

**LUCRARE DE LICENȚĂ**

**Absolvent:** Bolba-Mateescu Andrei-Ioan

**Coordonator:** Vasilescu Anca

Brașov

Iunie-Iulie 2023

Cuprins

[1. Introducere 2](#_Toc136808385)

[1.1. Scurtă descriere a temei 2](#_Toc136808386)

[1.2. Motivația alegere lucrării 3](#_Toc136808387)

[1.3. Contribuții personale 4](#_Toc136808388)

[1.4. Structura lucrării 7](#_Toc136808389)

[2. Abordarea temei 7](#_Toc136808390)

[2.1. Scopul aplicatiei 7](#_Toc136808391)

[2.2. Alte abordări în domeniu 11](#_Toc136808392)

[3. Managementul entităților 12](#_Toc136808393)

[3.1. Modelarea entităților 12](#_Toc136808394)

[3.2. Faza de implementare a bazei de date 12](#_Toc136808395)

[3.3. Managementul identităților si al utilizatorilor 12](#_Toc136808396)

[4. Arhitectura aplicației 12](#_Toc136808397)

[4.1. Separarea preocupărilor 12](#_Toc136808398)

[4.2. Arhitectura REST pentru designul unui API 16](#_Toc136808399)

[4.3. CRUD folosind verbe HTTP 18](#_Toc136808400)

[4.4. Statusuri de raspuns 20](#_Toc136808401)

[4.5. Modelul “Clean Arhitecture” definit de Uncle Bob 21](#_Toc136808402)

[4.6. Securitate 22](#_Toc136808403)

[5. Noțiuni teoretice și tehnologii folosite 24](#_Toc136808404)

[5.1. .NET 7 24](#_Toc136808405)

[5.2. Angular 29](#_Toc136808406)

[5.3. Microsoft SQL Server 34](#_Toc136808407)

[5.4. Securitatea datelor 37](#_Toc136808408)

[6. Prezentarea aplicației 37](#_Toc136808409)

[6.1. Introducere 37](#_Toc136808410)

[6.2. Structura aplicației 37](#_Toc136808411)

[6.3. Persistența datelor 37](#_Toc136808412)

[7. Concluzii și perspective de viitor 37](#_Toc136808413)

[7.1. Concluzii 37](#_Toc136808414)

[7.2. Posibilități de extindere 37](#_Toc136808415)

[8. Bibligrafie 38](#_Toc136808416)

# Introducere

## Scurtă descriere a temei

Primul pas pe care un student îl face pentru a începe o carieră în domeniul IT este, de cele mai multe ori, un internship. Și asta se poate observa foarte ușor pe piața job-urilor în domeniul specificat mai sus. Fie ca vorbim de site-uri precum eJobs, BestJobs sau chiar LinkedIn, se pot observa multe astfel de oferte de internship de la companii care caută studenți pe care să îi învețe ce au nevoie pentru a putea lucra in compania respectiva. Evident nu toți studenții ce reușesc să înceapă un internship reușesc să și obțină un post în acea companie. Acest lucru este cauzat de procesul de urmărire a evoluției studenților, ce este efectuat de un mentor ce le este atribuit in prima zi de lucru la noua companie.

Astfel tema aleasă și anume InternHub vine în ajutorul mentorilor si al managerilor pentru o mai buna urmărire a evoluției si pentru a ușura alegerea când vine vorba de oferirea unei oferte in cadrul companiei. InternHub este o aplicație unde mentori au grupuri cu studenți asignați lor si pot comunica mai ușor, pot vedea evoluția lor si le pot răspunde la întrebări.

Lucrarea își propune ușurarea procesul de urmărire si totodată înțelegere a bitumului de însușire de noi informații pentru fiecare student in parte prin oferirea de task-uri periodice, la care se adaugă provocările zilnice pe care le oferă mentori studenților pe care ii au sub observare. Sistemul de provocări zilnice are la baza idea de motivare prin procesul de gamification. Acest proces utilizează principii precum competiția, recompensele, punctele, clasamentele, provocările și elementele narative pentru a crește angajamentul, motivația și participarea într-o anumită activitate sau context. Astfel fiecare provocare dusă la bun sfârșit reprezintă un punct. Astfel la finalul luni se face un clasament pentru a vedea cine a fost mai activ. Astfel studenții se dezvolta, iar mentori vad rezultatele, le interpretează si pot lua o decizie mai ușor la finalul programului de internship.

In plus, aplicația dispune de un sistem de forum, asemănător lui StackOverflow, unde atât studenți, mentorii cat si administratori pot pune întrebări, nelămuriri sau chiar curiozități, iar ceilalți utilizatori, indiferent de rol pot adaugă răspunsuri. Pentru a știi care răspunsuri sunt bune si care nu, toate postările si răspunsurile pot fii votate in consecința cu upvote, daca se considera o întrebare sau un răspuns relevant, respectiv downvote daca nu este relevant. Astfel postarea are un punctaj care se calculează in felul următor: upvotes – downvotes, iar astfel ne putem da seama de calitatea postării.

Tot in intermediul aplicației se pot planifica ședințe, atât pentru grupuri, cat si individual. Platforma de susținere a acestor ședințe este la alegerea companiei care va utiliza in viitor acesta aplicație, sistemul din cadrul acestui soft are nevoie doar de un link către platforma aleasa, fie ea Google Meet, Zoom sau Teams. Toate au in comun acest avantaj al link ului pentru a intra in conferința.

Un alt feature cheie al aplicației este sistemul de chat, care facilitează comunicarea intre utilizatori, putând sa fie făcute si grupuri. Comunicarea este un aspect important in cadrul dezvoltării tinerilor aflați la început de drum in acest domeniu, astfel InternHub vine in ajutorul lor prin aceste sistem ce comunicare in timp real intre utilizatori.

Toate aceste caracteristici fac din InternHub nu doar o simplă aplicație, ci una care are un scop bine stabilit si care dispune de uneltele necesare pentru a duce la împlinirea scopului propus in timp alocat programului de internship din cadrul companiei ce utilizează acest soft.

## Motivația alegere lucrării

Ideea lucrarii a plecat de la stagiul meu de practica, unde ca in multe locuri nu este inca implementat un sistem de genul acesta, deoarece acest lucru tine fiecare companie in parte. Lispa unui astfel de sistem s-a simtit deoarece feedback-ul nu vine asa de usor cum ne-am imagina, iar existaente unui portal intern de genul lui InternHub ar fii facut o diferenta.

Astfel motivatia alegeri acestei teme este un motiv personal, deoarece simt ca m-ar fii ajutat la cand eram in practica. In plus o aplicatie de acest tip are o aplicabilitate reala in toate companiile de profil IT, care au genul acesta de programe de internship, fie ca sunt pentru stagiile de practica fie ca sunt programe de internship cu scopul pregatirii noilor angajati sa lucreze cu uneltele companiei pe proiecte reale folosind modul de lucru din cadrul companiei.

Utilitatea reala in cadrul unei companii este unul din motivele pentru care am ales sa gandesc, dezvolt si sa implementez o astfel de aplicatie. In plus este o aplicatie destul de unica, dar despre asta o sa vorbim mai multe in subcapitolul 2.2, unde o sa vedem ce alte abordari mai sunt pe piata si daca acestea exista, fiind o aplicatie care este nisata sa rezolve o anumita problema.

Un alt motiv pentru care am ales acesta tema este faptul ca nu este doar aplicatie care nu are o utilitate in viata reala, ci este o aplicatie care poate audce plus valoare unei companii prin dezvoltarea concomitenta a tuturor angajatilor care o folosesc. Este o aplicatie care are o utilitate relativ restransa, fiind bazata pe stagiile de practica din domeniul IT, dar ce poate fii usor moficata si pentru alte stagii precum contabilitate, inginerie etc. Acest lucru este posibili deoarece aplicatia nu are atat de multe caracteristici ce se regasesc doar in domeniul IT, ci are caracteristici ce se ragasesc prin orice domeniu. Astfel un sistem de chat este universal valabil pentru orice companie indiferent de domeniul de activitate. Sistemul de forum este si el valabil universal, diferenta fiind subiectele ce se discuta prin intermediul lui. Acest lucru nu tine de cod ci de ce topic-uri abordeaza oamenii. Exact la fel este si sistemul de challenge-uri. Partea de cod este exact la fel pentru orice arie in care poate fii compania. Putem continua sa discutam despre toate celelate feature-uri, dar nu este relevant, deoarece toate se bazeaza pe acceasi idee.

Avand in vedere toate mentionate mai sus, aplicatia InternHub este o aplicatia cu o utilitate restransa la o prima analiza, dar dupa o judecata mai amanuntita ne dam seama ca acesta poate fii folosite in mai multe domenii. Acest lucru face din acesta aplicatie una foarte utila adaugand, asa cum am mai spus, plus valoare in compania in care este utilizata prin mijloacele prezentate mai sus.

Asadar, motivatia principala pentru alegere acestei teme este utilitatea ei in mai multe domenii si outcome-ul pe care acesta il poate aduce in cadrul unei companii, daca este folosita la capacitatea ei adevarata. De asemenea merita metionat si ca o parte din motivatie este redata de exprienta personala in cadrul unui internship.

## Contribuții personale

Aplicatia are o structura de tip full-stack formata din Angular, ca si front-end si .NET ca si back-end. Legat de stocarea datelor am folosit Microsoft SQL Server, dar mai multe despre structura aplicatiei vom discuta in capitolul 6, la subcapitolul 6.2.

Atat pe partea de Angular, .NET cat si Microsft SQL Server eu am gandit, proiectat si dezvoltat componentele neceasre unei aplicatii de acest tip. Puteam folosi Firebase, dar nu avem acceasi libertate cand vine vorba de creearea ahritecturii back-end-ului, si manipularii requestu-urilor pentru fiecare componenta importanta.

Acum voi trece prin fiecare dintre cele 3 puncte pentru a prezenta si explica contributiile personale pentru acest proiect si pentru a intelege de ce am optat pentru o varianta in detrimentul altei variante.

* **Front-end (Angular)**

În cadrul dezvoltării aplicației InternHub, am jucat un rol important în elaborarea componentei de front-end. Acesta a implicat mai multe responsabilități și a necesitat utilizarea unei varietăți de abilități și cunoștințe.

Am proiectat și dezvoltat interfața utilizator, asigurându-mă că este ușor de utilizat și estetic plăcută. În acest proces, am folosit si dezvoltat cunoștințele mele de UX și UI pentru a crea o interfață care să fie intuitivă și eficientă pentru utilizatori.

Un aspect central al contribuției mele a fost crearea și gestionarea componentelor Angular si comunicarea lor, reusind in multe cazuri sa evit apelarea de requesturi care ar fii ingreunat aplicatia. Am reusit acest lucru prin manipularea DOM-ului (Document Object Model) prin pasarea datelor intre copmmonente cu unelte speicifce Angular. Am folosit acest cadru de lucru pentru a structura aplicația într-o serie de componente reutilizabile, care au îmbunătățit claritatea și manevrabilitatea codului.

Am implementat funcționalități ale utilizatorului, asigurându-mă că acestea sunt funcționale și eficiente. Acest lucru a inclus implementarea sistemelor de autentificare și autorizare, a formularelor și a altor funcționalități esențiale. Legat de partea de autorizare si autentificare vom discuta mai multe la capitolul 5, subcapitolul 5.4 Securitatea Datelor. Am incercat pe cat posibil sa evit duplicarea codului prin utilizarea unei componente pentru doua sau mai multe sarcini asemenatoare, precum adaugarea si editarea unei entitati in baza de date.

Pentru partea de front-end contributiile au fost notabile deoarece este o componenta complexa din punct de vedere atat al structurii cat si al design-ului. Legat de comunicarea cu API-ul s-a bazat in totitalitate pe pe verbe HTTP, mai multe in capitolul 4 Arhitectura aplicatiei, la subcapirolul 4.4 CRUD folosind verbe HPTTP.

* **API (Framework .NET)**

Pentru partea de back-end, am proiectat si dezvoltat un API de la 0 folosindu-ma de design pattern-ul MVC (Model-View-Controller), impreuna cu alte design pattern-uri care au avut ca si rezultat un cod curat, usor de inteles si de parcurs.

Legat de acest pattern, si anume MVC, vom discuta mai mutle in capitolul X, subcapitolul X.X, deoarece are un rol esential in proiectarea unei arhitecturi back-end eficient, usor de inteles urmand principile de Clean Code.

Unul dintre aspectele cheie ale contribuției mele a fost proiectarea și dezvoltarea API-urilor în .NET. Am utilizat aceste API-uri pentru a crea punți de comunicare între front-end și back-end, permițând transmiterea datelor între aceste două componente într-un mod eficient și securizat.

Securitatea este o parte cruciala in crearea unui API si de aceea am acordat o atentie marita acestui luri prin implementarea diverselor strategii de securitate pentru a ma asigura că datele utilizatorilor sunt protejate și că numai utilizatorii autorizați au acces la informații sensibile. Acest lucru este foarte relevant in cadrul aplicatiei InternHub, deoarece exista mai multe tipuri de utilizatori cu drepturi de access diferite si cu drepturi de scriere diferita. Acest lucru a inclus utilizarea de token-uri JWT (JSON WebToken), care permite doar utilizatorilor aplicatiei sa aiba access la resursele oferite de catre back-end, și implementarea de politici CORS. Legat de dreputurile de acees acestea se verifica in momentul in care reuest-ul ajunge in back-end si in fuctie de rang-ul utilizatorul de executa cererea sau se intoarce codul de erorare 401 Unauthorized.

Un alt aspect important de care s-a tinut cont in crearea back-end-ul a fost gerstionarea logicii de business si comunicarea cu baza de date. Acesta a implicat crearea și gestionarea entităților și a relațiilor între ele, precum și implementarea de operațiuni CRUD (Create, Read, Update, Delete) pentru manipularea datelor. Acest lucru s-a realizat prin implementarea design-pattern-ului repository. Un repository reprezintă o colecție de entități (de exemplu, obiecte) și oferă metode pentru accesarea și manipularea acestor entități. De exemplu, un repository poate oferi metode precum getAllElements(), getElementById(), createElement(), editElement(), deleteElement(), etc. O sa discutam mai multe despre acest pattern in caipolul 4, subacpitulul 4.1.

Pentru proiectarea bazei de date s-a folosit Entity Framewok, dupa modelul code-first, ceea ce a dus ca modele din cadrul API-ului sa fie voluminoase, avand infomatii care nu era necesare in totalitate. Pentre a rezolva acest lucru am ales sa folosesc obiecte de DTO (Data Transfer Object) care au in componenta doar informatiile necesare in absolut toate reqest-urile, avand cate un DTO pentru fie care model. Evident avand acest tip de obiecte este necesare si o mapare, care datorile tool-ului numit AutoMapper din ecosistemul .NET s-a realizat cu usurinta fiind necesare doar declarea relatiilor intre obiectele mapate.

Crearea unui API de la zero in detrimentul utilizarii back-end-ului de la FireBase are mai multe avantaje. Prin acestea se numara flexibilitatea modelarii entitatilor si modelarea request-urilor. Modelarea entitatilor este un punct important in cadrul unui back-end, deorece putem modela obiectul trimis catrea front-end dupa nevoile specifica fiecarui request. In plus un API creat de la zero faciliteaza testarea eventualelor bug-uri prin procesul de debugging, care nu este la fel de facil pentru un API de tipul FireBase.

* **Baze de date (Microsoft SQL Server)**

Asa cum am specificat mai sus pentru baza de date s-a utilizat Entity Framework impreuna cu pattern-ul code-first. Acest lucru ma ajuta sa am un management mai bun al bazei de date si faciliteaza schimbarile din baza de date, daca acestea sunt necesare, fie pentru adaugarea unui camp nou, fie pentru adaugarea unui feature nou, ce necesita o table sau mai multe in plus.

Asa cum am afirmat si mai sus am utilizat modelul de creare a bazei de date code-first, ceea ce a însemnat că schema bazei de date a fost generată în cod folosind migrari pentru a crea baza de date in Microsoft SQL Server. Acest lucru a necesitat evident si gandirea relatiilor intrea tabele prin chei straine.

De asemenea, am implementat diverse tehnici pentru a asigura securitatea datelor. Acest lucru a inclus criptarea datelor sensibile, implementarea de controale de acces la nivel de date și utilizarea de tranzacții pentru a asigura integritatea datelor. Un exemplu concret este stocarea datelor sensibile precum parola utilizatorului care este stocata sub forma de 2 valori: valoarea rezultate in urma criptari si cheia dupa care s-a realizat criptarea. Mai multe despre acest subiect o sa discutam in capitolul 5, la subcapitolul 5.4 Securitatea Datelor.

Contributia personala in crearea bazei de date a fost resprectarea pasilor de crearea a bazei de date, fiecare pas avand cerinte si nevoie speciale. O sa trecem pe scurt prin pasii parcursi in crearea bazei de date folosite in aplicatie InternHub:

Primul pas a fost indetificarea nevoilor care se refera la ce tipuri de date sunt necesare pentru fiecare entitate si am ales cea mai buna varianta pentru fiecare tip. Putin mai problematica a fost stocarea de imagini dar pana la urma am gasit o solutie buna si pentru acesta problema.

Apoi am ajuns la etapa de proeictare conceptuala a bazei de date. In aceasta este etapa am definit strcutura bazei de date, relatiile intre tabele si tipurile de relatii intre tabele. Aici am utlizat modele conceptuale, precum diagramele Entitate-Relație (ER), pentru a defini entitățile din baza de date și relațiile dintre acestea.

Dupa aceata etapa am ajuns la proiectarea bazei de date unde am transformat modelul conceptual intr-un model logic de date. Tot aici am definit structura tabelelor, a câmpurilor, a relațiilor și a restricțiilor.

Urmatorul pas a fost acele de implementarea bazei de date unde am creat migrarile specifica pentru ficare entitate dupa modelul Fluent Migrator. Schema bazei de date a fost implementată urmand sa fie populate cu date de inceput pentru a putea face etapa urmatoare, si anume faza de testare si optimizare a bazei de date. Aici am testat performanța, securitatea și funcționalitatea bazei de date.

Toate aceste etape reprezintă un ciclu continuu, întrucât baza de date a fost modifcata pentru a fii in conformitate cu nevoile aplicatiei, find nevoita să evolueze odată acestea. Pentru fiecare modificare adusa bazei de date s-au urmat aceste etape si de accea spunem ca pasii prezentati mai sus reprezinta un ciclu continuu.

Contributiile personale pentru acesta aplcatie sunt destul de mari si acest lucru se datoreaza complexitatii aplicatiei.

## Structura lucrării

Lucrarea este impartita in 8 capitole care contin in mare informatiile

* Capitolul 1: Introducere unde se prezinta in pe scurt tema aleasa, motivatie pentru tema aleasa urmata de contributiil perosonala si structura lucrarii.
* Capitolul 2: Abordarea temei, unde se prezinta scopul aplicatiei si se discuta eventuale alte implementari existente pe piata.
* Capitolul 3: Managementul entitatilor ce prezinta prezinta baza de date, fazele de impementare si managementul utilizatorilor.
* Capitolul 4: Arhitectura aplicatiei ce prezinte cum este impartita aplicatia prin separarea preocuparilor, tipurile de request-uri folosite si modelul Clean Arhitecture prezentat de Uncle Bob.
* Capitolul 5: Tehnologi folosite este probail una dintre cele mai interesante parti a lucrarii deoarece descrie partea practica.
* Capitolul 6: Prezentarea aplicatiei unde se prezinta mai in detaliu structura aplicatiei, comunicarea intre framework-uri si persistenta datelor.
* Capitolul 7: Concluzii si perspective de viitor unde se ajunge la o concluzie legata de acesta aplcaitie si se prezinta perspectivele de viitor ale acestei aplicatii.
* Capitolul 8: Bibliografie un se regasesc link-uri utile pe care le-am folosit in elaborarea lucrarii si a aplicatiei.

# Abordarea temei

## Scopul aplicatiei

Aplicatia are ca și scop principal facilitarea monitorizării evoluției studenților aflați în internship la o companie. Majoritatea își caută un internship pentru că reușesc să castige niste bani, sa prinda niste experinta in acest domeniu si totodata sa isi faca practica de specialitate, care este materie obligatorie in cadrul facultatilor. Astfel la finalizarea internship-ului, stundentul primeste foaia ce ii atesta ca a facut stagiul de practica si dupa o discutie cu mentorul aceste reuseste, daca a avut rezultate bune sa prinda un post de programator ajutor pana la finalizarea studiilor.

Dupa finalizarea programului de practica si obtinerea caietului de practica completat, stundentul merge cu acel caiet la profesor iar acest pune nota pentru acesta materie, fara sa stie exact daca rezultatele de pe acel caiet sunt reale sau nu. Astfel InternHub isi propune sa includa si profesorul coordonator al practicii de specialitate in acest proces prin accesul la evolutia studentului. Astfel el isi poate da seama daca studentul chiar a avut rezultate bune, sau mai putin bune.

In plus aplicatie isi propune nu doar monitorizarea acestor rezultate, ci si imbunatatirea studentilor de trec prin compania respectiva, prin feedback-ul oferit de mentor si prin ajutorul celorlalti din cadrul echipei de intern prin raspunsul proactiv la discutiile deschide pe forum de catre toate lumea.

Scopul forumul este acela de ajutor reciproc in cadrul companiei prin raspunsul oamenilor mai exprimentati la intrebarile celor aflati la inceput de drum. Astefel ei afla noi informatii pe care le pot citii toti internii din companie. Totodata ei pot da raspunsuri care vor fii apreciate de cei mai experimentati, sau care vor fii corectate pentru a nu continua cu informatii eronate.

Acesta oferă o platformă de comunicare și de partajare a cunoștințelor care poate îmbunătăți eficiența și productivitatea. Iată câteva motive specifice pentru care o companie ar putea dori să implementeze un astfel de forum:

* Partajarea cunoștințelor: Asa cum am spus anterior un sistem de forum ajuta la partajarea cunostintelor, ce raman scrise atata timp cat aplicatie ruleaza in cadrul companiei, si este Partajarea cunoștințelor: Un forum de acest gen permite angajaților să pună întrebări și să primească răspunsuri de la colegii lor. Acesta poate fi un mod eficient de a partaja cunoștințele și de a învăța de la ceilalți.
* Documentarea soluțiilor: Odată ce o întrebare a fost răspunsă pe forum, acea informație devine disponibilă pentru toți ceilalți angajați, indiferent de statulul lor in companie, fie ca sunt interni sau mentori. Aceasta poate ajuta la prevenirea duplicării eforturilor și la asigurarea că soluțiile la probleme comune sunt ușor accesibile.
* Implicarea angajaților: Forumul permit angajaților sa arate proactivitate cand vine vorba de cei aflati la inceput de drum, ce le pot devenii colegii dupa aceasta perioada. Interesul companiei este sa dezvolte acesti tineri si sa ramana la ei in companie. De aceea aceasta proactivitate este esentiala pentru asta, iar forumul faciliteaza acest pas. Acest lucru poate îmbunătăți satisfacția la locul de muncă și poate încuraja angajații să se dezvolte profesional.
* Creșterea eficienței: Prin furnizarea de răspunsuri la întrebări comune, forumul ajuta la economisirea timpului folosit pentru cautarea solutiilor la probleme comune si poate reduce stresul angajatilor.
* Îmbunătățirea comunicării: Forumul ajuta la îmbunătățirea comunicării în cadrul companiei in care este utilizat oferind un spațiu unde angajații pot discuta și pot colabora pe diverse subiecte.

Sistemul de chat este un sistem intalnit in aproape orice aplicatie de pe acest pamant, deoarece faciliteaza comunicarea intre oameni. Sistemul implement aplicația InternHub isi propune acesta eficienta in comunicare, avand posibilitatea comunicarii de tip 1-la-1 dar si pe grupuri de interes. Aceasta isi doreste sa fie un sistem utilizat nu doar pentru partea de munca din cadrul companiei dar si pentru parte de timp liber prin crearea unor grupuri cu interese comune, fie ca vorbim de sport, moda, arta culinara sau chiar partea auto.

In plus acest sistem ofera cateva benefici chiei ce il ajuta la indeplinirea scopului sau final si anume eficienta. Printre aceste puncte tari ale sistemului de chat se numara:

* Comunicare rapidă: Chat-ul permite angajaților să comunice rapid unul cu celălalt, fără a fi nevoie să trimită e-mailuri sau să inițieze apeluri telefonice. Aceasta poate îmbunătăți eficiența, deoarece permite angajaților să rezolve rapid problemele și să răspundă la întrebări.
* Colaborare: Chat-ul faciliteaza colaborarea între echipe sau între membrii echipei. Angajații discuta despre proiecte, împărtăsesc idei și primi input în timp real de la toti membri echipei. Acest lucru este foarte impotant pentru un stundet aflat pe pozitie de internship, deoarece are suport din partea unor oameni cu experienta. Astfel procesul de asumare al informatii devine mult mai facil.
* Documentarea conversațiilor: Spre deosebire de conversațiile telefonice sau față în față, chat-urile se pasteaza si se pot accesa ulterior. Acest lucru este util pentru urmarirea procesului de invatare al stundentului dar si pentru procesul de monitorizarea al unui proiect.
* Disponibilitate: Chat-ul permite angajaților să rămână conectați și să comunice eficient, indiferent de locație sau fus orar.Desigur acest lucru este posibil si telefon, dar acesta vine cu anumite costuri ce difera in functie de locul de unde este initiat apelul. Esita si varianta aplicatii lor de tip WhatsApp, dar oameni prefera sa separe viata profesionala de cea personala, astfel un sistem de chat poate ajuta la acest lucru. Acest lucru poate fi deosebit de util pentru companiile cu angajați sau echipe la distanță.
* Îmbunătățirea moralului angajaților: Chat-ul intern din cadrul companiei contribuie la îmbunătățirea moralului și a culturii companiei. El ofera un loc pentru angajați să socializeze și să creeze o atmosferă mai prietenoasă și mai colaborativă, ceea ce duce la un mediu de lucru sanatos ce ajuta in timp eficienta si calitatea proiectelor.
* Suport tehnic intern: Chat-ul poate fi utilizat pentru a oferi suport tehnic rapid pentru angajați, in special celor alfati in primele luni intr-un context de genul acesta. Fiind prima interactiune cu mediul de lucru intrebarile ce apar in minte celor aflati la inceput sunt inevitabile si au nevoie de raspuns rapid. Chat-ul reuseste sa faca acest lucru prin raspunul scris, care ramane acolo si poate fii o sursa de documentare pentru o alta problema. Acesta poate fi un canal eficient pentru rezolvarea problemelor tehnice sau pentru a răspunde la întrebări.

Sistemul de challenge-uri zilnice, avand un sistem de gamification in spate, in colaborarea cu cel de task-uri mai lungi duce la cresterea eficientei lucrului la proiecte si totodata duce la dorinta de acumulare de informatii noi prin finalizarea task-urilor.

Sistemul de challenge-uri zilnice se bazeaza pe ideea completarii lor de catre interni si oferirea de puncte de catre mentor. Aceste puncte se vor folosi la crearea unui clasament, ce se actulizeaza pe zi ce trece si care, la final, poate oferii un premiu primilor 3 angajati. Acest pas este la alegerea companiei, iar implementarea lui duce la motivarea stundetilor sa fie cat mai competitivi si sa completeze cat mai bine acest challenge pentru acumulare de puncte cat mai multe.

Asemanator lui Jira, sistem de task-uri isi propune motinorizarea evolutiei studentilor prin completarea treptata a lor. Progresul poate fii vazut atat de catre mentor cat si de catre profesul resposanibil de stagiul de practica din facultatea din care face parte studentul.

Cum Jira s-a dovedit a fii eficient de-a lungul timpului, sistemul de task-uri din cadrul aplicatiei InternHub isi propune cat de eficient se poate raportat la Jira. Acest sistem ofera numeroase beneficii printrea care:

* Urmărirea progresului: Sistemul de task-uri permite mentorilor si profesorului resposabil să urmărească progresul studentului in cadrul perioadei de practica. Astfel se pot sesiza anumite lacune pe care studentul le are si se pot remedia usor atat din partea mentorului din cadrul companiei cat si din partea profesorului.
* Transparența: Sistemul de task-uri oferă transparență asupra a ceea ce se lucrează, cine lucrează la ce, și care sunt termenele. Aceasta poate duce la o mai bună coordonare și la o mai bună înțelegere a sarcinilor de către toată echipa.
* Eficiența: Prin structurarea și organizarea muncii, un sistem de task-uri poate ajuta la creșterea eficienței și la reducerea riscului de a uita sau a neglija sarcini importante.
* Comunicarea și colaborarea: Sistemul de task-uri faciliteaza comunicarea și colaborarea în cadrul unei echipe. Membrii echipei pot adăuga comentarii la sarcini, pot împărtăși actualizări și pot colabora pentru a rezolva probleme.

Sistem de feedback este un sistem vital pentru evoluția oricarui angajat din cadru unei companii, deoacere prin feedback oamenii evolueaza si pot observa greseli pe care altfel nu le-ar fii sesizat la fel de usor.

Un sistem de feedback are numeroase avantaje atat in cadrul unei echipe de lucru la un proiect cat si in cadrul unui grup de oameni. Printre avantajele sistemului de feedback din cadrul aplcaitiei InternHub se enumera:

* Îmbunătățirea performanței: Feedback-ul regulat ajută oamenii să înțeleagă ce fac bine și unde au nevoie de îmbunătățire. Astfel un student care nu stie foarte multe despre acest domeniu fiind la inceput poate evolua usor prin implementarea feedback-ului primit de la oamenii mai experimentati. Acest lucru le permite să își îmbunătățească abilitățile și să crească în performanță.
* Dezvoltare profesională: Feedback-ul ghidaza dezvoltarea profesională a oamenilor, indicându-le zonele în care au nevoie de mai multă instruire sau experiență. Astfel mentori stiu unde trebuie sa puna accent, iar profesorul responsabil poate folosii aceste informatii pentru a ajuta studentul in dezvoltarea sa cat timp mai este in facultate.
* Rezolvarea problemelor: Prin feedback, problemele pot fi identificate și rezolvate rapid. Acest lucru poate duce la o îmbunătățire a eficienței și a productivității.
* Recunoașterea muncii bine făcute: Un sistem de feedback oferă oportunități pentru recunoașterea și aprecierea oamenilor pentru munca lor bine făcută, ceea ce poate duce la creșterea moralului și a angajamentului față de companie.
* Comunicare deschisă: Un sistem de feedback poate ajuta la crearea unei culturi de comunicare deschisă și onestă în cadrul companiei.

Astfel aplicatia nu are ca scop doar monitorizarea procesului de evolutie al unui student si usurarea alegerii daca se continua colaboarare sau nu, ci si dezvoltarea tuturor angajatilor ce utilizeaza aceasta aplicatie in cadrul companiei.

## Alte abordări în domeniu

Problema monitorizarii evolutiei angajtilor, fie ca vorbim de interni sau de angajati a mai fost abordata la cererea companiei care utilizeaza aplicatia. Este si normal ca aplicatiile de genul acesta sa difere fie prin feature-uri, fie prin modul de gandire ar ahritecturii componentelor ce o formeaza. Avand in vedere ca domeniul IT devine tot mai atrctiv, chiar si pentru reconversii profesionale, companiile isi doresc sa pastreze cei mai bun angajati. Astfel se apeleaza la tipul acesta de soft-uri care nu pot fii gasite pe piata, deoarece fiecare companie are nevoie diferite ce vin cu cerinte diferite si scenarii de utilizare tot mai difersificate.

Astfel companiile aleg sa angajeze studenti talentati cu dorinte de invatare in detrimentul experientei si sa ii invete pentru a putea lucra pentru proiectele companiei. Astfel internship este pratic ca un curs pe o perioada stiuta de la inceput si ce are ca finalitate simulararea unui proiect de demo. In tot acest timp este necesara o monitorizare a evolutiei, iar pentru acest lucru este necesar un soft de monitorizare. Practic exista si alte abordari al ideei de monitorizare a evolutiei, dar specific pentru stagiile de internship, fiind un lucru nisat, nu exista mai deloc abrodari.

In alte cuvinte aplicatie InternHub pleaca de ideea platformelor precum E-learning cu aplicabilitate in cdrul unei companii. Diferenta majora este ca in platforma InternHub nu exista note, exista doar feedback pe care cei alfati la inceput de drum il pot folosi pentru a se dezvolta si pentru a evolua in carierea pe care si-au ales-o.

Un concept forate la moda este acela de „gamification” care este procesul de utilizare a elementelor și tehnicilor de design de jocuri în contexte care nu sunt legate de jocuri, pentru a îmbunătăți implicarea utilizatorului, concentrarea sau învățarea. Aceasta este o tehnică populară în domenii precum educația, marketingul și sănătatea, unde poate ajuta la creșterea motivației, a angajamentului și a loialității utilizatorilor. Astfel fiecare task facut are un amonit punctaj, iar la final se creeaza un clasament general prin care oamenii sunt premiati pentru progresul lor.

Acest concept functionaza deoarece se baseaza pe concurenta, iar oamenii sunt fiinte competitive, ceea ii face sa isi doreasca sa termine ce au da facut cat mai repede si cat mai eficient, stiind ca la final exista o recompensa. Si simplul clasament are acelasi efect de starnire al oamenilor, care ajung in timp sa fie din ce in ce mai eficienti.

Asadar, aplicatia InternHub este o aplicatie originala ce are ca puncte de plecare principile pe care se bazeaza o aplicatie de monitorizare a evolutiei, la care se adauga principii ce se regasesc intr-o flatforma precum E-learning.

# Managementul entităților

## Modelarea entităților

## Faza de implementare a bazei de date

## Diagrame

## Managementul identităților si al utilizatorilor

# Arhitectura aplicației

## Separarea preocupărilor

Separarea preocuparilor in cadrul unei aplicatii este un element fundamental si reprezinta de asemeneaun principiu important in dezvoltarea software. Acesta se refera la impartirea codului in module astafel incat fiecare componenta sa aiba o singura resposabilitate. Avantajele care vin odata cu acesta separare sunt testarea mai usoara la care se adauga usurarea intretinerii. In plus acest principiu faciliteaza extinderea aplicatiei, pastrand modularitatea codului. Exista mai multe modalitati de realizare a acestui principiu. Iata cateva concepte:

1. **Arhtitectura Model-View-Controller(MVC).**
2. **Principiul responsabilității unice (Single Responsibility Principle – SRP).**
3. **Injectarea dependențelor (Dependency Inversion Principle – DIP).**
4. **Utilizarea design pattern-urilor.**
5. **Utilizarea modularizării.**
6. **URI-uri semantice.**
7. **Formatul de reprezentare**.

Acum ca avem prezentate conceptele putem discuta putin despre fiecare, si sa ii prezentam apartenenta la API-ul aplicatiei prezentate.

Legat de arhitectura MVC aplicatia este impartita in trei module principale, dupa cum ii spune si numele, si anume:

* **Model:** responsabil pentru gestionarea datelor
* **View:** responsabil pentru afisare datelor primite prin intermediul controller-ului. In plus el se ocupa de interactiunea cu utilizatorul final.
* **Controller:** responsabil pentru comunicarea intre model si view, precum si gersionarea evenimentelor utilziatorului.

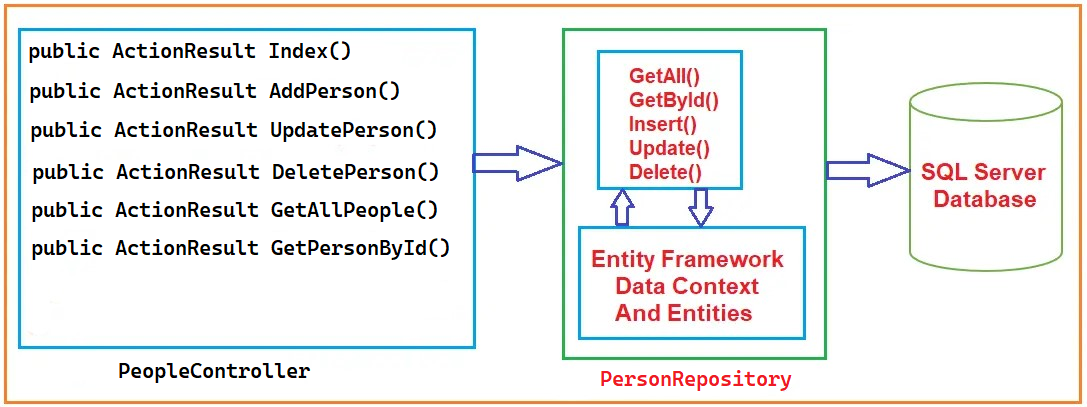
Fiind o aplicatie de tip full-stack, daor doua dintre cele trei elemente se regasesc in cadrul server-ului, si anume Model-ul si Controllere. In cazul aplicatiei InternHub, View-ul este repzentat de partea de front-end, care preia datele de la Controller si le expune utuilizatorul intr-un mod placut. In plus toate actiunel utilizatorului sunt monitorizare si gestiunate de catre client si trimise catre controller pentru manipularea acestor actiuni.

Acest principiu este legat de principul 5 care sa refera la impartirea aplicatiei in module logice si funtionale ce au in componenta elemente relevante. Aceasta poate fi realizată prin intermediul modulelor în Angular, pachetelor sau bibliotecilor separate în .NET sau prin organizarea fișierelor și directoarelor într-un mod coerent și modular. Acest lucru este vizibil in figurile 4.1.1 pentru Angular si 4.1.2 pentru .NET.

|  |  |
| --- | --- |
| **Figura 4.1.1 – Impartirea client-ului pe module logice si funcitonale** | **Figura 4.1.2 – Impartirea server-ului pe module logice si functionale** |

Astfel impartirea pentru partea de server se face prin intermdiul organizarii fisierelor si directoarelor in conformitate cu principiul 2, iar pentru partea de client acst lucru este posibili datorita pachetelor ce faciliteaza acesta separare. Am specificat mai sus ca sunt realizate in conformitate cu principiul 2 si anume pirncipiul resposabilitatii unice.

Acest principu se refera faptul ca fiecare componenta trebuie sa aiba o singura reponsabilitate, ce are ca si avantaj mentinerea codului mei usor si ajuta la imbunatatire si testare. Pentru partea de server acest principiu este resctat atat la niveleul directoarelor cat si la nivelul claselor. Totodata se poate observa imparitirea server-ului in conformitate cu ahritectura MVC, avand director atat pentru Model, cat si pentru controller. Datorita acestei impartiti modulare a solutiei back-end-ului este mult mai usoara utilizarea unor design pattern-uri, conform principiului 4.

Design pattern-ul utilizat, ce are un impact major asupra modularitatii aplictiei este repository. Acest design pattern actioneaza ca un intermediar intre restul aplicației și logica de acces la date. Practic acesta izoleaza toate codurile de acces la date de restul aplicatiei, avand ca avantaj principal usurinta modficarii codului cand este necesar. Acest lucru est posibil datorita faptului ca tot codul este gasit intr-un singur loc. De asemenea si testarea controlerelor devine mai usoara[-3]. Spre exemplu avem urmaroare diagrama:

**Figura 4.1.3 – Diagrama pentru controller si repository**

După cum puteți observa în diagrama de mai sus, acum controlerul People nu va comunica direct cu clasa contextului de date a Entity Framework. De asemenea, acum nu există interogări sau alte coduri de acces la date scrise în metodele de acțiune ale controlerului People, ele fiind realizate in interiorul repository-ului Person. Avem o diferenta de nume intre repository si controller care este cauzata de unul dintre principiile arhitecturii REST despre care vom discuta mai multe subapitolul urmator. Repository-ul Person utilizează clasa contextului de date a Entity Framework pentru a efectua operațiile CRUD. După cum puteți vedea, acum repository-ul Person are metode precum GetAll(), GetByID(), Insert(), Update() și Delete(). Aceste metode vor efectua operațiile tipice CRUD asupra bazei de date subiacente. Controlerul Employee utilizează aceste metode pentru a efectua operațiile necesare în baza de date.

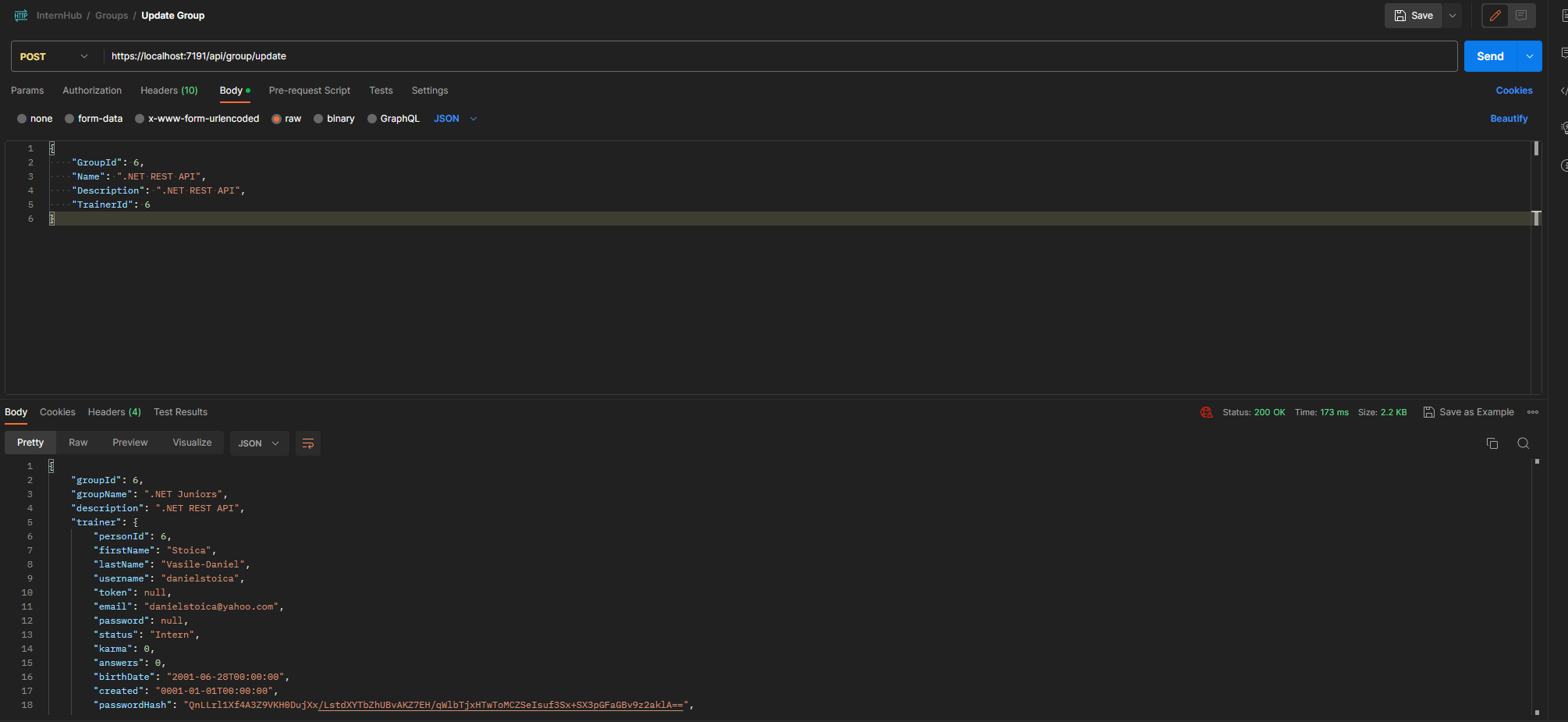
Un repository reprezintă pur și simplu o clasă definită pentru o entitate, care conține toate operațiile posibile în baza de date. In plus in aplicatia Intern Hub, pe partea de server, exista o interfata pentru fiecare repository. Toate interfetele sunt numite de tipul I + nume entitata + Repository, ir repository-urile sunt de forma + nume entitata + Repository. Acest lucru ajuta la un vizune de ansamblu mai buna pentru metodele existente in fiecare repository. De exemplu, un repository pentru o entitate Person va conține operațiile CRUD de bază și orice alte operații posibile legate de entitatea Person.

Un alt motiv pentru care am utilizat acesta modalitate si anume implementarea unei interfete tine de principiul 3 si anume Injectarea dependențelor (Dependency Inversion Principle – DIP). Astfel fiecare repository are injectat atat contextul cat si IMapper-ul, care se ocupa cu maparea obiectelor, iar unele mai au injectat si alte repository-uri, acolo unde exte cazul. Atat contextul cat si interfele sunt de tip private readonly. Pentru partea de controllere, avem injectate toate reporitory-urile de care avem nevoie. Numarul lor poate sa difere in functie de controller, unele avand nevoie de mai multe de o singura dependinta. Acest lucru se poate vede in figurile 4.1.4 pentru repository doar cu conextul si mapper-ul injectate, 4.1.5 pentru repository-uri care au si alte reporitory-uri pe langa contxt si 4.1.6 care reprezinta controller-ul si un numar variabil de repository-uri injectate.

|  |  |
| --- | --- |
| **Figura 4.1.4 – Repository cu conext si Mapper injectate** | **Figura 4.1.5 – Repository cu conext, Mapper si alte repository-uri injectate** |
| **Figura 4.1.6 – Controller cu repository-uri injectate** |  |

Un alt pricipiu prezent in aplicatia InternHub este URI semnatica care se ferera la faputl ca URI-urile ar truni sa fie semantica si sa reflecte actiunile si entitatile pe care le reprezinta. In plus ele ar trebui sa fie descriptiva si usor de inteles. Avem un exemplu de astefel de URL in figura 4.1.7.

Un ultim principiu al separari preocuparilor este formatul de reprezentare care se refera la faputl ca datele returnat ar trebui sa fie intr-un format standard, precum JSON sau XML. Pentru aplicatia prezentate formatul ales este JSON si este prezent atat pentru comunicarea de la server la client cat si pentru comunicarea clientului cu server-ul prin intermediul body-ului. Pentru ambele tipuri de comunicare avem reprezentarea in figura 4.1.7.



**Figura 4.1.7 – Formatul datelor in comunicarea client-server**

Conform figurii 4.1.7 URL este: <https://localhost:7191/api/group/update>. Este un URL semantic, usor de inteles. Doar uitandu-ne la el ne dam seama ca se face update la un grup, ce primeste datele ce trebuie actulizate prin intermediul unui body. Este un URL ușor de înțeles pentru dezvoltatori și utilizatori.

Legat de formatul de reprezentare acest lucru se poate observa usor din figura 4.1.7. Body-ul este de tip JSON, avand ca scop trimiterea de informatii catre server, iar raspusnul este de asemena de tip JSON.

Prin urmare, putem afirma ca API-ul aplicatiei prezentate tine cont de principile prezentate mai sus si urmeaza cele mai bune practici pentru a asigura un API eficient, ușor de utilizat și scalabil.

## Arhitectura REST pentru designul unui API

Mijloace API A pplication P rogramming I nterface și, la fel ca orice altă interfață, permite interacțiuni. În cazul unui API, acesta permite interacțiuni între sisteme, urmând un set de standarde și protocoale pentru a partaja caracteristici, informații și date . Cu alte cuvinte, oferă dezvoltatorilor posibilitatea de a construi și proiecta produse și servicii care vor comunica cu alte produse și servicii.

In programare termenul de arhitectura soft-ului se refera la un se de reguli de proiectare care identifică tipurile de componente și conectori ce pot fi utilizați pentru a compune un sistem sau un subsistem. Pipe and Filter, Layered, Push Based sunt cateva exemple comune de astfel de arhitecturi soft. Unul dintre cele mai intalnite arhitecturi este arhitectura REST ce are o abordare clinet-server, fara stare, in care reusrsele sunt vazuta ca fiind serviile web ce pot fi indentificate prin intermediul URl-ului. Modul prin care clinetii pot utiliza aceste resurse este prin intermediul metodelor HTTP, dar despre asta vom vorbii in capitolul 4, la subcapitolul 4.3 CRUD folosind verbe HTTP. In plus metodele HTTP pot realiza atat transferul continutul aplictiei cat si aciuni asupra resurselor. Un punct forte al arhitecturii REST este acela ca interactiunea intre el si protocolul HTTP se realizeaza fara intreruperi.

REST este un acronim de la **RE**presentational **S**tate **T**ransfer Transferul Stării Reprezentationale). Acesta nu este un termen nou. Acest stil arhitectural are ca scop principal centralizarea datelor ce urmeaza sa fie utilziate de catre diverse aplicatii client. REST este bazat pe un set de principii care descriu modul de definire si adresare al resurselor de retea. Roy Fielding este cel care a descris initial aceste principii in anul 2000.

Ca orice alta arhitectura, REST are de asemenea reguli si principii de porgramare care trebuie urmata pentru a obtine resultatele dorite de la aplicatie. Printre aceste reguli de enumera[-1]:

1. REST se bazeaza pe substantivul resursei, de aceea este indicat ca URL-ul sa se termine cu un substantiv
2. GET,POST,DELETE si UPDATE sunt verbele folosite pentru a indentifica o actiune. Pe lana acestea mai avea: Trace, Options, Connect si Patch
3. Aplicatia web trebuie organizata in resurse si verbe
4. Pentru consecvente trebuie folosit pluralul in URL-uri
5. Este necesara trimiterea unui cod corect pentru identificarea succes-ului sau a erorii.

Mai jos sunt cate exemple de URL-uri ce respect regule expuse mai sus:

* api/people GET Se obtin toate persoanele.
* api/people POST Se adauga o peroana noua.
* api/people/1 PUT Actualizați o persoana cu id = 1. Informatiile rederioate la persoana se trasnmit prin intermediul body-ului.
* api/people/1 DELETE Ștergeți o persoana cu id = 1
* api/people/popescumihai GET Obțineți o perosna cu username = popescumihai

Acum ca am discutat regulie pe care se bazeaza REST API, trebuie luate in considerare si cele 6 constrangeri arhitecturale tipice REST API[-1]:

1. **Interfata uniforma.** Acesta costrangere se refera la facputl ca orice resursa trebuie sa fie identifica in mod uni prin intermediul unui URL, folosind doar metode precum GET,POST,PUT sau DELETE. In plus este necesara prezenta unei metode unice de comunicare si server-ul, indiferent de tipul de dispozitiv pe care ruleaza aplicatia.
2. **Stateless.** Acesta constragere se refera la modul de gestionare a starii care trebuie facuta intotdeauna pe dispozitivul clientului, ci nu pe server.
3. **Cacheable.** Acesta constragere se refera de proiectare al resuselor astfel incat sa poate perimite stocarea în cache.
4. **Client-server.** Acesta constragere se refera faptul ca trebuie sa existe o delimitare clara intre client si server. Astefel fiecare sa aiba resposabilitatile sale. Spre exemplu, accesul la date si modificarea cestora este resposabilitatea serverului, iar elementele de UI/UX sunt resposabilitatea client-ului.
5. **Sistem stratificat.** Acesta constragere se refera faptul ca arhitectura REST permite utilizarea unei structuri stratificate compusă din mai multe niveluri de servere.
6. **Cod la cerere.** Acesta constragere se refera faptul ca de obicei forma sub care sunt trimise resursele la client este de tip JSON, dar server-ul poate trimite, cand este necesar, un cod executabil la cerere.

## CRUD folosind verbe HTTP

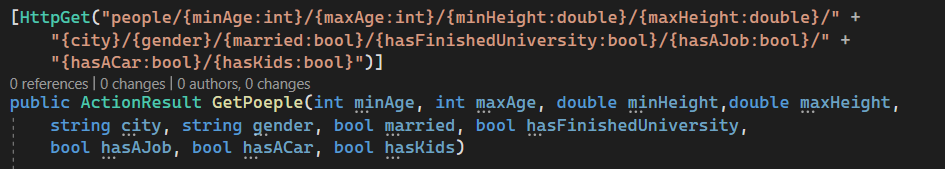
Modalitatea favorita de comunicare intre partle componente ale unei aplicatii full-stack, aici vorbind dspre forn-end si back-end, este indiscutabil HTTP. Comunicarea client-server se foloseste de acest protocol prin trimiterea de cereri, primind ca si raspuns unul specific HTTP, despre care vom vorbi mai in detalie in subcapitolul urmator. Astfel exista doua tipuri de mesaje:

* **Request** - cel trimis de la cilent catre server, ce are ca scop principal declasarea unei actiuni.
* **Raspuns** - cel trimis de la server catre client, ce are ca scop trimiterea ruapusnul dupa finalizarea actiunii declansate de request.

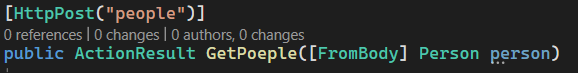
Aceste mesaje sunt sub forma unui text ASCII dispus, de cele mai multe ori, pe mai multe linii. Prima linie contine verbul HTTP ce este urmat de calea catre resursa la care ne dorim sa ajungem. Asta in cazul reqest-ului. Pentru rasuns pe prima linie gasim codul de raspuns si mesajul in sine.

Pentru descrierea actiunilor necesare efectuari opratiilor de tip CRUD se folosesc metodele HTTP echivalente. Singura exceptie este cazul in care dorim un request care sa aiba in componenta si un body. In cazul acesta nu vom putem folosi verbul GET deoarece acesta nu suprta un body, cel putin in cazul unor framework-uri de tip front-end, ci vom fii nevoiti sa apelam la verbul POST pentru a putea face acest lucru. Pentru cazul in care body-ul pe care vremsa il trimitem are putine campuri este de prefereat sa folosim GET, dar in cazul in acre avem multi astfel de parametri, insiruirea lui in cedrul URL-ului ar ingreuna atat citirea acestuia in scopul gasiri si rezolvarii unei posibile probleme, dar si ingreunarea citirii si urmariri fucntiei echivalente.[0]

Spre exemplu daca dorim sa primim ca raspuns de la API toate persoanele din baza de date care au varsta minima 18 ani, varsta maxima 50, intaltimea minima 150 cm, maxima 180 cm, din Brasov, sa fie barbat, sa fie casatorit, sa aiba facultatea terminata, sa aiba job, sa aiba o masia, si sa aiba copii. Pentru asta ave doua varinate. Figura 4.4.1 reprezinta funtie in cazul in care utilizam verbul GET, iar figura 4.4.2 reprezinta acelasi request dar realizat cu verbul POST. Se poate observa o diferenta notabila intre cele doua. In cazul acest chiar daca nu este cea mai corecta alegere verbul POST este o alegre mai bun in cazul request-urilor de acest fel. Este evident ca daca avem putini parametrii nu vom opta pentru acesta abordare, deoarece nu are nici un avantaj real pentru un numar mic de parametrii.



**Figura 4.3.1 – Request realizat cu verbul GET**



**Figura 4.3.2 – Request realizat cu verbul POST**

Pentru prima varinat un URL arat in felul urmator:

* <https://localhost:5421/api/people/18/50/150/180/Brasov/masculin/1/1/1/1/1>

Pentru cea de-a doua varinta un URL arata in felul urmator:

* [https://localhost:5421/api/people](https://localhost:5421/api/people1), urmat de body unde vom pune informatiile necesare.

Si in cazul acesta se poate oberva o incarcare inutila a URL-ului in primul caz, fata de al doilea un avem un URL simplu usor de urmarit si totodata usor de schimbat valorine care ne intereaza pentru acest request. In plus pentru genul acesta de probleme cu multi paramtrii nu este necesara scierea mai multe functii deoarece putem modifica body-ul pentru a ne satisface nevoile.

|  |  |  |
| --- | --- | --- |
| Actiune | SQL | HTTP |
| Create | Insert | PUT |
| Read | Select | GET |
| Update | Update | POST |
| Delete | Delete | DELETE |

Metodele HTTP, precum GET si POST, sunt utilizate pentru executarea actiunilor necesare in vederea crearii, citirii, acutalizarii si stergerii datelor in baza de date. Aceste operatii sunt cunoscute sub numele de operatii CRUD (create, read, update si delete). Fiind adesea folosite este evidenta existenta unei analogii cu operațiile în baza de date relațională SQL, ce se bazeaza pe un set comun de verbe prezentat in tabelul de mai jos.[0]

**Tabelul 1 – Relatie intre REST API si protocolul HTTP**

Un WebService RESTful se bazează pe metodele HTTP şi pe conceptele arhitecturii REST. Acesta defineşte tipic un URI de baza pentru resurse şi tipurile MIME suportate (XML, Text, JSON, Protocol Buffers, etc) şi setul de operaţii HTTP. Aceste metode standard sunt[0]:

* **GET** – Defineste accesul la citirea de datelor, evitand efectele secundare si modficarea lor in urma executiei cererii de tip GET.
* **PUT** – creeaza o resursa noua, si are ca si caracter princial idempotenta
* **DELETE** – asemea lui PUT, si DELETE este idempotenta si se ocupe cu stergerea resursei.
* **POST** – este asemanator cu PUT, avand in plus avantajul ca poate sa si modifica o resursa nu doar sa o creeze.

Un aspect important de mentionat pentru arhitectura HTTP este acela ca este independent fata de protocolul HTTP, ceea ce insemna ca arhitectura REST nu necesita utilizarea neapareta a protocolului HTTP.

## Statusuri de raspuns

Dupa cum am spus, fiecare request are parte si de un raspuns, care este sub forma unor coduri numerice care indica daca actiunea dorita a fost relizata cu succes sau nu. Actiunea declansata de request nu este intotdeauna executate cu succes, iar aceste statusuri de raspuns informeaza clientul pentru a stii ce trebuie sa faca.

Aceste coduri numerice se impart in categorii principale cu un inteles specific pentru fiecare dintre ele. Acestea se impart in 5 categorii pentru vom vedea in tabelul 2.

|  |  |  |
| --- | --- | --- |
| Cod numeric | Tip | Mesaj |
| 1xx | Informational | Cerere in curs de procesare sau informatii incomplete |
| 2xx | Succes | Cerere executata cu succes |
| 3xx | Redirectionare | Sunt necesare actiuni suplimentare in vederea finalizarii cererii |
| 4xx | Eroare a clientului | Cerea contine erori sau nu poate fii procesata |
| 5xx | Eroare a serverului | Serverul intampina probleme in rezolvarea cereri |

**Tabel 2 – Categorii de staturi de raspuns**

Acum ca stim in care categoriile de statusuri de raspuns putem trece si prin cateva exemple de status-uri de raspuns.

|  |  |  |
| --- | --- | --- |
| Cod numeric | Mesaj | Explicatie |
| 200 | OK | Cerere procesats cu succes |
| 201 | Created | Resursa creata cu succes |
| 400 | Bad Request | Cererea contine erori de sintaxa sau nu poate fii inteleasa |
| 401 | Unauthorized | Accesul restrctionat la resursa |
| 404 | Not Found | Resursa solicitata nu a fost gasita |
| 500 | Internal Server Error | Serverul intampina probleme in rezolvarea cereri |

**Tabel 3 – Exemple de statusuri de raspuns**

Evident că există și alte coduri de stare HTTP, iar fiecare cod are semnificația sa specifică în cadrul protocolului HTTP, iar fiecare aplicatie are modul ei de a le intelege si a le procesa. Aceste coduri de stare sunt esențiale pentru a comunica rezultatele cererilor între client și server. In cadul unei aplicatii este necesar ca aceste coduri sa fie intelesa si omterpreatre corect pentru a putea diagnostica si remedia eventualele probleme care pot aparea in cadrul comunicarii client-server.

## Modelul “Clean Arhitecture” definit de Uncle Bob

Robert C. Martin, cunoscut sub numele de Uncle Bob ne prezinta in cartea sa Clean Arhitecture: A Craftsman’s Guide to software Structure and Design [-4], un model de arhitectura pgragmatic, numirt sipmplu „Clean Arhitecture”, ce a evoluat din arhitectura Onion si din cea hexagonala.

Pe langa cartea publica in 2017, aceste mai are un blog numit The Clean Code Blog [-2], unde are publicatii inca din 2011 pana in prezent. Tot in blogul sau gasim o postare despre Clean Arhitecture, unde practic expune accelasi idei ca in catea sa, sensibil modificate, avand in vedere distanta de timp.

Astfel in 2012, in cadrul postari despre Clean Arhitecture acesta observa o gama intreaga de ideilegate de arhitectura sistemrlor printre care: arhitectura Hexagonala, Onion, Screaming DCI si BCE. Toate acesta au in comun obiectivul. Acela de separare al preocuparilor, subiect ce a fost in subcapitlol precedent. Toate au cel putin un strat pentru regulile de business și un alt strat pentru interfețe[-2].

Fiecare dintre aceste arhitecturi produce sisteme care sunt:

* **Independente de framework-uri.** Arhitectura curată nu se bazează pe existența unor biblioteci software complexe și pline de funcționalități. Acest lucru oferă libertatea de a utiliza framework-uri ca instrumente, fără a fi nevoie să adaptați sistemul la constrângerile impuse de acestea.
* **Testabile.** Testarea regulilor de bussines sa se poata face fara interventia interfetei, bazei de date, a serverului web sau ale altor elemente externe.
* **Independente de interfața utilizator.** Interfata utilizator trrebuie sa poate suferii modificari fara a afecta intregul sistem. O interfata web ar trebui sa poata fii inlocuita de o consolsa fara a modifica regulile de bussines.
* **Independente de baza de date.** Acelasi lucru si pentru baza de date. Acesta sa se poata modifca de la SQLServer la Mongo fara a afecta regulile de bussines.
* **Independente de orice agenție externă.** De fapt, regulile de business nu știu absolut nimic despre lumea exterioară.

Principiul de baza al acestei arhitecturi este ca dependinta straturilor superioare de cele inferioare, fara a fii afectate cele inferioare. In plus, straturile inferioare nu stiu despe acesta depedinta. Acest lucru faciliteza schimbarea componentelor sistemului in tandem cu extinderea individuala a fiecaruie strat. Un alt avantaj este acela ca datorita acestei separari ni se permite o scriere mai usoara a testelor unitae, fapt ce ajuta la un nivel de calitate superior. Diagrama ce urmeaza este o încercare de integrare a tuturor acestor arhitecturi într-o singură idee practică.



**Figura 4.5.1 – The Clean Arhitecture**

Tot Uncle Bob ne prezinta nite principi ce s-au dovedit a fii eficient in crearea unui cod usor de citit si de mentinut. Pe langa principiile SOLID, exista principii legate de denumirea variabilelor, de design si de metoda, alaturi de scrierea de teste automate.

Conform conceputlui de „Code Smells” al lui Uncle Bob, un cod care „miroase”, in sensul rau al cuvantului, este un cod care este greu de modificat, genereaza erori la cele mai mici schimbari, nu se pot refolosi bucati din cod si care are o complaxitate inutila care face codul greu de inteles.

Jason Taylor este un programator carea popularizat aplicaricarea acestui concept de „Clean Code” in cadrul conferintei internationale NDC. Acesta a creat un teamplate folosind CLI .Ner Core folosing urmatoarea comanda:

|  |
| --- |
| dotnet new --install Clean.Arhitecture.Solution.Teamplate |

Apoi pentru a crea o noua solutie s-a folsoit de comanda:

|  |
| --- |
| dotnet new ca-sln |

Apoi fiecare nivel este reprezentat de un proiect in Visual Studio.

## Securitate

Stim cu ca securitatea este un aspect important cand vine vorba de aplicatii. Aplicatia InternHub are 3 tipuri de utilizatori cu drepturi de acces diferite. Astfel este necesar sa nu lasam oameni neautorizati sa efectueze modificari nedorita in cadrul aplicatiei.

Acest subcapitol are ca scop prezentarea securitatii request-urilor, deorece sercuritatea datelor este prezentata in capitolul 5, subcapitolul 5.4 Securitatea datelor.

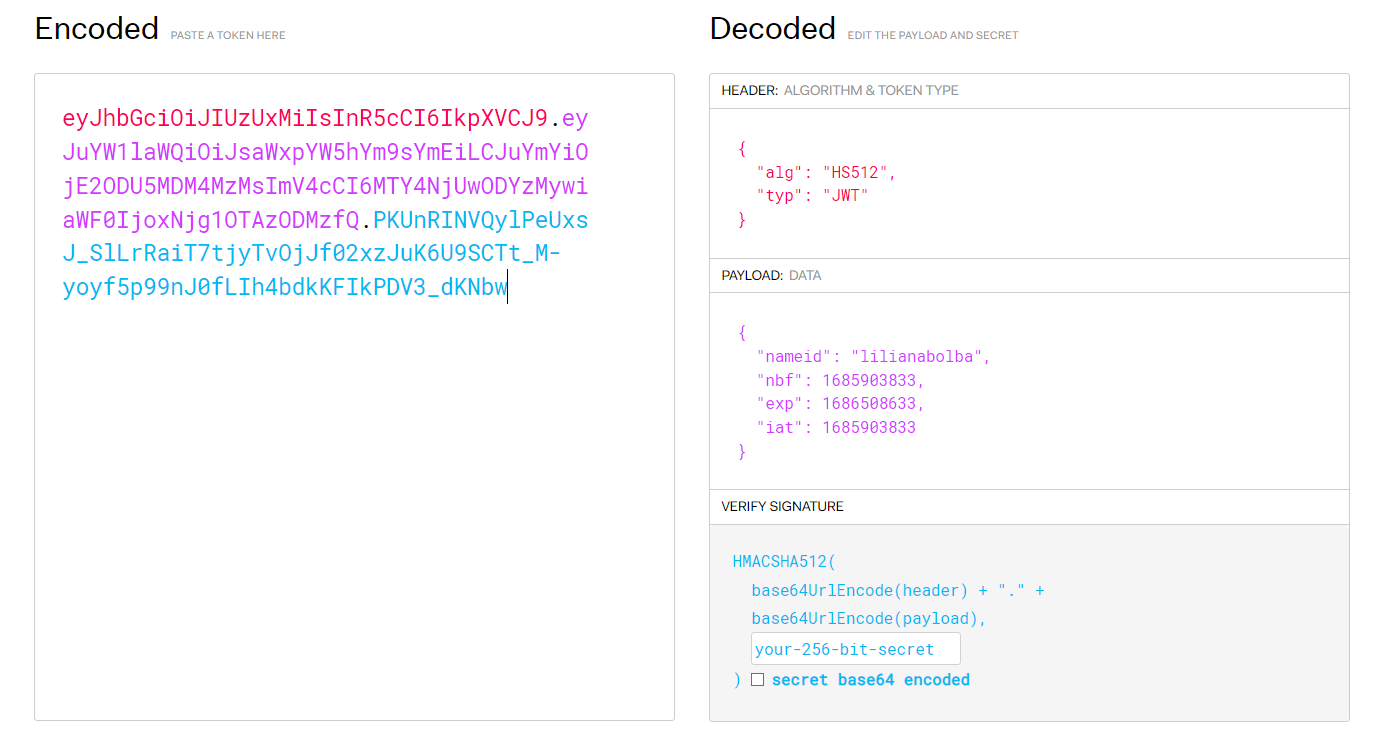
In cadrul aplicatia prezentata exista 2 tipuri de request-uri: cele care necesita autorizare, si cele care nu necesita autorizare. Cele care nu necesita autorizare sunt cele care ori nu au cum sa obtina token-ul de autorizare, precum request-ul de resetare al parolei sau requestu-uri ce au ca scop generarea unui astfel de token precum functia de log-in. In rest, toate celelate reques-uri necesita acest token de autorizare, ce este salvat in memoria cache, care se elibereaza odata cu delogarea utlizatorului, sau dupa ce expira.

In cadrul aplicatiei am ales sa folosesc autorizare sub cu token JWT(JSON Web Token), care este generat dupa username in momentul autentificarii utilizatorului.

* **JSON Web Token**

Tokenul JSON Web (JWT) este un standard propus pentru Internet care permite crearea de date cu semnătură opțională și/sau criptare opțională. Conținutul acestui token este reprezentat în format JSON și poate conține diverse afirmații (claims). Tokenurile sunt semnate fie cu o cheie secretă privată, fie cu o pereche de chei publice/privat.[-5]

In figura 4.6.1 avem exemplul unui token atat criptat cat si decriptat.



**Figura 4.6.1 – JWT criptat si decriptat**

Putem vedea ca token-ul este impartit in 3 parti. Prima parte reprezinta header-ul, unde avem informatii referitoare la agloritmul de critptare, HS512 in cazul de fata, si tipul sau care este JWT. Apoi este urmat de id care, pentru aplicatia InternHub este un string si anume username-ul persoanei logate, urmat de data la care a fost creat si data la care expire. Pentru InternHub toekn-ul a fost realizat cu o durata de viata de 7 zile, care se poate schimba foarte usor din clasa unde acesta a fost realizat. Ultima parte este semnatura token-ului care este relizata in HMACSHA512 si contine cheile criptate ale informatiilor de mai sus plus o cheie secreta de 521 biti.

Astfel prin intermediul acetui token doar utilizatori autorizati au dreptul la datele de pe server. Acest lucru este posibil prin intermediul unui Bearer token.

* **Bearer token**

Autentificarea Bearer este un mecanism de autentificare HTTP care utilizează bearer tokens pentru a oferi acces la resurse protejate. Tokenul bearer este generat de server și trebuie inclus în antetul de autorizare al cererilor clientului. Acesta este un șir de caractere criptice, de obicei generat de server în urma unei cereri de autentificare.[-6]

O astfel de autentificare se transmite prin intermendiul header-ului sub urmatoarea forma:

|  |
| --- |
| Authorization: Bearer <token> |

In exemplul de mai sus <token> reprezinta JWT generat la fiecare autenrificare a utilizatorului. Fiecare request contine in componenta sa acest header penru a putea avea acces la datele de de server.

Asta este pentru partea de client. Pentru partea de server, securizarea actiunilor echivalente reques-urilor se face prin internmeniul adnotarii [Authorize], ce se gaseste in AspNetCore si se acceaza incluzand urmatoare linie de cod:

|  |
| --- |
| using Microsoft.AspNetCore.Authorization; |

Astfel programatorul alege ce request-uri necesita autorizare si care nu. Lipsa acelei adnotari reprezinta faptul ca request-ul nu necesita autorizare. Un aspect impotant este ca putem avem o sigura actiune din tot controller ul care sa necesite sau sa nu necesita autorizare. Nu suntem obligati sa autorizaim tot controller-ul daca nu este cazul.

# Noțiuni teoretice și tehnologii folosite

Aplicația InternHub este o aplicație full-stack, din punct de vedere al dezvoltării, fiind formată din două parți: frontend și backend. Persistarea datelor a fost facută utilizând Microsoft SQ Server.

Pentru partea de backend a fost folosit framework-ul .NET cu C# ca si limbaj de programare. Motivarea pentru acesta alegere va fii în urmatoarele pagini, începând cu subcapitolul 5.1. pentru partea de backend.

Frontend-ul a fost realizat utilizând Angular și ca și limbaj de programare s-a folosit typescript. De asemenea motivarea pentru folosirea acestui framework o vom aborda la subcapitolul 5.2.

Pentru persistența datelor s-a folosit Microsoft SQ Server împreună cu Entity Framwrok pentru comunicarea backend-ului cu baza de date. Și pentru acesta avem motivarea alegerii la subcapitolul 5.3.

## .NET 7

* **.NET Framework**

.NET 7 este sucesorul lui .NET 6 și se focusează să fie unificat, modern, simple și de asemenea rapid. .NET 7 o să aibă suport tehnic pentru o perioadă de 18 luni, termen ce este standard in cadrul ecosistemului .NET. Lansat pe data de 8 noiembrie 2022, acesta va avea parte de suport tehnic până la data de 14 mai 2024.

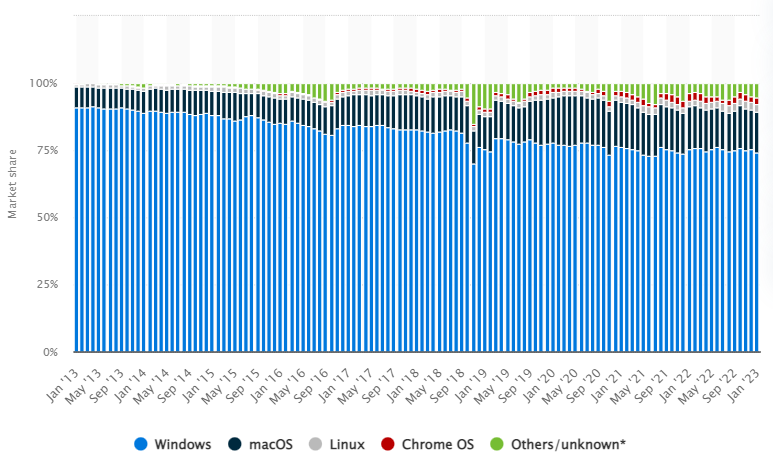
Printre noile caracteristici aduce de la .NET 6 la .NET 7 se enumeră performanța, serializarea JSON, matematica generică, expresii regulate, noi librării .NET, sdk .NET și de asemenea actulizări ce rezolva unele probleme ale subsistemelor ce folosesc ecosistemul .NET.

Pentru a întelege mai în detaliu ce înseamnă aceste noi caracteristici pentru .NET 7, o să le dezvoltăm puțin pe unele dintre cele prezentare mai sus:

* **Performața**: Performanța este una dintre caracteristicile cheie ale .NET 7, iar toate caracteristicile sale sunt concepute ținând cont de performanță. În plus, .NET 7 include îmbunătățiri destinate exclusiv performanței pentru următoarele îmbunătățiri: înlocuirea pe stivă (On-stack replacement), optimizare ghidată pe profil (Profile-guided optimization), imbunatățirea generării de cod pentru Arm64 și performanța îmbunatățită pentru timpul de rulare mono
* **Serializarea JSON:** .NET 7 include îmbunătățiri ale serializării System.Text.Json în următoarele domenii: serializarea polimorfică și suport pentru membrii necesari
* **Matematica generică**: .NET 7 și C# 11 includ inovații care vă permit să efectuați operații matematice în mod generic.Scrierea unei funții care are ca scop efectuarea unor operații matematice pentru două numere se simplifică, deoarece nu mai este nevoie supraîncărcarea metodei pentru fiecare tip dorit. Acum este posibilă scrierea unei metode generice în care parametrul tip este constrâns să fie un tip asemănător unui număr.
* **Expresii regulate**: Biblioteca de expresii regulate a .NET a înregistrat îmbunătățiri semnificative funcționale și de performanță în .NET 7 [1]

Ecosistemul .Net este un ecosistem dezvoltat si menținut de către Microsoft care rulează în principal pe Microsoft Windows. Dezvoltarea acestiu ecosistem a început la finalul anilor 1990, sub alt nume față de cel cunoscut astăzi. Inițial numit Next Generation Windows Services, era parte a strategiei .NET. Odată cu începutul anului 2000, primele versiuni beta ale lui .NET 1.0 au fost lansate, urmând ca in februarie 2002 să fie anunțată și totodată lansată prima versiune a .NET, și anume .NET 1.0. În prezent framework-ul se alfă la versiunea .NET 8, urmând ca în noiembrie 2023 să fie lansata versiunea .NET 8.

Cel mai mare avantaj al framework-ului .NET este că acceptă platforma Windows. Aproape toată lumea lucrează cu mașini Windows. [2]



**Figura 5.1.1 - Cota de piață globală deținută de sistemele de operare**

Conform figurii 5.1.1 se poate observa că sistemul de operare preferat în întreaga lume este windows cu un procetaj constant mai mare de 74% în ultimii 10 ani. [3]

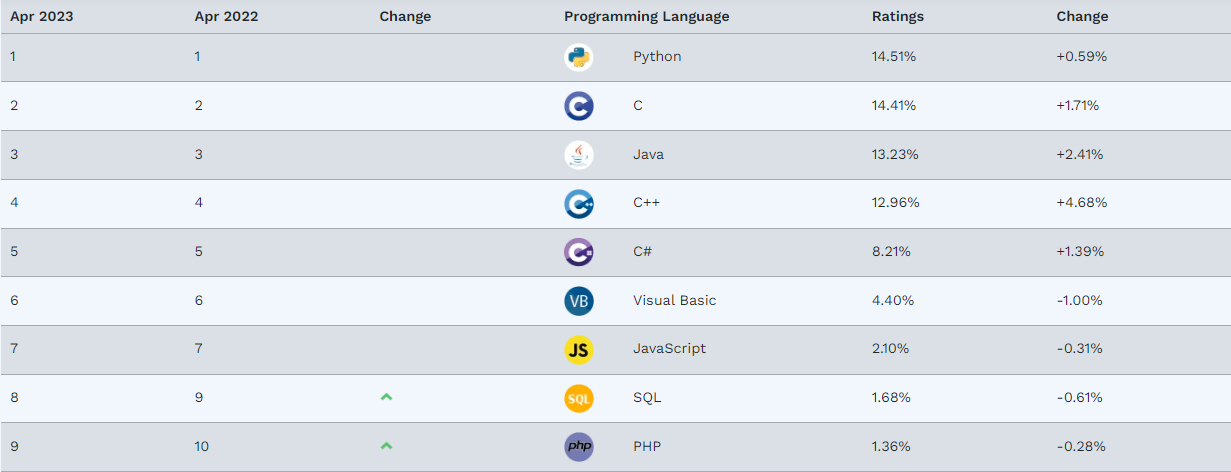
Un alt argument în favoarea ecosistemului .NET este constanța lansării de noi versiuni ale acestui ecosistem, și totodată menținerea suportului pentru versiunile lansate anterior versiunii curente, pentru a face mai ușoară trecerea de la o versiune la alta. La momentul actual acest timp de suport tehnic este de 6 luni după lansarea ultimei versiuni.

Pentru lucrarea în cauză am ales să lucrez cu .NET 7, deoarece este cea mai nouă și care are avantaje considerabile în fața altor opțiuni. Un alt motiv pentru .NET 7 este și incheierea suportului tehnic pentru .NET 3.1 în decembrie 2022, la puțin timp după lansarea lui .NET 7, iar tendința generală în cadrul companiilor este de a migra codul de la .NET 3.1 la .NET 6 sau .NET 7. Motivul pentru .NET 6 este că, asemenea lui .NET 3.1 este o versiune LTS (Long Term Suport), ceea ce îi oferă suport până în noiembrie 2024. [4]

Motivul pentru alegerea verisunii curent in detrimentul unei versunii LTS este, pe lângă cele prezentate la începutul acestui subcapitol este faptul că am dorit sa experimentez ultima versiune.

* **C#**

C# este un limbaj de programare care acceptă mai multe paradigme.Inițial lansat în 2000, acesta a ajuns deja la versiunea 11 lansată odată cu .NET 7 în noiembrie 2022. Asemănător lui .NET, C# primește contant actulizări și lansări de noi versiuni, astfel fiind o alegere optimă pentru partea de backend și totodată pentru frontend. [5]



**Figura 5.1.2 – TIOBE Programming Community Index**

Conform figurii 5.1.2 se poate observa că deși C# nu este cel mai popular limbaj de programare este în continuă creștere fiin al 5-lea cel mai utilizat limbaj de programare. În iunie 2001, la puțin timp după lansare avea un raiting de 0,84%, iar este la 8,21% cu o creștere de 1.39% în ultimul an. [6]

Legat de API, pentru acesta s-a folosit ASP.NET Core Web API. Conform stasticilor făcute de cei la stack overflow din 2022 legată de framewrok-uri de dezvoltare pentru aplicatii web ASP.NET Core și ASP.NET sunt destul de populare fiind imediat după Angular și Vue.js în rândul tuturor celor intervievați, în număr de 58 743, așa cum se poate observa în figura 5.1.3. În rândul programatoril experimentați ASP.NET Core urcă o poziție deasupra lui Vue.js, iar ASP.NET rămâne pe acceți pozitie. Numărul programatoril experimentați intervievași de către cei de la Stackoverflow este puțin mai mic, dar rămâne un numâr consderabil de persoane intervievate, în număr de 45 297 , așa cum se poate observa în figura 5.1.4. În schimb cei ce sunt la început de drum aleg mai puțin ASP.NET și ASP.NET Core. Numărul celor intervievați ce se află la început de carieră este de 4932, așa cum se poate observa în figura 5.1.5. Cu toate acestea primele 2 opțini sunt la fel, fie ca vorbim de utilizator cu experința fie că vorbim de cei la început de drum. [7]

|  |  |
| --- | --- |
| **Figura 5.1.3 – Toate răpunsurile la sondaj** | **Figura 5.1.4 – Răspunsurile programatorilor experimentați la sondaj** |
| **Figura 5.1.5 - Răspunsurile programatorilor aflați la început de carieră la sondaj** |  |

* **SignalR**

Sistemul de chat are nevoie de relationare in timp real in cadrul aplicatiei a fost utilizat SignalR, care este este o bibliotecă pentru dezvoltatorii ASP.NET care simplifică procesul de adăugare a funcționalității web în timp real la aplicații. Funcționalitatea web în timp real implică abilitatea unui server de a transmite instantaneu conținut către clienții conectați imediat cum devine acesta accesibil, în loc să aștepte o solicitare explicită pentru date noi de la un client.

Acesta bilbioteca poate fi folosita pentru adaugarea oricarui tip de fnctionalitate web „in timp real”, chiar daca exepmplul cel mai intalnit este chat-ul. Sistemul de notificare este un astfel de exemplu care functioneaza in timp real care se utilizeaza de SignalR. SignalR permite, de asemenea, tipuri complet noi de aplicații web care necesită actualizări de înaltă frecvență de la server, de exemplu, jocuri în timp real.

Un alt plus important al lui SignalR este faptul ca poate gestionarea conexiuni (de exemplu, evenimente de conectare și deconectare) și gruparea conexiunilor. Acest lucru este vital deoarece sunt momente in care nu toti utilizatorii trebuie sa fie afectati de schimbarea din aplicatie, ci doar un grup prestabilit de utiliztori.

In figura 5.1.6 se poate oberva o reprezentare grafica a procesului pe care il face SignalR cand vine vorba de actiuni in timp real.[8]



**Figura 5.1.6 Modul de functionare a lui SignalR**

SignalR este o bibliotecă care gestionează automat conexiunile și permite transmiterea de mesaje către toți clienții conectați simultan, similar unui chat. Poate și trimite mesaje clienților specifici, menținând o conexiune persistentă cu aceștia. Suportă funcționalitatea "server push", care permite codului server să apeleze codul client în browser prin intermediul Apelurilor de Procedură la Distanță (RPC). În plus, SignalR poate scala la mii de clienți folosind provideri de scalare, atât încorporați cât și externi.

Stim cu toti ca WebSocket-u este folosit pentru comunicare pe internet care permite schimbul bidirecțional de date între un client și un server. WebSocket este un protocol de comunicare care diferă de HTTP/HTTPS prin oferirea unui canal de comunicare bidirecțională și continuă. Spre deosebire de modelul solicitare-răspuns al HTTP/HTTPS, WebSocket menține o conexiune deschisă, ceea ce permite serverului și clientului să transmită date reciproc oricând, până când conexiunea este închisă. Aceasta este o tehnologie extrem de utilă în aplicații care necesită actualizări în timp real, cum ar fi chat-uri, jocuri online, sau alte aplicații interactiv.

SignalR folosește transportul WebSocket când este disponibil, dar se poate adapta la tehnologii mai vechi dacă este necesar. Avantajul folosirii SignalR constă în faptul că multe din funcționalitățile suplimentare necesare sunt deja implementate. Prin utilizarea SignalR, aplicația poate beneficia de WebSocket fără a necesita cod separat pentru clienții cu tehnologie mai veche. În plus, SignalR se ocupă de actualizările la WebSocket, asigurând o interfață consistentă pentru aplicație, indiferent de versiunea WebSocket.

## Angular

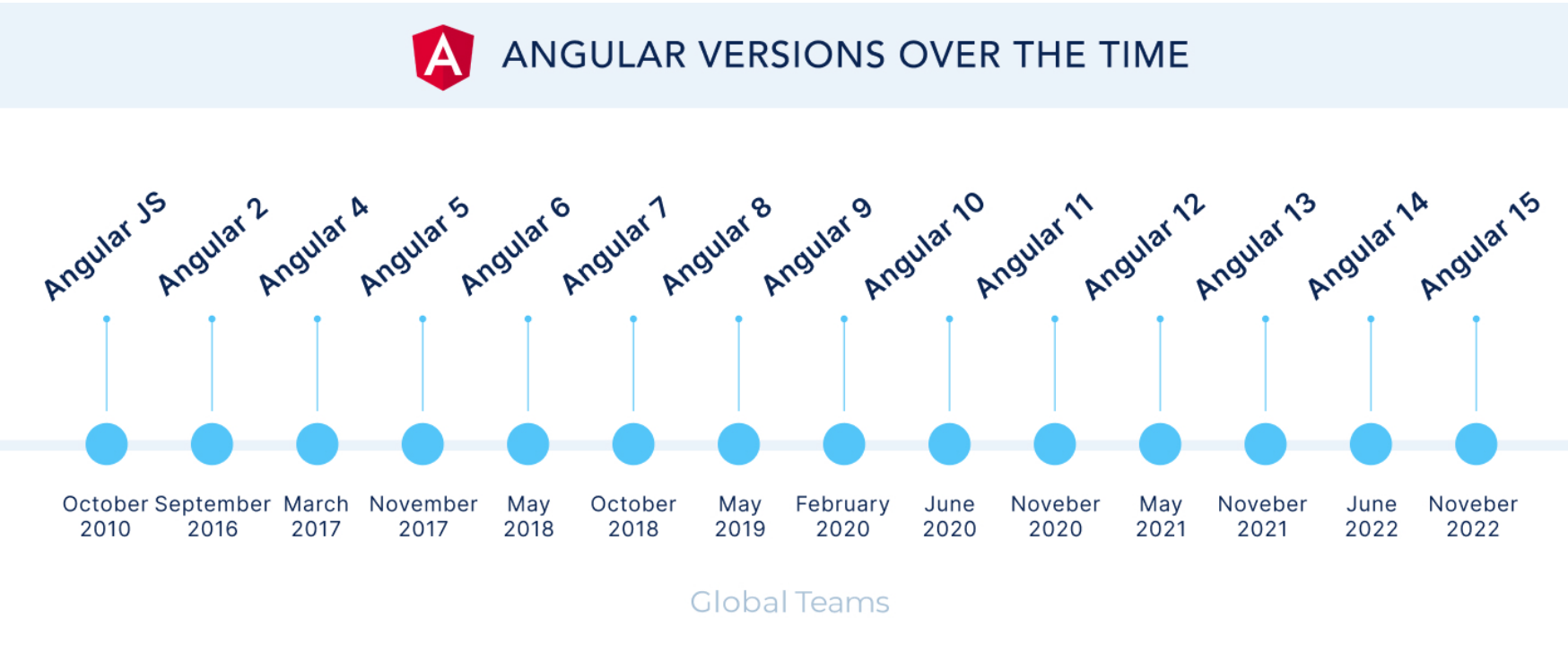
* **Angular**

Angular este un cadru (framework) de dezvoltare pentru construirea de aplicații web. A fost dezvoltat și este întreținut de Google. Dezvoltată în 2010, Angular a devenit o platformă de dezvoltare software extrem de populară și bine primită fiind cunoscut pentru abilitățile sale puternice de single-page application (SPA), adică aplicații care se încarcă o singură dată și apoi se actualizează dinamic pe măsură ce utilizatorul interacționează cu aplicația.

In favoarea lui Angular există multe motive care îi conferă superioritate în fața altor framework-uri disponibile pe piață cum ar fii React.js sau Vue.js. Codul său, ce poate fii JavaScript sau TypeScript, și caracteristicile structurale îl fac intuitiv, simplu de înțeles și ideal pentru dezvoltatorii front-end, avand in component si elemente de HTML/CSS ce utlizeaza aproape la fel ca si cand ar fii indepente. Prin utilizarea Angular, se pot creea soluții scalabile care să răspundă eficient nevoilor clienților tăi.

Un alt plus al lui Angular este lansarea de constanta de versiuni si de update-uri. Lansat in anul 2010, fiind la prima versiune a sa, a ajuns in luna mai a anului 2023 la a 16-a varianta. Dupa prima versiune a lui Angular a durat ceva pana la lasarea celei de-a doua versiuni, ce a aparut 6 ani mai tarziu in data de 14 septembrie 2016. După Angular 2, cadrul de lucru a adoptat versionarea semantică, ceea ce înseamnă că noile lansări:[9]

* + Apar regulat, la fiecare șase luni, cum se poate oberva in figura 5.2.1
  + Prezintă compatibilitate inversă - adică, o nouă versiune poate utiliza fișiere și date create de versiunile anterioare;
  + Includ șase luni de suport activ cu actualizări periodice și 12 luni de suport pe termen lung, în timpul cărora sunt lansate doar corecții importante.



**Figura 5.2.1 Versiunile de Angular de-a lungul timpului**

Astfel in luna mai a anului 2023 Angular a lasat versiunea 16, si planuieste lansarea versiunii 17 in luni noiembrie a aceluiasi an. Acest lucru inseamna ca Angular incerca sa tina pasul cu technologia, facilitatand utilizarea versiunilor pe perioade mai lungi de 6 luni.

Pe langa toate acestea Angular are si alte avantaje precum:[9]

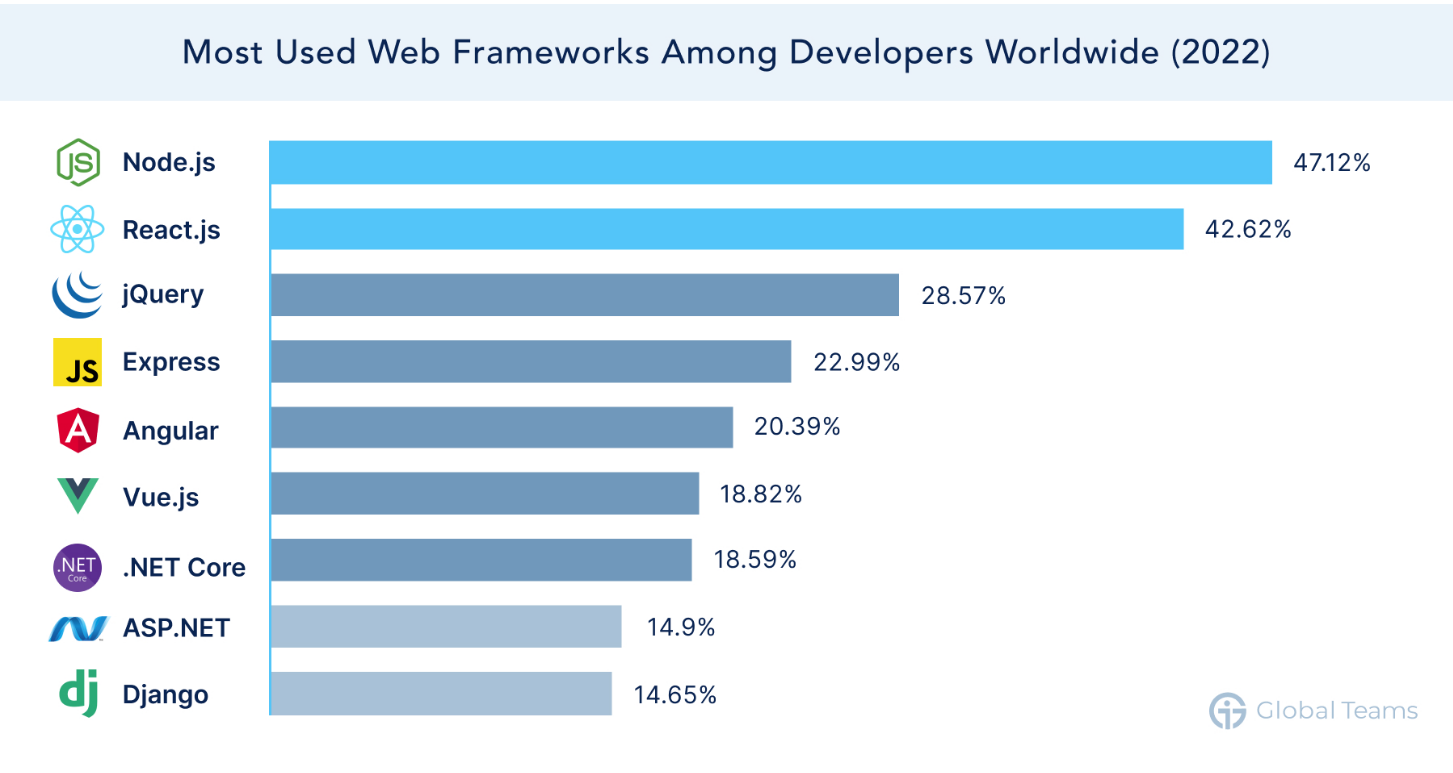
* + **Cross-Platform:** Unul dintre punctele tari ale lui Angular este facptul ca este o plaforma universala ce permite crearea atat de aplicatii web, cat si pentru dispozitivele mobile. Acesta are multe avantaje in comparatie cu celelate plaforma numite mai sus si este unul dintre cele mai populare framework-uri.
  + **Angular Material:** Angular a creat o bilbioteca open-source de componente UI care fac viata programatorilor mai usoara. Aceasta biblioteca faciliteaza crearea de interfete prietenoase si functionala in raport cu utilizatorul. Astfel se fgaciliteaza crearea de design-uri mobile atractive folosind componente reutilizabile ce ofera un aspect unitar în toate aplicațiile. Angular material faciliteaza constrcutia de aplicatii resposive prin utilizarea de componentele și widget-urile sale preconstruite. Acestea au o caracterista importanta si anume faptul ca sunt personalizabile, iar programatorul isi poate design-ul pentru aplicatia la care lucreaza.
  + **High Speed and Performance:** Aplicatiile ce utilizeaza Angular se incarca instantaneu, fiind aproape imposibil de egalat. Viteza de încărcare a platformei asigură că aplicatiile la care se lucreza sunt mereu active datorită caracteristicii lor de încărcare instantanee.. O caracteristică remarcabilă este că codul poate fi redat în CSS și HTML, iar aplicația este accesibilă de pe orice platformă, inclusiv .Net, NodeJs și PHP.
  + **Command Line Interface:** Interfața de Linie de Comandă (CLI) a Angular respectă practicile standard de dezvoltare front-end din industrie și include capacități unice, cum ar fi rutarea și suportul SCSS. Interfața CLI standard a Angular, cum ar fi ng-add și ng-new, facilitează pentru dezvoltatori găsirea caracteristicilor predefinite.

Figura 5.2.2 de mai jos prezinta toate celelalte beneficii ale ultilizarii lui Angular, sus fiind selectate cele mai important despre care s-a si discutat putin.[9]



**Figura 5.2.2 Beneficiile lui Angular**

Conform statisticilor din anul 2022 de pe site-ul StackOverflow, Angular devine din ce in ce mai apreciat de catre dezolvtatorii profesionisti, cu 20.39% utilizare in randul acestora. Acest lucru se poate obesrva in figura 5.2.3. A existat o tendință constantă pe piață pentru cadrul de lucru front-end Angular în ultimii doi ani.[9]



**Figura 5.2.3 Cele mai utilizate Framework-uri web in anul 2022**

Companii mari precum Microsoft, PayPal si Forbes au trecut pe Angular datorita numeroaselor avantaje pe care Angular le aduce in discutie. Pentru Gmail trecea la Anguar a insemnatoferirea unei simplitati greu de egalat cand vine vorba de inbox-urile personale. Evident ca si pentru PayPal avatajele nu au intarziat sa apara, oferind o constructie a paginilor web structurat, eficienta și care prezinta o evolutia a gestionarii datelor tranzactionate.

Asadar, avand in vedere toate cele mai sus Angular este un framework ce devine din ce in ce mai utilizat in cadrul companiilor si care ofera versiuni noi constant ducand la evolutia treptata a framework-ului.

* **Typescript**

TypeScript este un limbaj de programare open-source dezvoltat și menținut de Microsoft. Este un superset strict al JavaScript și adaugă tipizare statică și obiecte orientate pe obiecte la JavaScript. Fiind un limbaj de programare puternic tipizat, TypeScript permite o depanare mai ușoară (la momentul compilării), ceea ce este o abordare mai eficientă pentru codificarea aplicațiilor complexe.

Complexitatea proiectelor bazate pe JavaScript creste in mod exponenetial. Inital pe partea ce cleint era folosit exclusiv JavaScript ca si limbaj de programare. Insa, cu trecea timpului s-a obersavat ca JavScript se poate folosi si pentru partea de server.

Cu toate ca JavaScript are numeroase avantaje, acesta poate devenii usor confuz si implicat pe partea de server mai ales în cazul aplicațiilor de mari dimensiuni. Un alt minus al lui JavaScript este faptul ca a complicatpocesul de intretinere al aplicatiilor mai mari si mai complexe.

Perfecționările aduse browserelor și compatibilitatea dintre acestea necesită, de asemenea, modificări în ceea ce privește JavaScript-ul de bază, dar abordarea unui JavaScript pentru viitor nu este fezabilă. Prin urmare, TypeScript a fost introdus pentru a răspunde acestor nevoi specifice ale aplicației.[10]

Dar cu toate acestea, este oare TypeScript o alegere mai buna cand vine vorba de alegerea limbajului de programare pentru partea de front-end? Din felul in care l-am descris ar parea o alegere mai inspirata, fiind considerat o versiune mai buna. Desi am putea crede ca TypeScript va inlocuii JavaScript in viitorul apropiat. Ei bine, acest lucru nu se va intanpla deoacere JavaScript isi va avea locul.

Un factor care nu determina trecerea in viitorul apropiat de la JavaScript la TypeScript este complexitatea, care trebuie luata in considerare cand vine vorba de o asemenea schimbare.

Cand vine vorba de aplicatii mai simple, alegerea este la fel de simpla, deoarece JavaScript functioneaza foarte usor pe toate platformele. Pe cand compilarea codului TypeScript necesite resurse ale CPU-ul considerabil mai mari decat ar neceista JavaScript. Ocazional, poate fi nepotrivit sau excesiv să se folosească TypeScript pentru un anumit proiect.

TypeScript facilitează procesul de refacere a codului și pune în evidență tipurile într-un mod mai direct, ceea ce ajută dezvoltatorul să înțeleagă modul în care diferitele componente se conectează. Datorită capacității sale de a depana în timpul compilării, acesta constituie un beneficiu remarcabil pentru entitățile care administrează aplicații extinse și complexe.

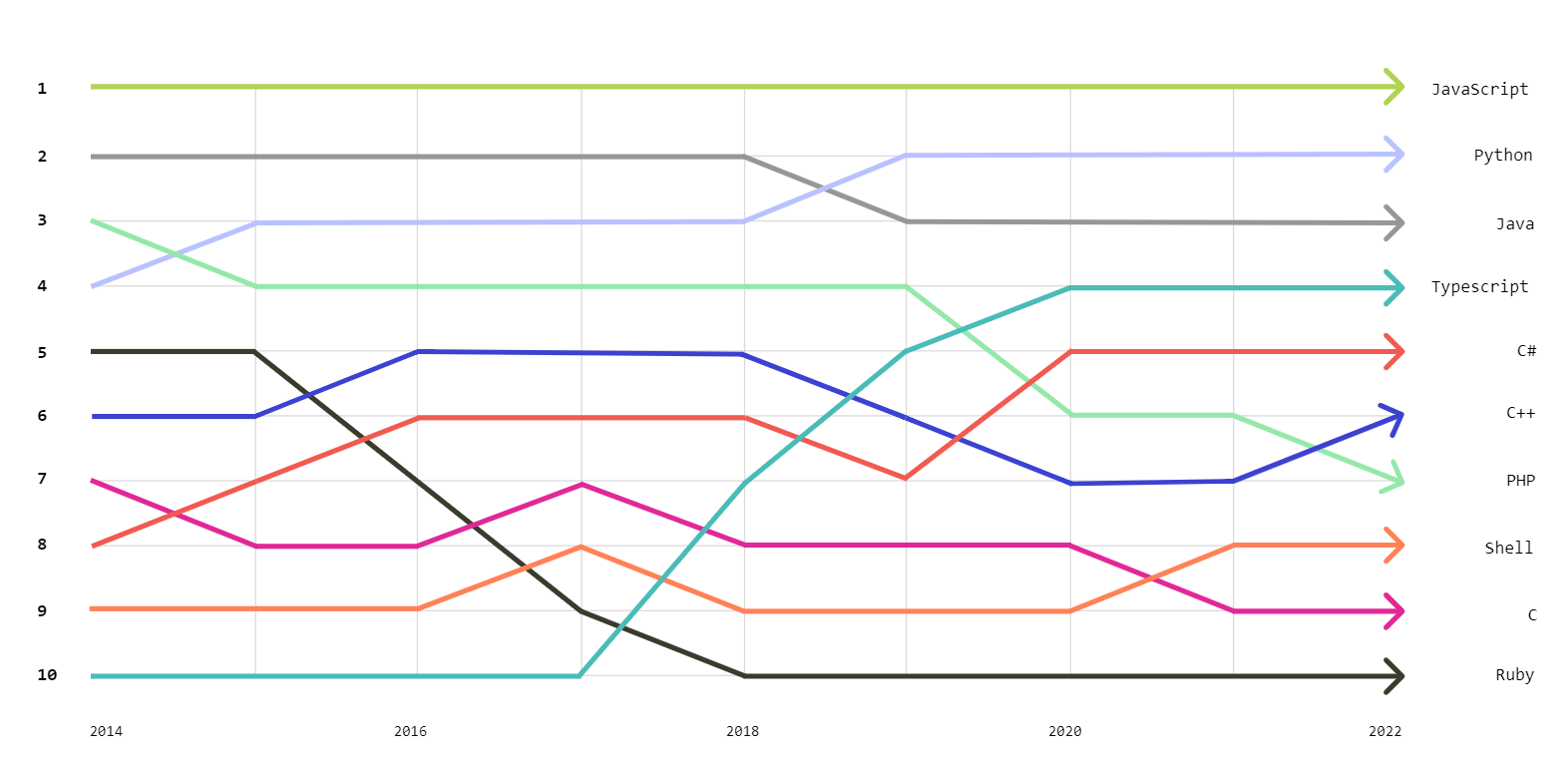
Configurarea TypeScript pentru orice proiect este simplă, iar unele framework-uri folosesc TypeScript în mod implicit. Un exemplu concret de framewoek ce foloseste ca limbaj implicit TypeScript-ul este Angular. Deci TypeScript are un avantaj considerabil in fata lui JavaScript la acest capitol. [10]

Diferența principală între JavaScript și TypeScript constă în faptul că JavaScript nu dispune de un sistem de tipuri. În JavaScript, variabilele pot schimba forma în mod aleator, în timp ce TypeScript în modul strict interzice acest lucru. Acest lucru face TypeScript mai ușor de gestionat și întreținut, în special cu o bază de cod mare. Acest lucru poate avea consecinte destul de mari o data cu implementarea de noi feature-uri care necesita ca datele sa fie de un anumit tip.

Următoarele sunt caracteristicile TypeScript:

* + TypeScript este un limbaj de programare open-source creat de Microsoft care adaugă tipuri pentru JavaScript. Este mai coplex in comparatie cu JavaScript care este un limbaj de programare la nivel înalt pentru scripting.
  + TypeScript este un limbaj cu un sistem de tipuri static. Adică, înainte de a utiliza variabilele și obiectele, trebuie să specificați tipul lor. Iar JavaScript nunecesita specificat tipul de date, fapt ce poate duce la anumite bug-uri sau chiar probleme ale datelor sensibile.
  + TypeScript extinde JavaScript cu tipuri opționale, clase și module.
  + TypeScript include instrumente pentru dezvoltarea de aplicații JavaScript la scară largă care nu depind de browser, gazdă sau sistem de operare.
  + TypeScript este compilat în JavaScript lizibil, conform standardelor.
  + Deoarece TypeScript extinde sintaxa JavaScript, orice program existent JavaScript va rula în TypeScript fără modificări.
  + Puteți instala compilatorul TypeScript pentru linia de comandă ca un pachet Node.js.

In cadrul figurii 5.2.4 se poate oberva o tendinta de trcere pe TypeScript de la lasarea lui pana in anul 2022, urcand treptat de pe positia a 10 in anul 2014 pe pozitia 7 in 2018, apoi pozitia 5 in 2019, iar din 2020 s-a mentinut pe pozitia a-4-a imediat dupa Java. Tot in figura 5.2.4 putem observa consistenta a positie de lider a lui JavaScript.[11] Tocmai de accea, TypeScript nu v-a inocuii complet JavaScript ci se va alege cel mai bun pentru un anumit proiecte luand in considerare cele prezentate mai sus.



**Figura 5.2.4 The top programming languages**

Avand in vedere cele prezentate pentru proiectul InternHub am decis sa utilizez TypeScript in detrimentrul lui JavaScript, motivul principal fiind complexitatea proiectului care poate fii maniplata cu o mai mare usurinta de catre TypeScript. De asemenea alt motiv este cresterea popularitatii sale in randul programatorilor mai experimentati.

## Microsoft SQL Server

Microsoft SQL Server este un sistem de gestionare de baze de date relaționale (RDBMS) produs de compania americană Microsoft Corp, folosind SQL ca limbaj de interogare alarturi de extensia T-SQL. Acest sistem de gestionare a bazelor de date este recunoscut în special pentru abilitatea sa de a gestiona baze de date de dimensiuni mari, fiind considerat potrivit pentru aplicatii mai complexe ce necesita o stocare mai mare de date. Cu toate acesta utlimele versiuni ale lui Microsft SQL Server beneficieaza de licenta variate, lucru ce include cu varianta gratuite, fapt ce faciliteaza lucru si cu baze de date mai mici.[12]

Un alt plus notabil pentru Microsoft SQL Server este suportul extins, ce ofera o solutie fiabila in vederea gestionarii eficienta a datelor in diverse medii si aplicatii. Astfel Micrsoft SQL Server este considera unul dintre cele mai buna sisteme de gestionare a basezlor de date avand costant lansari noi de varinate ce beneficieaz de suport pe o perioada destul de mare.[12]

Microsoft a lansat recent SQL Server 2022, ce se bazeaza pe versiunile anterioare cu scopul de a creea o platforma cu numeroase optiuni in ceea ce priveste limbajele de dezvoltare, tipurile de date, sisteme de oprare si mediile de lucru pe servere atat fizice cat si in cloud. [13]

Printre numeroasele beneficii se numara[14]:

* + **Continuitatea afacerii prin intermediul Azure:** Microsoft SQL serve ofera o gestionare eficineta pe cloudd, ceea ce asigura protectia datelor in fata oricaror dezastre cepot aparea. Astfel datele utilizatorilor nu vor fii pierdute in nici un caz.
  + **Analize fără întreruperi asupra datelor operaționale de pe serverele fizice:** Microsoft SQL Sever ofera posibilitatea generarii in timp real de statistici pentru toti utilizatorii prin intermediul lui Spark.
  + **Vizibilitate asupra întregului set de date:** Ajuta la gestionarea intregului volum de date prin prevenirea depasiri silozurilor de date prin intermediul lui Microsft PurView.
  + **Cea mai sigura baza de date din utlimii 10 ani:** Microsft SQL Server este evaluata ca fiind cel mai putin vulnerabil sistem de gerstiune al bazelor de date in urlimii 10 ani, folosind un mediu stabli pentru protectia datelor impotriva falsificarii.
  + **Performanță și disponibilitate lider în industrie:** Microsft SQL ofera performanță pentru interogări mai rapide, nefiind necesara modificarea codului, ceea ce dupe la aplicatii mai eficiente.

Avand in vedere cele prezentate mai sus este destul de clar de ce am ales sa folosesc Microsoft SQL Serve in detrimentul alter sisteme de gestionare al bazelor de date precum PostgreSQL. Evident ambele au avantaje si dezavantaje, dar pentru InternHub Microsoft SQL Server s-a mulat mai bine pe nevoile proiectului. Un astept important este faptul are un punct comun cu .NET, find creat de acceasi copmanie mama, astfel implementarea si comunicare este mult mai usoara.

Intr-un studiu facut de cei de la StackOverflow, ce are 63,327 de raspunsuri de la programatori cu diferite nivele de experinta reiese ca Microsoft SQL Server nu este la fel de preferat ca si PostgreSQL cand vine vorba de alegerea sistemul de gestiune al datelor, fiind abia pe pozitia 5 cu 26.78% in cadrul tuturor raspunsurilor. La capitolul programatori experimentati se schimba putin pozitia lui Microsft SQL Server, aceesta fiind pe pozitia a 4 cu 28.77%. Daca ne raportam la cei la invata sa proogrameze putem oberva ca mai bine de jumtate din cei care au raspuns la acest studiu aleg MYSql cu un procent de 58.4%. Aici Microsft SQL Server revine pe pozitia a 5-a cu 16.34%, iar PostgreSQL este o pozitie mai sus cu 25.54%.[15]

Aceste rezultate se pot observa in figurile de mai jos, si anume 5.3.1, 5.3.2 si 5.3.3, unde sunt toate rapsunsurile la studiu realizat de cei de la StackOverflow in anul 2022. In figura 5.3.1 se pot oberva toate raspunsurile la sondaj, celelalte doua fiind impartite pe programatori experimentati, figura 5.3.2, si cei care invata sa programeze figura 5.3.3.

|  |  |
| --- | --- |
| **Figura 5.3.1** - **Toate răpunsurile la sondaj** | **Figura 5.3.2** - **Răspunsurile programatorilor experimentați la sondaj** |
| **Figura 5.3.3 -** **Răspunsurile programatorilor aflați la început de carieră la sondaj** |  |

Pentru dezvoltatorii profesioniști, PostgreSQL abia a preluat locul întâi de la MySQL. Dezvoltatorii profesioniști sunt mai predispuși decât cei care învață să codeze să utilizeze Redis, PostgreSQL, Microsoft SQL Server și Elasticsearch. MongoDB este utilizat de un procent similar atât de către dezvoltatorii profesioniști, cât și de cei care învață să codeze, fiind a doua cea mai populară bază de date pentru cei care învață să codeze (după MySQL). Acest lucru are sens deoarece suportă un număr mare de limbaje și platforme de dezvoltare a aplicațiilor.[15]

Cu toate ca Microsoft SQL nu este in top 3 cele mai utilizate sisteme de gestiune, are multe avantaje care ajuta la dezvoltaea aplicatiei InternHub, avand grija de cea mai imporatnat parte a oricarei aplicatii, si anume securitatea datelor. In plus, asa cum am mai spus, avand producator comun cu framework-ul .NET, comunicare intre API si baza de date este mult mai usor de realizat si tot odata sigura.

Asadar, avand in vedere cele prezentate mai sus, Microsoft SQL Server reprezinta opitunea cea mai buna cand vine vorba de baze date, avand avantaje considerabile in fata celorlalte optiuni pentru proiectul de fata.

## Securitatea datelor

Stim cu toti ca securitatea datelor este un subiect important in cadrul oricarei aplicatii. In aplicatia prezentate avem si date sensibile ce trebuie stiute doar de utilizatorul in cazul, cum ar fii parola continui. Am ales un algoritm care cripteaza aceste valori dupa o cheie prestabilita in proiect. Avantajul algoritmului de criptare utlizat in aplicatie este HMACSHA512 care are ca si avantaj principal ca este destul de greu de decriptat. In cazul acesta accesul in contul personal se face prin generarea hash-ului si si al salt-ului cu valoarea transmisa de user. Daca aceste doua valori corespund cu cele din tabela atunci user-ul este autorizat, iar daca nu atunci nu i se permite accesul.

HMACSHA512 este un algoritm de hash bazat pe chei, conceput folosind funcția de hash SHA-512 și aplicat ca un Cod de Autentificare a Mesajului bazat pe Hash (HMAC). Procedura HMAC combină o cheie secretă cu datele mesajului și apoi calculează valoarea hash a combinației. Valoarea hash obținută este amestecată din nou cu cheia secretă și este supusă unui al doilea proces de hash. Hash-ul rezultat are o lungime de 512 biți.

HMAC poate fi utilizat pentru a verifica dacă un mesaj trimis prin intermediul unui canal nesigur a fost compromis, atât timp cât expeditorul și destinatarul dețin o cheie secretă comună. Expeditorul calculează valoarea hash pentru datele originale și trimite ambele, datele originale și valoarea hash, ca un singur mesaj. Destinatarul recalculează valoarea hash pe baza mesajului primit și verifică dacă HMAC-ul calculat corespunde cu HMAC-ul transmis.

Dacă valorile hash originale și cele calculate se potrivesc, mesajul este autentificat. Dacă nu se potrivesc, atunci fie datele, fie valoarea hash au fost modificate. HMAC oferă protecție împotriva modificărilor, deoarece pentru a schimba mesajul și a reproduce valoarea hash corectă este necesară cunoașterea cheii secrete.

HMACSHA512 permite utilizarea de chei de orice dimensiune și produce o secvență de hash cu o lungime de 512 biți.[16]

# Prezentarea aplicației

## Introducere

## Structura aplicației

## Persistența datelor

# Concluzii și perspective de viitor

## Concluzii

Avand in vedere faptul ca aplicatia InternHub nu este o aplicatie deloc simpla evident ca au aparut provari. Cea mai mare fiind gestiuonare pozelor, care sunt cautari lungi, am ajuns la conluzia ca este mult mai eficinta utilizarea unui serviciu de stocare in cloud al acestor informatii. O alta provocare a fost stocarea incercarilor de rezolvare ale challenge-urilor, deoarece acestea au fost stocare in baza de date ca fisiere .zip sau .rar, si trebuia aduse cu scopul descarii lor pe computer-ul personal.

Fiind primul proiect de tip full-stack, si a doua oara cand creez o aplicatie de front-end de la 0, prima fiind aplicatia de demo de la un curs de la Udemy, pot spune ca partea de front-end a reprezentat o provocare si ea. Chiar daca avem cunostinte de HTML/CSS si JavaScript a fost o provocare sa finalizez acesta aplicatie pe parte de front-end. In schimb pe partea de back-end a fost mult mai usor, doarece am mai creat API-uri, iar aici pot spune ca singura provocare a fost aceea legata de implementarea functionalitatii real-time, in special pentru sistemul de chat. Creea unui sistem de chat functional nu este in sine o actiune foarte complexa, dar partea pe cat de complexa pe atat de interesanta este utilizarea bibliotecii SignalR pentru realizarea unu sistem de chat real-time. Evident provocarea fost pentu toata implementarea real-time, dar totul a inceput de la sistemul de chat.

Astfel aplicatia a ajuns la forma ei actuala, avand functionalitati care sa faciliteze evolutia companiei in care va fii untilizata, respectand atat principiile de REST API cat si cele ale unei „Arhitecturi curate”(Clean Arhitecture).

## Posibilități de extindere

Dezvoltarea unei aplicatii nu se termina niciodata. Fie ca vorbim de trecerea la o noua versiune fie ca vorbim de un nou feature, o aplicatie este in continua dezvltare si imbunatatire. Aplicatie InternHub nu face exceptie acestei reguli. Putem lua in considerare urmatoarele imbunatatiri:

* Trecerea la Angular 16, care a aparut in timpul aplicatiei.
* Trecerea la .NET 8 cand o sa fie lansat, adica in noiembrie 2023.
* Crearea de teste unitare pentru cresterea calitatii si stabilitatii.
* Implementarea pattern-ului repository in tandem cu un code clean-up pe partea de client.
* Implementarea standardului Microsoft 0Data pentru a imbunatatii constrangerile REST.

Acesta sunt imbunatari ce pot fii aduse pe viitoa aplicatiei. Acum sa vorbim putin de cum mai poate fii dezolvtata aplicatie:

* Implementarea unui sistem de tip two-way-authentification.
* Un sistem de notificari pentru fiecare simbare relevanta in cadrul aplicatiei.
* Un sistem de logging pentru o urmarire mai buna a codului si pentru o identificare mai usorasa a erorilor.
* Implementarea unui sistem de celebrare a zilelor de nastere pentru utilizatorii aplicatiei
* Introducerea unui modul de rapoarte pentru a putea vedea mai bine evolutie fiecariu intern

Acestea sunt cateva idei de imbunatarire a aplicatiei. Pe parcursul dezvoltarii ei vor mai aparea cu siguranta. Cel putin trecerea la o noua versiune ori de cate ori este benifica va exista mereu.

# Bibligrafie

[-6] Bearer Authentication, <https://swagger.io/docs/specification/authentication/bearer-authentication/>

[-5] JSON Web Token, <https://en.wikipedia.org/wiki/JSON_Web_Token>

[-4] Robert C.Martin, Clean Arhitecture: A Craftsman’s Guide to Software Stucture and design, Editura Pearson, USA, 2007

[-3] Repository Design Pattern in C#, <https://dotnettutorials.net/lesson/repository-design-pattern-csharp/>

[-2] The Clean Architecture, <https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html>

[-1] O introducere în arhitectura REST, <https://www.opti.ro/ro/post/about-rest-apis>

[0] RESTful Web Services, <https://www.oracle.com/technical-resources/articles/javase/restful.html>

[1] What's new in .NET 7, <https://github.com/dotnet/docs/blob/main/docs/core/whats-new/dotnet-7.md>

[2] .NET, <https://en.wikipedia.org/wiki/.NET>

[3] Global market share held by operating systems for desktop PCs, from January 2013 to January 2023, <https://www.statista.com/statistics/218089/global-market-share-of-windows-7/>

[4] .NET and .NET Core Support Policy, <https://dotnet.microsoft.com/en-us/platform/support/policy/dotnet-core>

[5] C Sharp (programming language), <https://en.wikipedia.org/wiki/C_Sharp_(programming_language)>

[6] C# in 2023: The MOST POPULAR Programming Language?, <https://www.bytehide.com/blog/c-wants-to-become-the-most-popular-programming-language-in-2022>

[7] 2022 developer survey, <https://survey.stackoverflow.co/2022/#most-popular-technologies-webframe>

[8] Introduction to SignalR, <https://learn.microsoft.com/en-us/aspnet/signalr/overview/getting-started/introduction-to-signalr>

[9] What is the Angular Framework?, <https://www.blog.globalteams.ltd/what-is-the-angular-framework-new-features-and-updates-2023>

[10] Diferențele dintre Javascript și Typescript, <https://hitechglitz.com/romania/diferentele-dintre-javascript-si-typescript/>

[11] The top programming languages, <https://octoverse.github.com/2022/top-programming-languages>

[12] Microsoft SQL Server, <https://ro.wikipedia.org/wiki/Microsoft_SQL_Server>

[13] What's new in SQL Server 2022, <https://learn.microsoft.com/ro-ro/sql/sql-server/what-s-new-in-sql-server-2022?view=sql-server-ver16>

[14] SQL Server 2022, <https://www.microsoft.com/en-us/sql-server/sql-server-2022>

[15] 2022 developer survey, <https://survey.stackoverflow.co/2022/#most-popular-technologies-database-learn>

[16] HMACSHA512 Class, <https://learn.microsoft.com/en-us/dotnet/api/system.security.cryptography.hmacsha512?view=net-7.0>