|  |  |  |
| --- | --- | --- |
|  | 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**

1.1 Scopul 1

1.2 Motivația 1

1.3 Descriere generală a soluţiilor 1

**2** **FUNDAMENTE TEORETICE ŞI TEHNICE 2**

2.1 Tehnologii utilizate 2

*2.1.1* *Limbajul C# 4*

*2.1.2* *Platforma de dezvoltare .NET 5 4*

*2.1.3* *Entity Framework Core 4*

*2.1.4* *Limbaj de interogare structurat - MySQL 4*

*2.1.5* *Platforma Unghiular 4*

*2.1.6* *Cadrul Ionic 4*

*2.1.7* *Limbajul TypeScript 4*

*2.1.8* *Limbajul JavaScript 4*

*2.1.9* *HTML(HyperText Markup Language) 4*

*2.1.10* *SASS/SCSS(Syntactically Awesome Style Sheets) 4*

*2.1.11* *Sistemul de versionare Git 4*

2.2 Şabloane folosite 3

*2.2.1* *Repository 4*

*2.2.2* *Unit of Work 4*

*2.2.3* *Dependency Injection 4*

*2.2.4* *Singleton 4*

2.3 Pachete NuGet folosite 3

*2.3.1* *AutoMapper 4*

*2.3.2* *AspNetCore 4*

*2.3.3* *EntityFrameworkCore 4*

*2.3.4* *Swashbuckle 4*

2.4 Pachete npm folosite 4

*2.4.1* *Auth0/angular-jwt 4*

*2.4.2* *Balkangraph/familytree.js 4*

*2.4.3* *Ionic/angular 4*

*2.4.4* *Ionic/storage 4*

*2.4.5* *Ionic2-calendar 4*

*2.4.6* *Leaflet 4*

*2.4.7* *Ngx-image-compress 4*

*2.4.8* *Ngx-translate 4*

*2.4.9* *Rxjs 4*

*2.4.10* *Swiper 4*

2.5 Medii de dezvoltare 4

*2.5.1* *SQL Server Management Studio 4*

*2.5.2* *Visual Studio 2019 4*

*2.5.3* *Visual Studio Code 4*

2.6 Ilustrațiile 4

*2.6.1* *Figurile 4*

*2.6.2* *Tabelele 4*

*2.6.3* *Legenda (unei figuri/tabele) 6*

**3** **Sistemul de gestiune a arborelui genealogic 7**

3.1 Cerinţe 7

3.2 Cazuri de utilizare 7

3.3 Structura bazei de date 7

3.4 Arhitectura Soluţiilor 7

*3.4.1* *Arhitectura aplicaţiei backend 4*

*3.4.2* *Arhitectura aplicaţiei frontend 4*

**4** **Funcţionalităţi ale aplicaţiei 7**

4.1 CRUD Persoană 7

*4.1.1* *Vizualizare detalii persoană 4*

*4.1.2* *Adăugare persoană 4*

*4.1.3* *Editare persoană 4*

*4.1.4* *Ştergere persoană 4*

*4.1.5* *Adăugare şi schimbare poză persoană 4*

4.2 CRUD legături persoane 7

*4.2.1* *Vizualizare căsătorii 4*

*4.2.2* *Adăugare căsătorie 4*

*4.2.3* *Modificare căsătorie 4*

*4.2.4* *Ştergere căsătorie 4*

*4.2.5* *Vizualizare părinţi 4*

*4.2.6* *Vizualizare copii 4*

*4.2.7* *Adăugare părinte sau copil 4*

*4.2.8* *Ştergere părinte sau copil 4*

4.3 Vizualizare şi funcţionalităţi arbore 7

*4.3.1* *Vizualizare lisă persoane 4*

*4.3.2* *Vizualizare arbore genealogic 4*

*4.3.3* *Vizualizare calendar evenimente din arbore 4*

4.4 Funcţionalităţi utilizator 7

*4.4.1* *Înregistrare 4*

*4.4.2* *Autentificare 4*

*4.4.3* *Conectare 4*

*4.4.4* *Schimbare parolă 4*

*4.4.5* *Deconectare 4*

*4.4.6* *Vizualizare profil utilizator 4*

*4.4.7* *Editare profil utilizator 4*

4.5 CRUD ocupaţie 7

*4.5.1* *Vizualizare ocupaţii utilizator 4*

*4.5.2* *Adăugare ocupaţie 4*

*4.5.3* *Editare ocupaţie 4*

*4.5.4* *Ştergere ocupaţie 4*

4.6 CRUD educaţie 7

*4.6.1* *Vizualizare educaţii utilizator 4*

*4.6.2* *Adăugare educaţii 4*

*4.6.3* *Editare educaţie 4*

*4.6.4* *Ştergere educaţie 4*

4.7 Funcţionalităţi rude 7

*4.7.1* *Căutare utilizatori 4*

*4.7.2* *Cerere de rudenie 4*

*4.7.3* *Răspuns cerere de rudenie 4*

*4.7.4* *Vizualizare listă de rude 4*

*4.7.5* *Vizualizare detalii rudă 4*

*4.7.6* *Vizualizare arbore rudă 4*

*4.7.7* *Localizare rudă 4*

4.8 Aparenţă şi accesibilitate 7

*4.8.1* *Receptivitate 4*

*4.8.2* *Editare setări preferenţiale 4*

*4.8.3* *Editare aspect şi accesibilitate 4*

*4.8.4* *Selectare limbă 4*

*4.8.5* *Vizualizare titlu pagină curentă 4*

*4.8.6* *Notificări popup 4*

*4.8.7* *Suport prin email 4*

*4.8.8* *Apelare rudă 4*

*4.8.9* *Tutorial introducere 4*

**5** **Concluzii 8**

**6** **Bibliografie 9**

**7** **Referințe web 10**

**A.** **Codul sursă 11**

**B.** **Site-ul web al proiectului 12**

**C.** **CD / DVD 13**

**Index 14**

**LISTA FIGURILOR**

Figura 1. Selectarea prin click dreapta a opțiunii „Update field” 5

Figura 2. Actualizarea întregului tabel 5

**LISTA TABELELOR**

Tabelul 1. Nume de utilizatori și valorile rezumat ale parolelor acestora 5

* **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.

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, care 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, și opțional, după caz, data divorțului.

Informaţiile pe care fiecare persoană dintr-un arbore genealogic includ: nume, prenume, gen, ziua de naştere, ziua în care a murit (dacă este cazul), locul în care s-a născut, locul în care trăieşte, religia, naţionalitatea şi o poză. Dintre acestea, 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 despre educaţie, locul de muncă, număr de telefon, e-mail şi o biografie care . Niciuna din informații nu este obligatorie, fiecare utilizator putând decide ce dorește să adauge. Ocupațiile pe care le adaugă utilizatorul conțin informații despre postul ocupat, numele locului de muncă, și datele în care a început, respectiv încheiat să lucreze la locul de muncă respectiv. 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 informaţii despre aceștia, dar nu şi despre persoanele din arborii 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 asupra persoanei care reprezintă utilizatorul recunoscut ca rudă.

Fiecare utilizator poate alege să primească notificări despre ziua de naștere sau de nuntă a persoanelor din arborele său genealogic, notificări despre modificări în arborii rudelor.

* **Cazuri de utilizare**
* **Structura bazei de date**
* **Arhitectura soluţiilor**
* **Arhitectura aplicaţiei backend**
* **Arhitectura aplicaţiei frontend**
* **Convenții de redactare**

În această secțiune sunt detaliate convențiile de urmat în timpul editării lucrării.

* **Cerințe generale**

Cerințele generale sunt preluate din [Olt07] (lucrare pe care o recomandăm călduros spre citire candidaților noștri, înainte de demararea redactării tezei): „responsabilitatea tezei este în întregime a candidatului, atât în ceea ce priveşte conţinutul, cât şi forma. Lucrarea trebuie să aibă o organizare clară şi riguroasă, care să dovedească gândirea inginerească a candidatului. Ideile exprimate în lucrare trebuie să se înlănţuie conform unei logici clare. În acest sens, elementele de coerenţă şi de coeziune a textului trebuie folosite în mod corect. Ideile se organizează în paragrafe, redactate cu indentaţie şi fără spaţiu între ele. Nu fraza creează paragraful, ci ideea!

Stilul este extrem de important („Stilul este veşmântul gândului,” spune Samuel Johnson). Lucrarea trebuie redactată într-un limbaj ştiinţific adecvat domeniului de cercetare abordat. Se vor evita particularităţile limbajului colocvial. Nu sunt admise greşeli gramaticale de redactare (acord, punctuaţie, lexic etc.). Lucrarea normativă ce va fi avută în vedere în această privinţă este Dicţionarul ortografic, ortoepic şi morfologic al limbii române [DOOM05].

Candidatul are obligaţia de a verifica dacă datele, termenii folosiţi, numele proprii, citatele, titlurile (în limba română şi în alte limbi) sunt corecte.

Candidatul trebuie să fie consecvent în exprimarea ideilor, în folosirea termenilor, a numelor proprii, a datelor, precum şi a punctuaţiei şi a elementelor de structură a lucrării. Consecvenţa este necesară şi în privinţa tipurilor de evidenţieri grafice folosite (litere cursive, litere îngroșate sau sublinieri).

Termenii tehnici de origine străină neadaptaţi, consacraţi de lucrările de specialitate, nu se traduc, dar, dacă folosiţi o sursă bibliografică străină, puteţi încerca traducerea unor termeni noi, cu condiţia ca cei din limba de origine să fie prezenţi alături. În ambele cazuri, se recomandă scrierea acestor termeni cu litere speciale (de regulă italice).

La cele menţionate mai sus, adăugăm următoarea observaţie, care nu este deloc lipsită de importanţă: notarea semnelor diacritice româneşti este obligatorie. Un text românesc în care ă se confundă cu a nu face o impresie bună, dă o notă de neglijenţă.”

* **Structura documentului**

Lucrarea trebuie să conțină următoarele capitole:

* Coperta tezei;
* Pagina de titlu;
* Declarația de originalitate (completată și semnată de către autor);
* Formularul de înregistrare a enunțului temei lucrării (completat și semnat în solidar de către autor și coordonatorul științific);
* Referatul coordonatorului științific (completat și semnat de coordonator);
* Declarația de mulțumire a autorului (opțională);
* Cuprinsul lucrării;
* Lista figurilor;
* Lista tabelelor;
* Introducere;
* Conținutul propriu-zis al lucrării (capitolele constituente);
* Concluzii;
* Bibliografia;
* Referințele web;
* Anexele (index, codul sursă, site-ul web al aplicației, etc.).
* **Dimensiunile lucrării**

Nu pot fi acceptate lucrări cu un număr de pagini mai mic decât 50. Prin urmare se dau, cu titlu de recomandare, următoarele dimensiuni:

* 50 – 80 de pagini pentru o lucrare de licență;
* 60 – 100 de pagini pentru o lucrare de disertație.

În altă ordine de idei, recomandăm următoarea distribuție a paginilor pe diversele secțiuni/capitole [Olt07]:

* Introducerea reprezintă cca. 10 – 15% din lucrare;
* Conținutul reprezintă cca. 75 – 80% din lucrare;
* Concluziile reprezintă cca. 10%.
* **Elemente de tehnoredactare**

Pe scurt:

* dimensiunea paginilor va fi A4, 21 x 29,7 cm;
* marginile recomandate sunt: sus/jos/stânga/dreapta – 2,54 cm;
* fontul recomandat este Times New Roman;
* corpul literelor va avea dimensiunea de 11 puncte;
* spațiul dintre rânduri va avea dimensiunea de 1 rând și jumătate (1,5);
* indentarea unui paragraf se va face cu 1,27 cm;
* textul paragrafelor trebuie să fie aliniat în mod echilibrat stânga-dreapta (în eng. - *justify*);
* paginile trebuie să fie numerotate conform acestui șablon.
* **Formulele matematice**

Pentru redactarea formulelor matematice recomandăm utilizarea instrumentului Microsoft Equation Editor și importul lor (o data terminate) în Microsoft Word.

* **Ilustrațiile**
* **Figurile**

Partea grafică a lucrării are o pondere semnificativă în nota finală acordată lucrării candidatului. Se recomandă prin urmare acordarea unei atenții sporite tehnoredactării figurilor.

Actualizarea Listei Figurilor este obligatorie (procedura este similară cu cea exemplificată în secțiunea).

* **Tabelele**

Exemplificăm aici utilizarea tabelor. Pentru fiecare tabelă adăugată lucrării, autorul trebuie să prevadă și adăugarea unei legende (în eng., *Caption*). La final, este recomandată actualizarea listei figurilor. Pașii necesari pentru actualizarea listei tabelelor sunt:

* Click dreapta pe Lista Tabelelor. Va apărea un meniu similar cu cel din Figura 1.



**Figura 1. Selectarea prin click dreapta a opțiunii „Update field”**

* Selectarea opțiunii „Update entire table” conform cu Figura 2.



**Figura 2. Actualizarea întregului tabel**

* Verificarea fontului folosit pentru conținutul propriu-zis al Listei Tabelelor și alegerea fontului Times New Roman în caz de incongruență.

|  |  |  |
| --- | --- | --- |
| **Index** | **Nume utilizator** | **Valoarea rezumat a parolei (folosind SHA1)** |
| 1 | dpopescu | 8fb9e6763269ae7cba85f02668c3c32041bf00ed |
| 2 | eganea | 3b42ba8a586dd1589e949c28c9cf2810f7d65bb4 |
| 3 | mmarian | bfc01d16d1944f3b6caba515556713f4aeeb2d0b |

**Tabelul 1. Nume de utilizatori și valorile rezumat ale parolelor acestora**

În interiorul lucrării, tabelul poate fi citat folosind eticheta și numărul de ordine ale sale precum în exemplul acesta (vezi Tabelul 1).

* **Legenda (unei figuri/tabele)**

În cele doua secțiuni de mai sus s-au demonstrat două modele de legende atașate fiecărei tabele sau figuri. Microsoft Word permite modificarea, respectiv adăugarea de etichete noi corespunzătoare unui anumit tip de legendă.

* **Termeni de utilizare**
* **Autorii**

Acest șablon de document a fost creat de Marius Marian pentru colectivul Facultății de Automatică, Calculatoare și Electronică a Universității din Craiova și pentru uniformizarea structurii proiectelor de diplomă ale studenților săi.

* **Licența de utilizare**

Nu există restricții de utilizare. Documentul nu este constrâns de nicio licență. Sugestiile de îmbunătățire pot fi adresate autorului pe adresa de e-mail: marius.marian@cs.ucv.ro.

* **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 HYPERLINK "https://docs.microsoft.com/en-us/aspnet/core/tutorials/getting-started-with-swashbuckle?view=aspnetcore-6.0&tabs=visual-studio"& HYPERLINK "https://docs.microsoft.com/en-us/aspnet/core/tutorials/getting-started-with-swashbuckle?view=aspnetcore-6.0&tabs=visual-studio"tabs=visual-studio](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…

* **Site-ul web al proiectului**

Autorul prezintă în această anexă (opțională) site-ul web asociat proiectului său.

* **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