărei platforme.

### Limbajul TypeScript

TypeScript este un limbaj de programare dezvoltat și întreținut de Microsoft. Este un superset sintactic strict de JavaScript care adaugă tastare statică opțională limbajului. Este conceput pentru dezvoltarea de aplicații mari și transpilează în JavaScript. Deoarece este un superset de JavaScript, programele JavaScript existente sunt, de asemenea, programe TypeScript valide.

TypeScript poate fi folosit pentru a dezvolta aplicații JavaScript atât pentru execuție pe partea client, cât și pe partea server (ca și în cazul Node.js sau Deno). Sunt disponibile mai multe opțiuni pentru transpilare. Compilatorul implicit TypeScript poate fi utilizat sau compilatorul Babel poate fi invocat pentru a converti TypeScript în JavaScript.

### Limbajul JavaScript

JavaScript (JS) este un limbaj de programare ușor, interpretat sau compilat la timp, cu funcții de primă clasă. Deși este cel mai bine cunoscut ca limbaj de scripting pentru pagini web, multe medii non-browser îl folosesc și ele, cum ar fi Node.js, Apache CouchDB și Adobe Acrobat. JavaScript este un limbaj dinamic, bazat pe prototipuri, multi-paradigma, cu un singur thread, care acceptă stiluri orientate pe obiecte, imperative și declarative (de exemplu, programare funcțională).

### HTML (HyperText Markup Language)

HTML (HyperText Markup Language) este un limbaj de marcare care defineşte a structura unei pagini web și conținutul acesteia. HTML constă dintr-o serie de elemente, utilizate pentru a încadra sau încheia diferite părți ale conținutului pentru a face ca acesta să apară sau să acționeze într-un anumit fel. Etichetele care includ un cuvânt sau o imagine pot crea un hyperlink către altă parte, pot pune cuvinte în cursiv, pot face fontul mai mare sau mai mic și așa mai departe.

### SASS/SCSS (Syntactically Awesome Style Sheets)

CSS (Cascading Style Sheet) este un limbaj de scripting folosit pentru proiectarea paginilor web, fiind printre cele mai importante tehnologii web utilizate pe scară largă, împreună cu HTML și JavaScript.

SASS este cel mai matur, stabil și puternic limbaj de extensie CSS de grad profesional din lume. Este complet compatibil cu toate versiunile de CSS, astfel încât să poată fi utilizate fără probleme orice biblioteci CSS disponibile. Are mai multe caracteristici și abilități decât orice alt limbaj de extensie CSS existent, precum variabilele care ajută la scurtarea codului scris.

### Sistem de versionare Git

Git este un sistem de control al versiunilor distribuit gratuit și open source, conceput pentru a gestiona totul, de la proiecte mici la proiecte foarte mari, cu viteză și eficiență.

Git este ușor de învățat și are o amprentă mică, cu performanțe fulgerătoare. Deţiner funcții precum ramificare locală ieftină, zone de pregătire convenabile și fluxuri de lucru multiple.

## Şabloane folosite

### Repository

Şablonul de proiectare Repository acționează ca un intermediar sau un strat de mijloc între restul aplicației și logica de acces la date. Aceasta înseamnă că un model de depozit izolează tot codul de acces la date de restul aplicației. Funcționează cu entitățile de domeniu și realizează logica de acces la date. În modelul Repository, entitățile de domeniu, logica de acces la date și logica de afaceri vorbesc între ele folosind interfețe.

Realizează centralizarea logicii de acces la date, astfel încât mentenanța codului este mai ușoară, deoarece modificările vor fi făcute într-un singur loc. Un alt beneficiu este că testarea controlerelor devine ușoară, deoarece cadrul de testare nu trebuie să ruleze împotriva codului de acces la baza de date reală. Separă baza de date reală, interogările și altă logică de acces la date de restul aplicației. Logica de afaceri poate accesa obiectul de date fără a avea cunoștințe despre arhitectura de acces la date de bază.

### Unit of Work

Şablonul Unitate de lucru este folosit pentru a grupa una sau mai multe operațiuni (de obicei operațiuni CRUD din baza de date) într-o singură tranzacție sau „unitate de lucru”, astfel încât toate operațiunile fie să treacă, fie să eșueze ca o singură unitate. Înseamnă că pentru o anumită acțiune a utilizatorului, toate tranzacțiile precum inserarea/actualizarea/ștergerea și așa mai departe sunt efectuate într-o singură tranzacție, mai degrabă decât efectuarea mai multor tranzacții în baza de date. Aceasta înseamnă că o unitate de lucru implică operațiuni de inserare/actualizare/ștergere, toate într-o singură tranzacție.

### Dependency Injection

Dependency Injection (DI) este un şablon de proiectare software care perminte dezvoltarea unui cod slab cuplat prin realizarea inversării controlului între clase și dependențele acestora.

Intenția Dependency Injection este de a face codul care mai uşor de întreținut, prin reducerea cuplării strânse dintre componentele software, a dependențelor hard-coded dintre clase prin injectarea acestor dependențe în timpul rulării în loc de timpul de proiectare din punct de vedere tehnic.

O dependență este un obiect de care depinde un alt obiect. Injecția de dependență poate fi utilizată şi într-un mod înlănțuit, fiecare dependință solicitând la rândul său propriile dependențe. Containerul rezolvă dependențele din grafic și returnează serviciul complet rezolvat. Setul colectiv de dependențe care trebuie rezolvat este de obicei denumit arbore de dependență, grafic de dependență.

### Singleton

Şablonul de proiectare singleton în este unul dintre cele mai populare şabloane. În acest model, o clasă are o singură instanță în program care oferă un punct global de acces la ea. Un singleton este o clasă care permite să fie creată doar o singură instanță a lui însuși și de obicei oferă acces simplu la acea instanță.

Implementarea unui model singleton are următoarele caracteristici: constructor unic privat și fără parametri, clasa sigilată, variabilă statică pentru a păstra o referință la o singură instanță creată, o metodă publică și statică de a obține referința la instanța creată.

Acest şablon poate implementa interfețe, poate fi încărcat leneș și are inițializare statică, ajută la ascunderea dependențelor, oferă un singur punct de acces la o anumită instanță, fiind ușor de întreținut. Totuşi implementarea acestui şablon face testarea unitară puțin dificilă, deoarece introduce o stare globală într-o aplicație şi reduce potențialul de paralelism în cadrul unui program prin blocare.

## Pachete NuGet folosite

### AutoMapper

AutoMapper este un mapper obiect-obiect. Maparea obiect-obiect funcționează prin transformarea unui obiect de intrare de un tip într-un obiect de ieșire de alt tip. AutoMapper oferă câteva convenții interesante pentru a elimina munca murdară de a afla cum să fie făcută maparea de la tipul A la tipul B. Atâta timp cât tipul B urmează convenția stabilită de AutoMapper, este nevoie de aproape zero configurație pentru a mapa două tipuri.

AutoMapper oferă o configurare simplă a tipurilor, precum și o testare simplă a mapărilor. Maparea poate apărea în multe locuri dintr-o aplicație, dar mai ales la granițele dintre straturi, cum ar fi între straturile UI/Domeniu sau straturile Serviciu/Domeniu. Preocupările unui strat sunt adesea în conflict cu preocupările din altul, astfel încât maparea obiect-obiect duce la modele segregate, în care preocupările pentru fiecare strat pot afecta doar tipurile din acel strat.

### AspNetCore

ASP.NET Core este un cadru open-source și multiplatformă pentru construirea de aplicații moderne conectate la internet bazate pe cloud, cum ar fi aplicații web, aplicații IoT și backend-uri mobile.

Din acest cadru au fost utilizate următoarele pachete NuGet:

* Microsoft.AspNetCore.Authentication: a fost utilizat la procesul de determinare a identității unui utilizator în aplicaţie
* Microsoft.AspNetCore.Cryptography.KeyDerivation: utilizat la crearea hashing-ului pentru parola utilizatorului
* Microsoft.AspNetCore.Http: utilizat pentru trimiterea şi primirea de formulare fişier prin intermediul cererilor http

### EntityFrameworkCore

Entity Framework (EF) Core este o versiune ușoară, extensibilă, open source și multiplatformă a tehnologiei populare de acces la date Entity Framework.EF Core permite dezvoltatorilor .NET să lucreze cu o bază de date folosind obiecte .NET, elimină necesitatea majorității codului de acces la date care de obicei trebuie scris şi acceptă multe motoare de baze de date.

În EF Core, DbContext are o metodă virtuală numită onConfiguring, care va fi apelată intern de EF Core și, de asemenea, va trece într-o instanță optionsBuilder și puteți utiliza acel optionsBuilder pentru a configura opțiunile pentru DbContext. OptionsBuilder are metoda UseSqlServer; se așteaptă un șir de conexiune ca parametru. Odată ce aveți un model, puteți utiliza migrările pentru a crea o bază de date.

Au fost utilizate urmatoarele pachete NuGet din acest cadru:

* Microsoft.EntityFrameworkCore.Design conține toată logica în timp de proiectare pentru Entity Framework Core. Este codul la care apelează toate instrumentele (cmdleturi PMC precum Add-Migration, dotnet ef & ef.exe).
* Microsoft.EntityFrameworkCore.SqlServer permite utilizarea Entity Framework Core cu Microsoft SQL Server (inclusiv SQL Azure), precum şi MySQL în cazul acestui proiect.
* Microsoft.EntityFrameworkCore.Tools este o suită de instrumente EF Core pentru Consola Manager de pachete NuGet în Visual Studio care activează comenzi utilizate în mod obișnuit: Add-Migration, Bundle-Migration, Drop-Database, Get-DbContext, Get-Migration

### Swashbuckle

Swashbuckle este un proiect open source pentru generarea de documente Swagger pentru API-uri web care sunt construite cu ASP.NET Core.

Cele trei componente de bază utilizate sunt:

* Swashbuckle.AspNetCore.Swagger: un model de obiect Swagger și middleware pentru a expune obiectele SwaggerDocument ca puncte finale JSON;
* Swashbuckle.AspNetCore.SwaggerGen: un generator Swagger care construiește obiecte SwaggerDocument direct din rutele, controlerele și modelele dvs. De obicei, este combinat cu middleware-ul final Swagger pentru a expune automat JSON Swagger;
* Swashbuckle.AspNetCore.SwaggerUI: o versiune încorporată a instrumentului Swagger UI. Acesta interpretează Swagger JSON pentru a construi o experiență bogată, personalizabilă pentru descrierea funcționalității API-ului web.

## Pachete NPM folosite

### Auth0/angular-jwt

Biblioteca auth0/angular-jwt oferă un HttpInterceptor care atașează automat un token web JSON la solicitările HttpClient.

Această bibliotecă nu are nicio funcționalitate pentru implementarea autentificării utilizatorilor și regăsirea JWT-urilor pentru început. Aceste detalii vor varia în funcție de configurația aplicaţiei., dar în cele mai multe cazuri, va fi folosită o solicitare HTTP obișnuită pentru a autentifica utilizatorii și apoi vor fi salvate JWT-urile lor în stocarea locală sau într-un cookie.

Auth0 vă ajută la:

* adăugarea de autentificare cu mai multe surse de autentificare, fie rețele sociale, cum ar fi Google, Facebook, Microsoft Account, LinkedIn, GitHub, Twitter, Box, Salesforce, printre altele, sau sisteme de identitate de întreprindere precum Windows Azure AD, Google Apps, Active Directory, ADFS sau orice identitate SAML Furnizor.
* adăugarea de autentificare prin baze de date mai tradiționale cu nume de utilizator/parolă;
* adăugare suport pentru conectarea diferitelor conturi de utilizator cu același utilizator;
* suport pentru generarea de jetoane web Json semnate pentru apelurile la API-uri și a transmite identitatea utilizatorului în siguranță;
* analize despre cum, când și unde se conectează utilizatorii;
* extragere date din alte surse și adăugare la profilul utilizatorului, prin regulile JavaScript.

### Balkangraph/familytree.js

Creați aplicația pentru arborele genealogic cu biblioteca JavaScript BALKAN FamilyTreeJS. Arborele genealogic numit și o genealogie sau o diagramă genealogică, este o diagramă care reprezintă relațiile de familie într-o structură convențională de arbore.

Printe funcţionalităţile acestui pachet se numără: suport date locale și date de la distanță (JSON), efecte de extindere/reducere, aliniere diagramă în 8 orientări, pan și zoom, personalizare a nodului unei persoane, căutare, bară de defilare, lazy loading, export.

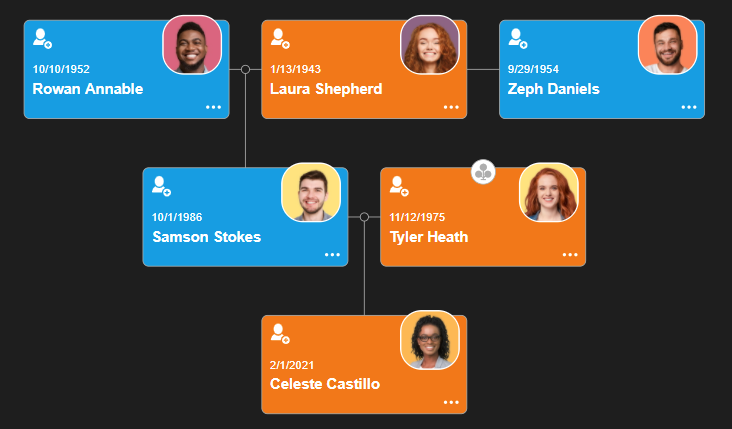


Figura 1. Exemplu vizualizare arbore

Family Tree JS rulează pe orice server care acceptă HTML, se poate rula şi local dintr-un sistem de fișiere, nu depinde de nicio bibliotecă JavaScript terță parte, funcționează în toate browserele moderne și folosește SVG pentru redarea grafică.

Printe componentele cheie utilizate la reprezentarea arborelui se numără:

* node: are este sursa de date şi are nevoie obligatoriu proprietatea „id”;
* pids: sunt id-urile partenerului şi reprezintă legătura dintre doi parteneri (soție și soț);
* mid: id-ul mamei;
* fid: id-ul tatălui;
* genre: genul, care poate fi masculin sau feminin;
* nodeBinding: prin intermediul acesteia se pot declara mai multe surse de date care vor fi folosite în şablonul arborelui (precum numele, şi ziua de naştere în şablonul anterior).

### Ionic/angular

Ionic este des utilizat alături de Angular. În timp ce componentele de bază au fost scrise pentru a funcționa ca o bibliotecă de componente web de sine stătătoare, pachetul @ionic/angular face integrarea cu ecosistemul Angular ușoară.

Prin instalarea pachetului @ionic/angular se facilitează utilizarea cadrului Ionic într-un proiect Angular. Acest pachet vine cu toate componentele Ionic Framework, serviciile și caracteristicile specifice Angular, dar şi învelișuri specifice Angular pentru @ionic/core. Pachetul @ionic/angular include toate funcționalitățile pe care dezvoltatorii Angular s-ar aștepta de la Ionic şi se integrează cu bibliotecile de bază Angular, precum routerul Angular.

S-a verificat că pachetul npm @ionic/angular nu are probleme de vulnerabilități cunoscute și licență lipsă.

### Ionic/storage

Pachetul @ionic/storage este un modul de stocare cheie-valoare simplu pentru aplicațiile Ionic. Acest utilitar folosește cel mai bun motor de stocare disponibil pe platformă fără a fi nevoie să interacționeze direct cu acesta, ci doar de o anumită configurație.

Începând cu versiunea 3, această bibliotecă acceptă orice proiect JavaScript (versiunile vechi fiind acceptate doar Angular), iar funcționalitatea specifică Angular a fost mutată într-un nou pachet @ionic/storage-angular, fiind şi acesta instalat.

Implicit, Ionic Storage va folosi o bază de date indexată (IndexedDB) și localstorage unde este disponibil. Pentru a utiliza SQLite pentru stocarea nativă, vor trebui realizaţi câţiva paşi în plus.

Pentru aplicații sensibile la securitate care necesită criptare, de la versiunea 3 se acceptă criptarea prin Ionic Secure Storage, acesta depozit de date de înaltă performanță, pregătit pentru întreprindere, cu suport SQL sau cheie/valoare și care oferă criptare AES pe 256 de biți.

### Ionic2-calendar

Pachetul ionic2-calendar conţine un modul special conceput pentru afişarea unui calendar şi a evenimentelor din acesta. Pentru a configura calendarul, aveți este nevoie de o serie de evenimente care pot fi încărcate dintr-un API și câteva setări de bază pentru calendar, precum modul și data curentă. Modul specifică vizualizarea calendarului și poate fi sub formă lună, săptămână și zi. Se poate localiza și o anumită limbă, prin utilizarea registerLocaleData din Angular și se poate importa orice pachet de limbă de care este nevoie.

Funcțiile de bază care pot fi adăugate la calendar sunt:

* next/back: mută vizualizarea calendarului în următoarea lună/săptămâna/zi în funcție de modul curent selectat
* onViewTitleChanged: va fi apelat de calendar și reflectă titlul actual al vizualizării, care poate fi personalizat și cu un formatator special
* onEventSelected: apelat atunci când se face clic pe o intrare din calendar;
* eventSource conţine evenimentele ce vor fi afişate în calendar, într-un format specific, şi poate fi modificată oricând, schimbările în calendar realizându-se în timp real

Utilizând CSS este o modalitate de a-l personaliza, iar o altă modalitate excelentă de a schimba aspectul calendarului este utilizarea șabloanelor personalizate pentru anumite părți ale calendarului.

### Leaflet

Leaflet este cea mai importantă bibliotecă JavaScript open-source pentru hărți interactive pentru dispozitive mobile. Având aproximativ 42 KB de JS, are toate caracteristicile de cartografiere de care au nevoie majoritatea dezvoltatorilor.

Leaflet este conceput pentru simplitate, performanță și utilizare. Funcționează eficient pe toate platformele desktop și mobile majore, poate fi extins cu o mulțime de plugin-uri, are un API frumos, ușor de utilizat și bine documentat și un cod sursă simplu și lizibil.

În mod implicit, toate interacțiunile cu mouse-ul și atingerea de pe hartă sunt activate și are comenzi de zoom și atribuire, acestea putând fi configurate la creere.

O hartă creată cu leaflet este alcătuită din mai multe straturi, fiecare cu un scop diferit. Aceste straturi sunt:

* Stratul de plăci(tile layer) oferă aspectul hărţii şi poate fi setat utilizând şablonul URL pentru imaginile de plăci al unui furnizor, textul de atribuire şi nivelul de zoom.
* Marcatoare(markers) pentru poziţiile geografice
* Poligoane şi cercuri
* Ferestre popup care sunt ataşate la un anumit obiect şi în care pot fi afişate orice informaţie despre în format HTML

De fiecare dată când se întâmplă ceva în Leaflet, obiectul corespunzător trimite un eveniment la care se poate subscrie cu o funcție, astfel permite reacţia imediată la interacțiunea utilizatorului.

### Ngx-image-compress

Pachetul ngx-image-compress este o utilitate Angular folosită pentru compresia fişierelor imagini la o dimensiune satisfăcătoare, care poate fi aleasă, înainte de încărcarea acestora pe un server.

Folosind funcția de încărcare și comprimare compressFile() se oferă control asupra procesului de compresie. Se poate efectua o singură încărcare cu comprimare automată la o dimensiune maximă prestabilită. Oferă suport şi pentru încărcarea mai multor fişiere.

Acest servici foloseşte Renderer2 care transformă imaginea folosind încrustația HTML canvas. Angular menține componenta și vizualizarea sincronizate folosind șabloane, legarea datelor și detectarea modificărilor.

### Ngx-translate

Una dintre cele mai populare biblioteci open-source i18n, ngx-translate, permite să se definească traduceri pentru aplicaţii și să se comute între ele în mod dinamic. Poate fi folosit fie un serviciu, o directivă sau o conductă (pipe) pentru a gestiona conținutul tradus.

Pentru realizarea traducerii în cadrul aplicaţiei utilizând ngx-translate sunt necesare 2 pachete, şi anume:

* @ngx-translate/core care conține rutinele de bază pentru traducere: TranslateService și câteva conducte;
* @ngx-translate/http-loader care încarcă fișierele de traducere de pe serverul web.

Pentru utilizare trebuie setată limba implicită a aplicației folosind setDefaultLang('en') din serviciul de traducere. Fiecare limbă care poate fi utilizată la traducere este stocată într-un fișier .json separate, fiind prin convenţie în directorul i18n .

### Rxjs

Reactive Extensions for JavaScript, sau RxJS, este o bibliotecă JavaScript care utilizează observabile pentru programarea reactivă. Poate fi folosit cu alte biblioteci și cadre JavaScript și se integrează bine în Angular. Paradigma reactivă poate fi utilizată în multe limbi diferite prin utilizarea bibliotecilor reactive. Aceste biblioteci sunt API-uri descărcate care oferă funcționalități pentru instrumente reactive. RxJS este o bibliotecă reactivă folosită pentru a implementa programarea reactivă pentru a face față implementării asincrone, apelurilor inverse și programelor bazate pe evenimente.

RxJS are câteva caracteristici de bază care ajută la gestionarea implementării asincrone:

* Observabilele(Observable) permit publicarea evenimente. Observabilele au două metode: abonare(subscribe) și dezabonare(unsubscribe). Un observabil se execută prin abonarea la acesta. Observabilele modelează un flux de evenimente.
* Observatorul (Observer) este un obiect cu metode next(), error() și complete() care este apelat atunci când există o interacțiune cu observabilul. Ele sunt obiectele care subscriu la observabile.
* Abonamentul(Subscription) la observabil va declanșa observabilul să se execute.
* Operatorul (Operator) este o funcție care ne permite să efectuăm anumite acțiuni asupra evenimentelor executate de observabile.
* Subiectul(Subject) este același cu un EventEmitter. Este un observabil care transmite informații către observatori.
* Planificatorul (Scheduler) se ocupă de execuția abonamentelor.

Biblioteca RxJS este excelentă pentru gestionarea sarcinilor asincrone. Are o colecție mare de operatori în filtrare, tratare a erorilor, condițional, creare, multicast și multe altele. Este acceptat de JavaScript și TypeScript. Acesta poate fi folosit cu alte biblioteci și cadre JavaScript, este conceput pentru a permite noi funcționalități şi nu are dependențe de terți.

### Swiper

Swiper este cel mai modern glisor tactil mobil gratuit, cu tranziții accelerate hardwar. Este destinat să fie utilizat în site-uri web mobile, aplicații web mobile și aplicații mobile native/hibride. Swiper nu este compatibil cu toate platformele, este un glisor tactil modern care se concentrează doar pe aplicații/platforme moderne pentru a aduce cea mai bună experiență și simplitate.

Swiper, împreună cu alte componente excelente, face parte din Framework7 - un cadru complet pentru construirea de aplicații iOS și Android. Swiper este, de asemenea, o componentă implicită de glisare în cadrul Ionic.

## Medii de dezvoltare

### SQL Server Management Studio (SSMS)

Pentru stocarea informaţiilor despre arborii genealogici ai utilizatorilor am folosit mediul SQL Server Management Studio.

SQL Server Management Studio (SSMS) este un mediu integrat pentru gestionarea oricărei infrastructuri SQL, de la SQL Server la Azure SQL Database. SSMS oferă instrumente pentru a configura, monitoriza și administra instanțe ale SQL Server și baze de date.

SSMS a fost utilizat pentru a monitoriza și actualiza componentele de nivel de date utilizate de aplicație și pentru a crea interogări și scripturi și gestiona baza de date.

### Visual Studio 2019

Pentru realizarea soluţiei de gestionare a datelor arborelui genealogic a fost utilizat mediul de dezvoltare integrat (IDE) Visual Studio 2019.

Aceasta este o aplicație bogată în caracteristici care poate fi utilizată pentru multe aspecte ale dezvoltării software. Visual Studio IDE facilitează editarea, depanarea, construirea și publicarea aplicației. Visual Studio include un editor de cod care acceptă componenta de completare a codului numită IntelliSense, cât și refactorizarea codului.

Acceptă plug-in-uri care extind funcționalitatea la aproape fiecare nivel, inclusiv adăugarea de suport pentru sistemele de control sursă și adăugarea de noi seturi de instrumente.

### Visual Studio Code

Pentru realizarea soluţiei de interacţiune cu utilizatorul şi afişare a arborelui genealogic a fost folosit editorul Visual Studio Code.

Visual Studio Code este un editor de cod simplificat cu suport pentru operațiuni de dezvoltare, cum ar fi depanarea, rularea sarcinilor și controlul versiunilor. Acesta își propune să ofere doar instrumentele de care are nevoie un dezvoltator pentru un ciclu rapid de creare a codului-depanare și lasă fluxuri de lucru mai complexe pentru IDE-uri mai complete.

# Sistemul de gestiune al arborelui genealogic

## Cerinţe

Pentru acest proiect a fost luată în considerare o structură clasică pentru gestionarea arborelui genealogic, în care fiecare persoană din arbore poate avea doar 2 părinţi, o mamă de gen feminin, şi un tată de gen masculin, mai multe căsătorii doar cu persoane de gen opus, şi mai mulţi copii.

Fiecare utilizator al aplicaţiei are un arbore genealogic unic pe care doar acesta îl poate modifica prin adăugare, editare sau ştergere de persoane sau legături între persoane. Aceste legături pot fi de tipul căsătorii sau părinte-copil. La legăturile părinte-copil se poate specifica dacă persoanele sunt înrudite biologic sau nu (spre exemplu copil adoptat). La legăturile de tip căsătorie nu se poate specifica acest lucru, considerând implicit că persoanele nu sunt înrudite, însă trebuie specificat data căsătoriei, având opțional, după situaţie, posibilitatea de a specifica şi data divorțului.

Informaţiile pe care fiecare persoană dintr-un arbore genealogic le includ sunt: nume, prenume, gen, ziua de naştere, după caz şi ziua în care a murit, locul în care s-a născut, locul în care trăieşte, religia, naţionalitatea şi o poză o persoanei. Dintre aceste informaţii numele, prenumele și data nașterii sunt obligatorii.

Toţi utilizatorii aplicaţiei moştenesc şi informaţiile persoanei care îi reprezintă în arborele lor genealogic, şi au în plus informaţii precum număr de telefon, e-mail, o biografie, educaţiile şi locurile lor de muncă. Înafară de e-mail, niciuna din informații nu este obligatorie, fiecare utilizator putând decide ce dorește să adauge la profil. Ocupațiile pe care le adaugă utilizatorul conțin informații despre postul ocupat, numele locului de muncă, și datele în care a început şi respectiv încheiat să lucreze la locul de muncă. Educația are informații despre nivelul acesteia(grădiniță, primară etc.), numele instituției de învățământ, data înrolării și data absolvirii.

Utilizatorii aplicaţiei vor putea căuta alți utilizatori pentru a vedea profilul acestora, dar nu vor avea acces la informaţiile despre persoanele din arborii genealogici ai acestora. În urma unei cereri de recunoaştere a rudeniei dintre 2 utilizatori, care a fost în prealabil acceptată, ambii utilizatori vor avea access de vizualizare a informaţiilor din arborele celuilalt, cât şi o referinţă în propriul arbore genealogic asupra persoanei care reprezintă utilizatorul recunoscut ca rudă.

Fiecare utilizator poate alege să primească notificări despre ziua de naștere şi de nuntă a persoanelor din arborele său genealogic, cât şi despre modificări în arborii rudelor. Tot drept notificări vor apărea şi cererile de rudenie dintre utilizatori, cât şi răspunsul la aceste cereri.

Fiecare utilizator va putea verifica aniversările din arborele său genealogic prin intermediul unui calendar care va conţine toate aniversările de zile de naştere şi de nuntă de pe tot parcursul anului.

Un utilizator va putea să vadă ultima locaţie cunoscută a fiecărei rude din aplicaţie, având posibilitatea de a refuza să ofere locaţia celorlalţi.

## Cazuri de utilizare

Majoritatea funţionalităţilor aplicaţiei vor fi disponibile doar în urma unei autentificări.

Când utilizatorul nu este autentificat, acesta are posibilitatea de a se înregistra în aplicaţie cu un e-mail valid, să vadă pagina de introducere a aplicaţiei, să seteze diverse caracteristici despre accesibilitate, precum tema, limba şi evidenţierea link-urilor.

În urma autentificării, utilizatorul va avea acces la toate funcţionalităţile afişate în diagrama de utilizare, cu menţiunea că o parte dintre acestea pot fi realizate doar când acesta accesează informaţii din arborele său genealogic, ci nu din arborele unei rude. Funcţionalităţile la care utilizatorul nu are acces când vede informaţii din arborele genealogic al unui alt utilizator care îi este rude sau chiar profilul acesteia sunt: operaţiile CRUD pentru persoană, căsătorie, părinte, copil, educaţie, ocupaţie, cât şi schimbarea de poză şi editarea profil. Când utilizatorul al cărui informaţii doreşte să le vadă nu îi este rudă, acesta nu va putea nici vizualiza arborele genealogic, lista de persoane din arbore sau detaliile oricărei persoane din acesta, cât nici anumite informaţii despre utilizator precum e-mail-ul şi numărul de telefon.

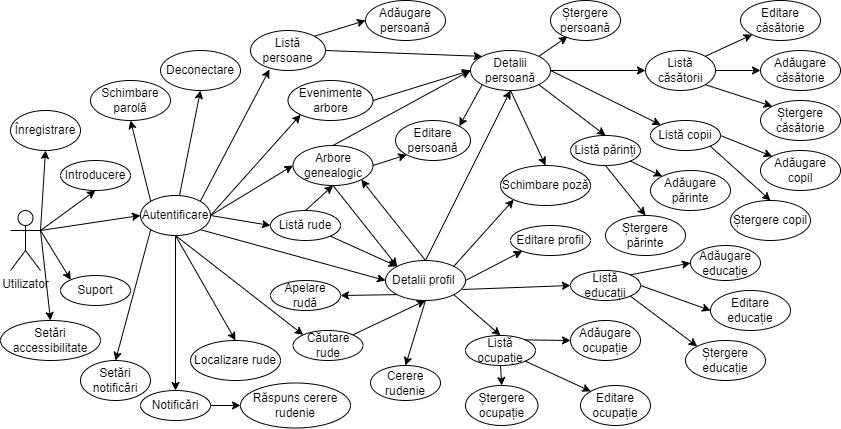


Figura 2. Diagrama cazurilor de utilizare

## Structura bazei de date

Baza de date este stocată într-un server MySQL care poate fi direct accesată folosing SSMS.

Acestă bază de date a fost creată pe baza tehnicii de programare code-first, în care prima dată se vor creea clase care să reprezinte tipurile de entităţi, care în urma adăugării de migrări folosind pachetul EF Core şi updatării bazei de date, se vor creea sau modifica tabelele din această bază de date.

Baza de date a suportat mai multe modificări pe parcursul dezvoltării, actual aceasta fiind alcătuită din 15 tabele, dintre care majoritatea legăturilor gravitează în jurul tabelelor Person şi User.

Tabelul Person deţine informaţiile despre fiecare persoană din toţi arborii genealogici. Pentru a diferenţia din ce arbore face parte fiecare persoană, la creerea unei persoane i se ataşează cheia străină Tree\_ID. Aceasta marchează arborele unic pe care îl deţine fiecare utilizator al aplicaţiei.

Tabelul Image păstrează informaţii de bază despre pozele persoanelor, cea mai importantă informaţie fiind file\_name, care oferă şi calea unde a fost salvată imaginea în sistemul de fişiere pe care rulează aplicaţia.

Tabelul Location deţine este folosit pentru stocarea locului în care s-a născut sau trăieşte fiecare persoană.

Tabelele Religion şi Nationality şi EducationLevek au valori predefinite despre posibilele variante pe care le poate alege utilizatorul la crearea sau editarea unei persoane sau a profilului personal.

Tabelele Marriage şi ParentChild sunt folosite pentru a stoca legăturile dintre persoanele unui anumit arbore genealogic. Fiecare dintre aceste tabele deţin câte 2 referinţe către entităţi de tip Person a putea crea legăturile dintre persoane. Adiţional acestea conţin informaţii specifice tipului de legătură: pentru căsătorii începutul şi finalul acestora, iar pentru legături părinte-copil dacă acestea sunt rude de sânge sau nu.

Cele 3 tabele Person, Marriage şi ParentChild sunt principalele tabele pe baza cărora se construieşte arborele genealogic. Toate acestea deţin 2 câmpuri suplimentare pentru stocarea momentului creării şi al editării lor, care vor fi utilizate la notificările care informează în privinţa modificării arborelui unei rude.

Tabela User deţine informaţii specifice utilizatorului aplicaţiei, care pot fi împărţite în informaţii despre cont (username-ul, şi cele 2 câmpuri pentru parola cu hash şi salt), preferinţe utilizator (notify\_birthdays, notify\_updates, share\_personal\_info, share\_location), şi informaţii vizibile pentru rude(email, phone\_number, about).

Tabela Position păstrează ultima locaţie cunoscută a fiecărui utilizator al acestei aplicaţiei, dacă acesta a permis accesul la gps-ul dispozitivului pe care îl foloseşte pentru a accesa aplicaţia.

Tabelele Education şi Occupation au informaţii despre educaţiile şi ocupaţiile unui utilizator.

Tabelul Request are toate cererile de rudenie care au fost trimise între 2 utilizatori, cât şi răspunsul la aceaste cereri.

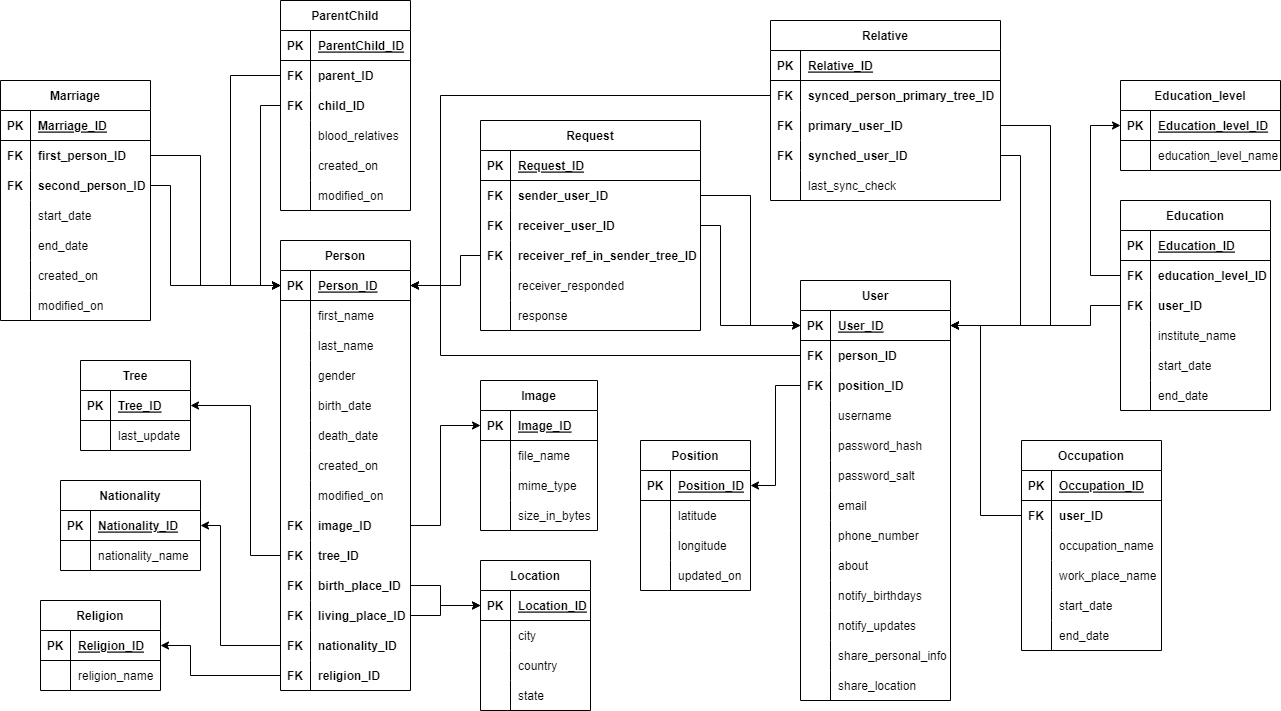


Figura 3. Structura bazei de date

## Arhitectura soluţiei backend

Soluţia de backend pentru aplicaţie a fost realizată folosind mediul de dezvoltare Visual Studio 2019, şi a fost împărţită în 4 proiecte: API, Business, Data, Domain. Fiecare dintre aceste proiecte are un rol specific.

### Proiectul API

API este proiectul de bază, aceasta pornind aplicaţia de backend propriu-zisă şi fiind locul unde se realizează legătura cu agenţii externi. Printre cele mai importante clase regăsim clasa Program care conţine metoda Main în care se creează builder-ul pentru aplicaţie şi se creează baza de date în cazul în care aceasta încă nu a fost creeată.

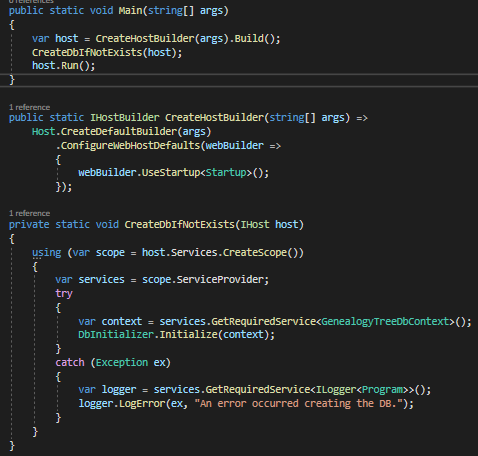


Figura 4. Clasa Program

Altă clasă importantă este Startup care conţine 2 metode: una care conţine configurările pentru context, servicii, mapper, unit of work, opţiunile pentru formulare, controllere, swagger şi smtp; iar cealaltă care conţine pipeline-ul pentru cererile HTTP.



Figura 5. Configurarea serviciilor



Figura 6. Configurarea pipeline-ului

Un controller este responsabil pentru controlul modului în care un utilizator interacţionează cu aplicaţia, acesta conţinând punctele de acces ale aplicaţiei cu logica specifică pentru fluxul datelor şi determinând răspunsul pe care îl va primi utilizatorul în urma unei cereri din browser.

Controllerele aplicaţiei sunt: AuthController, CommonController, EducationController, MarriageController, OccupationController, ParentChildController, PersonController, RelativeController, UserController.

Fişierul appsettings.json conţine setările de configurare ale proiectului, cum ar fi şirul pentru conexiunea cu baza de date, alte variabile globale, cum ar fi configurarea serviciului SMTP şi calea directorului în care se stochează fişierele încărcate de către utilizatori.

Tot în proiectul API se regăsesc şi două atribute pentru validarea dimensiunii maxime a imaginii încărcate şi pentru autorizarea utilizatorului la accesarea unui endpoint.

La pornirea aplicaţiei se va porni o pagină în care avem acces la Swagger, de unde se pot accesa toate punctele de acces disponibile ale aplicaţiei, cât şi structura obiectelor folosite la transmiterea informaţiilor.

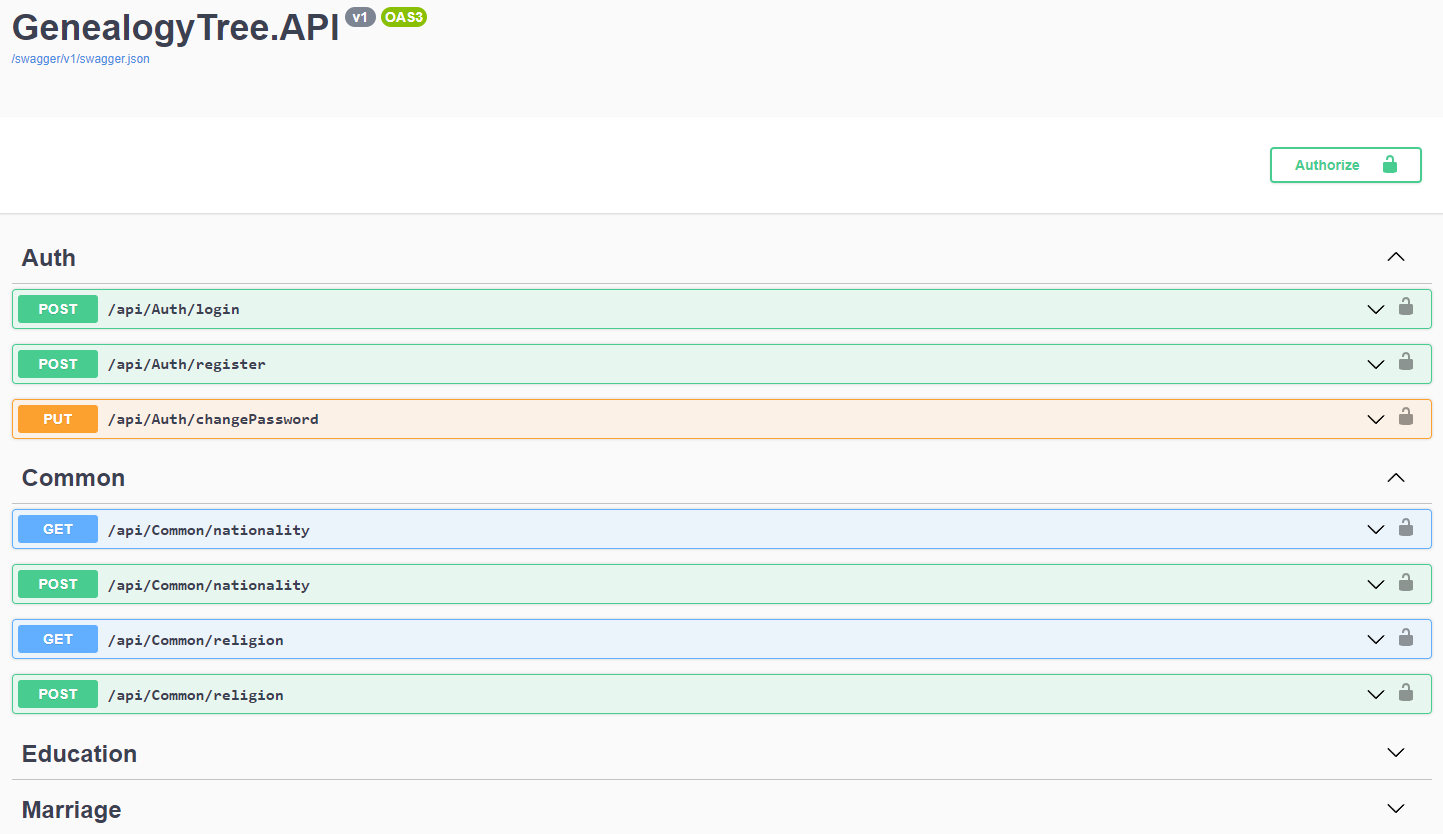
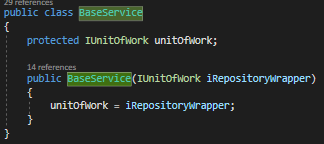


Figura 7. Interfaţa Swagger

### Proiectul Business

Proiectul bussiness conţine implementările serviciilor aplicaţiei, generarea token-ului de autentificare, şablonul pentru e-mail-ul de suport alături de câteva clase pentru informaţiile despre e-mail, profilul de mapare al obiectelor şi o clasă de constante şi o clasă pentru fişierele imagine.

Toate serviciile implementează o interfaţă specifică lor şi extind clasa de bază BaseService care conţine Unit Of Work.



Figură 8. Clasa BaseService

Serviciile sunt clase create de către programator care conţin întreaga logică din spatele unui apel la aplicaţia de backend. Servicii soluţiei backend sunt: AuthService, EducationService, FileManagementService, ImageService, LocationService, MarriageService, NationalityService, OccupationService, ParentChildService, PersonService, RelativeService, RequestService, TokenService UserService. Toate aceste servicii lucrează cu una sau mai multe tipuri de entităţi specificat în numele claselor, cu excepţia a 3 servicii:

* FileManagementService care gestionează scrierea, ştergerea şi returnarea fişierelor imagine încărcate de către utilizatori;
* TokenService utilizat la crearea token-ului, validarea acestuia şi preluarea drepturilor de utilizator din acesta;
* EmailService care are rolul de a trimite mail-ul cu cererea de suport a utilizatorului către contul de gmail special creat pentru această aplicaţie.

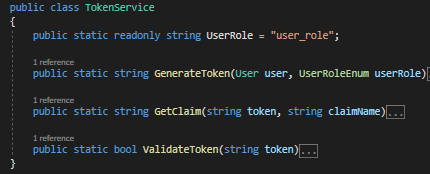


Figura 9. TokenService

Token-ul generat este de tip JWT( JSON Web Token) care conţine validitatea, numele utilizatorului, email-ul, id-ul şi rolul acestuia, toate aceste informaţii fiind criptate cu algoritmul de criptare HmaSha256.

Clasa de constante JWT este utilizată pentru a stoca informaţii despre emitentul, audienţa şi cheia secretă a JWT-ului.

Clasele Hash şi Salt sunt utilizate pentru securitatea parolei la criptarea şi validarea acesteia. Clasa Hash conţine o funcţie hash unidirecţională care transformă un şir de caractere, în cazul acesta fiind parola introdusă de utilizator la înregistrare, autentificare sau schimbarea parolei, într-o serie de simboluri de lungime fixă. Clasa Salt are rolul de a genera un şir de caractere aleatoare care să fie ataşate parolei înainte de a-i aplica funţia hash. În acest fel se pot preveni atacurile de dicţionar.

Clasa EmailLanguageHelper are rolul de a seta conţinutul mesajului care va fi transmis prin email într-o limbă specifică, română sau engleză.

Clasa MappingProfile este utilizată pentru configurarea mapărilor între entitate - DTO(Data Transfer Object), DTO - entitate sau DTO – DTO, care vor fi folosite în serviciile aplicaţiei.

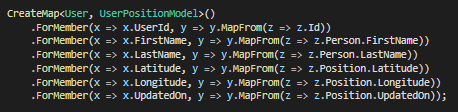


Figura 10. Exemplu mapare folosind IMapper

Clasa statică FileExtensions este utilizată la preluarea informaţiilor dintr-ul fişier de tip IFormFile şi crearea unui DTO de tipul ImageFile care să conţină informaţiile specifice imaginii.

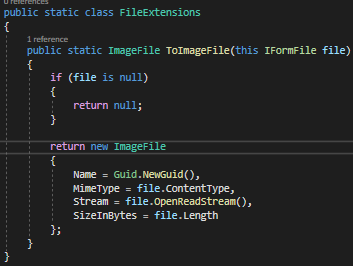
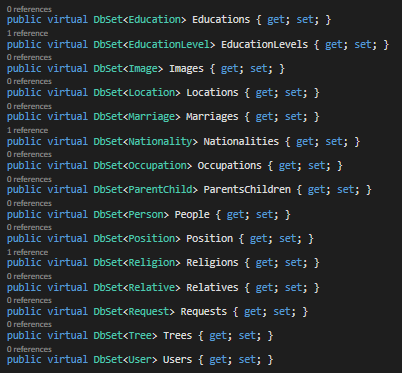


Figura 11. Clasa FileExtensions

### Proiectul Data

În proiectul Data se regăsesc implementările contextului bazei de date, a unităţii de lucru (UnitOfWork), ale repertoriilor (repositories), cât şi directorul în care se salvează imaginile încărcate de utilizatori, migrările şi un iniţializator al bazei de date.

Contextul bazei de date implementează interfaţa specifică şi extinde clasa de bază DbContext. În mod logic, un DbContext se mapează la o anumită bază de date care are o schemă pe care DbContext o înțelege. Acesta conţine proprietăţi de tip DbSet<T> pentru fiecare tabel din baza de date, care poate fi accesat pentru a gestiona entităţile existente în aceasta.



Figură 12: Proprietăţi DbSet în Clasa GenealogyTreeDbContext

Tot în această clasă se găseşte builder-ul care se configurează pentru a specifica legăturile dintre tabele şi tipul acestora, dar şi ce entităţi să fie automat incluse la regăsirea unei entităţi.

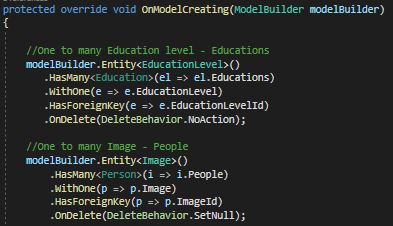


Figura 13: Specificare legături între tabele

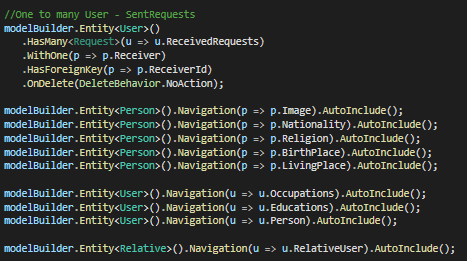


Figura 14: Includere automată a entităţilor

Unitatea de lucru (UnitOfWork) implementează interfaţa sa specifică, conţine câte un repertoriu pentru fiecare tabel din baza de date şi implementează şablonul singleton.

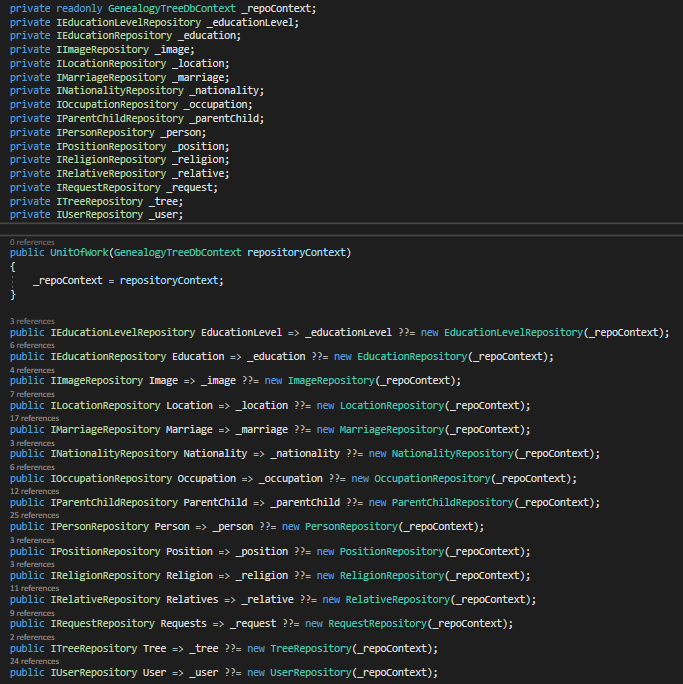
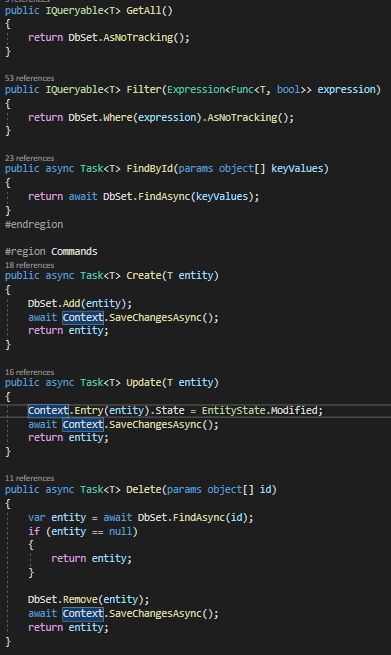


Figura 15: Clasa UnitOfWork

Clasa abstractă RepositoryBase implementează interfaţa sa specifică şi conţine metodele generice de acces şi modificare a unei entităţi din baza de date.



Figură 16: Metode generice în RepositoryBase

Toate repertoriile implementează interfaţa lor specifică şi extind clasa abstractă RepositoryBase, având specificat drept parametru de tip generic tipul entităţii pentru care s-a creeat repertoriul.

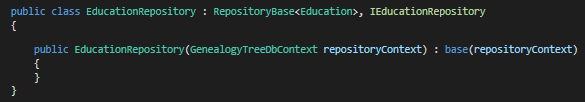


Figura 17: Exemplu repertoriu

Clasa DbInitializer verifică dacă anumite tabele sunt populate cu entităţi, iar în caz negativ va genera entităţi pentru acestea. Această clasă este utilizată pentru tabelele care trebuie să aibe opţiuni predefinite, şi anume EducationLevel, Religion şi Nationality.

Directorul de migrări conţine atât migrările care au fost făcute pentru a creea sau modifica structura bazei de date, cât şi un fişier snapshot care conţine toate informaţiile despre structura curentă a bazei de date.

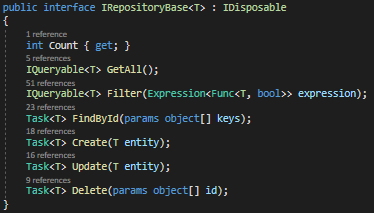
### Proiectul Domain

În proiectul Domain se află clasele DTO(Data To Object), clasele entităţi care reprezintă structura entităţilor din tabelele bazei de date, enumeratoarele folosite şi toate interfeţele pentru repertorii, servicii, context şi unitatea de lucru.

Clasele entităţi sunt utilizate pentru reprezentarea exactă a structurii entităţilor din tabelele bazei de date. Deoarece în acest proiect s-a folosit abordarea code-first, tabelele au fost generate pe baza acestor clase.

Clasele DTO (Data To Object) folosite pentru a primi sau răspunde la cereri http când nu se doreşte să se cunoască toate câmpurile sau să se folosească aceeaşi structură asemenea entităţilor stocate în baza de date.

Interfaţa IRepositoryBase conţine amprentele funcţiilor generice ale operaţiilor care vor putea fi realizate pe tabelele din baza de date.



Figură 18: Interfaţa IRepositoryBase

Fiecare interfaţă pentru repertoriu moşteneşte această metodă, având ca parametru de tip generic entitatea pentru care s-a creeat acest repertoriu



Figură 19: Exemplu moştenire IRepositoryBase

Interfeţele pentru servicii conţin amprentele metodelor publice care vor fi accesate de endpoint-uri.

Interfaţa pentru unitatea de lucru are doar interfaţa fiecărui repertoriu.

## Arhitectura soluţiei frontend

Directorul sursă al soluţiei de frontend conţine

* Fişierul index.html în care este randat programatic tot conţinutul aplicaţiei
* Fişierul global.scss în care sunt aplicate stiluri asupra tuturor fişierelor html componente
* Fişierul main.ts care creează mediul pe care să ruleze aplicaţia
* Directorul theme care conţine fişierul scss în care se configurează tema proiectului
* Directorul environments care conţine fişiere cu clasele configurate pentru mediul proiectului
* Directorul assets în care se regăsesc fişiere js, audio, .scss, iconiţe, imagini şi json pentru internaţionalizarea limbii
* Directorul app care conţine toate componentele care fac parte din cadrul aplicaţiei

În directorul app se află modulul aplicaţiei cu configurări ale acesteia, rutarea modulului cu căile de acces pentru paginile aplicaţiei, directorul app-component care conţine prima componentă angular care se încarcă în cadrul aplicaţiei şi directorul modules.

Directorul modules conţine mai multe module, fiecare dintre aceste module este utilizat pentru o anumită funcţionalitate a aplicaţiei.

Un director modul al aplicaţiei are structura următoare:

* Directorul components conţine componente care pot fi adăugate în cadrul unei pagini. Toate aceste componente trebue declarate în cadrul fişierului clasei modulului pentru a putea fi utilizate în cadrul aplicaţiei. O componentă este alcătuită dintr-un fişier typescript care conţine partea de logică, aceasta gestionând datele; un fişier html pentru a afişa grafic informaţiile din fişierul typescript; şi un fişier scss folosit pentru îmbunătăţirea vizuală a conţinutului fişierului html. Într-o componentă se pot utiliza şi alte componente.
* Directorul models are clase de modele, enumerabile sau constante specifice modulului, utilizate pentru tansferul de date între componente sau pentru comunicarea cu aplicaţia de backend.
* Directorul pages conţine paginile specifice modulului şi sunt similare cu caracteristicile componentelor, singura diferenţă constând în faptul că paginile reprezintă nivelul cel mai de sus, în cadrul acestora randându-se tot conţinutul unei anumite pagini html, pe când componenta este doar o parte din cadrul acestei pagini. În cazul anumitor module, acest director poate să lipsească, deoarece nu conţin pagini care să poată fi afişate.
* Directorul directives conţine directive specifice modulului care pot îndeplini diverse funcţii în fundal, acestea având doar un fişier de typescript. Şi acesta poate să lipsească în anumite module.
* Directorul services are unul sau mai multe servicii utilizate în vederea procurării de informaţii, de obicei de la aplicaţia de backend prin accesarea de puncte de acces.
* Fişierul de rutare conţine căile pentru pagini care pot fi accesate în cadrul modulului. În cazul unui modul care nu are pagini ce pot fi afişate, aces fişier va lipsi.
* Fişierul clasei modulului este locul în care se declară toate componentele şi paginile implementate în modul, se exportă componentele declarate care pot fi folosite şi în cadrul altor module şi se importă modulele din care trebuie utilizate anumite componente.

Structura prezentată mai sus poate fi puţin diferită în funcţie de modul, unele directoare putând să lipsească.Modulul componentei de bază a aplicaţiei conţine configurările pentru aplicaţie.



Figura 20: Modulul AppModule



Figura 21: Exemplu rutare modul

În acesta se adaugă:

* Modulele utilizate la nivelul cel am înalt al aplicaţiei, precum modulul pentru rutare, modulul pentru componentele din cadrul Ionic, modalitatea de stocare a datelor care poate fi în ordinea disponibilităţii driverul cordova, bază de date indexată sau depozitul local, şi 2 module create care vor fi ulterior explicate.
* Furnizorii pentru rutarea aplicaţiei şi pentru setarea titlului paginilor.

În cele ce urmează va fi descris fiecare modul al soluţiei de frontend şi scopul acestuia.

### Modulul shared

Acest modul pune la dispoziţie diverse componente şi servicii care să fie utilizate în cadrul celorlalte module, toate componentele declarate în acesta fiind exportate în acest scop. Din această cauză, nu conţine nicio pagină proprie care să poată fi accesată, neavând rutare sau director pentru pagini.

Modulul shared conţine 2 servicii: common şi utils. Serviciul common este utilizat pentru a trimite cereri către punctele de acces din controllerul cu acelaşi nume din soluţia de backend. Serviciul utils este folosit pentru a determina clasa css pentru card-urile folosite pentru persoane în aplicaţie şi crearea url-ul imaginii pentru persoane.

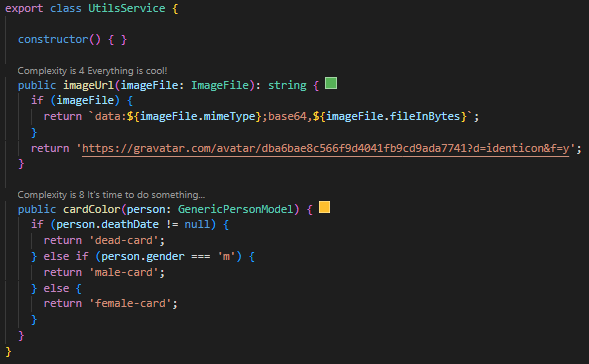


Figura 22: Clasa UtilsService

Drept componente, acest modul conţine şabloane reutilizabile pentru casete de input, selecţie din listă, selecţie a zilei din calendar, selecţia limbii, selecţie a referinţei unei persoane din cadrul cererilor de rudenie, butoane de legătură pentru persoane care sunt şi utilizatori.

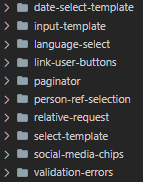


Figura 23: Şabloane reutilizabile

Ca modele, avem o clasă CommonObject care reprezină cel mai simplu model, care are ca parametri doar id şi name, o clasă pentru fişiere tip imagine, un clasă filtru pentru scroll infinit şi o clasă pentru locaţia utilizatorului.

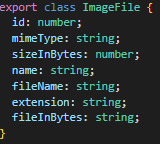


Figura 24: Clasa ImageFile

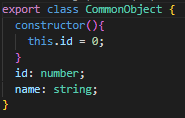


Figura 25: Clasa CommonObject

### Modulul core

Modulul core nu conţine pagini, astfel neavând fişier de rutare. Acesta conţine numai 2 componente pentru navigarea între paginile aplicaţiei.

Precum sugerează şi numele, acest modul conţine alte fişiere de bază pentru aplicaţie: servicii, gărzi, interceptori, furnizor de zonă, manipulator de erori, şi mai multe modele pentru transferul de date.

Gărzile sunt clase care au rolul de a verifica dacă utilizatorul poate accesa anumite pagini ale aplicaţiei. Acestea implementează de regulă una din interfeţele CanLoad sau CanActivate. În cazul acestui proiect, toate gărzile implementează CanLoad, aceste gărzi fiind:

* AuthGuard care permite accesul la pagină în urma validării că utilizatorul este autentificat în aplicaţie. Această gardă este utilizată pentru a restricţiona toate paginile care nu ar trebui să fie accesate de utilizatori neautentificaţi.
* AutoLoginGuard care permite accesul la pagină numai dacă utilizatorul nu este autentificat, iar dacă acesta este autentificat, este redirecţionat la pagina de profil a utilizatorului. Această gardă este folosită pentru paginile de logare şi înregistrare, care nu ar trebui să poată fi accesate când utilizatorul este autentificat.
* IntroGuard este utilizat pentru a verifica dacă utilizatorul a văzut sau nu pentru prima dată intro-ul aplicaţiei. Dacă nu l-a văzut, acesta este redirecţionat către pagina de intro.

Interceptorii sunt utilizaţi pentru a intercepta toate cererile sau răspunsurile http de la soluţia backend şi a le modifica sau a efectua diverse operaţii utilizând informaţiile date de aceste cereri/răspunsuri. Aceşti interceptori implementează interfaţa HttpInterceptor şi sunt următorii:

* AppHttpInterceptor este folosit pentru a seta tipul conţinutului, pentru a gestiona posibilele erori returnate de backend şi pentru a afişa mesajele de succes sau eroare după caz.
* AuthInterceptor este folosit pentru a modifica cererea trimisă prin adăugarea headerului de autorizare a utilizatorului care va conţine jwt-ul primit în urma autentificării cu succes.
* LoadingInterceptor are scopul de a afişa efectul de încărcare asupra paginii la trimiterea unei cereri http în cazul în care nu s-a setat explicit acunderea acestuia, şi de a înlătura acest efect la primirea răspunsului la cerere.

Manipulatorul de erori se ocupă cu detecţia de erori la nivelul aplicaţiei de frontend, în cazul apariţiei uneia, acesta o va loga în consolă şi va încerca să reîncarce pagina în vederea rezolvării ei.

Serviciile care pot fi regăsite în acest modul sunt: AlertService, AuthService, BaseService, DataService, LoadingService, LocaleService

Clasa abstractă de bază pentru serviciile care se ocupă de cererile http se regăseşte în acest modul. Aceasta este moştenită de către toate serviciile al căror scop este să trimită sau să primească date de la soluţia de backend şi are metode generice pentru aproape orice tip de cerere.

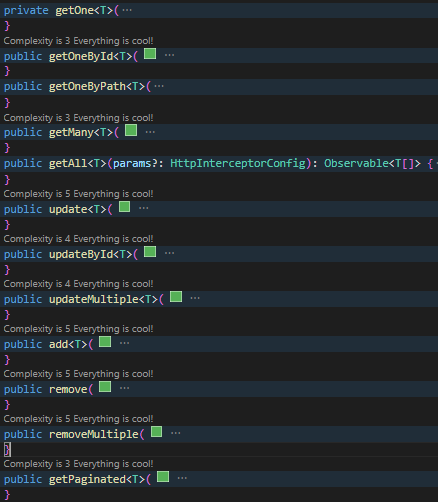


Figura 26: Metode generice din BaseService

Serviciul de zonă LocaleService are rolul de a observa limba selectată de utilizator şi de a seta în funcţie de aceasta zona pentru toate obiectele de tip dată care sunt folosite în cadrul aplicaţiei.

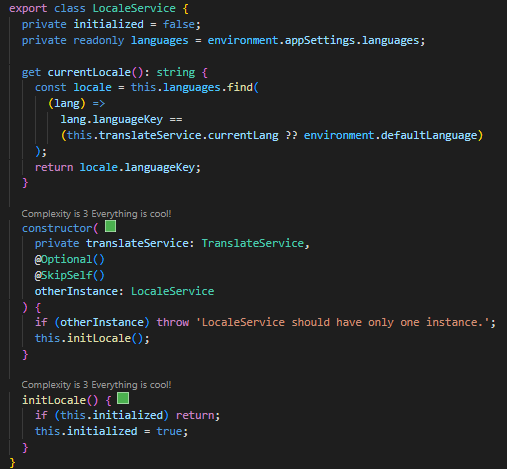


Figura 27: Clasa LocaleService

Serviciul de încărcare LoadingService gestionează adăugarea şi înlăturarea efectului de încărcare din cadrul aplicaţiei.

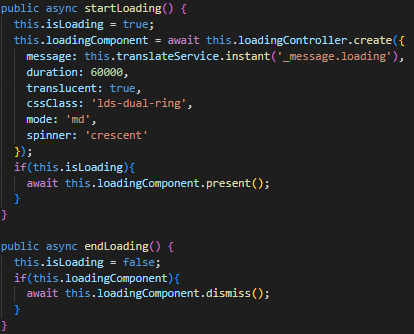


Figura 28: Metode din clasa LoadingService

Serviciul de alerte se ocupă cu afişarea pop-up-ului de confirmare al unei acţiuni, precum ştergerea din arborele genealogic şi afişarea alertelor sub formă de toaster în partea inferioară a paginii. Aceste alerte sunt de 4 tipuri: succes, informare, avertisment şi eroare. Fiecare dintre aceste tipuri are câte o metodă specifică pentru afişare produce şi un efect sonor la momentul apariţiei.

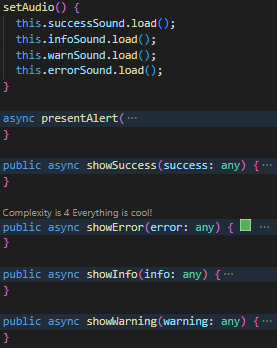


Figura 29: Tipuri de alerte în AlertService

Pentru crearea toaster-ului care va fi afişat, acestea utilizează o metodă generică createToast.

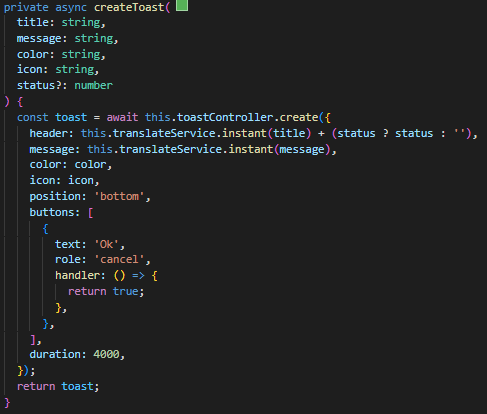


Figura 30: Metoda generică createToast

Serviciul pentru date DataService se ocupă cu gestionarea informaţiilor stocate într-un anumit tip de depozit ( IndexedDB/LocalStorage/CordovaDriver) pe dispozitivul utilizatorului. Printre informaţiile stocate se numără: utilizatorul conectat, token-ul utilizatorului, limba, setările preferenţiale şi setările de accesibilitate şi de aspect.

Acest serviciu se foloseşte clasa de depozitare Storage oferită de pachetul ionic/storage care necesită o iniţializare preferenţială.

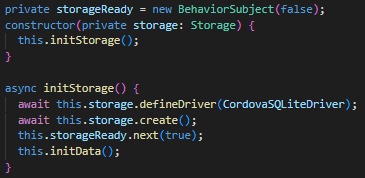


Figura 31: Iniţializare Storage

Are de asemenea 3 metode generice prin care se poate modifica, prelua sau şterge orice informaţie de pe dispozitiv.

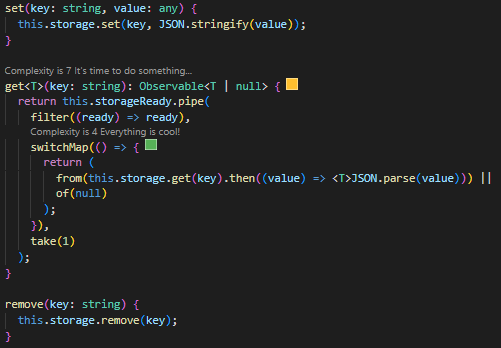


Figura 32: Metode generice în DataService

Serviciul de autentificare AuthService interacţionează cu soluţia de backend, mai exact cu punctele de acces ale controller-ului cu acelaşi nume AuthController. Totodată acesta se ocupă şi cu deconectarea utilizatorului, unde nu este nevoie de o cerere către backend, şi cu setarea timpului de exprirare a token-ului în urma unei autentificări reuşite şi la reîncărcarea paginii.

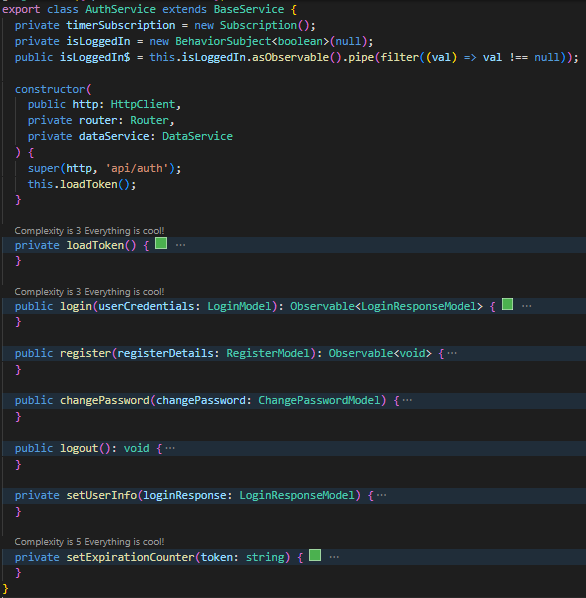


Figura 33: Clasa AuthService

### Modulul person

Modulul person conţine tot ce ţine de crearea, editarea şi ştergerea informaţiilor din arborele genealogic, cât şi pagină pentru vizualizarea informaţiilor unei persoane cu toate legăturile acesteia (copii, părinţi şi căsătorii).

Pentru a realiza o separare între porţiunea care gestionează tot ce ţine de persoană şi porţiunile care se ocupă de legăturile persoanei, în interiorul acestui modul se regăsesc încă 2 module: marriage pentru căsătorii şi parent-child pentru copii şi părinţi, fiecare cu câte un serviciu specific pentru a realiza conexiunea cu aplicaţia backend. Pentru a accesa componentele specifice gestiunii legăturilor, în fişierul modulului person au fost importate printre altele şi modulele marriage şi parent-child. Neavând pagini specifice, aceste 2 module nu au fişier de rutare, ci doar modulul person are.

### Modulul user

În acest modul se regăsesc paginile de autentificare, înregistrare şi de profil al utilizatorului, alături de toate aproape toate componentele utilizate în cadrul acestor pagini. Asemănător modulului person, şi acest modul conţine alte două module: occupation pentru gestiunea ocupaţiilor utilizatorului şi education pentru gestiunea educaţiei, tot cu câte un serviciu specific.

### Modulul genealogy

Acest modul este folosit pentru afişare sub formă de arbore genealogic al persoanelor dintr-un arbore, afişarea listei cu toate persoanele din arbore, cât şi un calendar cu aniversările persoanelor din arbore. Pentru fiecare scop menţionat mai sus există câte o componentă care va fi afişată într-un tab al paginii, care poate fi selectat din partea inferioară.

### Modulul relative

Modulul relative conţine funcţionalităţile de vizualizare a tuturor utilizatorilor recunoscuţi ca rude, căutarea altor utilizatori în aplicaţie pentru a-i adăuga drept rude, şi vizualizarea pe hartă a ultimei locaţii cunoscute pentru fiecare rudă. Similar cu modulul genealogy, şi funţionalităţile menţionate anterior sunt dispuse în cadrul unui tab al paginii pentru rude.

### Modulul notifications

Acest modul conţine doar pagina dedicată notificărilor şi patru componente care reprezintă şabloane pentru cele 4 tipuri de notificări posibile: notificare aniversare, notificare cerere de rudenie, notificare răspuns cerere, notificare modificări în arborele rudei.

### Modulul settings

Modulul settings conţine şi el doar pagina de setări şi trei componente pentru setarea preferinţelor utilizatorului, setarea aspectului şi accesibilităţii aplicaţiei şi schimbarea parolei.

### Modulul support

Modulul de suport este unul dintre cele mai simple module, având o singură pagină dedicată pentru înregistrarea cererii de suport a utilizatorului.

### Modulul intro

Similar cu modulul suport, şi acest modul are doar o pagină care conţine mai multe slide-uri informative pentru utilizatorii noi ai aplicaţiei.

# Funcţionalităţi ale aplicaţiei

## Funcţionalităţi Persoană

### Vizualizare detalii persoană

Pentru vizualizarea detaliilor unei persoane un utilizator poate accesa pagina de detalii a persoanei dată de ruta /person/details/:id, unde id reprezintă identificatorul unic al persoanei. Această rută va fi atribuită mai multor butoane şi carduri de persoană din aplicaţie pentru a o putea accesa, fiind disponibilă numai pentru utilizatorul care gestionează arborele în care se află persoana sau este rudă cu acest utilizator.

În cadrul acestei paginii de detalii sunt mai multe informaţii, însă referindu-ne strict la detaliile persoanei, acestea include: numele şi prenumele, genul, naţionalitatea, religia, ziua de naştere, ziua decesului dacă este cazul, locul naşterii, locul în care trăieşte şi poza persoanei. Aceste informaţii sunt primite de la punctul de acces api/Person/{id} de tip GET. Dacă persoana curentă are atribuit şi un utilizator, aceasta are şi un buton care va redirecţiona către pagina de profil al utilizatorului.

### Adăugare persoană

Adăugarea de persoană se realizează de pe o pagină special creeată în acest sens ce poate fi accesată din lista de persoane din arbore care va fi explicată în cadrul secţiunii 4.3.1 şi având calea /person/create.

În formularul de creere de persoană există diverse câmpuri pentru prenume, nume, gen, naţionalitate, religie, ziua de naştere, ziua decesului, cât şi ţara, judeţul şi oraşul pentru locul de naştere şi locul în care locuieşte. Dintre toate acestea, numele, prenumele, genul şi ziua de naştere sunt obligatorii, restul putând să lipsească.

Pentru selecţia naţionalităţii şi religiei se vor realiza 2 apeluri în prealabil la punctele de acces GET api/common/nationality şi api/common/religion care vor returna pentru acestea toate opţiunile din baza de date sub forma de obiecte CommonObject (explicat la secţiunea 3.5.1). Acestea vor popula cele 2 dropdown-uri pentru selecţie.

După completarea cu informaţiile dorite, la apăsarea butonului Save se va trimite o cerere către punctul de acces api/Person de tip POST cu informaţiile din formular. Aici se va atribui pentru persoana ce urmează să fie creeată identificatorul arborelui genealogic al utilizatorului care doreşte să creeze noua persoană, apoi se va creea în baza de date în tabela Person o nouă entitate, iar în tabela Location încă 2 entităţi. În urma creeării persoanei, utilizatorul va fi automat redirecţionat către pagina de detalii a persoanei.

### Editare persoană

Pentru editarea unei persoane se va accesa aceeaşi pagină ca şi cea destinată creeării unei persoane, însă folosind ruta person/edit/:id, unde id este identificatorul persoanei care se doreşte a fi editată. Această pagină poate fi accesată:

* din pagina de detalii a persoanei, în cazul în care persoana face parte din arborele genealogic al utilizatorului curent, când va fi vizibil un buton de editare;
* din pagina de vizualizare a arborelui genealogic, unde va fi disponibil pentru cardul fiecărei persoane din arbore câte un buton de editare care va fi disponibil tot numai dacă persoana face parte din arborele utilizatorului curent.

La încărcarea paginii se va verifica pe baza cărei rută s-a accesat pagina, astfel încât pentru ruta de editare să se seteze o variabilă care să indice faptul că suntem în stadiul de editare şi să se apeleze punctul de acces GET person/{id} pentru a primi informaţiile existente în baza de date care urmează a fi modificate. Formularul de editare este acelaşi cu cel de creere al unei persoane, cu diferenţa că la editare nu se mai poate schimba genul ales iniţial pentru persoană.

După completarea formularului, utilizaroul va putea salva modificările, lucru ce va trimite o cerere la punctul de acces PUT api/Person/{id} având ataşat informaţiile din formular cu care să se modifice persoana.

### Ştergere persoană

Ştergerea unei persoane poate fi realizată doar din cadrul paginii de detalii a respectivei persoane. Pentru ştergere va fi disponibil un buton doar în cazul în care utilizatorul curent are dreptul de a edita această persoană şi nu are un utilizator ataşat. După apăsarea butonului de ştergere va fi cerută o confirmare pentru a evita ştergerea accidentală a persoanelor. La confirmare se va trimite o cerere către punctul de acces DELETE person/{id}, care apoi va realiza ştergerea tuturor legăturilor persoanei (căsătorii, părinţi şi copii), pentru ca după să se şteargă şi persoana propriu-zisă. În urma ştergerii utilizatorul va fi redirecţionat către lista de persoane.

### Adăugare şi schimbare poză persoană

Adăugarea şi schimbarea de poză a unei persoane poate fi realizată atât din cadrul paginii de detalii a persoanei, cât şi din cadrul paginii de profil al utilizatorului, şi acesta fiind o persoană care are o poză.

În aceste pagini, în cazul posibilităţii de editare, utilizatorul va putea apăsa butonul de lângă poza de profil pentru a deschide un modal de unde va selecta o poză pe care să o adauge sau să o înlocuiască pe cea exitentă.

După selecţia pozei, folosind serviciul ImageCompress, oferit de pachetul ngx-image-compress, se va realiza o compresie a pozei încărcate care va reduce dimensiunea pozei până la un maxim de 400x400 pixeli, având o calitate de compresie de 90%. În acest fel se va reduce cantitatea excesivă de date transmise între aplicaţia de frontend şi backend pentru cererile Http care includ şi imagini.

Prin apăsarea butonului de confirmare a schimbării pozei, se va transmite o cerere către punctul de acces POST api/Person/photo/{personId}, care va salva local un fişier pentru imaginea încărcată şi va creea o nouă entitate cu informaţii despre imaginea salvată, precum numele, dimensiunea şi tipul acesteia, apoi va modifica id-ul imaginii persoanei cu id-ul personId să fie cel al imaginii recent salvate.

Este de menţionat că la momentul schimbării unei poze se va face o verificare a tuturor referinţelor din baza de date către poza salvată local, iar în cazul în care nu s-a găsit niciuna, vechea poză va fi ştearsă automat.

## Funcţionalităţi Legături persoane

### Vizualizare listă căsătorii

Vizualizarea listei de căsătorii a unei persoane este realizată din pagina de detalii a persoanei. În urma unei cereri către punctul de acces api/Marriage/person/{personId} vor fi primite informaţii despre toate căsătoriile pe care le are o persoană.

Pentru a regăsi aceste căsătorii din baza de date, în metoda specifică din serviciul pentru căsătorii se vor realiza 2 interogări, deoarece în tabel sunt 2 câmpuri pentru referinţele persoanelor căsătorite (FirstPersonId şi SecondPersonId). Trebuie verificate ambele câmpuri dacă se potriveşte oricare din cele două cu id-ul persoanei pentru care căutăm şi să întoarcem doar informaţiile despre cealaltă persoană. Urmează să se realizeze o mapare în funcţie de persoana a cărei informaţii au fost întoarse din baza de date şi să se atribuie fiecărei persoane poza sa, dacă are.

După primirea informaţiilor în pagina de detalii a persoanei, lista cu toate căsătoriile vor fi afişate, fiecare dintre acestea într-un card care oferă informaţii despre numele persoanei cu care este sau a fost căsătorită, intervalul de timp în care au fost căsătorite şi o poză a acestei persoane. De asemenea, acest card are link care la apăsare va redirecţiona utilizatorul către pagina de detalii a persoanei selectate.

### Adăugare căsătorie

În pagina de detalii a persoanei, la finalul listei cu căsătorii, în cazul în care utilizatorul are drepturi de editare a acesteia, se va situa un buton prin intermediul căruia un modal va apărea pe ecran pentru adăugarea unei noi căsătorii pentru persoana a cărei pagină de detalii o observăm.

În urma apariţiei acestui modal, o cerere se va trimite către punctul de acces api GET api/ParentChild/parentSpouceOptions/{personId} care va căuta toate posibilele opţiuni din care utilizatorul poate alege pentru o nouă căsătorie. Metoda utilizată pentru găsirea acestor opţiuni este comună cu posibilele opţiuni pentru găsirea opţiunilor pentru părinţi, deoarece aceleaşi constrângeri vor fi aplicate în ambele cazuri.

Metoda specifică din serviciu va returna doar persoanele care nu sunt înrudite bazate pe strămoşi şi cu care nu este deja căsătorit. Pentru a găsi aceste persoane, trebuie mai întâi să fie găsite persoanele care sunt înrudite bazate pe strămoşi, acestea fiind mai apoi excluse din lista tuturor persoanelor din arbore. Persoanele înrudite bazate pe strămoşi sunt toate persoanele care sunt predecesori ai persoanei pentru care căutăm, cât şi toate persoanele care sunt urmaşi ale acestor persoane.

În diagrama următoare este o exemplificare a opţiunilor, fiecare culoare reprezentând: mov - persoana pentru care căutăm, roşu - predecesorii persoanei, galben - toţi urmaşii tuturor predecesorilor persoanei, verde – toate perssoanele care nu sunt înrudite bazate pe strămoşi.

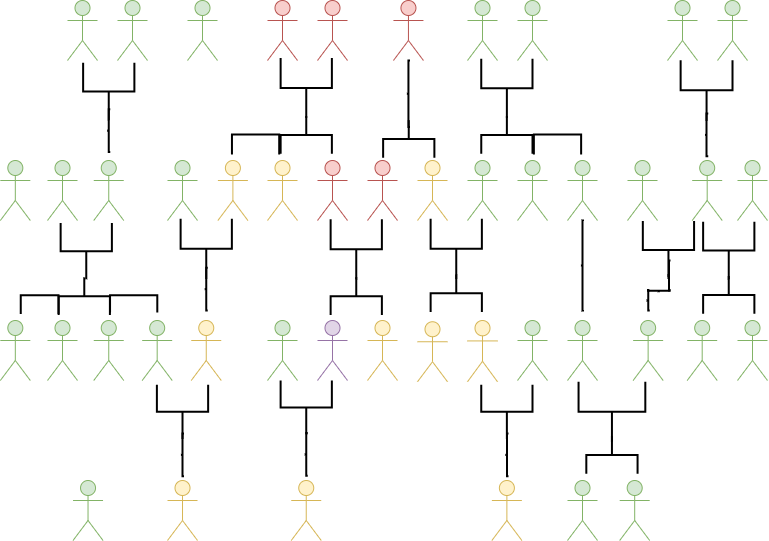


Figura 34: Diagramă opţiuni pentru căsătorie sau părinte

După ce metoda returnează posibilile persoane pentru căsătorie, pentru fiecare dintre acestea se va afişa într-o listă un card selectabil care conţine poza şi numele persoanei. Utilizatorul trebuie să completeze de asemenea şi ziua căsătoriei, după care va putea apăsa butonul de salvare care va trimite o cerere către punctul de acces POST api/Marriage care va creea o nouă entitate în tabelul Marriage din baza de date. La primirea răspunsului de la această metodă, lista cu căsătorii va fi actualizată cu noua căsătorie creeată prin reapelul la punctul de acces GET api/Marriage/person/{personId} descris la secţiunea 4.2.1.

### Modificare căsătorie

În cadrul fiecărui card pentru căsătoriile unei persoane din pagina ei de detalii, pentru un utilizator cu are drepturi de editare, va fi disponibil un buton prin intermediul căruia un modal va apărea pe ecran care va permite editarea unei căsătorii a acestei persoane. În acest modal pot fi observate cele 2 persoane căsătorie cu nume, prenume şi imagine şi se pot edita doar zilele de începere, respectiv încheiere căsătorie, nu şi persoanele. În urma apăsării butonului de salvare se va trimite o cerere cu modificările efectuate către punctul de acces PUT api/Marriage care se va ocupa cu salvarea noilor valori. La primirea răspunsului lista de căsătorii din pagina de detalii a persoanei se va actualiza cu noile informaţii.

### Ştergere căsătorie

Similar cu modificarea unei căsătorii, pentru ştergere va exista în cadrul fiecărui card pentru căsătorie câte un buton de ştergere în cazul în care utilizatorul are permisiuni de modificare a căsătoriei. Pentru a evita ştergerea accidentală a unei căsătorii, în urma apăsării butonului de ştergere va apărea un popup de confirmare a acţiunii. Dacă se confirmă, o cerere către punctul de acces DELETE api/Marriage/{id} va fi trimisă către backend. Aici se va executa ştergerea din baza de date a entiăţii din tabelul Marriage cu id-ul. La primirea răspunsului la cerere, se va actualiza lista cu căsătorii a persoanei.

### Vizualizare listă copii

Vizualizarea listei de copii a unei persoane se realizează în pagina de detalii a persoanei, unde se va executa o cerere către punctul de acces GET api/ParentChild/children /{personId} pentru a primi informaţii despre toţi copii persoanei în cauză.

Pentru a returna aceste toţi copiii unei persoane din baza de date, în metoda specifică din serviciul ParentChild se va realiza o interogare care va returna toate entităţile de tip ParentChild în care id-ul părintelui coincide cu cel al persoanei, cât şi detalii despre copii din aceste entităţi (Child). Aceste entităţi vor fi în continuare mapate la un model generic care să conţină informaţii despre copil, cât şi id-ul entităţii ParentChild din care a fost procurată această informaţie. La fiecare copil găsit, se va căuta şi se ataşa poza stocată local a acestuia, dacă exită.

În urma returnării datelor, lista de copii va fi vizibilă pentru utilizator, fiecare copil având câte un card care conţine numele, prenumele şi imaginea acestuia. Fiecare dintre aceste carduri sunt apăsabile şi vor redirecţiona utilizatorul către pagina de detalii a copilului.

### Vizualizare listă părinţi

Similar cu vizualizarea listei de copii, şi lista de părinţi poate fi obserată din cadrul paginii de detalii persoană. Pentru a afişa aceste informaţii se va efectua un apel la punctul de acces GET api/ParentChild/parents/{personId} care va returna o listă care va conţine până la maxim 2 persoane, şi anume mama şi tatăl persoanei pe a cărei pagină de detalii ne aflăm.

În metoda specifică pentru returnarea părinţilor unei persoane din cadrul serviciului ParentChild, opus faţă de returnarea copiilor din baza de date, se va realiza o interogare care va returna toate entităţile de tip ParentChild în care id-ul copilului coincide cu cel al persoanei, cu detalii despre părinţii din aceste entităţi (Parent). Totuşi şi aceste entităţi vor trece prin acelaşi proces de mapate la un model generic la fel ca informaţiile despre copii, însă vor conţine informaţii despre părinte. Fiecărui părinte găsit i se va ataşa, dacă are, şi poza stocată a acestuia.

În urma returnării datelor, lista de părinţi poate fi văzută de către utilizator în aproape acelaşi fel ca lista de copii. Fiecare părinte are câte un card cu numele, prenumele şi imaginea acestuia, iar în urma apăsării acestuia îl va redirecţiona pe utilizator către pagina de detalii a părintelui.

### Adăugare părinte sau copil

Adăugarea de părinte şi copil sunt foarte asemănătoare, singura diferenţă constând în câmpul în care se pune id-ul persoanei pe a cărei pagină de detalii ne situăm şi id-ul persoanei selectate pe modelul de pe fronend înainte de trimiterea cererii cu acest model la acelaşi punc de acces pentru ambele scenarii.

Pentru un utilizator care are drepturi de editare, la finalul listei de copii a persoanei va fi disponibil un buton de adăugare copii. În lista de părinţi este o situaţie puţin diferită, deoarece butoanele vor fi disponibile doar pentru părintele pe care persoana nu îl are. Dacă în lista de părinţi nu se regăseşte o persoană de gen masculin care să fie recunoscut ca tată, atunci un buton pentru adăugare tată va fi vizibil. Dacă în lista de părinţi nu se regăseşte o persoană de gen feminin care să fie recunoscută ca mamă, atunci un buton pentru adăugare mamă va fi vizibil. Dacă sunt părinţi de ambele genuri, atunci niciun buton nu va fi disponibil.

La apăsarea oricărui dintre butoanele de adăugare tată şi adăugare mamă se va lansa un modal pentru selecţia părintelui şi se va realiza o cerere către backend la punctul de acces GET api/ParentChild/parentSpouceOptions/{personId}. Logica din spatele acestui apel a fost anterio descrisă la secţiunea 4.2.2.

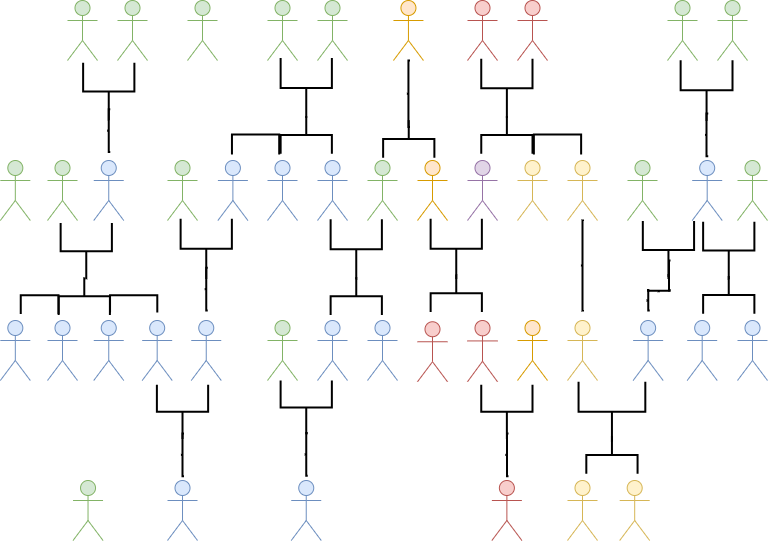
În urma răspunsului la această cerere se va realiza o filtrare bazată pe genul persoanei în funcţie de butonul de adăugare părinte care a fost selectat astfel: dacă a fost apăsat butonul de adăugare tată, se va aplica o filtrare pentru a păstra doar persoanele returnate de sex masculin, iar dacă s-a apăsat butonul de adăugare mamă se va aplica o filtrare pentru păstrarea persoanelor de sex feminin.

După aplicarea filtrării, fiecare persoană va fi afişată sub formă de card selectabil cu nume, prenume şi imagine. Când se detectează selecţia unei persoane, butonul de salvare va fi accesibil, care va produce la apăsare un apel către punctul de acces POST api/ParentChild, având id-ul persoanei selectate drept id-ul părintelui şi id-ul persoanei pe a cărei pagină de detalii ne situăm drep id-ul copilului. În metoda specifică din cadrul serviciului ParentChild se va creea o nouă entitate în baza de date cu informaţiile oferite.

La apăsarea butonului de adăugare copil însă, se efectua o cerere la punctul de acces GET api/ParentChild/childrenOptions/{personId} care va returna toate opţiunile de selecţie drept copil pentru persoana pe a cărei pagină de detalii ne aflăm. Metoda folosită pentru găsirea acestor opţiuni va returna doar persoanele care nu sunt înrudite bazate nici pe descendenţi, nici pe predecesori, cu care nu sunt sau au fost căsătoriţi şi care nu au deja un părinte de acelaşi gen cu persoana cu identificatorul personId. Metoda de găsire a persoanelor care nu sunt înrudite bazate pe descendenţi a fost explicată la secţiunea 4.2.2.

Pentru a găsi persoanele care nu sunt înrudite bazate pe descendenţi, trebuie găsite persoanele care sunt înrudite bazate pe descendenţi, fiind apoi excluse din lista tuturor persoanelor din arbore. Persoanele înrudite bazate pe descendenţi sunt toate persoanele care sunt urmaşi ai persoanei pentru care căutăm, cât şi toate persoanele care sunt predecesori ale acestor persoane. După găsirea persoanelor neînrudite bazate pe descendenţi şi a persoanelor înrudite bazate pe predecesori, se realizează o intersecţie a acestora, apoi o filtrare pentru a exclude şi persoanele care au deja un părinte cu acelaşi gen sau care au fost deja căsătorite cu persoana pentru care se caută opţiunile pentru copii.

În diagrama următoare sunt exemplificate opţiunile, fiecare culoare a persoanelor reprezentând: mov - persoana pentru care căutăm, roşu - predecesorii sau descendenţii direcţi ai persoanei, portocaliu- toţi predecesorii tuturor descendenţilor persoanei, galben - toţi descendenţii tuturor predecesorilor persoanei, albastru – persoanele care au deja părinte de acelaşi gen, verde – toate persoanele cu care nu este înrudită.



Figură 35: Diagramă opţiuni pentru copil

După ce metoda returnează posibilile persoane pentru a fi copil, pentru fiecare persoană va fi afişat câte un card selectabil cu nume, prenume şi imagine. În urma detecţiei selecţiei unei persoane, butonul de salvare va putea fi apăsat pentru a produce un apel către punctul de acces POST api/ParentChild, având id-ul persoanei selectate drept id-ul copilului şi id-ul persoanei pe a cărei pagină de detalii ne situăm drept id-ul părintelui. Fiind acelaşi punct de acces precum cel utilizat la adăugarea unui părinte, va rezulta executarea aceluiaşi algoritm de creere a entităţii ParentChild.

### Ştergere părinte sau copil

Ştergerea unui părinte sau a unui copil sunt foarte similare, deoarece se va realiza utilizând acelaşi punct de acces DELETE api/ParentChild/{id}.

Pentru a putea şterge un copil sau părinte este important ca utilizatorul să aibe permisiunea de editare. În cazul afirmativ, în interiorul fiecărui card al unui părinte din lista de părinţi şi în interiorul fiecărui card al unui copil din lista de copii va exista un buton de ştergere a legăturii respective.

În urma apăsării butonului de ştergere va fi necesară confimarea utilizatorului pentru a evita ştergerea accidentală a unei legături cu un copil sau un părinte. După confirmarea ştergerii se va realiza cerere la punctul de acces de ştergere anterior menţionat, urmând ca o metodă din serviciul specific să se ocupe de ştergerea entităţii de tip ParentChild din baza de date. La primirea răspunsului de la punctul de acces lista de părinţi sau copii va fi actualizată, după caz.

## Funcţionalităţi Arbore

### Vizualizare listă persoane din arbore

Pentru vizualizarea listei de persoane din arborele genealogic al utilizatorului conectat sau a arborelui genealogic al unei rude se poate accesa pagina dată de ruta /tree/list sau /tree/list/:id. Această rută va fi atribuită unui buton în cadrul fiecărei pagini de detalii a persoanei care face parte din arborele genealogic al utilizatorului conectat sau al unui utilizator rudă cu cel conectat, cât şi butonului pentru schimbarea tabului din pagina pentru arborele genealogic.

La momentul acesării acestei pagini, se va trimite o cerere către punctul de acces GET api/Person/treeList/{treeId} în urma căruia se vor returna toate persoanele care au atribuit identificatorul pentru arborele genealogic treeId. Pentru fiecare persoană nu se vor returna toate informaţiile acesteia, ci doar câteva considerate mai importante pentru identificarea şi diferenţierea persoanelor.

În urma returnării răspunsului de la punctul de acces, în cadrul paginii se va popula cu câte un card apăsabil pentru fiecare persoană, acesta conţinând numele prenumele şi poza persoanei. La apăsarea acestui card de persoană, utilizatorul va fi redirecţionat către pagina de detalii a persoanei. În plus, dacă persoana este atribuită şi unui utilizator, acest lucru va fi marcat prin prezenţa unui buton la care este legată pagina de profil a utilizatorului care este reprezentat de persoana respectivă. Dacă utilizatorul atribuit persoanei este chiar utilizatorul autentificat, butonul acestuia va fi marcat diferit faţă de celelalte.

Tot în cadrul acestei pagini se regăseşte şi un câmp special conceput pentru filtrarea listei de persoane în funcţie de numele acestora, cât şi un buton de adăugare persoană, care la momentul apăsării redirecţiona utilizatorul la pagina de creere persoană, descrisă anterio în secţiunea 4.1.2.

### Vizualizare arbore genealogic

Vizualizarea persoanelor sub formă de arbore genealogic este accesibilă la ruta /genealogy/tree şi /genealogy/tree/:id. Această rută este atribuită butonului specific vizualizării arborelui din bara de navigaţie, care este accesibil doar când utilizatorul este conectat, cât şi butonului de selecţie a tabului din pagina pentru arborele genealogic. Vizualizarea arborelui genealogic reprezintă componenta care este afişată în mod implicit de către aplicaţie. Tot în cadrul acestui tab se poate vizualiza şi arborele genealogic al unei rude.

În urma accesării tabului pentru arborele genealogic, o cerere va fi efectuată către punctul de acces GET api/Person/treeData/{treeId}. Aici o metodă din serviciul Person va fi utilizată pentru preluarea din baza de date a informaţiilor despre toate persoanele din arborele al cărui identificator este treeId, cât şi părinţii, căsătoriile şi legătura acestora cu alţi utilizatori-rude. După regăsirea informaţiilor se va realiza o mapare în urma căreia se vor păstra câteva informaţii de bază despre persoană, precum nume prenume, genul, data naşterii, poza şi id-ul; id-urile persoanelor cu are acesta a fost căsătorit, id-ul tatălui şi al mamei acestuia şi posibilul id al utilizatorului care reprezintă această persoană. Aceste informaţii sunt apoi returnate către frontend.

După răspunsul la cerere, datele primite vor fi remapate în funcţie de informaţiile fiecărei persoane la un nou model care va fi folosit la popularea componentei JavaScript FamilyTree, care face parte din pachetul Balkangraph/familytree.js, utilizată la afişarea arborelui genealogic. Înainte de a fi generată acestă componentă se vor seta şabloanele păţilor componente arborelui genealogic care vor fi afişate, folosind o clasă de constante care conţine valorile acestora în format html. Aceste şabloane sunt următoarele:

* defs pentru definţiile de bază ale structurii şi aspectului arborelui;
* field\_0 pentru aspectul şi mărimii scrisului pentru nume;
* field\_1 pentru aspectul zile de naştere al persoanei;
* field\_2 pentru butonul atribuit persoanelor care reprezintă şi utilizatori
* img\_0 pentru încadrarea imaginii persoanei;
* node pentru aspectul cardului unei persoane decedate;
* gene\_male.node pentru aspectul cardului unei persoane de sex masculin;
* gene\_female.node pentru aspectul cardului unei persoane de sex feminin.

După setarea şabloanelor, se va creea obiectul FamilyTree cu configurările pentru acesta:

* acţiuni pe mouse (onClick, onScroll);
* tema arborelui (luminous sau întunecat);
* şablonul care să fie folosit (gene – cel creeat anterior);
* câmpurile pe baza cărora să se caute persoane (name şi born - nume şi ziua de naştere);
* setările pentru toolbar (zoom, layout, fullscreen);
* meniul arborelui care va conţine link-uri pentru pagina de detalii a persoanei, iar dacă utilizatorul are persmisiuni şi pentru pagina de editare persoană;
* legături noduri pentru informaţiile persoanei (field\_0, field\_1, field\_2, img\_0).

După efectuarea configurărilor, obiectul creat FamilyTree va primi datele persoanelor mapate anterior pentru a fi afişate.

După încărcarea datelor, în pagină vor fi afişate persoanele sub formă de arbore, fiecare având câte un card şi legături cu alte persoane. Cardul unei persoane conţine imaginea persoanei, numele şi prenumele, un meniu din care se poate accesa pagina de detalii a persoanei, iar dacă utilizatorul are permisiuni de editare atunci şi pagina de editare a persoanei, un buton care duce la pagina utilizatorului atribuit persoanei respective (dacă există) şi un în funcţie de situaţie un buton de expansiune cardurilor persoanelor rude în imediata vecinătate (părinţi, copii).

Ca funcţionalităţi ale paginii sunt

* mişcarea planşei pe care este afişat arborele în oricare dintre cele 4 direcţii carteziene prin tragerea cu mouse-ul;
* zoom in şi zoom out folosind rotiţa de scroll;
* comutare fullscreen on şi off folosind un buton specific;
* câmp de căutare bazată pe nume şi zi de naştere pentru găsirea unei persoane.

### Vizualizare evenimente din arbore

Vizualizarea evenimentelor din arborele genealogic este accesibilă la ruta /genealogy/events. Această rută este atribuită butonului de selecţie a tabului din pagina pentru arborele genealogic. Această pagină conţine un calendar în care sunt marcate toate aniversările persoanelor din arborele genealogic, şi anume aniversările de zi de naştere şi aniversările de căsătorie.

În urma accesării tabului pentru vizualizarea evenimentelor din arbore, o cerere este trimisă către punctul de acces GET api/Person/events/{treeId} din backend. Ajuns aici, metoda GetEventsInTree din serviciul Person va fi utilizată pentru preluarea din baza de date a tuturor persoanelor care au ziua decesului nesetată, adică încă trăiesc şi îşi pot serba ziua, şi toate căsătoriile care nu au setată ziua încheierii căsătoriei pentru toate persoanele ale căror identificator pentru arbore este treeId. După regăsirea acestor informaţii, se vor realiza mapări pentru fiecare tip de eveniment. Ambele tipuri de evenimente se vor mapa la acelaşi model, acesta având trei câmpuri: pentru setarea zilei evenimentului, tipul evenimentului şi persoanele afectate de acesta. Pentru aniversările de tip zi de naştere se va adăuga într-o listă a persoanelor afectate numai utilizatorul care îşi serbează ziua. Pentru aniversările de tip căsătorie se vor adăuga în lista persoanelor afectate ambele persoane care sunt căsătorite. Informaţiile pentru persoanele afectate sunt numele, prenumele, identificatorul şi imaginea acesteia.

La primirea răspunsului cu evenimentele arborelui de la soluţia backend, acestea sunt remapate la un format care va putea fi adăugat în calendarul din pachetul ionic2-calendar, folosit pentru afişarea evenimentelor. Deoarece se vor afişa şi anii aniversării, datele primite de la backend şi mapate trebuie păstrate neschimbate, şi o nouă listă cu evenimente este folosită pentru adăugarea evenimentelor în calendar. Aceste evenimente adăugate în calendar se modifică bazat pe evenimentele originale în funcţie de anul la care se uită utilizatorul în calendar. S-a realizat şi o filtrare pentru ca atunci când utilizatorul se uită în trecut să nu vadă evenimente înainte de a se întâmpla, spre exemplu dacă persoana X este născută în 2003 şi utilizatorul se uită pe calendarul din 2002, aniversarea zilei de naştere nu va apărea în calendar.

În pagina calendarului ziua curentă va putea fi identificată având fundal verde, iar cea selectată de către utilizator având fundal galben. Dacă o zi are evenimente, aceasta va conţine un dreptunghi albastru care să semnaleze acest lucru. Numărul de evenimente dintr-o anumită zi va fi şi el indicat prin dungi de culoare verde.

În urma selecţiei unei zile cu evenimente, sub calendar va fi pentru fiecare dintre acestea câte o descriere care va conţine numele şi imaginea/imaginile persoanei sau persoanelor care se serbează. Imaginile fiecărei persoană este apăsabilă şi va redirecţiona utilizatorul la pagina de detalii.



Figura 36: Calendar aniversări

## Funcţionalităţi Utilizator

### Înregistrare

Înregistrarea unui utilizator nou se va realiza în pagina de înregistrare de la ruta /user/register, care poate fi accesată folosind butonul de pe pagina de autentificare şi butonul din bara de navigare destinată acesteia. Această pagină este disponibilă doar pentru utilizatorul neautentificat în aplicaţie.

După accesarea paginii de autentificare, două cereri vor fi trimise către punctele de acces GET api/Common/nationality şi api/Common/religion, explicate anterior în secţiunea 4.1.2.

În această pagină utilizatorul va completa informaţii care vor fi utilizate pentru crearea entităţii Person care o va reprezenta în arbore explicate în secţiunea 4.2.1, cât şi informaţii specifice utilizatorului, şi anume adresa de e-mail, numărul de telefon, numele de utilizator şi parola. Există mai multe constrângeri de completare pentru e-mail, nume utilizator şi parolă.

E-mail-ul introdus trebuie atât să aibe un format specific de e-mail, cât şi să nu fie utilizat de un alt utilizator. Prin utilizarea unei directive care implementează AsyncValidator se vor valida în momentul tastării aceste constrângeri prin testarea e-mail-ului cu ajutorul unui regex şi apelul la punctul de access GET api/User/emailAvailable care va determina dacă email-ul este deja utilizat de un alt utilizator, apoi va returna rezultatul.

Numele de utilizator ca să fie valid trebuie să aibe cel puţin 6 caractere şi să nu fie deja utiliza de un alt utilizator, acest lucru verificându-se folosind o altă directivă care va trimite o cerere la punctul de acces GET api/User/usernameAvailable care determină starea de utilizare a acestuia.

Pentru validarea parolei este utilizat un alt regex pentru a verifica să conţină o literă mare, o literă mică, o literă mare, un număr şi minim 8 caractere. De asemenea, utilizatorul va trebui să reintroducă parola în alt câmp pentru a o confima.

În urma completării şi validării tuturor câmpurilor, butonul de înregistrare va putea fi accesat pentru a trimite o cerere către punctul de acces POST api/Auth/register cu informaţiile introduse de utilizator.

Pe partea de backend se vor creea entităţi pentru persoană şi utilizator, folosind clasele Hash şi Salt descrise în secţiunea 3.4.1 pentru a cripta parola aleasă de utilizator.

După confimarea înregistrării cu succes, utilizarorul va fi redirecţionat către pagina de autentificare.

### Autentificare

Autentificarea utilizatorului se realizează de pe pagina cu ruta user/login, care poate fi accesată folosind butonul de pe pagina de înregistrare şi butonul situate în bara de navigare. Această pagină este disponibilă doar pentru utilizatorul neautentificat în aplicaţie.

În cadrul acestei pagini se regăsesc două câmpuri destinate completării cu numele de utilizator şi parola celui care doreşte să se autentifice. După completare, la apăsarea butonului de autentificare o cerere va fi transmisă către punctul de acces POST api/Auth/login.

Pe backend se va realiza o comparare, folosind o metodă din clasa Hash, între parola introdusă de utilizator şi hash-ul şi salt-ul parolei stocate în baza de date. În urma validării, se va genera un JSON Web Token(JWT) utilizând serviciul TokenService (explicat la secţiunea 3.4.2).

Primit JWT-ul de către frontend, acesta va fi depozitat pe dispozitivul utilizatorului folosind serviciul DataService (explicat la secţiunea 3.5.2), cât şi informaţiile despre utilizator din acesta. După stocarea informaţiilor, se va realiza o redirecţionare către pagina dedicate vizualizării arborelui genealogic al utilizatorului.

### Schimbare parolă

Schimbarea de parolă poate fi realizată din pagina de setări a aplicaţiei care poate fi accesată pe ruta /user/settings. În această pagină se regăseşte un buton pentru schimbarea parolei, care odată apăsat va deschide un modal unde i se va cere utilizatorului introducerea parolei curente, noua parolă aleasă şi confirmarea acesteia. Pentru noua parolă există aceleaşi constrângeri de validitate precum cele pentru parola cerută la înregistrarea utilizatorului. După completarea informaţiilor, utilizatorul poate confima modificările pentru a trimite o cerere la punctul de acces PUT api/Auth/changePassword.

Pe backend se va realiza validarea parolei identic cu validarea din procesul de conectare, însă în urma confirmării validităţii se vor creea şi se vor salva în baza de date hash-ul şi salt-ul noii parole.

### Deconectare

Deconectarea utilizatorului se poate realiza din pagina de setări a aplicaţiei, prin apăsarea unui buton care are drept efect ştergerea token-ului stocat pe dispozitivul utilizatorului şi redirecţionarea acestuia pe pagina de autentificare.

De asemenea, deconectarea utilizatorului se realizează şi automat după expirarea JWT-ului stocat, care conţine data exactă în secunde a momentului din care acesta nu mai este valid. Când utlizatorul se conectează şi ori de câte ori utilizatorul conectat reîncarcă o pagină a aplicaţiei, serviciul authService verifică dacă există un JWT stocat local şi dacă acesta mai este valid. În caz afirmativ, se va seta un timer care la momentul expirării va deconecta automat utilizatorul.

### Vizualizare profil utilizator

Pentru vizualizarea profilului utilizatorului conectat sau a altui utilizator al aplicaţiei se poate accesa pagina dată de ruta /user /profile sau /user/profile /:id, unde id reprezintă identificatorul unic al utilizatorului. Aceasta este atribuită mai multor butoane şi carduri pentru utilizator sau rudă din cadrul aplicaţiei pentru a o putea accesa, fiind disponibilă oricărui utilizatorul al aplicaţiei.

În cadrul acestei paginii de profil sunt diverse informaţii înpărţite în 4 categorii în funcţie de specifitate: despre utilizator, despre persoana care reprezintă utilizatorul, despre ocupaţiile utilizatorului şi despre edicaţia utilizatorului. Informaţiile despre persoană sunt aceleaşi care pot fi regăsite şi în pagina de detalii a persoanei atribuite utilizatorului. Informaţiile specifice utilizatorului sunt e-mail-ul, numărul de telefon şi descrierea. Pentru a prelua aceste informaţii se apelează unul dintre următoarele 2 puncte de acces: api/User/info /{username} pentru informaţiile utilizatorului conectat sau api/User/{id} pentru informaţiile oricărui alt utilizator al aplicaţiei. Aceste două căi returnează acelaşi tip de obiect, însă cea de-a doua returnând doar informaţiile permise de utilizatorul al cărui identificator unic este reprezentat de id.

### Editare profil utilizator

Pentru editarea propriului profil, utilizatorul curent conectat va putea apăsa din pagina sa de profil butonul de editare profil disponibil doar pentru acesta, care va produce afişarea pe ecran a unui modal destinate editării profilului.

În acest modal se va putea modifica numărul de telefon, descriere şi e-mail-ul propriu, după care la apăsarea butonului de salvare se va trimite o cerere către punctul de acces PUT api/User/{id} care va salva aceste modificări în baza de date.

## Funcţionalităţi Ocupaţie

### Vizualizare listă ocupaţie

Vizualizarea listei de ocupaţii a unui utilizator se realizează din pagina de profil a acestuia. În urma cererii către unul dintre punctele de acces api/User/info /{username} sau api/User/{id} menţionate la secţiunea 4.4.5, vor fi primite şi informaţii despre toate ocupaţiile pe care le-a avut sau încă le are.

La primirea informaţiilor de la backend, lista cu toate ocupaţiile va fi afişată, fiecare dintre acestea având câte un card în care se specifică meseria practicată, locul de muncă şi perioada în care acesta a lucrat.

### Adăugare ocupaţie

În pagina de profil a utilizatorului, în cazul în care este pe pagina cu propriul profil de utilizator, la finalul listei cu ocupaţii se va situa un buton prin intermediul căruia un modal va apărea pe ecran pentru a îşi putea adăuga o nouă ocupaţie.

În interiorul acestui modal, sunt patru câmpuri care pot fi completate de către de utilizator, acestea fiind: numele postului de lucru, numele firmei în cadrul căreia lucrează sau a lucrat, ziua în care a început să lucreze şi ziua în care a încheiat să lucreze. Dintre aceste câmpuri doar ziua de încheiat să lucreze este opţională. După completarea informaţiilor necesare, utilizatorul poate apăsa butonul de salvare care trimite o cerere la punctul de acces POST api/Occupation cu datele despre ocupaţie, de unde se va creea o nouă entitate de tip Occupation care va referenţia utilizatorul care a trimis cererea. La primirea confirmării de creere cu succes, lista de ocupaţii va fi actualizată cu noile informaţii.

### Editare ocupaţie

În cadrul fiecărui card pentru ocupaţiile unui utilizator din pagina sa proprie de profil va fi disponibil câte un buton care la apăsare va declanşa apariţia pe ecran a unui un modal care va permite editarea ocupaţiei utilizatorului. Acest modal este acelaşi cu cel folosit la crearea unei ocupaţii, putându-se edita toate informaţiile din cadrul acestuia, singura diferenţă constând în faptul că în urma apăsării butonului de salvare se va trimite o cerere cu modificările ocupaţieie către alt punct de acces, şi anume PUT api/Occupation utilizat la salvarea noilor câmpuri pentru ocupaţia existentă. La primirea răspunsului lista de ocupaţii din pagina de profil se va actualiza cu noile informaţii.

### Ştergere ocupaţie

Asemănător cu editarea unei ocupaţii, în cazul în care utilizatorul se află pe pagina propriului profil, pentru ştergere va exista în cadrul fiecărui card de ocupaţie câte un buton de ştergere. Pentru evitarea potenţialelor ştergeri accidentale, în urma apăsării butonului de ştergere a unei ocupaţii va fi necesară confirmarea intenţiei într-un popup care va apărea pe ecran. După confirmare se trimite o cerere către backend la punctul de acces DELETE api/Occupation /{id}. Din acest loc se va efectua ştergerea din ocupaţiei cu identificatorul dat de id din baza de date. În urma primirii răspunsului la cerere, se va actualiza lista cu ocupaţii a utilizatorului.

## Funcţionalităţi Educaţie

### Vizualizare listă educaţii

Vizualizarea listei de edutaţie a unui utilizator se realizează din pagina de profil a acestui utilizator. Informaţiile despre educaţii sunt disponibile în urma cererii către unul dintre punctele de acces GET api/User/info /{username} sau GET api/User/{id} menţionate la secţiunea 4.4.5.

La recepţionarea răspunsului de la aplicaţia de backend, în interiorul cardului specific listei educaţii, pentru fiecare educaţie din lista de educaţii primită, se va fi afişa câte un card care va conţine informaţii despre nivelul de educaţie absolvit, instituţia de învăţământ la care a fost şi perioada în care acesta a frecventat.

### Adăugare educaţie

În pagina propriului profil de utilizator, la finalul listei cu de educaţii este un buton utilizat pentru adăugare de noi educaţii. La apăsarea acestuia, un modal va apărea pe ecran în care se vor putea completa informaţii despre noua educaţie care doreşte să se adauge.

După apariţia modalului, va fi trimisă o cerere către punctul de acces GET api/Education/levels, acesta returnând o listă de tipul CommonObject cu toate posibilele opţiuni de selecţie ale nivelului de educaţie.

În modal se regăsesc patru câmpuri care pot fi completate, şi anume: numele instituţiei de învăţământ, nivelul de educaţie care are ca opţiuni valorile returnate de către punctul de access GET api/Education/levels, ziua începerii studiilor şi ziua încheierii studiilor la unitatea de învăţământ respectivă. Dintre aceste câmpuri doar ziua de începere a studiilor este obligatorie. În urma completării formularului, la apăsare butonului de salvare se va trimite o cerere la punctul de acces POST api/Education, în urma căreia se va creea o nouă entitate de tip Education care ve fi legată de utilizatorul emitent al cerererii. După primirea răspunsului că educaţia a fost creeată cu succes, lista de educaţii este actualizată.

### Editare educaţie

Pentru fiecare card al unei educaţii a utilizatorului care îsi vizionează propria pagină de profil, va exista câte un buton pentru editarea respectivei educaţii. La apăsarea acestuia, un modal va apărea pe ecran şi se va efectua o nouă cerere către punctul de acces GET api/Education/levels descris la secţiunea anterioară.

Pentru crearea unei educaţii a fost utilizat acelaşi modal, având aceleaşi câmpuri care pot fi editate. Pentru confirmarea salvării modificărilor se va apăsa butonul de salvare se va trimite o cerere cu informaţiile modificate ale educaţiei către punctul de acces PUT api/Education care va salva noile atribute ale educaţiei existente. În urma primirii răspunsului de salvare cu success de la backend, în pagina de profil se va actualiza lista de educaţiii cu noile informaţii.

### Ştergere educaţie

Când utilizatorul se află pe pagina propriului profil, similar cu butonul de editare, va exista şi un buton de ştergere a unei educaţii pentru fiecare card care reprezintă o educaţie a utilizatorului. În vederea prevenirii ştergerilor accidentale informaţii, un popup care va necesita confirmarea utilizatorului va apărea în urma apăsării butonului de ştergere a unei educaţii. În cazul confirmării ştergerii, se va trimite o cerere către punctul de acces DELETE api/Edication /{id}, de unde se va efectua ştergerea propriu-zisă din baza de date a educaţiei cu identificatorul dat de id.

După confirmarea ştergerii, lista de carduri de educaţii din pagina de profil a utilizatorului se va actualiza.

## Funcţionalităţi Rude

### Căutare utilizatori

Pentru căutarea de utilizatori care folosesc aplicaţia se poate accesa pagina dată de ruta /relative/search. Această rută este accesibilă folosind butonul pentru schimbare a tabului din pagina de rude.

În pagină există un câmp pentru căutarea persoanelor care poate fi folosit pentru a filtra utilizatorii bazat pe numele şi prenumele acestora. După o secundă de la introducerea de caractere în bara de căutare se va executa un apel către punctul de acces GET api/User/find care va returna primii 15 utilizatori găsiţi pe baza căutării.

Utilizatorii returnaţi de către apel vor fi afişaţi sub bara de căutare, fiecare având propriul card cu informaţii precum nume, prenume, imagine şi zi de naştere. Aceste carduri sunt şi apăsabile, la acţionarea lor utilizatorul fiind redirecţionat către pagina de profil al utilizatorului respectiv.

Dacă în urma primului apel nu s-au returnat toate persoanele care au fost regăsite în baza de date după aplicarea filtrelor, când utilizatorul navighează până la finalul listei cu persoane returnate se va efectua un nou apel către acelaşi punct de acces, dar de această dată returnându-se următorii 15. La modificarea filtrului pentru utilizatori, lista acestora se va reseta, astfel iar primii 15 utilizatori vor fi returnaţi de backend. Funcţionalitatea de încărcare a mai multor obiecte când se ajunge la finalul paginii se numeşte şi infinite scroll, şi a fost implementată folosind un filtru care să menţioneze câte obiecte să întoarcă, peste câte dintre acestea să sară şi numele bazat pe care se face căutarea.

### Cerere rudenie

Pe pagina de profil a utilizatorilor se poate adăuga şi o cerere de rudenie între utilizatori. Imediat după accesarea paginii de profil un apel se va face la punctul de acces GET api/Relative/check/{userId}. Acesta va verifica în ce stare este utilizatoul pe a cărui profil ne situăm faţă de utilizatorul conectat în aplicaţie şi va returna unul din posibilele verdicte:

* Relatie de rudenie: ambele persoane sunt deja rude, astfel butonul de adăugare cerere nu va fi vizibil deoarece nu se poate adăuga o nouă cerere pentru rudele cunoscute;
* S-a trimis o cerere de rudenie: nici în acest caz nu va apărea butonul, fiindcă nu ar avea rost să se mai trimită o a doua cerere;
* Fără relaţie de rudenie: numai în acest caz butonul va fi vizibil pentru adăugare cerere

Prin apăsarea butonului de adăugare rudă va apărea un modal pentru a selecta referinţa utilizatorului care se doreşte să fie adăugat ca rudă cu persoana care îl va reprezenta în arborelei utilizatorului curent autentificat.

După momentul apariţiei acestui modal, o cerere se va tansmite către punctul de acces GET api/Person/withoutRelative/{treeId}. În serviciul Person se va apela o metodă care va filtra persoanele din arborele genealogic determinat prin identificatorul treeId pentru a le returna doar pe acelea care nu au deja existentă vreo referinţă către un alt utilizator. În urma determinării persoanelor care nu au referinţă către utilizatori, acestea vor fi returnate către frontent pentru a fi folosită.

Cu răspunsul primit de la backend se va afişa câte un card selectibil pentru fiecare persoană fără rudă ataşată în arborele utilizatorului. Având o persoană selectată în listă se poate apăsa butonul de save care va produce o nouă cerere către punctul de access POST api/Relative/request. Aici se va apela o metodă din serviciul Relative care va creea în baza de date o nouă entitate de tipul Request. La primirea răspunsului de creere cu succes, butonul de trimitere cerere de rudenie va dispărea, deoarece acum există deja o cerere de rudenie în aşteptare pentru utilizatorul pe a cărui pagină ne situăm.

### Răspuns cerere de rudenie

Toate cererile de rudenie pe care un utilizator le-a primit pot fi găsite în pagina de notificări a aplicației care se află pe ruta /notifications. Acest loc poate fi accesat utilizând butonul specific din bara de navigație.

Fiecare cerere de rudenie este afișată sub forma unui card apăsabil care conține numele și prenumele utilizatorului care a trimis cererea, poza de profil a acestuia și două butoane, unul pentru acceptarea cerererii, iar celălalt pentru refuzul acesteia. La apăsarea cardului cererii, se va realiza o redirecționare către pagina de profil a utilizatorului emitent al cererii pentru a putea vedea mai multe informații despre acesta înainte de a accepta sau a refuza cererea acestuia.

La apăsarea butonului de acceptare a cererii de rudenie, un modal va apărea pe ecranul utilizatorului curent pentru a selecta persoana din arborere propriu care reprezinta utilizatorul emitent al cereri. Similar cu selecția persoanei pentru creearea unei cereri, în acest modal este câte un card pentru fiecare persoană din arborele propriu care încă nu este reprezentată de niciun utilizator. După selecția unei persoane, se poate apăsa butonul de salvare care va trimite o cerere către punctul de acces PUT api/Relative/request/{id}. Acest punct de acces pentru ambele cazuri, de refuz sau accepare a cererii, actualizându-se entitatea Request care identificatorul dat de id bazat pe răspunsul utilizatorului. În cazul acceptării cererii de rudenie, se vor creea încă două entități Relative pentru a păstra pentru fiecare utilizator informațiile despre restul utilizatorilor care îi sunt rude.

La apăsarea butonului de refuzare a cererii, un popup pentru confirmarea alegerii va apărea pe ecran. După confirmare se va realiza aceeași cerere către punctul de acces PUT api/Relative/request/{id}, în urma căruia doar salvându-se răspunsul la cerere.

Utilizatorul emitent al cererii va putea să vadă răspunsul la cerere tot în pagina de notificări, unde va fi un card care va specifica răspunsul utilizatorului receptor. Dacă receptorul a acceptat cererea, un buton de vizualizare a arborelui genealogic al receptorului va fi disponibil în interiorul cardului. Neținând cont de ce răspuns a fost oferit, există și un buton care este utilizat pentru a șterge răspunsul din pagina de notificări. La apăsarea acestui buton se va trimite o cerere către punctul de acces DELETE api/Relative/request/{id} pentru a șterge entitatea Request din baza de date, aceasta îndeplinindu-și scopul de cerere.

### Vizualizare listă de rude

Vizualizarea listei cu toți utilizatorii recunoscuți ca rude se poate realiza din cadrul paginii determinate de ruta /relative/list, aceasta fiind accesibilă folosind butonul specific din meniul de navigare al aplicației și butonul de selecție a tabului din pagina de rude. Componenta pentru afișarea listei de rude este cea afișată în mod implicit în cadrul paginii de rude.

.La momentul acesării acestei componente, se va realiza un apel cătrepunctul de acces GET api/Relative/{userId}. La accesarea acestuia se va căuta în tabelul Relative din baza de date toate entitățile care au în câmpul PrimaryId identificatorul utilizatorului care a trimis cererea, apoi returnând informațiile despre utilizatorul recunoscut ca rudă.

După returnarea rudelor, în conținutul paginii se va afișa pentru fiecare dintre acestea câte un card apăsabil care va redirecționa utilizatorul conectat către pagina de profil a rudei. Din cadrul paginii de profil a rudei acesta va putea accesa lista de persoane din arborele său genealogic sau chiar vizualiza arborele genealogic al acestuia.

### Localizare rude

Localizarea rudelor pe hartă se poate observa utilizând ruta /relative/location. Această rută poate fi accesată utilizând butonul de selecție a localiției a tabului din pagina de rude.

Odată ajunși pe pagină, aplicația îi va cere utilizatorului permisiunea de a accesa locația sa folosind gps-ul dispozitivului, urmând apoi să se inițializeze harta folosită din pachetul npm leaflet.

La inițializare se va seta zona vizualizată inițial să fie deasupra României și aspectul panourilor afișate pe ecran, după care se va actualiza locația utilizatorului curent pe hartă, pentru acesta setându-se o iconiță verde și un popup care să îi ofere informații despre locația sa. Urmează să se trimită un apel către punctul de acces GET api/Relative/position/{userId} pentru a primi locațiile rudelor.

În backend se vor căuta toate rudele utilizatorului cu identificatorul userId, care au permis utilizarea gps-ului de către aplicație cel puțin o dată. După identificarea acestor rude, se va realiza o mapare care va include informații despre ultima poziție cunoscută a utilizatorului, data și ora ultimei actualizări a locației, numele, prenumele și poza utilizatorului. Aceste informații vor fi apoi returnate pentru soluției de frontend.

Odată primite de către frontend pozițiile rudelor, se va actualiza harta prin adăugarea sau actualizarea acestora pe hară, fiecare rudă având câte o iconiță de culoare mov și un popup accesibil la apăsare care conține numele, prenumele, poza și momentul ultimei actualizări a rudei respective.

De asemenea, de fiecare dată când aplicaţia va detecta schimbarea poziţiei geografice a acestuia, un apel cu noile sale coordonate va fi efectuat la punctul de acces PUT /api/User/position/{id}. Din acest punct se va executa o metodă în urma căreia noua poziţie a utilizatorului conectat va fi actualizată cu noile coordonate şi cu momentul de timp când s-a efectuat această actualizare.

Printre funcționalitățile hărții se numără posibilitatea mutării acesteia la tragerea cu mouise-ul în oricare dintre cele 4 direcţii carteziene cu mouse-ul și zoom-in/zoom-out la acționarea rotiței mouse-ului.

## Aparenţă şi accesibilitate

### Editare setări preferenţiale

Editarea setărilor preferenţiale se poate realiza din pagina de setări a aplicaţiei care poate fi accesată utilizând ruta /user/settings sau butonul de setări din bara de navigaţie.

Există patru setări preferenţiale pe care le poate alege utilizatorul, şi anume: notificarea de modificări în arborii rudelor, notificarea de aniversări, împărtăşirea e-mail-ului şi a numărului de telefon cu celelalte rude şi împărtăşirea locaţiei curente cu celelalte rude. Toate aceste 4 setări pot fi activate sau dezactivate prin acţionarea unui comutator atribuit lor. După modificarea a cel puţin uneia dintre aceste preferinţe, un buton de salvare va apărea care va permite salvarea opţiunilor. La acţionarea acestui buton un apel se va efectua către punctul de acces PUT api/User/settings/{id}, în urma căruia preferinţele utilizatorului vor fi salvate în baza de date.

De fiecare dată când utilizatorul curent va dori să acceseze pagina de profil a unui alt utilizator, în metoda care se ocupă cu preluarea datelor din baza de date se va verifica dacă utilizatorul a căror date se returnează permite să împartă informaţii despre e-mail şi număr de telefon. Dacă acesta nu permite acest lucru, nu vor fi returnate cele 2 date menţionate. Pe acelaşi principiu se bazează şi restul setărilor preferenţiale. În funcţie de valorile acestora, utilizatorul va putea sau nu să vadă în pagina de notificări dacă a sunt aniversări în ziua curentă sau actualizări în arborii genealogici ai rudelor, iar în pagina de localizare locaţia altor rude în funcţie de alegerile lor.

### Editare aspect şi accesibilitate

În pagina de setări se poate modifica şi aspectul general al aplicaţiei, utilizatorul putând modifica 4 butoane de comutare. Acestea au următoarele funcţii de comutare: mod întunecat, inversare a culorilor, vizualizare în alb-negru şi evidenţare a link-urilor şi butoanelor cu acţiuni.

La acţionarea oricăruia dintre aceste butoane de comutare, valorile acestora vor fi salvate prin intermediul serviciului DataService pe dispozitivul utilizatorului, astfel păstrându-se aspectul la următoarele accesări ale paginii. Componenta de la cel mai înalt nivel, AppComponent, observă aceste modificări pentru aspect pentru a le aplica sub formă de clase css asupra tot conţinutului aplicaţiei.

Ca accesibilitate avem şi caracteristicile non-funcţionale de notificări prin toaster, modificare dinamică a titlului paginii şi receptivitatea aplicaţiei. Notificările folosind toastere vor apărea în partea inferioară a aplicaţiei pentru a confirmarea executării cu succes a unei modificări în baza de date, pentru a semnala o eroare sau pentru a informa sau atenţiona utilizatorul.

Folosind serviciul Title din pachetul @angular/platform-browser, la fiecare modificare a rutei se va modifica şi titlul afişat în tab-ul paginii web.

Receptivitatea aplicaţiei constă în capacitatea acesteia de a se mula fără impedimente pe mai multe dimensiuni ale ecranului, păstrând un aspect vizual plăcut chiar şi pe dispozitivele mobile.

### Selectare limbă

Selectarea limbii se va realiza din pagina de setări a aplicaţiei, unde există un meniu derulant cu 2 opţiuni pentru selecţia limbii: română şi engleză. Limba implicită a aplicaţiei este engleză, însă în urma schimbării valorii din meniul derulant, se va salva noua limbă folosind serviciul DataService pe dispozitivul utilizatorului, pentru a fi folosită noua limbă aleasă ori de câte ori utilizatorul accesează site-ul.

La schimbarea limbii serviciul de traducere observă acest lucru, modificând imediat şi acesta traducerile aproape tuturor informaţiilor de pe site.

Printre informaţiile care se modifică în funcţie de limba selectată se numără: conţinutul textual al paginilor aplicaţiei, butoanele din pagini, butoanele din bara de navigare, mesajele din notificările toaster, antetul paginii, titlul din tabul browser-ului, butoanele specifice taburilor din pagina arborelui genealogic si pagina rudelor, butoanele din interiorul arborelui genealogic, descrierile de evenimente din calendarul de aniversări, lunile şi zilele săptămânii din calendarele aplicaţiei, popup-urile ataşate utilizatorilor din pagina de localizare a rudelor, popup-urile de confirmare, mesajul din loader-ul aplicaţiei şi cererea de suport trimisă.

### Notificări

Notificările aplicaţiei se regăsesc în pagina de notificări de pe ruta /notifications care poate fi accesată folosind butonul din meniul de navigare. La accesarea acestei pagini se va trimite o cerere către punctul de acces GET api/User/notifications/{userId} care va creea notificări de 4 tipuri: modificări în arborii rudelor, aniversări de zi de naştere sau căsătorie în ziua curentă, cereri de rudenie (secţiunea 4.7.2), şi răspuns la cererile de rudenie(secţiunea 4.7.3).Toate aceste notificări vor fi dispuse sub formă de carduri.

Pentru notificarea de modificări în arborul genealogic al rudei, se va verifica pentru ruda utilizatorului dacă există persoane sau legături între persoane care au fost adăugate sau modificate mai recent decât ultima verificare a rudei respective. În cazul în care există, acestea vor fi adăugate într-o listă, care va conţine informaţii despre persoanele afectate şi tipul modificării. În pagină un card de notificare modificări va conţine informaţii cu toate modificările din arborele unei rude, acesta având şi un buton pentru înlăturarea notificării. La apăsarea acestuia se va trimite o cerere la punctul de acces PUT api/Relative/markChanges/{relativeId} care va actualiza cu data curentă momentul ultimei verificări a modificărilor rudei cu identificatorul relativeId.

Pentru notificarea de aniversări zile de naştere în backend se vor determina persoanele ale căror aniversare şi căsătoriile care au fost realizate în ziua curentă. Pe partea de frontend se va afişa câte un card care va conţine informaţii fie despre aniversarea zilei de naştere a unei persoane, fie despre aniversarea nunţii a două persoane.

O dată la un minut, aplicaţia verifică dacă exită noi notificări pentru utilizator, afişând în dreptul butonului pentru notificare din meniul de navigare un contor.

### Suport prin email

Există o pagină dedicată pentru suport email care poate fi accesată folosind butonul specific din bara de navigare pentru a accesa ruta /support. Pe această pagină există un formular cu trei câmpuri pe care trebuie să le completeze un utilizator pentru a trimite o cerere de suport în aplicaţie. Cele trei câmpuri sunt: tipul cererii care are un meniu derulant cu 4 opţiuni, specificul cererii şi descrierea acesteia.

În urma completării cererii se apela punctul de acces POST api/User/support, de unde va fi executat în cadrul serviciului EmailService procesul de trimitere e-mail folosind serviciul SMTP. Această cerere va fi trimisă către adresa de e-mail a aplicaţiei [contact.genealogy.tree@gmail.com](mailto:contact.genealogy.tree@gmail.com), care va avea trecut în CC e-mail-ul utilizatorului emitent. Conţinutul e-mail-ului este în limba selectată de către utilizator în aplicaţia de frontend.

### Pagina de introducere

Pagina de introducere este accesibilă la ruta /intro, la care se poate ajunge utilizând butonul din meniul de navigare. Pe acestă pagină sunt afişate 4 slidere cu scop informativ pentru un utilizator nou.

În cazul în care utilizatorul nu a mai accesat site-ul, acesta va fi imediat redirecţionat la pagina cu introducerea, urmând să se depoziteze pe dispozitivul utilizatorului un flag cu ajutorul serviciului DataService care să semnaleze că a văzut introducerea.

# Concluzii

Autorul prezintă concluziile sale…

# Bibliografie

Bibliografia va fi ordonată alfabetic dupa eticheta fiecărei element (de ex. DOOM05 în lista de mai jos este o etichetă). Etichetele materialelor consultate vor fi formatate folosind:

* primele litere ale primului autor urmate de cele două cifre semnificative ale anului apariției materialului, sau
* dintr-un acronim popular al lucrării respective, urmat din nou de cele două cifre semnificative ale anului apariției.

[DOOM05] – *Dicţionarul ortografic, ortoepic şi morfologic al limbii române*, Editura Univers Enciclopedic, Bucureşti, 2005

# Referințe web

Recomandăm și aici respectarea regulilor enunțate pentru secțiunea 5.

[Alm08] – Pedro de Almeida, Patrik Fuhrer, Documentation Guidelines for Diploma and Master Thesis, Universitatea din Fribourg, Elveția, 2008, disponibil on-line la adresa http://diuf.unifr.ch/drupal/softeng/teaching/guidelines

[Olt07] – Th. Olteanu, C. Albu, *Ghid pentru redactarea lucrării de diplomă sau a disertaţiei de masterat*, Universitatea Română de Arte și Științe „Gheorghe Cristea”, 2007, disponibil via web la adresa http://www.ugc.ro/tpl/GHID REDACTARE DIPLOMA LICENTA.pdf

<https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/>

<https://docs.microsoft.com/en-us/dotnet/core/introduction>

https://docs.microsoft.com/en-us/ef/core/

<https://en.wikipedia.org/wiki/MySQL>

<https://angular.io/features>

<https://angular.io/guide/what-is-angular>

<https://en.wikipedia.org/wiki/TypeScript>

<https://developer.mozilla.org/en-US/docs/Web/JavaScript>

<https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics>

<https://sass-lang.com/>

<https://git-scm.com/>

<https://www.c-sharpcorner.com/article/repository-design-pattern-in-asp-net-mvc/>

<https://www.c-sharpcorner.com/UploadFile/b1df45/unit-of-work-in-repository-pattern/>

<https://docs.microsoft.com/en-us/dotnet/core/extensions/dependency-injection>

<https://www.c-sharpcorner.com/UploadFile/8911c4/singleton-design-pattern-in-C-Sharp/>

<https://docs.automapper.org/en/stable/Getting-started.html>

<https://docs.microsoft.com/en-us/aspnet/core/security/authentication/?view=aspnetcore-6.0>

<https://docs.microsoft.com/en-us/aspnet/core/tutorials/getting-started-with-swashbuckle?view=aspnetcore-6.0&tabs=visual-studio>

<https://github.com/auth0/angular2-jwt>

<https://github.com/BALKANGraph/FamilyTreeJS>

<https://github.com/ionic-team/ionic-storage>

<https://devdactic.com/ionic-5-calendar-modal/>

<https://leafletjs.com/examples/quick-start/>

<https://github.com/dfa1234/ngx-image-compress>

<https://www.codeandweb.com/babeledit/tutorials/how-to-translate-your-angular7-app-with-ngx-translate>

<https://www.educative.io/blog/rx-js-angular>

<https://swiperjs.com/>

https://docs.microsoft.com/en-us/visualstudio/ide/?view=vs-2022

https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver16

# Codul sursă

În această anexă se adaugă codul sursă al aplicației…

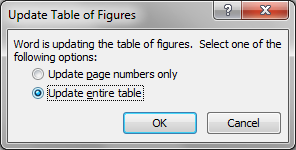


Figura 1. Actualizarea întregului tabel

# Site-ul web al proiectului

https://github.com/bogdandr14/Genealogy-Tree

# CD / DVD

Autorul atașează în această anexă obligatorie, versiunea electronică a aplicației, a acestei lucrări, precum și prezentarea finală a tezei.



# Index

B

Bibliografie 9

C

CUPRINSUL xi

D

Dimensiuni 3

F

Figuri 4

Formulele matematice 4

I

Ilustrațiile 4

L

Legenda 6

LISTA FIGURILOR xii

LISTA TABELELOR xiii

R

Referințe web 10

S

Structura documentului 2

T

Tabele 5