

**PROIECT DE DIPLOMĂ**

**Conducător științific:**

**Conf. Dr. Ing. Popescu Vlad**

**Absolvent:**

**Cursaru David**

**BRAȘOV, 2024**

**Departamentul: Electronică și calculatoare**

**Programul de studii: Calculatoare**

***Cursaru David***

**Proiectarea și Implementarea unei Aplicații de Monitorizare a Sănătății fizice și a Nutriției - Health Hub**

**Conducător științific:**

Conf. Dr. Ing. Popescu Vlad

Brașov, 2024

|  |  |
| --- | --- |
| **FIȘA PROIECTULUI DE DIPLOMĂ** | |
| Universitatea Transilvania din Braşov | Proiect de diplomă nr. .......... |
| Facultatea de Inginerie Electrică şi Ştiința Calculatoarelor |
| Departamentul de ............................................................................... | Viza facultății |
| Programul de studii:  **..................................................................** | Anul universitar:  2023 – 2024 |
| Candidat:  **Numele si Prenumele** | Promoția:  2024 |
| Conducător ştiințific:  **Titulatura. Numele și prenumele** |  |
| **PROIECT DE DIPLOMĂ** | |
| Titlul lucrării*: .................................................* | |
| Problemele principale tratate: | |
| Locul şi durata practicii: | |
| Bibliografie:   1. .......................... 2. .......................... 3. ........................... | |
| Aspecte particulare:  - ............................................................;  - ...........................................................  (desene, aplicații practice, metode specifice etc.) | |
| Primit tema la data de: .............................................. | |
| Data predării lucrării: ................................................. | |
| Director departament, Conducător științific,  *Titulatura. Numele și prenumele Titulatura. Numele și prenumele* | |
| Candidat,  *Numele și prenumele* | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PROIECT DE DIPLOMĂ – VIZE** | | | | | |
| Data vizei | Capitole/ problemele analizate | | | | Semnătura conducătorului științific |
|  | ..... | | | |  |
|  | ..... | | | |  |
|  | ..... | | | |  |
|  | ...... | | | |  |
|  | **Verificare procentaj similaritate Turnitin (total ≤25%și o singura sursa ≤10%)** | | | |  |
| **APRECIEREA ŞI AVIZUL CONDUCĂTORULUI ȘTIINȚIFIC** | | | | | |
| (*aprecierea lucrării/ proiectului se face prin raportare la cerințele privind elaborarea şi redactarea stabilite pe PS/ facultate; pot fi utilizate instrumente de evaluare, grile cu criterii unitare de apreciere adoptate de facultate/ departament*) | | | | | |
| Data:  ................................ | | ADMIS pentru susținere/ RESPINS | | Conducător științific  *Titulatura. Numele și prenumele*  *...............................(semnatura)* | |
| **AVIZUL DIRECTORULUI DE DEPARTAMENT** | | | | | |
| Data:  ................................ | | ADMIS pentru susținere/ RESPINS | | Director departament,  *Titulatura. Numele și prenumele*  *..............................(semnatura)* | |
| **SUSȚINEREA PROIECTULUI DE DIPLOMĂ** | | | | | |
| Sesiunea: ................................. | | | | | |
| Rezultatul  susținerii | | | PROMOVAT cu media: | | |
| RESPINS **cu** refacerea lucrării | | |
| RESPINS **fără** refacerea lucrării | | |
| Președinte de comisie,  *Titulatura. Numele și prenumele*  *.................................(semnatura)* | | | | | |

F04-PS7.6-01/ed.2,rev.2

**Cuprins**

Cuprins

[Lista de acronime 5](#_Toc156948401)

[1 Introducere 6](#_Toc156948402)

[1.1 Tema proiectului 6](#_Toc156948403)

[1.2 Scopul aplicației 6](#_Toc156948404)

[1.3 Obiective principale 6](#_Toc156948405)

[1.3.1 Obiective software 6](#_Toc156948406)

[1.3.2 Obiective de documentare 6](#_Toc156948407)

[1.4 Structura lucrării 6](#_Toc156948408)

[2 Tehnologii și metodologie 6](#_Toc156948409)

[2.1 Cadre de lucru 6](#_Toc156948410)

[2.1.1 Angular 6](#_Toc156948411)

[2.1.2 ASP .Net core web API 10](#_Toc156948412)

[2.1.3 Microsoft SQL Server 12](#_Toc156948413)

[2.2 Limbaje de programare 14](#_Toc156948414)

[2.2.1 TypeScript 14](#_Toc156948415)

[2.2.2 C# 15](#_Toc156948416)

[2.3 Medii de programare 17](#_Toc156948417)

[2.3.1 Visual Studio Code 17](#_Toc156948418)

[2.3.2 Visual studio 2022 18](#_Toc156948419)

[2.4 Tactici și tehnici de dezvoltare 18](#_Toc156948420)

[2.4.1 Single page application 18](#_Toc156948421)

[2.4.2 Comunicarea HTTP 19](#_Toc156948422)

[2.4.3 JSON Web Token 21](#_Toc156948423)

[2.4.4 Metoda Code First 22](#_Toc156948424)

[2.4.5 Repository Pattern 24](#_Toc156948425)

[3 Structura și dezvoltarea aplicației 24](#_Toc156948426)

[4 Bibliografie 25](#_Toc156948427)

[Rezumat 26](#_Toc156948428)

[Abstract 27](#_Toc156948429)

[Anexa 1 28](#_Toc156948430)

[Anexa 2 29](#_Toc156948431)

[Anexa 3 31](#_Toc156948432)

[DECLARAȚIE PRIVIND ORIGINALITATEA 33](#_Toc156948433)

# Lista de acronime

# Introducere

## Tema proiectului

## Scopul aplicației

## Obiective principale

### Obiective software

### Obiective de documentare

## Structura lucrării

# Tehnologii și metodologie

## Cadre de lucru

### Angular

Angular este un cadru de dezvoltare cu sursă deschisă, creat de echipa Angular de la Google și lansat pe 14 septembrie 2016. A reprezentat o evoluție semnificativă față de predecesorul său, AngularJS, fiind o rescriere completă și introducând îmbunătățiri esențiale, cum ar fi limbajul TypeScript, sintaxa și directivele, precum și concepte avansate precum încărcarea amânată (lazy loading).

Angular se utilizează în principal pentru dezvoltarea aplicațiilor web single-page, furnizând o interacțiune cu utilizatorul fluidă, fără a necesita reîncărcarea întregii pagini. Este apreciat la nivel global pentru versatilitatea sa și este utilizat într-o gamă variată de proiecte, de la cele academice și educaționale până la aplicații complexe cu impact global.

Pentru a profita de capacitățile Angular, o înțelegere de bază a programării web în JavaScript, CSS și HTML este necesară. Fără aceste abilități fundamentale, navigarea în complexitățile dezvoltării aplicațiilor web cu Angular poate prezenta provocări semnificative. În secțiunile următoare, voi prezenta pașii necesari pentru a configura Angular pe computerul personal.

Înainte de a începe instalarea Angular, este important să avem urmatoarele instrumente instalate:

* **Node.js**: Un mediu de execuție introdus în 2009 de Ryan Dahl, folosind JavaScript. Este adoptat pe scară largă de o comunitate vastă de programatori pentru dezvoltarea serverelor de aplicații web.
* **npm (Node Package Manager):** Un instrument folosit de programatori pentru a descoperi, descărca, instala și utiliza module în mod efficient și rapid.

Următorul pas implică deschiderea unei ferestre de terminal și executarea comenzii pentru a instala Angular CLI:



Figura 1.0: Comandă instalare Angular CLI

Odată ce aceste componente fundamentale sunt în loc, inițierea unui nou proiect se realizează foarte ușor cu următoarea comandă:



Figura 1.1: Comandă creare proiect Angular

Un aspect crucial al Angular este accentul pus pe componente, servind drept blocuri fundamentale pentru orice aplicație Angular și fiind instrumentale în crearea interfeței utilizatorului. Componentele gestionează și prezintă datele în mod eficient, fiind responsabile de interacțiunea cu utilizatorului. O componentă Angular constă în trei fișiere esențiale: un fișier TypeScript care reprezintă o clasă ce incapsulează logica și proprietățile specifice acelei clase, un fișier de șablon HTML unde putem folosi aceste proprietăți pentru a le afișa în interfața cu utilizatorul, și un fișier CSS pentru stilizare.

Două avantaje semnificative evidențiază utilizarea răspândită și necesitatea componentelor: modularitatea și reutilizarea. Componentele permit programatorilor să le utilizeze împreună cu alte componente, facilitând crearea unei interfețe mai intricate și estetic plăcută. Componentele au capacitatea de a reacționa la evenimente, cum ar fi apăsarea unei taste pe tastatură sau un clic de mouse, declanșând acțiuni conexe.

Angular simplifică generarea de componente fără conținut cu o singură comandă, traducându-se în economii semnificative de timp. Comanda pentru generarea unei componente în Angular este formulată astfel:

Un aspect crucial al Angular este accentul pus pe componente, servind drept blocuri fundamentale pentru orice aplicație Angular și fiind instrumentale în crearea interfeței utilizatorului. Componentele gestionează eficient și prezintă datele, asumându-și responsabilitatea pentru interacțiunea utilizatorului. O componentă Angular constă în trei fișiere esențiale: un fișier TypeScript care găzduiește logica, un fișier de șablon HTML și un fișier CSS pentru stilizare.

Două avantaje semnificative justifică utilizarea răspândită și necesitatea componentelor: modularitatea și reutilizarea. Componentele permit programatorilor să le utilizeze împreună cu alte componente, facilitând crearea unei interfețe mai complexe. Componentele au capacitatea de a reacționa la evenimente, cum ar fi apăsarea unei taste pe tastatură sau un click de mouse, declanșând acțiuni conexe.

Angular simplifică generarea de componente fără conținut cu o singură comandă, arătând încă odată avantajele oferite de un cadru de lucru în termeni de eficaciatate și fluiditate a procesului de dezvoltare. Comanda pentru generarea unei componente în Angular este formulată astfel:



Figura 1.2: Comandă generare componentă Angular

Mai departe putem vedea mai jos imagini cu conținutul unei componente în Angular:

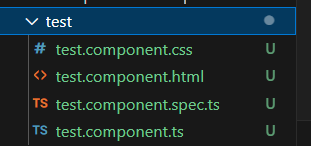


Figura 1.3: Structura unei componente Angular



Figura 1.4: Fișier HTML dintr-o componentă Angular



Figura 1.5: Fișier TypeScript dintr-o componentă Angular

Până în acest moment, am explorat posibilitățile oferite de Angular și beneficiile asociate utilizării acestei tehnologii în dezvoltarea unei aplicații. Cu toate acestea, detaliile anterioare referitoare la componente și comenzi reprezintă doar punctul de plecare. Deși acestea constituie un fundament esențial pentru crearea unei aplicații web, Angular oferă mai multe resurse pentru a eficientiza și accelera procesul de dezvoltare, iar printre acestea se numără Angular Material. Angular Material este o bibliotecă în care se află componente și stiluri predefinite, create special pentru a se integra ușor și rapid în aplicații Angular. Inițial dezvoltată de Google în 2014, această bibliotecă beneficiază constant de îmbunătățiri, actualizări și noi funcționalități din partea echipei Angular. Similar cu Angular, Angular Material este supus unor actualizări regulate, la intervale de câteva luni, impunând necesitatea de a fi la curent cu versiunile disponibile. Aplicațiile construite cu Angular Material se pot adapta cu ușurință pe diverse platforme, inclusiv Android, iOS, Windows și MacOS, datorită versatilității tehnologiei. Pentru a beneficia de Angular Material în cadrul proiectului meu, a fost suficient să execut următoarea comandă în terminalul corespunzător proiectului și să selectez opțiunile oferite:



Figura 1.6: Comandă adaugare Angular Material în proiect

### ASP .NET Core Web API

Plaftorma Web .NET Core face parte din cadrul de lucru ASP.NET Core, și este folosit pentru a dezvolta servicii web API.

Înainte de a merge mai departe și să intru în detalii legate de acest cadru de lucru ar fi mai bine să explic ce este o aplicație de tip Web API mai întâi. API este prescurtarea de la Application Programming Interface, care s-ar traduce Interfață de Programare a Aplicațiilor. O aplicație Web API este o aplicație care oferă servicii și funcționalități accesibile prin intermediul protocoalelor și standardelor web. Într-o aplicație de acest tip, funcționalitățile sunt expuse prin intermediul unor puncte finale de acces numite endpoint-uri care pot fi apelate de alte aplicații sau servicii. Aplicația primeste cereri HTTP, cum ar fi GET, POST, PUT, DELETE, de la clienti și returnează răspunsuri HTTP care conțin datele solicitate.

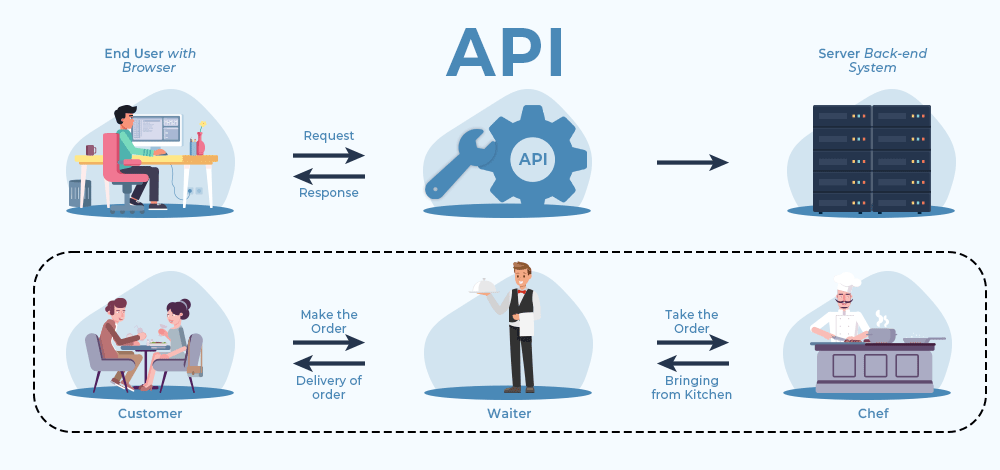


Figura 1.7: Intrefața de programare a aplicațiilor

ASP.NET Core Web API este un cadru de lucru puternic și flexibil dezvoltat de Microsoft, care permite crearea de servicii web RESTful scalabile și interoperabile. Această tehnologie face parte din ecosistemul ASP.NET și oferă dezvoltatorilor un set robust de instrumente și biblioteci pentru construirea și implementarea aplicațiilor web bazate pe arhitectura API.

Acest cadru de lucru se bazează pe fundamentul ASP.NET și permite dezvoltatorilor să creeze servicii web care pot fi consumate de diverse clienți, inclusiv:

* aplicații web,
* aplicații mobile,
* aplicații de tip SPA (Single Page Application),
* dispozitive IoT (Internet of Things) și multe altele.

Unul dintre punctele forte ale acestui cadru de lucru este abilitatea sa de a returna date într-un format standardizat, cum ar fi JSON sau XML , oferind astfel o interacțiune eficientă între server și client. Caracteristica centrală a framework-ul de dezvoltare ASP.NET este reprezentată de suportul său pentru arhitectura REST . Aceasta implică utilizarea standardului HTTP și a metodelor sale (GET, POST, PUT, DELETE) pentru a crea, citi, actualiza și șterge resursele expuse prin intermediul API-ului. Prin adoptarea unei abordări bazate pe resurse și folosind conceptele de identificator al resursei (URI ) și de reprezentare a resursei, pot proiecta și implementa servicii web scalabile și ușor de utilizat.

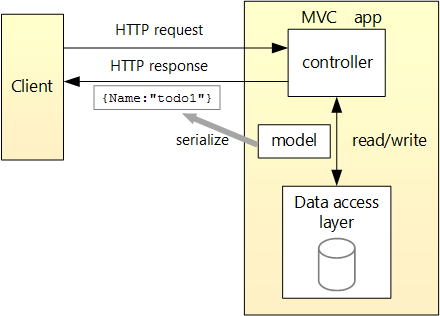


Figura 1.8: Modul de funcționare al ASP .Net core web API

Cadrul Web ASP.NET Core facilitează, de asemenea, crearea de servicii web securizate, prin integrarea cu mecanisme precum autentificarea și autorizarea. Prin utilizarea sistemului de autentificare și autorizare bazat pe token-uri, pot proteja resursele expuse prin API și pot controla accesul utilizatorilor sau aplicațiilor la acestea.

Un alt aspect important al acestui cadru de lucru este flexibilitatea sa. Acesta poate fi utilizat împreună cu diverse tehnologii și biblioteci, cum ar fi Entity Framework pentru accesul la baze de date, biblioteca Newtonsoft.Json pentru serializarea și deserializarea datelor în format JSON, și multe altele. De asemenea, este compatibil cu diferite formate de date, cum ar fi XML și JSON, permițând astfel integrarea cu o varietate de clienți și servicii.

Platforma Web API .NET Core pune la dispoziție, de asemenea, un set bogat de instrumente pentru testare și depanare. Prin intermediul unor instrumente precum Postman sau Swagger, pot testa și interoga serviciile API, pot crea documentație automată și pot monitoriza traficul și performanța acestora.

### Microsoft SQL Server

Microsoft SQL Server reprezintă un sistem de administrare a bazelor de date relaționale (RDBMS) cu o utilizare extensivă în industrie, datorită versatilității și documentației sale detaliate și cuprinzătoare. Este recunoscut ca unul dintre cele mai valoroase sisteme de gestionare a bazelor de date existente.

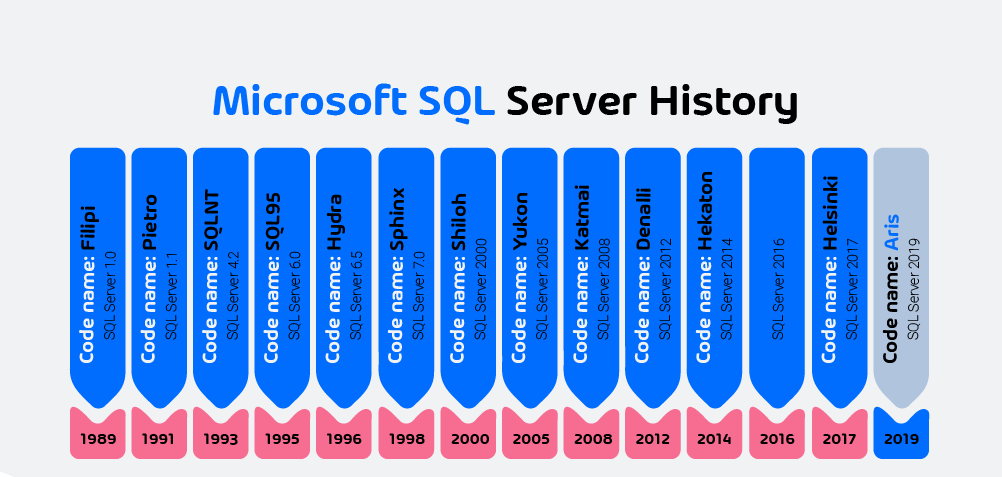
 [1] [1]

Figura 1.9: Istoria versiunilor Microsoft SQL Server

În etapa inițială, SQL Server a fost conceput în decursul anilor '80 de către Sybase Inc., iar în prezent, este sub administrarea și dezvoltarea continuă a Microsoft. De-a lungul decadelor, SQL Server a suferit un proces de evoluție și îmbunătățire constantă, transformându-se într-un produs solid și de încredere pentru gestionarea datelor în diferite contexte de afaceri. „Securitatea este esențială în orice sistem de stocare și procesare a datelor, iar SQL Server s-a mândrit că este cea mai securizată bază de date în ultimii opt ani, conform bazei de date a Institutului Național de Standarde și Tehnologie(NIST). SQL Server susține securitatea și conformitatea în mediul enterprise prin funcționalități de securitate precum Transparent Data Encryption, Auditing, Row-Level Security, Dynamic Data Masking și Always Encrypted.” [1]

Transparent Data Encryption este o caracteristică puternică a SQL Server care criptează automat datele în timpul stocării pe disc, asigurând protecția acestora chiar și în situația în care dispozitivul de stocare este compromis. Aceasta asigură confidențialitatea datelor și respectarea cerințelor de conformitate cu privire la securitatea informațiilor.

Auditing este o funcționalitate esențială pentru monitorizarea și urmărirea activităților desfășurate în baza de date. SQL Server permite configurarea auditării la nivel de bază de date și de tabele, oferind informații detaliate despre acțiunile efectuate asupra datelor.

Row-Level Security permite definirea politicilor de securitate la nivel de rând, permițând accesul și vizualizarea datelor doar pentru utilizatorii autorizați. Aceasta oferă un control fin asupra accesului la informații și protejează datele sensibile împotriva accesului neautorizat.

Dynamic Data Masking permite ascunderea sau mascarea datelor sensibile în timp real, în funcție de drepturile de acces ale utilizatorului. Aceasta previne expunerea accidentală a informațiilor confidențiale în aplicații sau rapoarte și asigură protecția datelor sensibile.

Always Encrypted este o funcționalitate avansată care permite criptarea datelor sensibile, inclusiv cheile de criptare, astfel încât acestea să fie păstrate în mod sigur chiar și atunci când sunt în tranzit sau stocate în medii nesigure. Aceasta oferă un nivel înalt de confidențialitate și protecție a datelor.

Prin intermediul acestor funcționalități de securitate, SQL Server asigură protecția datelor și respectarea standardelor și cerințelor de securitate în medii enterprise. Aceasta face din SQL Server o alegere potrivită pentru aplicații și sisteme care necesită un nivel înalt de securitate și conformitate.

În lumea actuală, gestionarea eficientă a datelor este esențială pentru succesul organizațiilor de orice dimensiune. Un sistem de gestiune a bazelor de date puternic și bogat în funcționalități este cheia pentru a asigura stocarea, manipularea și administrarea corespunzătoare a acestor date. Unul dintre aceste sisteme este Microsoft SQL Server.

SQL Server, dezvoltat de Microsoft, oferă o platformă robustă pentru gestionarea datelor într-un mod eficient și sigur. Este recunoscut pentru caracteristicile sale avansate de securitate, opțiunile de scalabilitate și suportul pentru disponibilitatea înaltă și recuperarea în caz de dezastru. Aceste calități fac din SQL Server o alegere populară pentru organizații de diferite dimensiuni, de la întreprinderi mici până la corporații mari.

Cartea „Microsoft SQL Server 2019: A Beginner's Guide” de Dusan Petkovic, publicată în 2020, servește drept ghid pentru începători în lumea SQL Server. Această lucrare cuprinde conceptele fundamentale ale SQL Server, interogările SQL, proiectarea bazelor de date și sarcinile de administrare. Prin intermediul acestui ghid, pot învăța să utilizez SQL Server în mod eficient și să mă familiarizez cu principiile și practicile esențiale ale gestionării bazelor de date.

„SQL Server este un sistem puternic și bogat în funcționalități pentru gestionarea bazelor de date relaționale, dezvoltat de Microsoft. Oferă o platformă robustă pentru stocarea, gestionarea și manipularea datelor, fiind o opțiune populară pentru organizații de toate dimensiunile. Cu funcționalitățile sale avansate de securitate, opțiunile de scalabilitate și suportul pentru disponibilitate înaltă și recuperare în caz de dezastre, SQL Server oferă o soluție cuprinzătoare pentru gestionarea datelor. Această carte servește ca ghid pentru începători în utilizarea SQL Server, acoperind conceptele fundamentale, interogările SQL, proiectarea bazelor de date și sarcinile de administrare.”[2]

## Limbaje de programare

### TypeScript

TypeScript este un limbaj de programare open-source dezvoltat de Microsoft, bazat pe JavaScript. A fost creat pentru a extinde JavaScript prin adăugarea de tipuri statice și funcționalități specifice dezvoltării de aplicații de mari dimensiuni și complexitate.

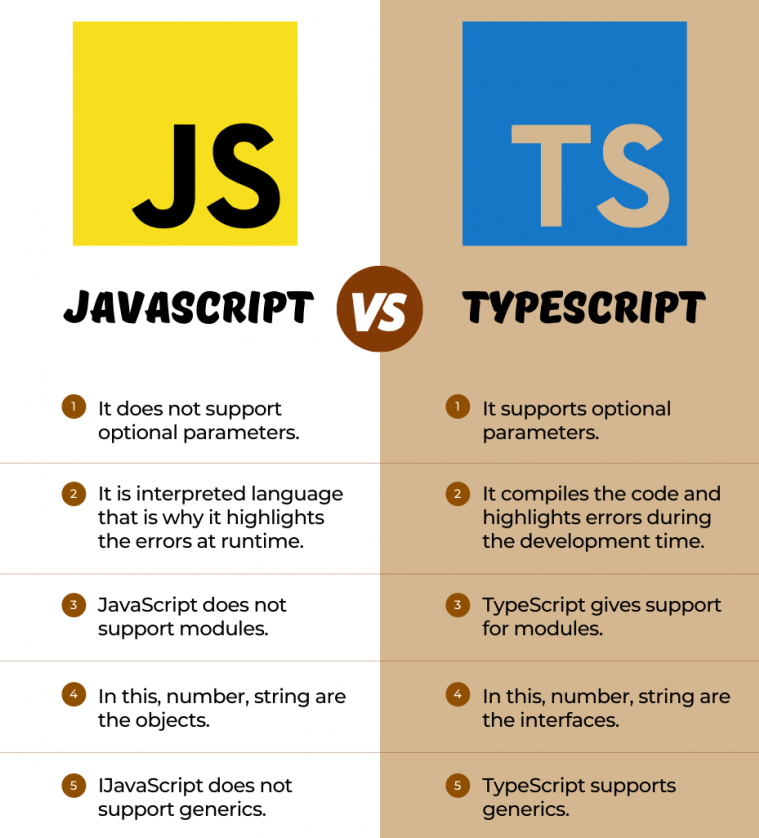


Figura 2.0: JavaScript vs TypeScript

Unul dintre principalele avantaje ale TypeScript este posibilitatea de a adăuga tipuri de date la variabile, funcții și obiecte. Aceasta aduce beneficii semnificative în dezvoltarea software, deoarece oferă un sistem puternic de verificare a tipurilor în timpul compilării și elimină o serie de erori comune de tipuri care pot apărea în JavaScript.

TypeScript aduce o serie de funcționalități suplimentare față de JavaScript, cum ar fi tipurile de date avansate(inclusiv tipuri personalizate), interfețe, clase, moștenire, module și multe altele. Aceste caracteristici îmbunătățesc productivitatea dezvoltatorului, oferind suport pentru dezvoltarea de aplicații scalabile și ușor de întreținut.

Un alt aspect important al TypeScript este că acesta este transpilat în JavaScript standard, ceea ce înseamnă că codul TypeScript poate fi rulat pe orice browser sau platformă care suportă JavaScript.

TypeScript a devenit tot mai popular în comunitatea dezvoltatorilor datorită beneficiilor pe care le aduce în dezvoltarea aplicațiilor web și a aplicațiilor pe partea de server. Este folosit în special în cadrul cadrelor de lucru și bibliotecilor populare, precum Angular, React și Vue.js.

Cu ajutorul TypeScript, pot scrie cod mai sigur, mai ușor de întreținut și mai scalabil, beneficiind totodată de avantajele ecosistemului JavaScript.

### C#

C# este un limbaj de programare modern, orientat pe obiecte, dezvoltat de Microsoft în anul 2000. A fost conceput pentru a fi un limbaj puternic, versatil și ușor de înțeles, adaptat dezvoltării aplicațiilor de tip enterprise, web și mobile.

C# face parte din familia de limbaje de programare C și este strâns legat de platforma .NET. Acesta beneficiază de un sistem de tipuri puternic, gestionarea automată a memoriei și o sintaxă clară și expresivă, ceea ce îl face alegerea preferată pentru dezvoltarea de aplicații robuste și scalabile.

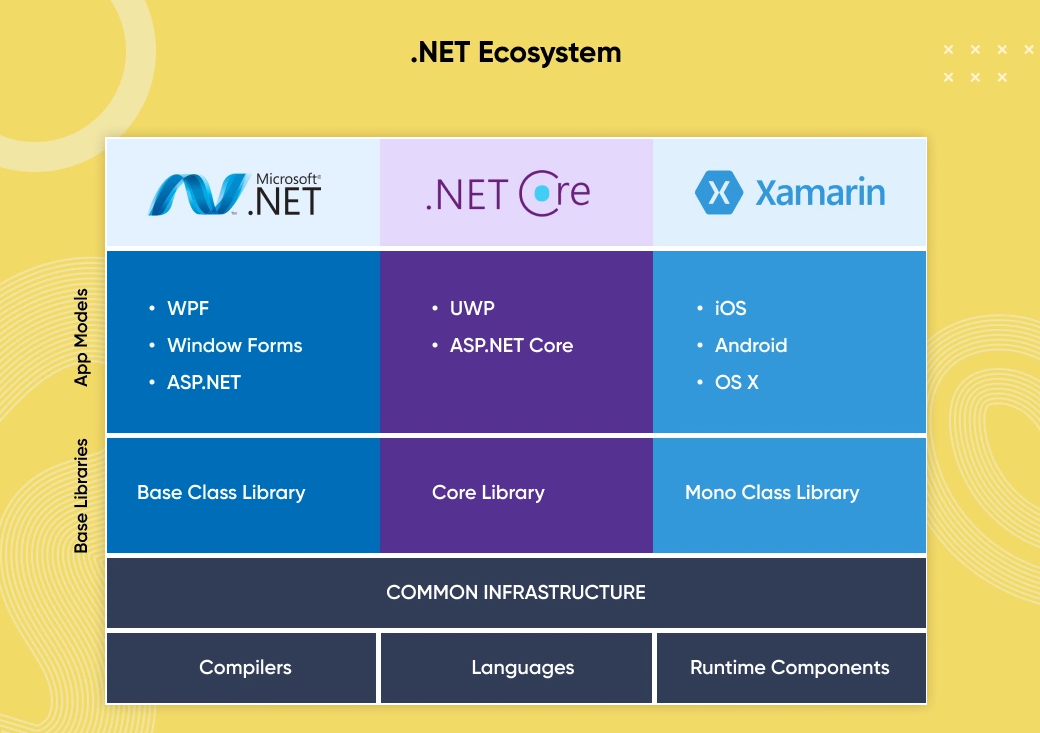


Figura 2.1: Ecosistemul .NET

Una dintre caracteristicile-cheie ale limbajului C# este orientarea sa pe obiecte. Acesta imi permite să modelez structuri de date și să creez obiecte care să interacționeze între ele prin intermediul conceptelor de clasă, moștenire, încapsulare și polimorfism. Aceasta facilitează dezvoltarea de cod modular și extensibil, care poate fi mai ușor de întreținut pe parcursul timpului.

Un alt aspect important al limbajului C# este tipizarea sa statică. Acest lucru înseamnă că toate variabilele și expresiile trebuie să fie declarate cu un tip de date specific înainte de a fi utilizate și că tipurile nu pot fi schimbate ulterior. Tipizarea statică aduce beneficii precum detectarea mai ușoară a erorilor de tip în timpul compilării și oferă o mai mare siguranță și claritate în cod.

C# oferă, de asemenea, suport puternic pentru programarea asincronă. Acest lucru este deosebit de util în aplicațiile care trebuie să răspundă la evenimente și să desfășoare operații de lungă durată fără a bloca firul principal de execuție. Prin utilizarea cuvântului cheie "async" și a bibliotecii de tipuri "Task", pot crea operații asincrone și pot beneficia de o performanță sporită în aplicațiile lor.

C# este, de asemenea, strâns integrat cu platforma .NET, ceea ce înseamnă că pot avea acces la o gamă largă de biblioteci și cadre de lucru pentru dezvoltarea aplicațiilor. Aceasta include .NET Framework, care este utilizat pentru dezvoltarea de aplicații Windows, precum și .NET Core, care este utilizat pentru dezvoltarea de aplicații cross-platform. Există și cadre de lucru specifice domeniului, cum ar fi ASP.NET pentru dezvoltarea aplicațiilor web și Xamarin pentru dezvoltarea aplicațiilor mobile.

C# este utilizat într-o gamă largă de domenii, inclusiv dezvoltarea de aplicații desktop, aplicații web, aplicații mobile, jocuri, sisteme înglobate și multe altele. Comunitatea dezvoltatorilor C# este activă și oferă suport și resurse pentru îmbunătățirea cunoștințelor și abilităților în acest limbaj.

Cu toate acestea, C# este un limbaj în continuă evoluție, iar Microsoft lansează în mod regulat noi versiuni și actualizări pentru a adăuga funcționalități noi și pentru a îmbunătăți performanța și productivitatea dezvoltatorilor. Prin urmare, rămâne un limbaj viu și actual în industria software, oferind posibilități nelimitate pentru crearea de aplicații moderne și inovatoare.

## Medii de programare

### Visual Studio Code

Visual Studio Code (VS Code) este un editor de cod sursă dezvoltat de Microsoft, care s-a bucurat de o popularitate rapidă în rândul comunității dezvoltatorilor. Am ales să utilizez Visual Studio Code pentru dezvoltarea aplicației noastre datorită numeroaselor beneficii pe care le oferă. Unul dintre avantajele majore ale utilizării Visual Studio Code este experiența de lucru eficientă și rapidă. Cu ajutorul completării automate a codului și a evidențierii sintaxei, am putut scrie codul mai repede și mai precis. Aceasta mi-a permis să creez aplicația într-un timp mai scurt, fără a face compromisuri în ceea ce privește calitatea.

Un alt instrument care mi-a fost de mare ajutor în dezvoltarea aplicației este GitHub Copilot. Acesta este un sistem de inteligență artificială dezvoltat de GitHub și OpenAI, care oferă sugestii și completări de cod bazate pe contextul și nevoile mele. Cu ajutorul GitHub Copilot, am putut scrie cod mai rapid și mai eficient, beneficiind de recomandări inteligente și adaptate specificului proiectului meu.

### Visual studio 2022

Visual Studio 2022 este cea mai nouă versiune a mediului de dezvoltare integrat (IDE) oferit de Microsoft, având un set bogat de funcționalități și îmbunătățiri care îi susțin utilizatorii în dezvoltarea software. Această versiune aduce o serie de avantaje și facilități pentru programatori, în special pentru proiectele bazate pe platforma ASP.NET Core Web API.

Unul dintre aspectele cheie ale Visual Studio 2022 este suportul avansat pentru cadrul de lucru Entity Framework (EF). EF este o tehnologie dezvoltată de Microsoft care facilitează lucrul cu bazele de date relaționale într-un mod simplu și eficient. Prin intermediul Visual Studio 2022, dezvoltatorii pot utiliza instrumentele EF pentru a defini modelele de date, a efectua operații de interogare și manipulare a datelor și a realiza migrări ale schemelor bazei de date. Aceasta aduce un nivel ridicat de productivitate și ușurință în dezvoltarea și gestionarea bazei de date.

## Tactici și tehnici de dezvoltare

### Single page application

Single Page Application (SPA) este o abordare modernă în dezvoltarea aplicațiilor web, care se diferențiază de modelul tradițional de navigare între pagini multiple. Într-o SPA, întreaga aplicație este încărcată o singură dată în browser, iar toate interacțiunile ulterioare cu serverul sunt gestionate prin intermediul comunicării asincrone cu servicii web.

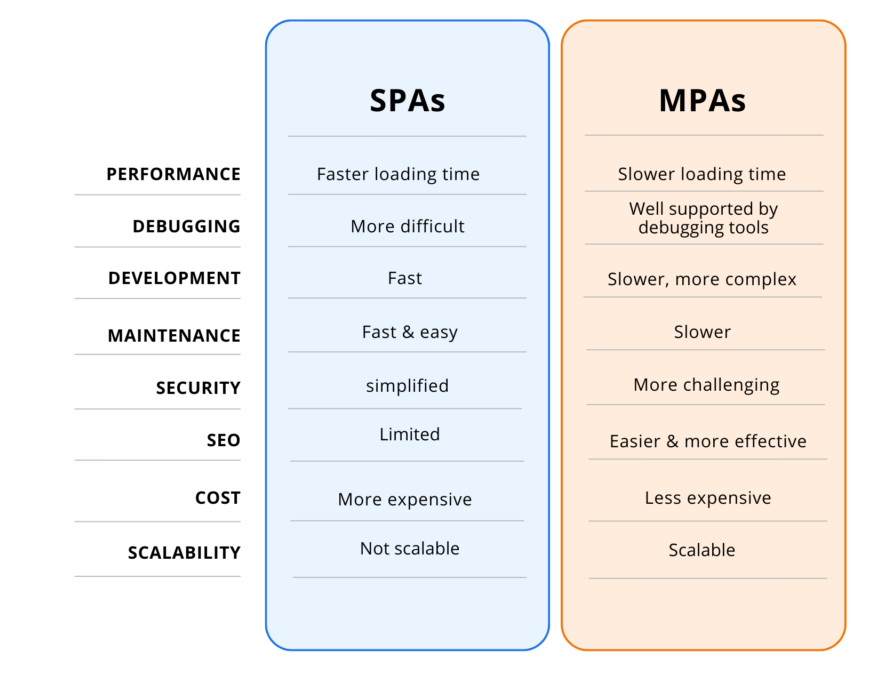


Figura 2.2: Single Page Application vs Multiple Page Application

Principala caracteristică a unei SPA este că toate schimbările de conținut și de stare sunt realizate dinamic, fără a se reîncărca întreaga pagină. În schimb, se utilizează tehnici de manipulare a DOM pentru a actualiza și reafirma secțiunile specifice ale aplicației, oferind astfel o experiență fluidă și interactivă utilizatorului.

Una dintre principalele avantaje ale unei SPA este viteza și reactivitatea, deoarece paginile nu trebuie reîncărcate în mod constant. Acest lucru conduce la o interacțiune mai fluidă și la o utilizare îmbunătățită. De asemenea, o SPA permite încărcarea diferită a resurselor, ceea ce duce la o utilizare mai eficientă a băndei de internet.

O altă caracteristică importantă a SPA este reutilizarea codului și separarea clară între frontend și backend. SPA-urile utilizează API-uri pentru a comunica cu serviciile web și pentru a obține sau a trimite date. Aceasta permite dezvoltatorilor să creeze și să mențină logică de afișare și de interacțiune în frontend, în timp ce serverul se ocupă de prelucrarea datelor și de logica de afaceri în backend.

Pentru a realiza o SPA, se utilizează de obicei un cadru de lucru de dezvoltare a frontend-ului, cum ar fi Angular, React sau Vue.js. Aceste cadre oferă un set de instrumente și biblioteci care facilitează gestionarea stării aplicației, rutarea, manipularea DOM-ului și comunicarea cu serviciile web.

Cu toate acestea, există și câteva aspecte de luat în considerare atunci când se dezvoltă o SPA. Deoarece întreaga aplicație este încărcată o singură dată, inițializarea inițială a aplicației poate necesita mai mult timp, iar gestionarea memoriei și a resurselor trebuie abordată cu atenție pentru a evita problemele de performanță.

### Comunicarea HTTP

Comunicarea HTTP (Hypertext Transfer Protocol) reprezintă un protocol standard utilizat pentru transmiterea datelor între client și server pe World Wide Web. În contextul arhitecturii REST (Representational State Transfer), comunicarea HTTP joacă un rol central în interacțiunea între client și serviciul web.

Principala caracteristică a comunicării HTTP în cadrul unei arhitecturi RESTful este reprezentată de utilizarea metodelor HTTP pentru a accesa și manipula resursele. Cele mai comune metode utilizate în acest context sunt:

* GET: Această metodă este utilizată pentru a obține o resursă specifică de la server. De exemplu, un client poate utiliza o cerere GET pentru a obține informații despre un produs specific de pe un site de comerț electronic.
* POST: Metoda POST este folosita pentru a trimite date către server pentru a crea o nouă resursă. De exemplu, atunci când un utilizator completează un formular de înregistrare pe un site web, datele vor fi trimise prin intermediul unei cereri POST pentru a crea un nou cont.
* PUT: Metoda PUT este utilizată pentru a actualiza o resursă existentă sau pentru a crea o resursă nouă într-un anumit loc. De exemplu, un client poate utiliza o cerere PUT pentru a actualiza detaliile unui produs dintr-un magazin online.
* DELETE: Metoda DELETE este utilizată pentru a șterge o resursă specifică de pe server. De exemplu, un client poate utiliza o cerere DELETE pentru a șterge un articol dintr-un blog.

Pe lângă metodele menționate mai sus, comunicarea HTTP în cadrul arhitecturii RESTful se bazează pe alte elemente importante:

* URI (Uniform Resource Identifier): Reprezintă un identificator unic care identifică resursa la care se face referire în cerere. De exemplu, într-o cerere GET, URI-ul poate indica adresa URL a unei pagini web sau a unui serviciu web.
* Header-uri HTTP: Acestea conțin informații suplimentare despre cerere sau răspuns, cum ar fi tipul de conținut acceptat sau trimis, informații de autentificare, cache-control și multe altele.
* Body: Acesta reprezintă conținutul datelor trimise sau primite în cadrul unei cereri sau a unui răspuns HTTP. De exemplu, într-o cerere POST, corpul poate conține datele formularului trimise de utilizator.

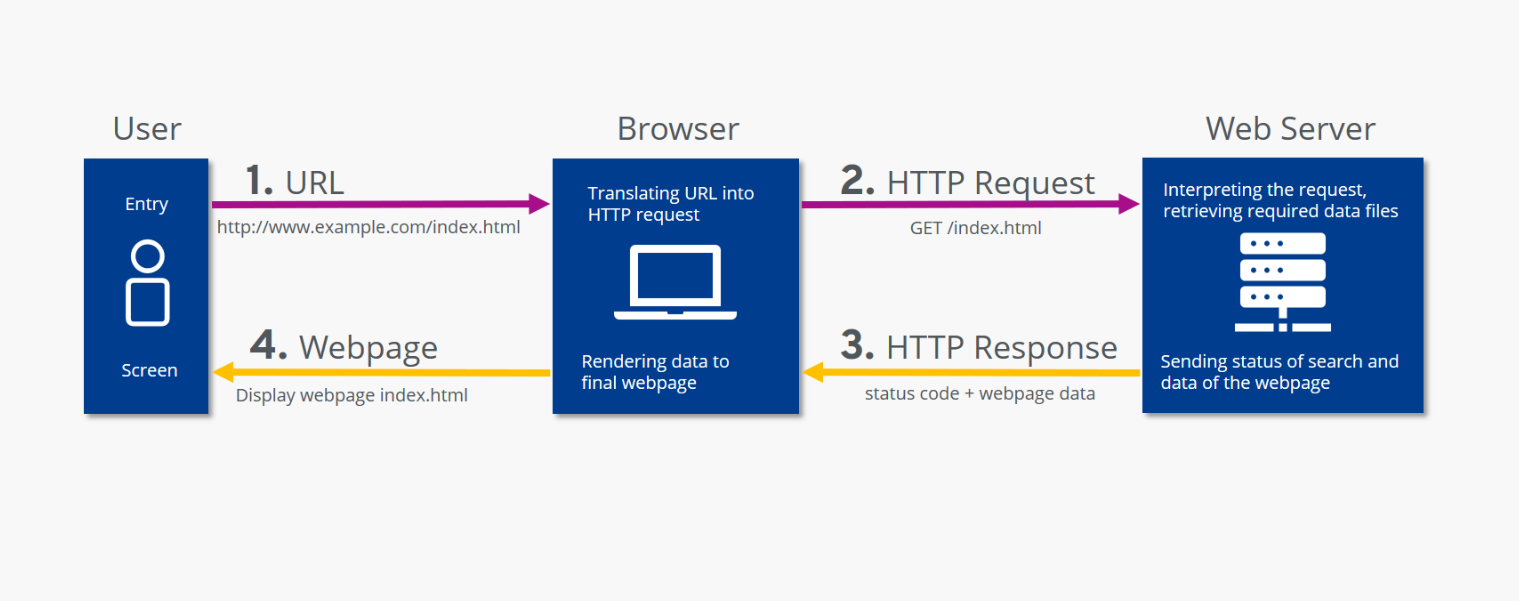


Figura 2.3: Comunicarea HTTP

Comunicarea HTTP în arhitectura RESTful facilitează interoperabilitatea și integrarea între sisteme diferite. Folosind metodele HTTP standard și respectând principiile REST, pot crea servicii web scalabile, ușor de utilizat și independente de platformă.

De asemenea, comunicarea HTTP oferă suport pentru diferite formate de date, cum ar fi JSON și XML, permițând astfel transmiterea și recepționarea datelor într-un mod flexibil și eficient.

### JSON Web Token

JSON Web Token (JWT) este o tehnologie de securitate care permite transmiterea sigură a datelor între diferite entități într-un format compact și autenticat. Este utilizat în mod obișnuit în aplicațiile web și mobile pentru autentificare și autorizare.

JWT este compus din trei părți principale: header, payload și semnătură. Header-ul conține informații despre tipul de token și algoritmul de criptare utilizat. Payload-ul conține datele pe care doresc să le transmit, cum ar fi informații despre utilizator sau detalii de autorizare. Semnătura este rezultatul aplicării unui algoritm de criptare asupra header-ului, payload-ului și unei chei secrete, asigurând astfel autenticitatea și integritatea tokenului

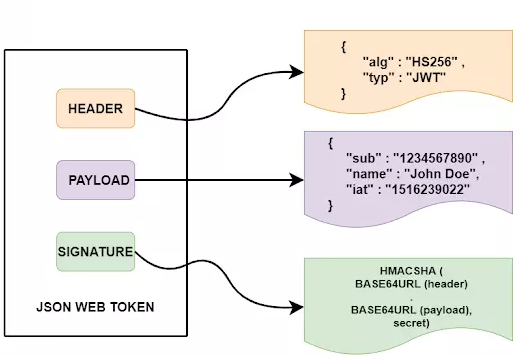


Figura 2.4: Structura JWT

Unul dintre principalele avantaje ale JWT este faptul că poate fi verificat rapid și eficient de către părțile implicate, fără a necesita interogarea unei baze de date sau un server de autentificare. Acest aspect face ca JWT să fie o soluție scalabilă și eficientă în aplicațiile distribuite.

JWT este folosit în principal pentru autentificare și autorizare. După ce un utilizator se autentifică cu succes, un token JWT este generat și trimis către client. Acesta poate fi inclus în cererile ulterioare către server pentru a confirma identitatea utilizatorului. Serverul poate verifica și valida token-ul primit pentru a autoriza accesul la resursele protejate.

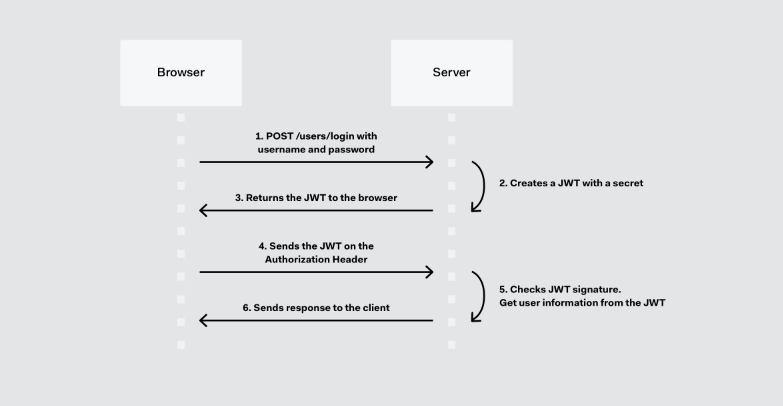


Figura 2.5: Modul de funcționare al JWT

Un alt aspect interesant al JWT este că poate transporta informații suplimentare despre utilizator, numite și claim-uri. Aceste claim-uri pot conține privilegii speciale, roluri, date de profil sau alte informații relevante pentru aplicație.

Este important de menționat că securitatea JWT depinde de păstrarea în siguranță a cheii secrete utilizate pentru semnătură. Aceasta trebuie protejată și accesibilă doar entităților autorizate. De asemenea, dimensiunea și complexitatea token-ului trebuie luate în considerare, deoarece acesta este trimis cu fiecare cerere, ceea ce poate afecta performanța aplicației.

### Metoda Code First

Metoda Code First este o abordare de dezvoltare utilizată în cadrul de lucru Entity Framework (EF) pentru gestionarea bazei de date în aplicații .NET. Ea pune accent pe definirea și configurarea modelelor de date în codul sursă al aplicației, urmând ca baza de date să fie generată automat pe baza acestor modele.

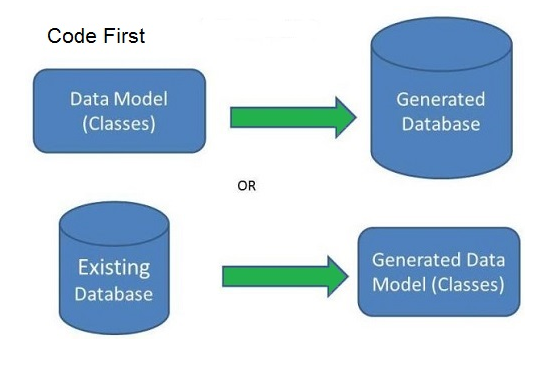


Figura 2.6: Code First vs Database First

Procesul de utilizare a metodei Code First implică următorii pași:

1. Definirea modelelor de date: definesc clase în codul sursă al aplicației pentru a reprezenta entitățile și relațiile din domeniul de lucru. Aceste clase pot fi decorate cu atribute specifice pentru a configura aspecte precum cheile primare, relațiile între entități și restricțiile.
2. Crearea contextului de bază de date: creez o clasă care extinde clasa DbContext din Entity Framework. Această clasă reprezintă contextul de bază de date și conține seturile de date (DbSet) pentru fiecare entitate. De asemenea, se poate configura conexiunea la baza de date și alte opțiuni specifice.
3. Generarea și aplicarea migrărilor: După ce modelele de date au fost definite, Entity Framework permite generarea automată a migrărilor. Acestea reprezintă schimbările necesare în schema bazei de date pentru a reflecta modelele de date. Migrările pot fi apoi aplicate pentru a crea sau actualiza baza de date.
4. Utilizarea bazei de date: După generarea bazei de date, pot utiliza contextul de bază de date pentru a interacționa cu datele. Acest lucru include operații precum adăugare, actualizare, ștergere și interogare a entităților.

Metoda "Code First" oferă un nivel ridicat de control și flexibilitate în gestionarea bazei de date în aplicații .NET. Dezvoltatorii pot defini modelele de date în mod explicit și personalizat, iar Entity Framework se ocupă de generarea schemelor și migrărilor necesare. Acest lucru facilitează dezvoltarea rapidă și iterativă, oferind totodată posibilitatea de a adăuga modificări și funcționalități suplimentare pe măsură ce aplicația evoluează.

### Repository Pattern

Repository Pattern este o strategie de proiectare utilizată în dezvoltarea software pentru a organiza și gestiona interacțiunea cu baza de date într-o aplicație. În loc să scriu cod specific pentru fiecare operațiune de acces la date, acest șablon imi permite să definesc o interfață comună pentru accesul la date și să implementez această interfață în clase separate numite repository-uri.

Interfața repository definește metodele generice pentru a crea, citi, actualiza și șterge datele din baza de date. Fiecare repository concret implementează aceste metode în funcție de logica specifică a entităților și a mecanismului de stocare. Astfel, repository pattern imi oferă un nivel de abstractizare care separă logica de afaceri de detalii specifice ale bazei de date.

Am ales să utilizez Repository Pattern în cadrul aplicației mele pentru a obține o structură eficientă și modulară a accesului la date.

Unul dintre principalele motive pentru care am optat pentru Repository Pattern este reutilizabilitatea și flexibilitatea pe care o oferă. Prin intermediul acestui pattern, pot defini interfețe clare și abstracte pentru operațiile de acces la date, cum ar fi crearea, citirea, actualizarea și ștergerea (CRUD). Astfel, pot implementa diferite repository-uri care să utilizeze diferite tehnologii de stocare a datelor, precum baze de date relaționale sau non-relaționale, fără a afecta logica de afaceri a aplicației.

# Structura și dezvoltarea aplicației

# Rezumat

(o descriere sumară, în limba română, a principalelor aspecte tratate în proiect – maxim 300 de cuvinte)

# Abstract

(short description, in english, of main aspects treated in the project)

# Anexa 1

UNIVERSITATEA Transilvania din braşov

FACULTATEA \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Cerere de înscriere la examenul de \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

I. Date personale ale candidatului/ candidatei comunicate în scopul prelucrării necesare pentru organizarea examenului de finalizare studii

1. Date privind identitatea persoanei

Numele de naştere:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Numele (dacă este cazul):\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prenumele: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ CNP \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. Sexul: 🞏Feminin 🞏 Masculin

3. Data şi locul naşterii:

Ziua / luna / anul \_\_\_\_\_\_\_\_ /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_

Locul (localitate, județ, țara) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. Prenumele părinților:

Tata: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Mama:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. Domiciliul stabil: Localitatea \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, jud. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Cod poştal \_\_\_\_\_\_,

str. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ nr.\_\_\_, bloc \_\_\_, sc.\_\_, et. \_\_, ap.\_\_,

Telefon \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, mail \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

II. Date privind şcolarizarea

6. Sunt absolvent(ă) promoția: \_\_\_\_\_\_ / \_\_\_\_\_\_\_ (anul înmatriculării / anul absolvirii)

7. Mențiuni privind şcolarizarea: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

8. Programul de studii \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

9. Durata studiilor \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

10. Forma de învățământ absolvită: 🞏IF 🞏IFR 🞏ID

🞏Fără taxă 🞏 Cu taxă

11. Solicit înscrierea la examenul de \_\_\_\_\_\_\_\_\_\_, sesiunea \_\_\_\_\_\_\_\_\_\_\_\_\_\_ anul \_\_\_\_\_\_\_\_\_\_

12. Lucrarea/ Proiectul de \_\_\_\_\_\_\_\_\_\_pe care o susțin are următorul titlu:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

13. Conducător ştiințific:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

14. Susțin examenul de \_\_\_\_\_\_\_\_\_\_\_\_(pentru prima oară, a doua oară - dupăcaz) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

15. Menționez că sunt de acord cu afişarea rezultatelor examenului conform art.15 alin.9/art.18 alin.9 din OMENCS nr.6125/2016 modificat prin OMEN nr.5643/2017.

SEMNĂTURA, Verificat,

Secretar facultate

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(*numele şi prenumele, semnătura*) F05-PS 7.6-01/ed.2,rev.2

# Anexa 2

Dosar de înscriere la examenul de diplomă/disertație

Pentru înscrierea la examenul de licență/ diplomă/ disertație, absolvenții trebuie să depună la secretariat următoarele acte:

1. Cerere de înscriere la examen
2. Declarație pe proprie răspundere privind prelucrarea datelor cu caracter personal în cadrul procedurii de organizare a examenului de licență/diplomă/disertație
3. Certificat de naştere, în copie legalizată sau în copie simplă care a fost certificată „Conform cu originalul” de către persoana autorizată din secretariatul facultății, în baza prezentării actului în original;
4. Certificat de căsătorie (dacă este cazul), în copie legalizată sau în copie simplă care a fost certificată „Conform cu originalul” de către persoana autorizată din secretariatul facultății, în baza prezentării actului în original;
5. Ordin al rectorului de schimbare a numelui absolventului (dacă este cazul);
6. Diplomă de bacalaureat sau echivalentă cu aceasta, în copie legalizată sau în copie simplă, certificată „Conform cu originalul” de către persoana autorizată din secretariatul facultății, în baza prezentării actului în original – pentru examenul de licență/diplomă;
7. Diploma de licență sau diplomă de inginer şi anexa la diplomă, în copie legalizată sauîn copie simplă care a fost certificată „Conform cu originalul” de către persoana autorizată din secretariatul facultății, în baza prezentării actului în original – pentru examenul de disertație;
8. Certificat de competență lingvistică (numai pentru examenul de licență sau de diplomă), eliberat de instituția organizatoare sau de o altă instituție specializată, națională sau internațională, recunoscută de instituția organizatoare.

Pentru absolvenții proprii, competențele lingvistice certificate prin notele din registrul matricol la o limbă străină de largă comunicare internațională sunt recunoscute de Departamentul de Lingvistică teoretică şi aplicată, fără a mai fi necesar un certificat de competență lingvistică ataşat la dosar.

Pentru absolvenții proprii care susțin examenele de licență/diplomă la alte instituții de învățământ superior, precum şi pentru absolvenții altor instituții de învățământ superior care susțin examenele de licență/diplomă la UTBv, existența în dosar a certificatului de competență lingvistică este obligatorie.

1. 2 fotografii color, recente, dimensiunea ¾ cm, pe hârtie fotografică;
2. Carte de identitate sau paşaport, în copie;
3. Copie a Scrisorii de acceptare la studii / Ordinului MEN sau Atestatului de echivalare (dacă este cazul);
4. Chitanța de plată a taxei de examen (dacă este cazul);
5. Declarație pe proprie răspundere privind originalitatea lucrării de licență/ proiectului de diplomă/disertației;

Absolvenții proveniți de la alte instituții de învățământ superior vor depune documentele prevăzute la pt.1-13, la care se adaugă:

1. Suplimentul la diplomă, eliberat de instituția de învățământ de stat sau particular superior absolvită, din care să rezulte, pentru fiecare semestru şi an de studii, disciplinele promovate, numărul de ore prevăzut pentru fiecare curs, aplicații, lucrări practice – separat, forma de verificare (examen, colocviu, proiect, verificare), creditele şi notele obținute. şi o copie a Suplimentului la diplomă, certificată „conform cu originalul” de către facultatea care o eliberează;
2. Adeverință eliberată de instituția de învățământ de stat sau particular superior absolvită, din care să rezulte calitatea de absolvent, întocmită în conformitate cu Ordinul ................................

Documentele se depun la secretariatul facultății într-un dosar plic de carton, pe care se înscriu:

* Numele şi prenumele absolventului;
* Programul de studii
* Facultatea
* Sesiunea
* Promoția

*Notă: Certificarea conformității cu originalul a copiilor după actele de identitate/de stare civilă şi a actelor de studii se face de către angajații desemnați din cadrul facultății, în baza prezentării documentului în original.*

# Anexa 3

**Grila de evaluare a proiectelor de absolvire**

**Atenție!**

Notarea la proiectul de diplomă va cuprinde o notă pentru formatul tipărit al lucrării de diplomă, o notă pentru prezentarea orală și o notă acordată pentru răspunsul la întrebări. Fiecare notă va avea o anumită pondere în nota finală acordată în funcție de comisia de examinare. Pentru partea tipărită a proiectului de diploma comisia de examinare va acorda punctaje conform criteriilor de mai jos în anumite ponderi stabilite de către fiecare comisie de examinare:

1. Fond

1.1.Originalitate

Indicatori de performanță

* Originalitatea ideii proiectului.
* Moduri originale de rezolvare hardware/software/netware.
* Raport între materialul elaborat de student şi material preluat din bibliografie/webografie, etc.

1.2.Nivel ştiințific

Indicatori de performanță

* Claritatea obiectivelor propuse.
* Calitatea documentării stadiului actual în domeniul proiectului.
* Calitatea şi corectitudinea bibliografiei/webografiei.
* Claritatea realizărilor din proiect (existența schemelor explicative).
* Calitatea proiectării hardware/ software/netware.
* Simularea/emularea/experimentarea/testarea-validareasistemelor/proceselor implementate.
* Relevanța concluziilor în urma realizării proiectului.

1.3.Complexitate

Indicatoride performanță

* Gradul de complexitate al realizării hardware/software/netware.
* Gradul de adâncime al realizării (proiect complet nou sau dezvoltare pornind de la un proiect/concept anterior).
* Complexitatea simulărilor/experimentelor(program de simulare utilizat, gradul de precizie al simulării, rezultate obținute).

1.4.Nivel de implementare

Indicatoride performanță

* Gradul definalizare în raport cu obiectivele propuse.
* Complexitatea funcționalității
* Demonstrarea funcționalității,punere în funcțiune a machetei sau aparatajului, rularea unor programe, etc.

**Verificare antiplagiat**

* **Nu sunt acceptate lucrări cu procent de similaritate Turnitin general mai mare de 25%.**
* **Nu sunt acceptate lucrări cu procent de similaritate Turnitin mai mare de 10% dintr-o singură sursă.**
* **Viza coordonatorului pentru antiplagiat pe baza raportului Turnitineste obligatorie în ultima casetă de vize din fișa proiectului.**

2.Formă

Indicatoride performanță

* Forma grafică în ansamblu, calitateafigurilor, calitatea prezentărilor multimedia.
* Corectitudine în utilizarea limbii române/engleze; traducerea părților de text din figuri în limba română/engleză, cu excepția fișelor de catalog și a altor documente scanate; explicitarea acronimelor din alte limbi.
* Încadrarea în cerințele de editare şi în cerința de cantitate amaterialului.

|  |
| --- |
| DECLARAȚIE PRIVIND ORIGINALITATEA **LUCRĂRII DE LICENȚĂ / PROIECTULUI DE DIPLOMĂ / DISERTAȚIEI** |
| UNIVERSITATEA TRANSILVANIA DIN BRAŞOV  FACULTATEA INGINERIE ELECTRICĂ ȘI ȘTIINȚA CALCULATOARELOR  PROGRAMUL DE STUDII ........................................................................................................................................ |
| NUMELE ŞI PRENUMELE........................................................................................................................................  PROMOȚIA...................................................................  SESIUNEA .................................................................... |
| TEMA LUCRĂRII / PROIECTULUI/ DISERTAȚIEI  ........................................................................................................................................................................................  ........................................................................................................................................................................................  CONDUCĂTOR ŞTIINȚIFIC ...................................................................................................................................... |
| Declar pe propria răspundere că lucrarea de față este rezultatul muncii proprii, pe baza cercetărilor proprii şi pe baza informațiilor obținute din surse care au fost citate şi indicate conform normelor etice, în textul lucrării/proiectului, în note şi în bibliografie.  Declar că nu s-a folosit în mod tacit sau ilegal munca altora şi că nici o parte din teză/proiect nu încalcă drepturile de proprietate intelectuală ale altcuiva, persoană fizică sau juridică.  Declar că lucrarea/ proiectul nu a mai fost prezentat(ă) sub această formă vreunei instituții de învățământ superior în vederea obținerii unui grad sau titlu ştiințific ori didactic.  În cazul constatării ulterioare a unor declarații false, voi suporta rigorile legii. |
| Data: ...................................  Absolvent  ...................................  (*nume, prenume, semnătură* ) |
|  |
|  |
|  |

F07-PS 7.6-01/ed.2,rev.2