

UNIVERSITATEA „ALEXANDRU IOAN CUZA” IAȘI

FACULTATEA DE INFORMATICĂ



LUCRARE DE LICENȚĂ

MasterIT

propusă de

Diana Ilisei

Sesiunea: iulie, 2019

Coordonator științific

Prof. Colab. Florin Olariu

UNIVERSITATEA „ALEXANDRU IOAN CUZA” IAȘI

FACULTATEA DE INFORMATICĂ

MasterIT

Diana Ilisei

Sesiunea: *iulie, 2019*

Coordonator științific

Prof. Colab. Florin Olariu

Cuprins

Introducere	4
Context.....	4
Motivație	5
Obiective generale	6
Scurtă descriere a soluției	6
Abordare tehnică.....	7
Limbaje și medii de programare	7
Instrumente software.....	8
1. Contribuții	11
2. Descrierea problemei	12
3. Descrierea soluției.....	13
Structura aplicației	13
<i>Backend</i>	13
<i>Frontend</i>	20
Funcționalități	22
Concluziile lucrării.....	33
Bibliografie	34
Anexe	36

Introducere

Context

Considerat un punct de start eficient și la îndemâna studenților și proaspăt absolvenților de facultate, internship-ul a prins tot mai mult teren în țara noastră, fiind util atât pentru companii cât și pentru viitorii angajați.

Alături de voluntariat, experiența unui internship plătit sau neplătit a devenit un *must-have* pentru CV-ul unui IT-ist la început de drum și nu numai. Acest fapt este dovedit de creșterea de la an la an a numărului de oferte disponibil la începutul fiecărei vacanțe de vară.

Ce este și ce oferă un internship în IT?

Un internship este un program de studii și practică în cadrul unei companii, acesta fiind dedicat studenților din domeniul informaticii. Ca interni, aceștia din urmă au posibilitatea de a observa îndeaproape etapele realizării unui proiect, se acomodează cu ritmul și atmosfera dintr-o companie și pun în aplicare noțiunile dobândite în cadrul facultății. Durata unui astfel de program variază de la 2 luni până la 6 luni, dar majoritatea firmelor de IT planuiesc internship-uri care nu se suprapun cu calendarul unui an universitar, astfel încât studentul să își poată continua și finaliza studiile la timp.

Cum se desfășoară?

În cele mai multe cazuri, internii sunt recrutați printr-un proces asemănător celui de angajare (test tehnic, interviu cu un angajat al departamentului de resurse umane, interviu tehnic), sunt împărțiți în echipe și li se asignează un angajat existent în firmă care le va fi mentor. Echipele vor lucra la un proiect stabilit de mentori, sarcinile pentru terminarea lui fiind structurate în funcție de metodologia de dezvoltare software adoptată de firmă. De asemenea, echipele vor avea oportunitatea de a lucra cu produse software utilizate în mod obișnuit într-o companie de profil.

De regulă, înaintea lucrului la proiect, există o perioadă delimitată de câteva săptămâni în care internii au parte de cursuri practice în care învață concepte și noțiuni noi din aria pentru care au aplicat.

La sfârșitul internship-ului, internii cu rezultate foarte bune pot primi oferte de muncă, acest lucru fiind cel mai mare beneficiu adus companiilor deoarece își asigură angajați competenți și pregătiți pentru sarcinile de zi cu zi.

Motivație

Am decis să dezvolt o aplicație web dedicată programelor de internship în urma experienței personale pe care am dobândit-o după ce am terminat anul al doilea de facultate când am fost angajată ca intern într-o companie locală și în urma discuțiilor purtate cu mai mulți colegi care au participat la rândul lor la internship-urile oferite de câteva companii din Iași. Astfel am adunat informații, păreri, sugestii, am aflat care sunt punctele comune pro și contra din pachetul de programe de internship și am decis să fac o aplicație web care să le fie utilă atât companiilor cât și internilor.

Pe de o parte, din experiența proprie am realizat că în unele cazuri sincronizarea echipei de interni cu mentorul poate fi anevoioasă din pricina mai multor factori: mentorul, fiind angajat al firmei respective, are sarcinile lui de îndeplinit, ședințe și discuții anunțate sau de ultim moment la care trebuie să participe. De asemenea, se mai poate întâmpla ca atât mentorul cât și internii să poată uita sau amâna lămurirea unor probleme cu care cei din urmă s-au confruntat pe parcursul procesului de învățare.

Pe de altă parte, din discuțiile pe care le-am purtat cu colegii care au participat la diverse programe de internship, mi-am dat seama că o altă îmbunătățire adusă acestora ar fi existența unui sistem facil de evaluare a internilor cât și o metodă de a integra echipa mai ușor în fluxul de lucru al companiei.

Obiective generale

Pe baza a ceea ce am enunțat mai sus, un prim obiectiv ar fi crearea unui sistem de memento-uri prin care mentorul și echipa sa să fie notificați pentru a lua parte la ședințele organizatorice din cadrul internship-ului.

De asemenea, aplicația trebuie să pună la dispoziție o modalitate de a crea conturi speciale de mentori și conturi de interni, cu atribuții și funcționalități diferite.

Un alt obiectiv este urmărirea unei metodologii anume a fluxului de lucru, *Agile*, cu extinderea sa, *Scrum*, despre care se pot găsi informații în Anexa 1. Astfel, mentorul poate vizualiza progresul echipei după fiecare sprint Scrum, dar poate să primească remarci în legătură cu ce s-a întâmplat în decursul sprint-ului trecut. Motivul pentru care am ales să construiesc o aplicație care să îmbunătățească mediul de lucru dintr-o echipă *Scrum* de interni este pentru că această metodologie este cea mai răspândită în companii.

Mai mult decât atât, aplicația trebuie să ofere un sistem ușor de folosit de gestionare a echipelor de interni, mentorul având posibilitatea să vizualizeze participanții care nu sunt asigurați unei echipe cât și membrii echipei sale.

Un alt obiectiv ar fi asigurarea unui sistem de evaluare a internilor structurat în funcție de aria pentru care au aplicat la internship pe baza căruia aceștia își aproximează nivelul de cunoștințe dobândite, iar mentorii pot decide de pe urma lui ce interni ar putea fi angajați la terminarea internship-ului.

Scurtă descriere a soluției

Prezenta lucrare vine în sprijinul ambelor părți, atât al companiilor, reprezentate prin intermediul mentorilor, cât și al internilor deoarece, pe de o parte, ajută la organizarea mai eficientă a echipelor și a planificării ritmului de lucru, iar pe de altă parte, oferă o modalitate de testare tehnică a posibililor viitori angajați. De asemenea, lucrarea este dedicată programelor de internship ce implementează metoda de gestionare a proiectelor Scrum a metodologiei Agile descrise succint în Anexa 1.

Aplicația vine ca o completare a soft-urilor utilizate zilnic într-o companie pentru organizarea eficientă a sarcinilor (e.g. Jira), comunicarea efectivă a echipelor (e.g. Slack),

noutatea fiind faptul că pune accent pe partea de mentorat din cadrul unei companii, oferă un sistem de coordonare a participanților și de testare a cunoștințelor dobândite în cadrul internship-ului.

Fiind o aplicație web, MasterIT are o structură bazată pe arhitectura client-server, astfel avem:

- Backend-ul aplicației este un REST API (Anexa 2) construit în NodeJS, utilizând și framework-ul Express (Anexa 3).
- Frontend-ul aplicației este format din componente Vue.js îmbunătățite de Vuetify care comunică cu serverul prin apeluri asincrone Axios (Anexa 4).
- Baza de date: nestructurată sau NoSQL, făcută în MongoDB (Anexa 5).

Abordare tehnică

Limbaje și medii de programare

NodeJS și Express (Anexa 3)

NodeJS este un mediu de lucru pe server open-source, cross-platform, care folosește ca limbaj de programare JavaScript, implicit și TypeScript.

Express este un framework minimalist, care simplifică implementarea unei arhitecturi de tip *MVC*¹ și furnizează soluții pentru crearea și particularizarea rutelor, manipularea cererilor HTTP.

Vue și Vuetify (Anexa 4)

Pe partea de client a aplicației, am decis să folosesc un framework ce prinde tot mai mult teren în 2019 deoarece reușește să combine beneficiile aduse de framework-urile existente în producție (ReactJS și Angular), Vue.js. De asemenea, Vue.js are o dimensiune considerabil mai mică (18 – 21 kB), are o structură simplă care se pliază atât pe proiectele mici cât și pe cele complexe. Un alt plus ar fi documentația foarte detaliată de care dispune, documentație necesară celor care învață cum să construiască o aplicație web folosind Vue.js.

¹ *Model – View – Controller*: <https://www.codecademy.com/articles/mvc>

Vuetify este un UI (*User Interface*) framework care îmbină specificațiile de la *Google Material Design*² cu elementele definitorii dintr-o aplicație creată cu Vue.js.

MongoDB (Anexa 5)

MongoDB este o bază de date NoSQL bazată pe documente, deci fiecare înregistrare este de fapt un document JSON a cărui structură poate varia, indexarea și interogarea acestora fiind realizate mai ușor, permițând accesul facil și rapid la date. Documentele sunt de fapt rândurile dintr-o tabelă SQL, iar cele din urmă sunt reprezentate de colecții în MongoDB.

Instrumente software

Postman (Figura 1)

Postman este un instrument software de verificare a unui REST API care simulează cererile HTTP venite de la client la server și afișează rezultatul acestora. Cu ajutorul lui am testat API-ul pe care l-am implementat pe partea de server și am salvat într-o manieră organizată apelurile către endpoint-urile API-ului, așa cum se poate observa în figura de mai jos.

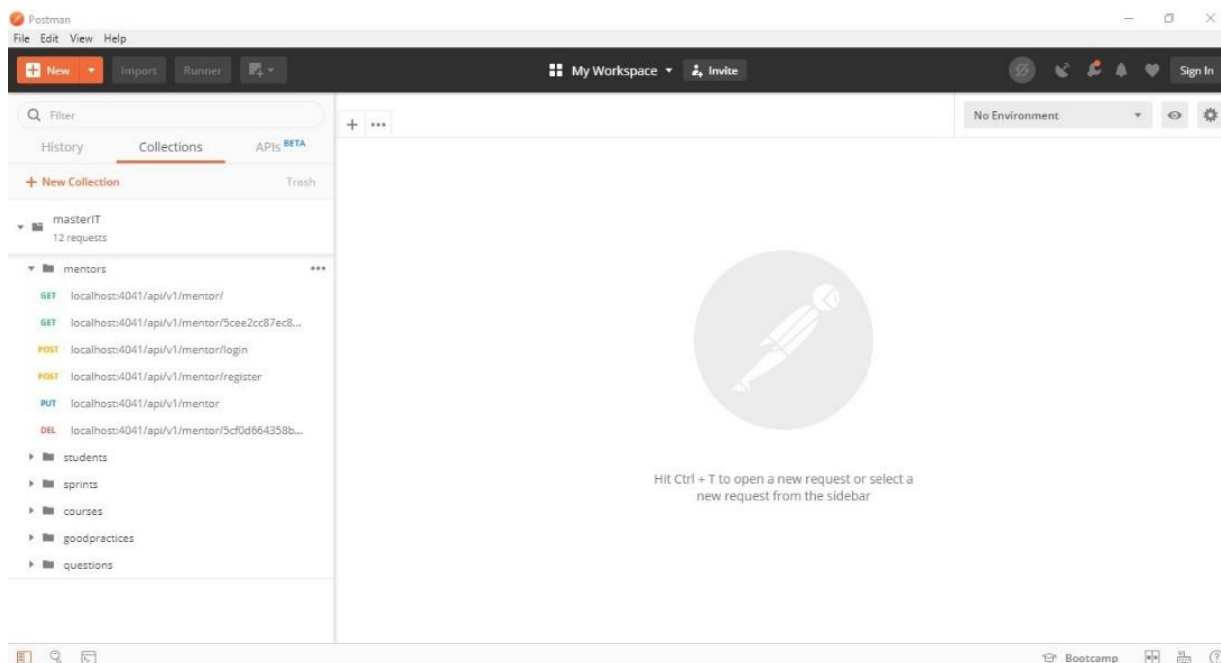


Figura 1 - Utilizarea aplicației Postman pentru testarea API-ului de pe server

² <https://material.io/design/introduction/#principles>

MongoDB Compass Community (Figura 2)

MongoDB Compass Community este interfața grafică (*GUI – Graphical User Interface*) pe care am folosit-o pentru a manipula datele din baza de date MongoDB. Aceasta este varianta gratuită a programului MongoDB Compass care oferă mai multe funcționalități. În figura de mai jos se poate observa structurarea documentelor dintr-o colecție NoSQL. Cu ajutorul acestei aplicații se pot realiza operații CRUD direct pe date.

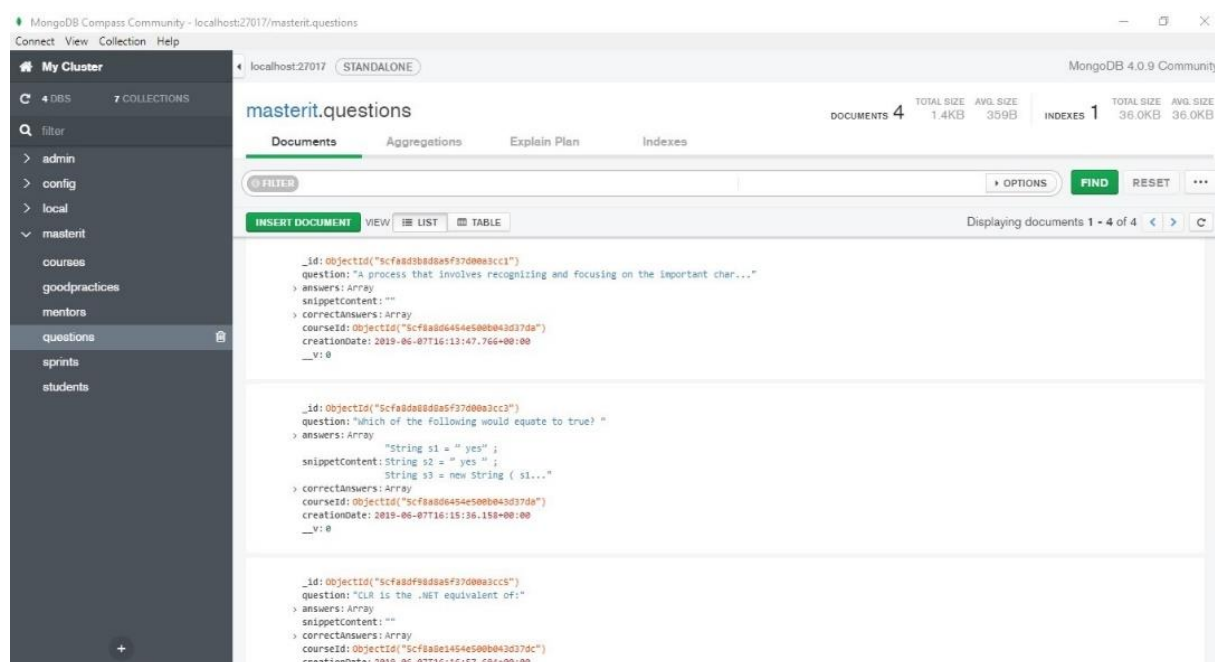


Figura 2 - Utilizarea MongoDB Compass pentru manipularea datelor

Visual Studio Code

Visual Studio Code este un editor gratuit de cod creat de Microsoft. Este foarte utilizat pentru construirea și depanarea aplicațiilor, deoarece oferă evidențierea sintaxei codului pentru o multitudine de limbaje de programare și are integrat sistemul de versionare Git. Cu ajutorul acestui IDE am reușit să implementez atât proiectul de backend cât și cel de frontend.

C4Model & Structurizr (Figura 3)

Modelul C4 a fost creat ca o metodă ajutătoare pentru echipele de dezvoltare software pentru a descrie arhitectura software, atât în timpul sesiunilor de proiectare cât și pentru a

documenta retrospectiv codul. Este un mod de a crea hărți ale structurii codului, pe nivele diferite de detaliere.

Structurizr este un editor online ce implementează acest model într-un mod interactiv și ușor de utilizat. Am folosit Structurizr pentru a realiza diagramele bazei de date și cele ce prezintă structura în ansamblu și în detaliu a proiectului.

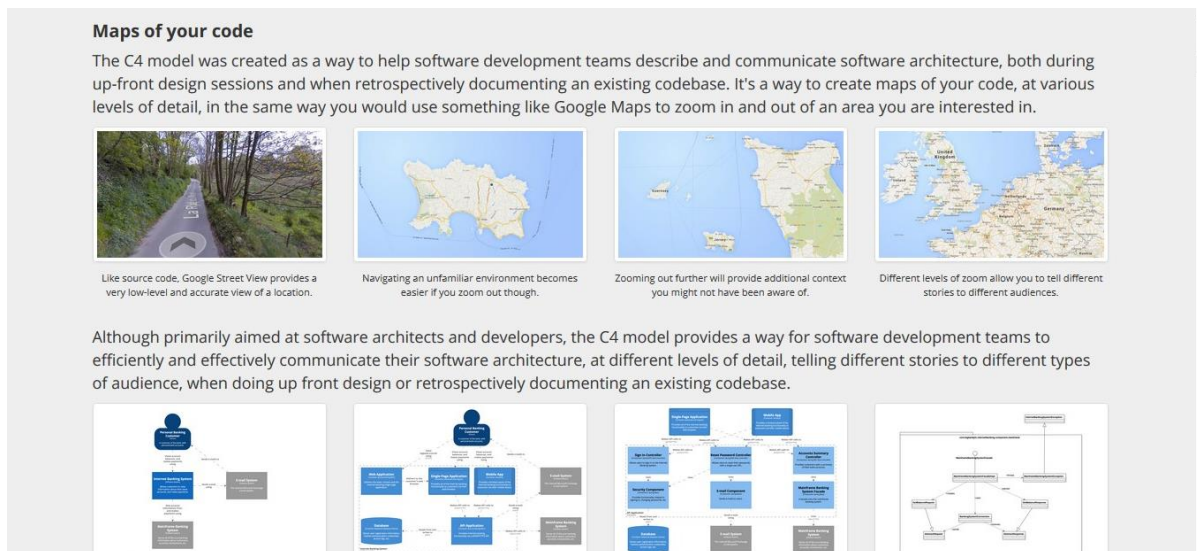


Figura 3 - Captură de ecran de pe <https://c4model.com/>

Git & Git Bash

Git este un sistem distribuit de versionare a controlului ce rulează pe majoritatea platformelor și ajută la păstrarea istoricului dezvoltării aplicațiilor cât și la coordonarea eficientă a sarcinilor împărțite în echipe mari.

Git Bash este o aplicație pentru sistemul de operare Windows ce oferă un terminal pentru comenzile Git. Cu ajutorul acestuia am încărcat în contul git fiecare progres pe parcursul aplicației și am revenit la versiuni anterioare, dacă a fost cazul.

1. Contribuții

Așa cum am menționat și în introducere, ideea acestei aplicații a venit, în principal, după experiența personală pe care am avut-o anul trecut în cadrul unui program de internship. Observând în mod subiectiv, din perspectiva unui intern, cam ce ar fi de îmbunătățit și mai apoi ascultând opiniile colegilor despre experiențelor lor, am sintetizat principalele puncte comune de rezolvat și am încercat să le conectez. Pe urmă, am căutat alte aplicații care ar putea reprezenta o rezolvare parțială sau completă a problemelor găsite însă nu am găsit niciuna dedicată programelor de internship.

De asemenea, am dorit să aflu care este perspectiva din punctul de vedere al unui mentor, astfel l-am întrebat pe domnul profesor coordonator pe ce să pun accent atunci când voi stabili funcționalitățile aplicației și am hotărât împreună care vor fi utilitățile principale.

Am început proiectarea aplicației prin stabilirea tehnologiilor folosite, tehnologii descrise în capitolul anterior și în anexe. Pe urmă, am schițat schema bazei de date și structura server-ului, ținând cont de bunele practici specifice limbajului de programare ales și de ce am aprofundat în cadrul facultății (design pattern-uri, organizarea eficientă a codului etc).

Pentru o mai bună vizualizare a componentelor aplicației am ales să schițez și partea de client pentru a nu rata implementarea unei funcționalități pe backend în timpul dezvoltării.

A urmat lucrul propriu-zis timp de câteva luni în care am testat, implementat și îmbunătățit aplicația. Pe parcursul acestei perioade am primit feedback constant de la domnul profesor coordonator și câteva sfaturi utile ce mi-au ușurat munca.

Astfel, MasterIT a fost creată cu ajutorul unei suite de instrumente software și tehnologii noi și utilizate la scară largă, precum și a cunoștințelor învățate din câteva materii din facultate: Ingineria Programării, Rețele de Calculatoare, Tehnologii WEB, Dezvoltarea Aplicațiilor WEB la Nivel de Client, Programare Orientată pe Obiect.

2. Descrierea problemei

Internship-ul este o tehnică destul de recentă dar foarte utilizată și folositoare prin care marile companii de dezvoltare software și nu numai recrutează posibili viitori angajați capabili pentru cerințele lor. Firmele le oferă studenților mai multe programe, iar aceștia își aleg unul pe placul lor, trec prin procesul de recrutare și devin, sau nu, interni. Internship-ul e un proiect win-win în care ambele părți au de câștigat, internii – experiență, cunoștințe și prieteni noi, iar firmele – angajați potriviți și instruiți.

Însă, pe parcursul acestui proces de acomodare a internului cu noul mediu, cu un program de lucru part-time, de 4-6 ore, sau chiar full-time, de 8 ore, pot apărea probleme de comunicare, nelămuriri etc. Fiind începători și poate neînțelegând unde se îmbină cunoștințele teoretice cu practica efectivă, internii pot fi debusolați și din neatenție sau chiar din cauza stresului ce intervine într-o astfel de situație, pot uita de întâlnirile programate cu mentorul echipei. Bineînțeles, se poate întâmpla și invers, adică mentorul să nu poată participa la ședințele cu echipa sa din varii motive.

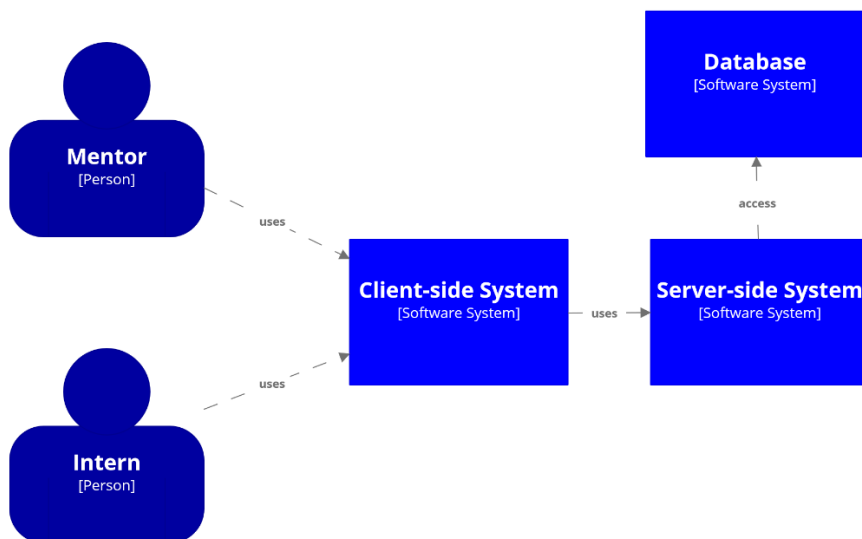
O altă problemă ar fi impactul psihologic pe care o evaluare printată pe hârtie o are asupra internilor: aflându-se într-un mediu profesionist, aceștia nu vor să dezamăgească sau să fie sub nivelul așteptărilor, așa că un intern se poate simți nesigur pe sine în fața unei evaluări scrise care poate provoca un stres nejustificat. Astfel, din punctul lor de vedere, evaluarea este un proces de natură eliminatorie. Mai mult decât atât, corectarea acelor evaluări reprezintă un timp consumat pentru mentor, timp în care se putea ocupa de responsabilitățile postului său sau de pregătirea echipei.

De asemenea, feedback-urile și bunele practici pe care echipa hotărăște să le urmeze sunt notate pe hârtii și acest fapt poate duce la pierderea neintenționată sau uitarea acelor reguli stabilite pe parcursul timpului.

Nu în ultimul rând, având în vedere faptul că mentorul este responsabil cu progresul echipei sale, acesta are nevoie de un sistem de a înregistra și vizualiza evoluția membrilor pentru a crea o echipă coezivă și care prezintă o ascensiune treptată.

3. Descrierea soluției

Structura aplicației



System Landscape diagram

Diagram created with Structurizr - your software architecture documentation hub | Workspace last modified: Tue Jun 25 2019 00:22:18 GMT+0300 (Ora de vară a Europei de Est)

Figura 4 - Diagrama Level 1

Backend

Atunci când am schițat structura generală a serverului am ținut cont de câteva noțiuni și tehnici utile pe care le-am învățat pe parcursul facultății și care m-au ajutat la proiectarea aplicației:

- Principiile *SOLID*, în special *Dependency Injection* și *Inversion of Control*³: implementarea acestora a condus la decuplarea codului și la separarea atribuțiilor.

³ <https://samueleresca.net/2017/07/inversion-of-control-and-unit-testing-using-typescript/>

- Diagrame UML: utile pentru a vizualiza structura generală și detaliată a întregului sistem sau a unei componente specifice.
- Arhitectura MVC a unei aplicații web.

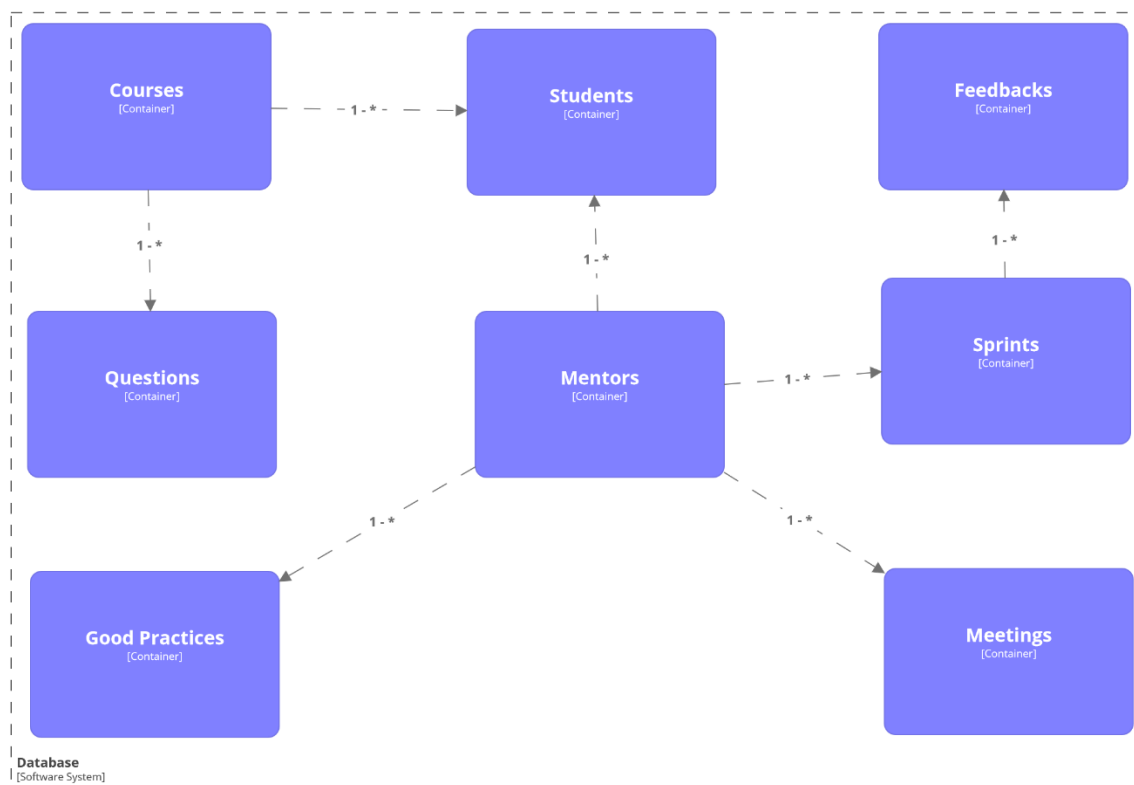
Pe baza acestora am conceput o ierarhie de directoare și fișiere în felul următor (Figura II.3):

În directorul `models` există toate entitățile necesare reprezentării datelor din baza de date (spre exemplu, `Course`, `Question`, `Mentor`, `Sprint` etc). Fiecare entitate este descrisă cu ajutorul claselor `TypeScript` și asociată cu entitatea corespunzătoare din `MongoDB` folosind modelul `Typegoose`:

- `Mentor`: descrie entitatea necesară unui cont de mentor ce are detalii personale, email, parolă, componența echipei și un câmp ce stabilește dacă acel cont este activat de un administrator sau nu.
- `Intern`: deține informațiile personale ale unui intern, ce curs urmează în cadrul internship-ului, din ce echipă face parte, dacă a realizat evaluarea tehnică și ce scor a luat.
- `Course`: păstrează denumirea fiecărei arii disponibile în internship.
- `Sprint`: este entitatea specifică unui sprint, descriind când începe și când se termină, precum și câte puncte a primit inițial și care a fost rezultatul final. Un sprint aparține unui singur mentor.
- `Question`: conține detaliile necesare pentru afișarea unei întrebări din evaluarea tehnică, astfel avem un corp al întrebării care poate conține o porțiune de cod, avem lista cu posibilele răspunsuri și cea cu răspunsurile corecte. Fiecare întrebare are un singur curs asignat.
- `Meeting`: reprezintă entitatea ce se ocupă cu stocarea memento-urilor pentru ședințele Scrum și conține denumirea întâlnirii, data și ora acesteia. O entitate `Meeting` are un singur mentor asignat.
- `Feedback`: deține fiecare întrebare adresată într-un proces de feedback precum și răspunsurile aferente. Un `Feedback` aparține unui singur `Sprint`, implicit, unui singur mentor.

- GoodPractice: utile pentru păstrarea bunelor practici stabilite de mentor împreună cu echipa sa. O astfel de entitate aparține unui singur mentor.

Relațiile dintre aceste entități sunt descrise în figura de mai jos.



Container diagram for Database

Diagram created with Structurizr - your software architecture documentation hub | Workspace last modified: Mon Jun 24 2019 23:13:44 GMT+0300 (Ora de vară a Europei de Est)

Figura 5 - Schema bazei de date din MongoDB

În directorul repositories se află interfața ICrudRepository (Figura 7) în care sunt declarate toate operațiile de bază CRUD. Această interfață va fi extinsă de interfețele corespunzătoare fiecărei entități, denumite în funcție de numele lor (i.e ICourseRepository, IQuestionRepository etc), iar în acestea vor exista, dacă este cazul, funcții specifice (pentru Mentor este posibilă necesitatea unei metode getByEmail, Figura 8). Toate aceste interfețe sunt, la rândul lor, implementate de clasele repository (CourseRepository, QuestionRepository etc).

În directorul config există fișierul de configurare a conexiunii cu baza de date, precum și container-ul necesar pentru implementarea tehnicii Inversion of Control asupra interfețelor și claselor (Figura 10).

În directorul controllers se află toate endpoint-urile API-ului împărțite în funcție de entitatea pe care se face operația specificată în antetul cererii HTTP. Practic, fiecare controller este punctul de acces din exterior către funcționalitățile serverului.

În directorul middleware există funcțiile de autentificare și autorizare a utilizatorilor, funcții implementate respectând conceptul JWT (*JSON Web Token*⁴). Procesul realizat de această metodă de logare/authentificare este redat în figura următoare:

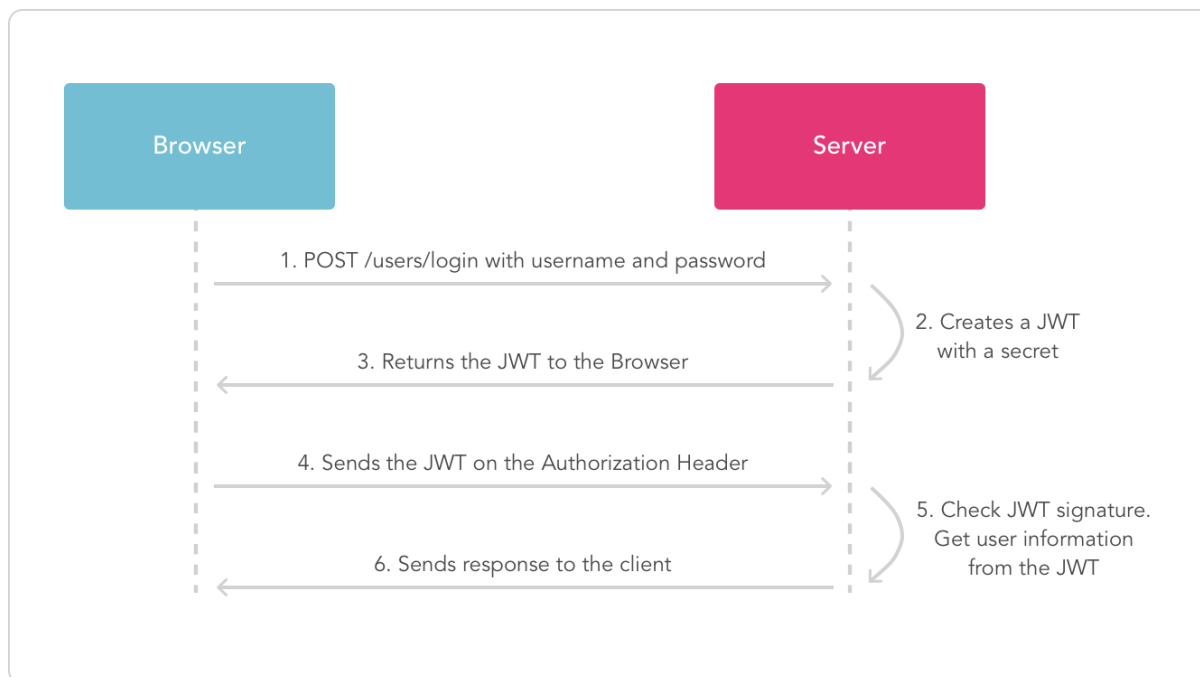


Figura 6 - Sistemul de login bazat pe JWT - <https://jwt.io/introduction/>

Fișierul server.ts stabilește configurarea serverului de NodeJS.

Fișierul app.ts este inima întregului proiect de backend deoarece pune laolaltă toate funcționalitățile descrise mai sus, stabilește conexiunea cu baza de date potrivit configurărilor și expune rutele la care clienții pot face cereri HTTP.

⁴ <https://jwt.io/>


```

lib ▸ repository ▸ TS ICrudRepository.ts ▸ ...
1  import { Typegoose } from 'typegoose';
2
3  export abstract class ICrudRepository<T extends Typegoose> {
4      abstract getAll(): Promise<T[]>;
5
6      abstract getById(id: String): Promise<T>;
7
8      abstract add(document: T): Promise<T>;
9
10     abstract update(id: String, document: T): Promise<T>;
11
12     abstract delete(id: String): Promise<T>;
13 }

```

Figura 7 - ICrudRepository.ts

```

lib ▸ repository ▸ TS IMentorRepository.ts ▸ IMentorRepository
1  import { ICrudRepository } from "../ICrudRepository";
2  import { Mentor } from "models/Mentor";
3
4  export abstract class IMentorRepository extends ICrudRepository<Mentor> {
5      abstract getByEmail(email: string): Promise<Mentor>;
6  }

```

Figura 8 - IMentorRepository.ts



Figura 9 - Structura simplificată a directoroarelor

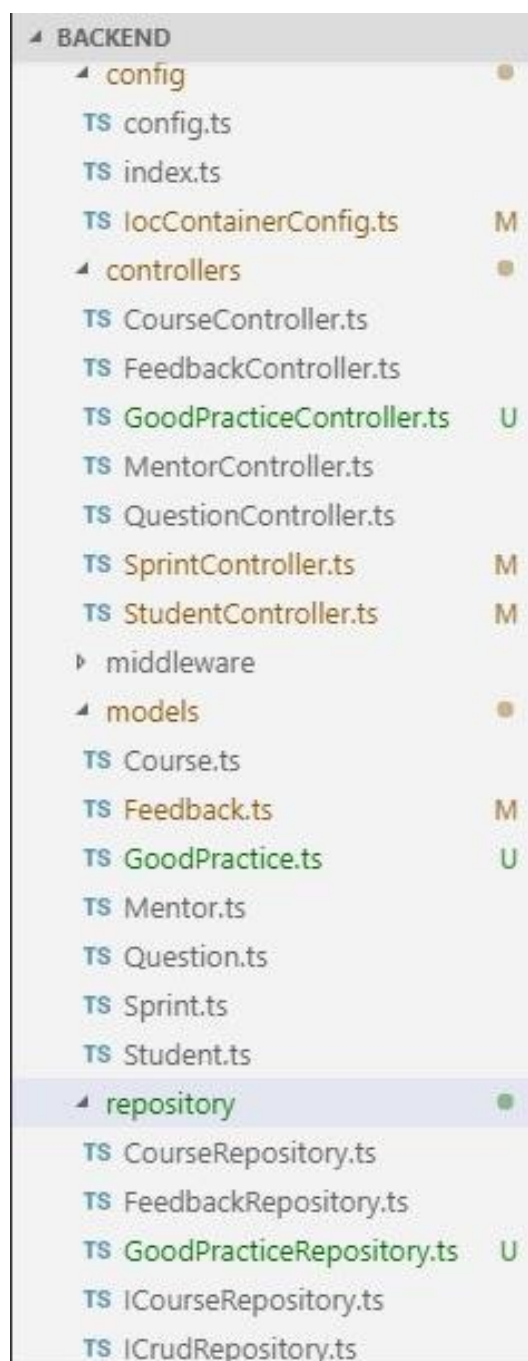
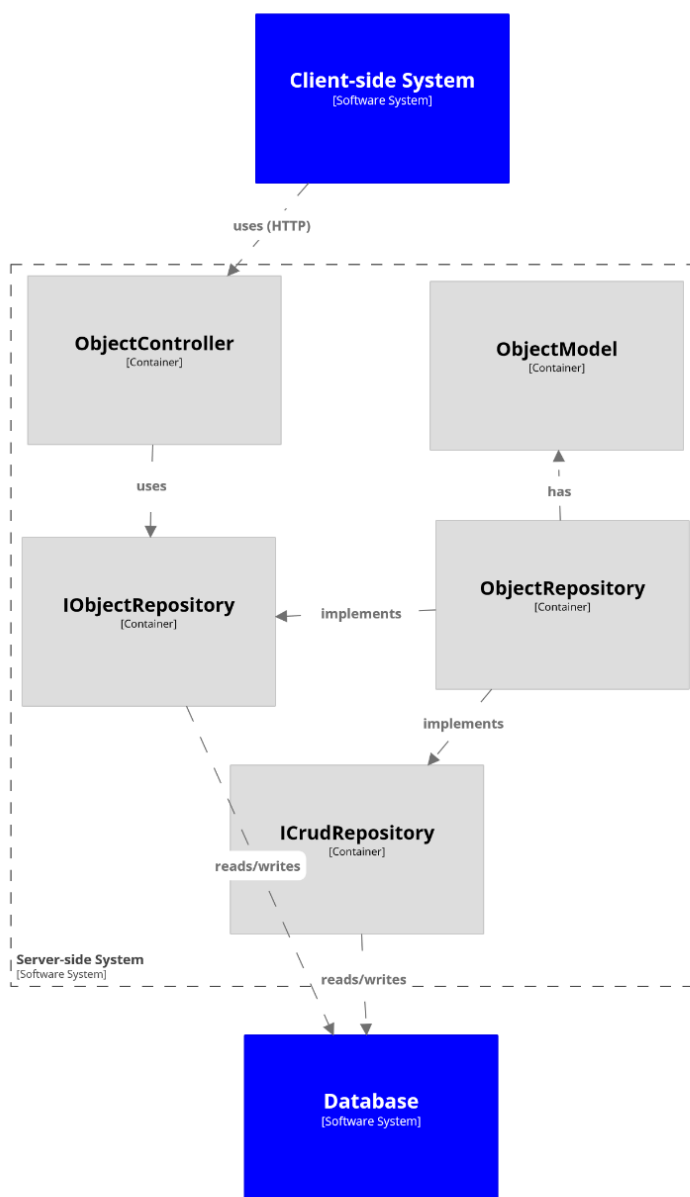


Figura 10 - Structura fișierelor de pe server

În diagrama UML de mai jos (Figura II.5) se pot vedea modul cum sunt tratate cererile HTTP venite de la client. Object definește fiecare entitate ce există în program, iar descrierea acestuia, i.e. ce proprietăți are, este cuprinsă în ObjectModel.

Clientul face o cerere la ruta unui controller care la rândul lui va apela operația corespunzătoare din repository și în cele din urmă se accesează baza de date pentru a salva modificările aduse de repository.



Container diagram for Server-side System

Diagram created with Structurizr - your software architecture documentation hub | Workspace last modified: Fri Jun 14 2019 12:49:43 GMT+0300 (Ora de vară a Europei)

Figura 11 - Diagrama UML

Frontend

Proiectul de client (Figura 12 și Figura 13) urmărește o structură în general specifică oricărui proiect făcut în Vue.js. Poate fi împărțit în funcție de tipurile de utilizatori, Mentor și Intern:

- În directorul components sunt toate fișierele .vue care compun paginile specifice utilizatorilor cât și bara de navigare care este particularizată pentru Mentori și Interni.
- În directorul views se regăsesc paginile principale accesate: mentor, intern și pagina de logare/creare cont. În view-uri sunt randate componentele.
- În directorul store sunt fișiere pentru configurarea și accesarea store-ului *Vuex*⁵ pe care l-am adăugat în aplicație pentru a face disponibile din orice componentă anumite date necesare.
- În fișierul utils.js sunt câteva funcții folosite în componente diferite.
- Fișierul router.js implementează vue-router, setează rutele către view-uri și componente.
- main.js este startul aplicației, locul unde se declară dependențele în obiectul de tip Vue.
- App.vue este view-ul principal al aplicației, în acesta fiind randate toate view-urile sau componentele disponibile.
- În directorul api-services găsim o structură asemănătoare cu cea din controllers din backend, în sensul în care există fișiere pentru fiecare tip de entitate din baza de date. Fișierele conțin operațiile CRUD implementate cu *Axios*⁶ și făcute vizibile în proiect sub altă denumire. Astfel, în cazul în care se va schimba o rută de pe server, în proiectul de client vom schimba într-un singur loc, în repository-ul corespunzător. Un exemplu de serviciu disponibil se regăsește în Figura 14.

⁵ <https://vuex.vuejs.org/>

⁶ <https://github.com/axios/axios>

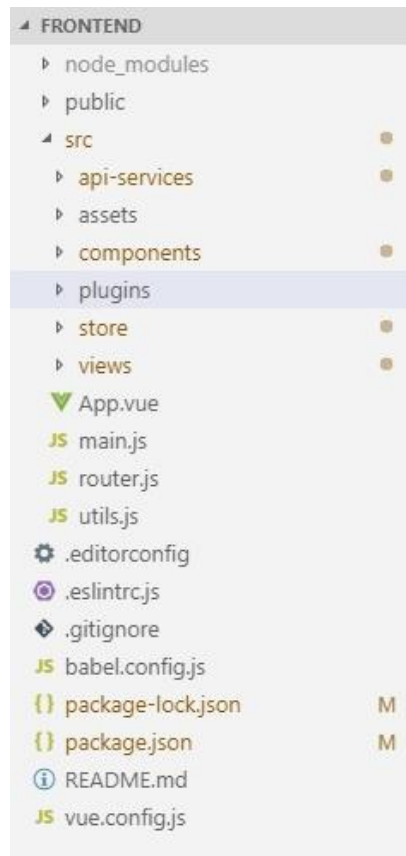


Figura 12 - Structura proiectului de client

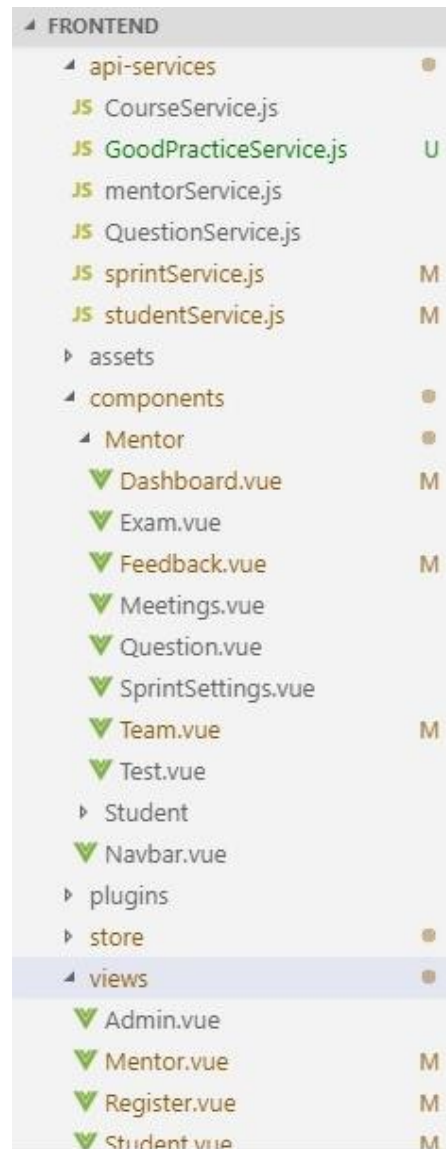



Figura 13 - Ierarhia fișierelor din proiectul de client



```

3  const RESOURCE_NAME = '/api/v1/question';
4
5  export default {
6    getAll(token) {
7      return axios({
8        method: 'get',
9        url: `${RESOURCE_NAME}/all`,
10       headers: { Authorization: `Bearer ${token}` }
11     });
12   },
13   getById(id, token) {
14     return axios({
15       method: 'get',
16       url: `${RESOURCE_NAME}/${id}`,
17       headers: { Authorization: `Bearer ${token}` }
18     });
19   },
20   getByCourseId(id, token) {
21     return axios({
22       method: 'get',
23       url: `${RESOURCE_NAME}/course/${id}`,
24       headers: { Authorization: `Bearer ${token}` }
25     });
26   },
27   create(question, token) {
28     return axios({ method: 'post', url: `${RESOURCE_NAME}`, data: question, headers: { Authorization: `Bearer ${token}` } });
29   },
30 }

```

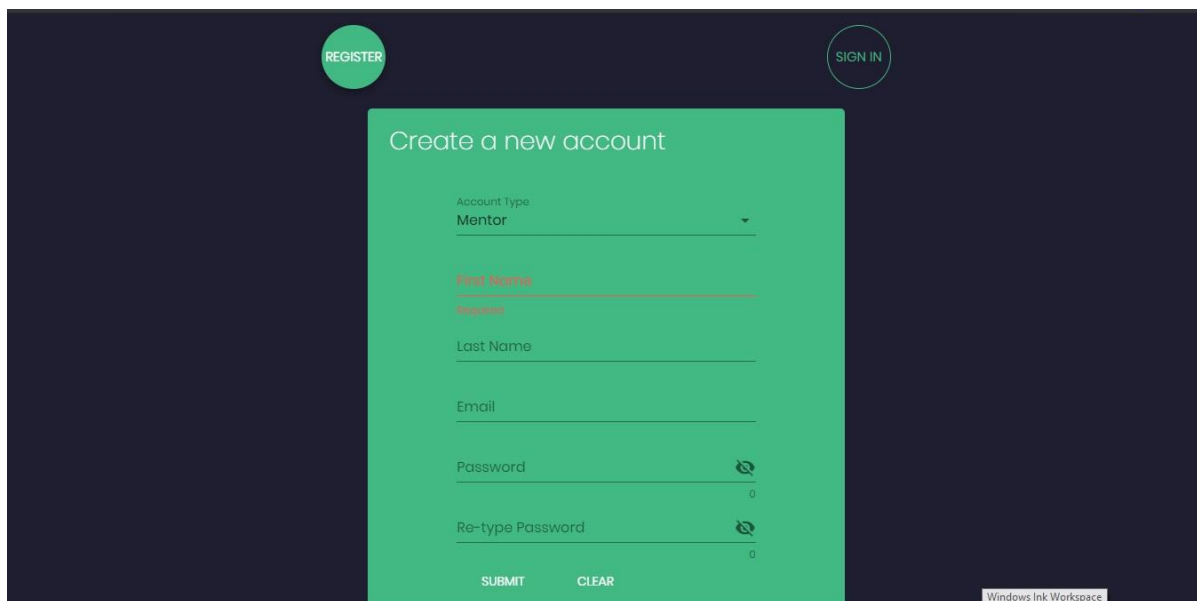
Figura 14 - O parte din QuestionRepository ce conține apelurile Axios către server

Funcționalități

Creare cont/logare

Crearea contului (Figura 15) și logarea (Figura 16): aceste operațiuni se fac din pagina de start a aplicației, atât pentru Mentor cât și pentru Intern. Utilizatorul care își creează un cont Mentor va trebui aprobat de către administratorul aplicației care deține lista cu toți mentorii înregistrați până în acel moment. Dacă nu primește aprobare, utilizatorul nu va putea accesa contul.

Pe de altă parte, contul Student poate fi creat fără aprobare, de îndată ce se salvează datele în baza de date, utilizatorul este redirecționat către Dashboard. Atât formularul de înregistrare cont cât și cel de logare au implementate validări ale textului pentru toate tipurile de input-uri (email, parola etc).



The screenshot shows a dark-themed user interface with a central light green registration form. At the top of the form, the title "Create a new account" is displayed. Below the title, there is a dropdown menu for "Account Type" with "Mentor" selected. The form includes input fields for "First Name", "Last Name", "Email", "Password", and "Re-type Password". The "Password" and "Re-type Password" fields have a small icon to the right of the input field. At the bottom of the form, there are two buttons: "SUBMIT" and "CLEAR". In the top left corner of the dark background, there is a circular button labeled "REGISTER". In the top right corner, there is a circular button labeled "SIGN IN". A small "Windows Ink Workspace" watermark is visible in the bottom right corner of the dark background.

REGISTER SIGN IN

Create a new account

Account Type
Mentor

First Name

Last Name

Email

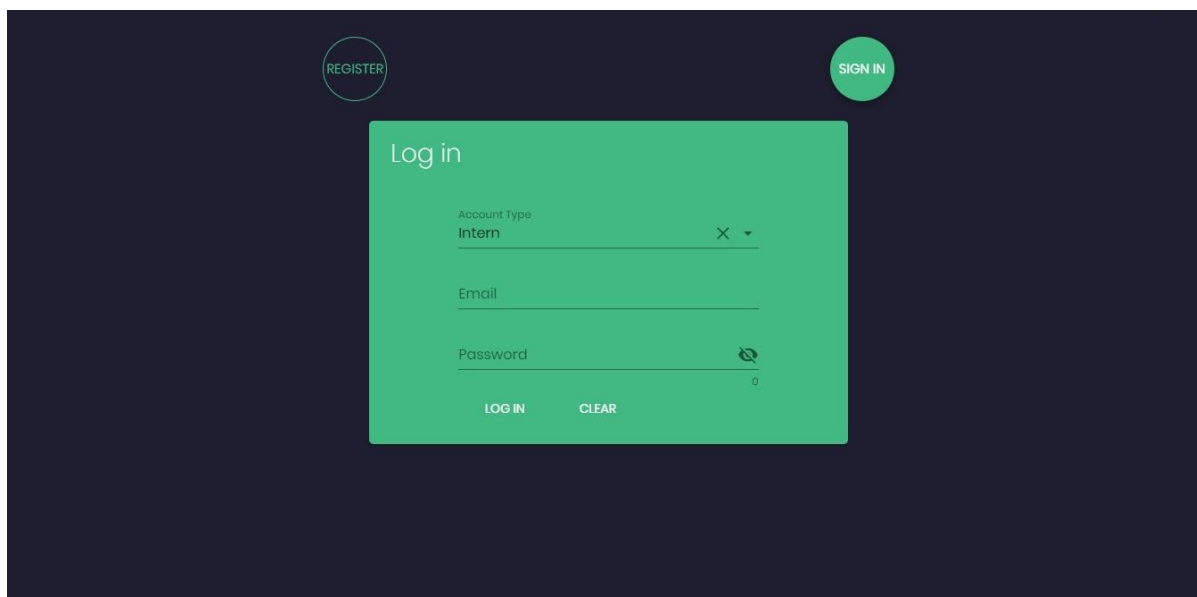
Password

Re-type Password

SUBMIT CLEAR

Windows Ink Workspace

Figura 15 - Înregistrare cont Mentor



The screenshot shows a dark-themed user interface with a central light green login form. At the top of the form, the title "Log in" is displayed. Below the title, there is a dropdown menu for "Account Type" with "Intern" selected. The form includes input fields for "Email" and "Password". The "Password" field has a small icon to the right of the input field. At the bottom of the form, there are two buttons: "LOG IN" and "CLEAR". In the top left corner of the dark background, there is a circular button labeled "REGISTER". In the top right corner, there is a circular button labeled "SIGN IN".

REGISTER SIGN IN

Log in

Account Type
Intern

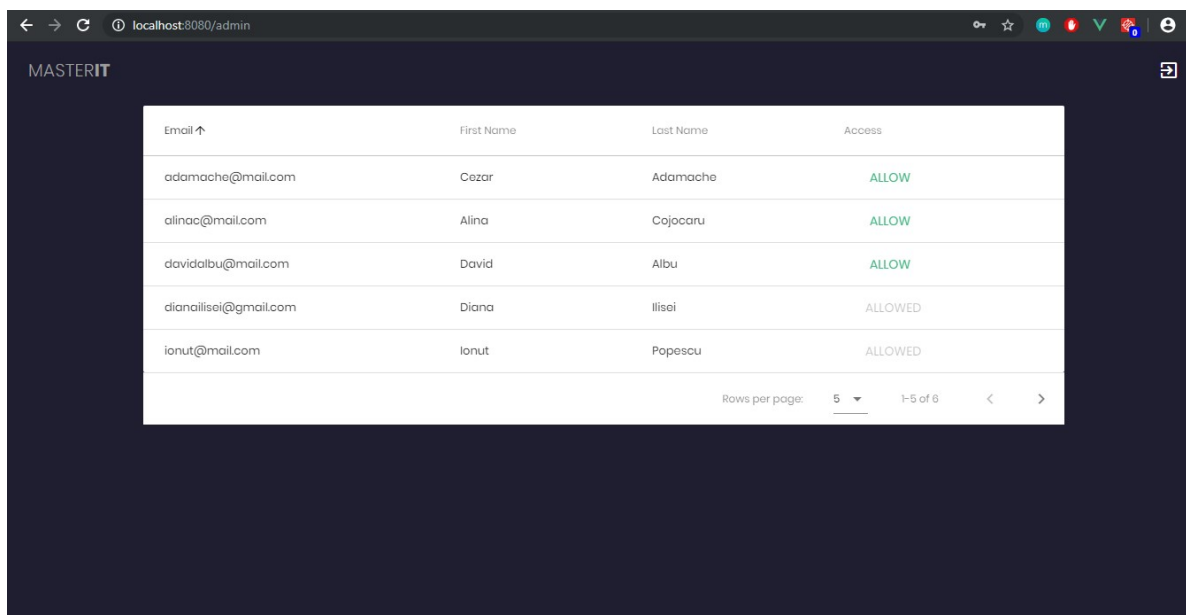
Email

Password

LOG IN CLEAR

Figura 16 - Logare cont Intern

Contul de administrator (Figura 17) conține o listă cu toți cei care și-au făcut cont de mentor. Aceștia necesită aprobare pentru a restricționa accesul la funcționalitățile oferite de acest tip de cont.



The screenshot shows a web browser window with the URL 'localhost:8080/admin'. The page title is 'MASTERIT'. It displays a table of mentors with the following data:

Email ↑	First Name	Last Name	Access
adamache@mail.com	Cezar	Adamache	ALLOW
alinac@mail.com	Alina	Cojocaru	ALLOW
davidalbu@mail.com	David	Albu	ALLOW
dianailisei@gmail.com	Diana	Ilisei	ALLOWED
ionut@mail.com	Ionut	Popescu	ALLOWED

At the bottom right of the table, there is a pagination control showing 'Rows per page: 5' and '1-5 of 6'.

Figura 17 - Cont administrator

În cele ce urmează voi prezenta mai întâi funcționalitățile oferite în contul Mentor iar apoi cele dedicate Internului.

Dashboard

Odată logat, un mentor are posibilitatea de a vizualiza în pagina Dashboard (Figura 18) detalii despre sprint-ul curent, câte minute mai sunt până la următoarea ședință Scrum și lista cu membrii echipei sale. În partea stângă a paginii există un meniu care poate fi minimizat.

Toate aceste informații pot fi editate în paginile următoare.

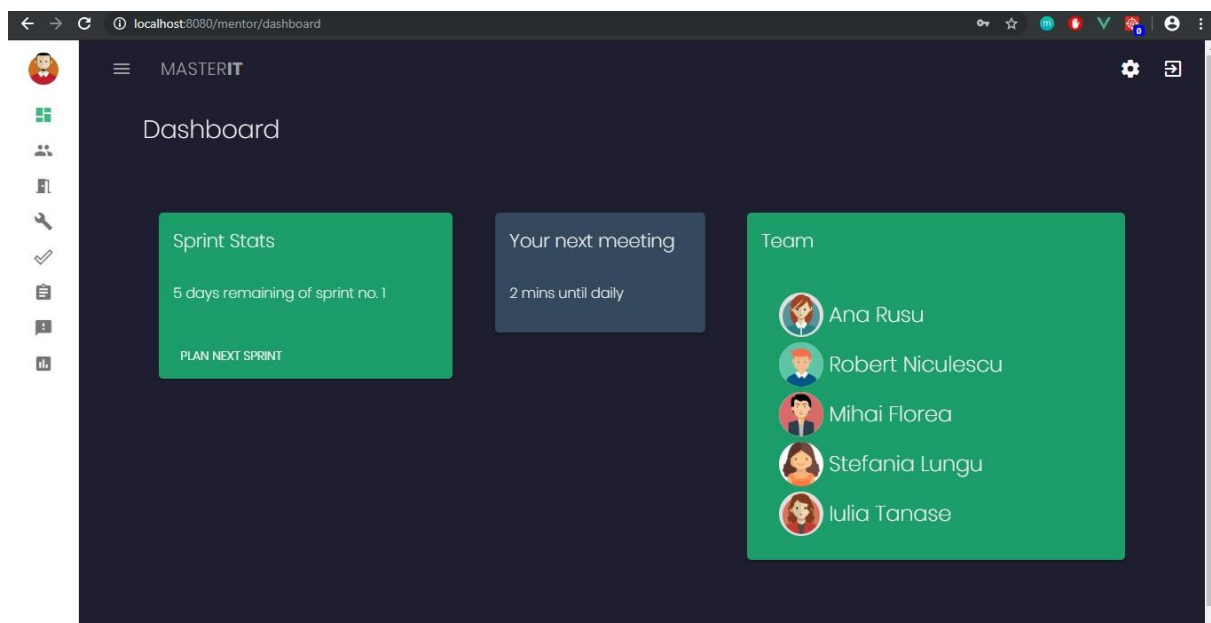


Figura 18 - Dashboard Mentor

Team

Următoarea pagină din meniu este Team. Aceasta conține 2 secțiuni: în prima (Figura 19), sunt membrii echipei cu denumirea cursului pentru care au aplicat la internship, precum și un buton de ștergere a internului din echipă. Cea de-a doua secțiune cuprinde lista cu internii fără echipă, fiecare având un buton de adăugare în echipă (Figura 20).

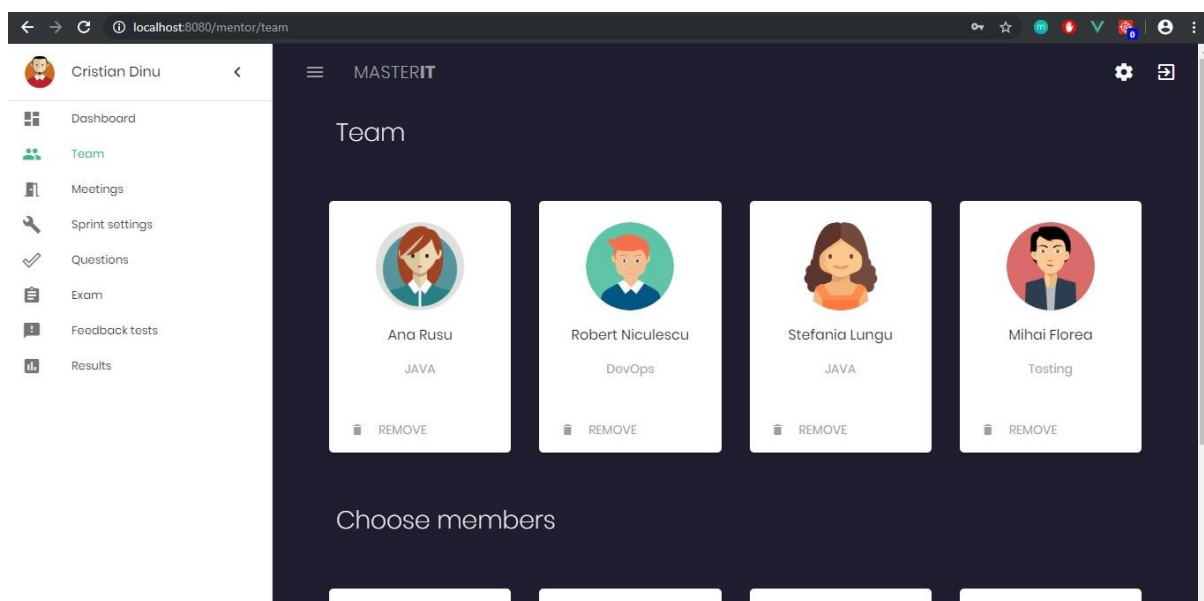


Figura 19 - Pagina Team - membrii echipei

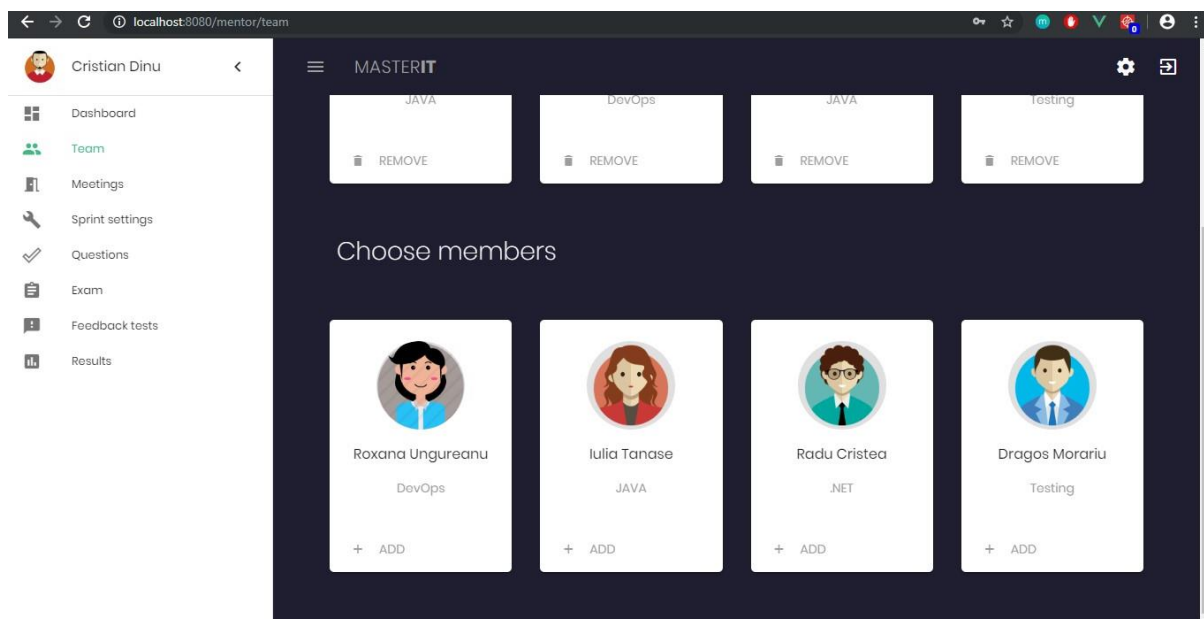


Figura 20 - Pagina Team - internii fără echipă

Meetings

Din această pagină mentorul poate să seteze data, respectiv ora următoarelor ședințe Scrum. Astfel, spre exemplu, dacă se setează ora pentru Daily Meeting, atât mentorul cât și membrii echipei sale vor primi o notificare cu câteva minute înaintea orei stabilite pentru a se pregăti corespunzător (Figura 21).

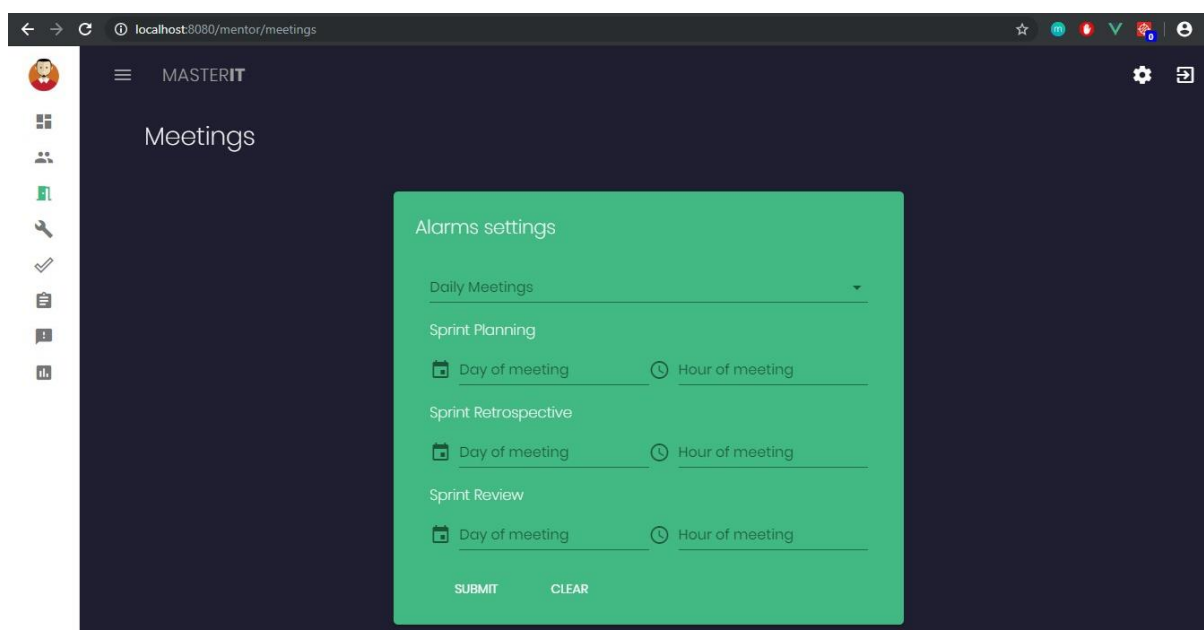


Figura 21 - Meetings

Sprint Settings

Fiind o aplicație ce urmărește metodologia Scrum, o altă caracteristică fundamentală este împărțirea timpului efectiv de lucru al unui proiect în sprint-uri. Termenul vine din sport și reprezintă acele curse de viteză pe perioade scurte de timp. S-au observat numeroase beneficii ale introducerii sprint-urilor și implicit a implementării metodei Scrum în cadrul echipelor de dezvoltare software: concentrare sporită, comunicare eficientă și transparență, facilitarea prioritizării sarcinilor, team building.

MasterIT oferă posibilitatea de a organiza echipa și modul de lucru în manieră Scrum, adică mentorul poate să creeze un sprint nou, atribuindu-i un număr de puncte (story points⁷) și data de început și de final (Figura 22) . De asemenea, există și opțiunea de a încheia sprint-ul curent, stabilind dacă s-a terminat mai devreme sau nu și numărul de story points pe care echipa le-a realizat (Figura 23).

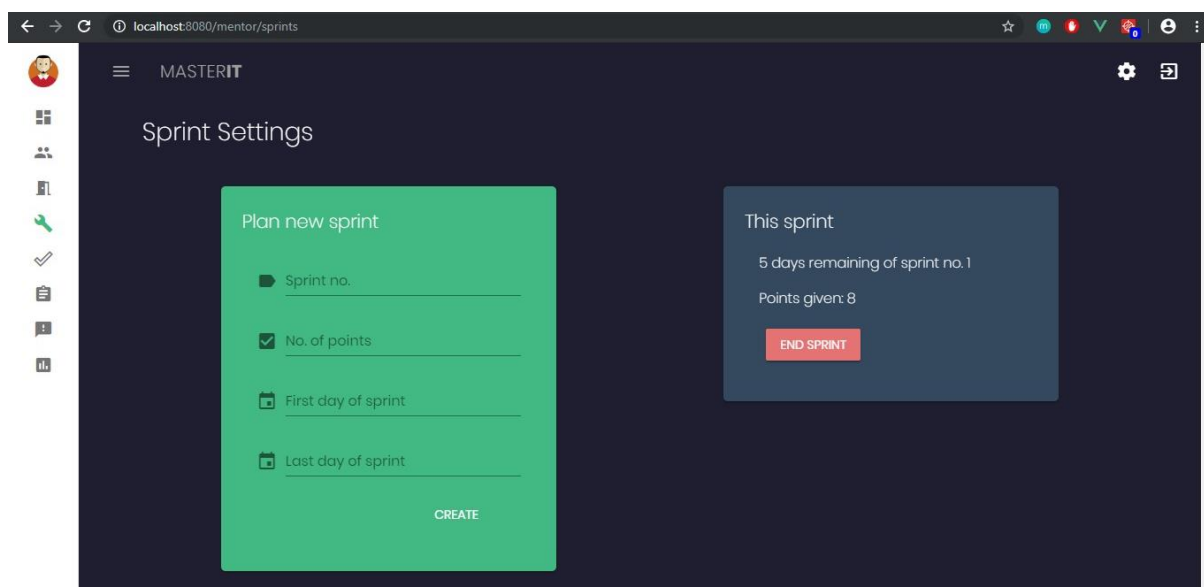


Figura 22 - Pagina Sprint Settings

⁷ Un story point este o unitate de măsură abstractă pentru stabilirea dificultății realizării unor sarcini. (<http://www.tothenew.com/blog/how-to-estimate-story-points-in-agile/>)

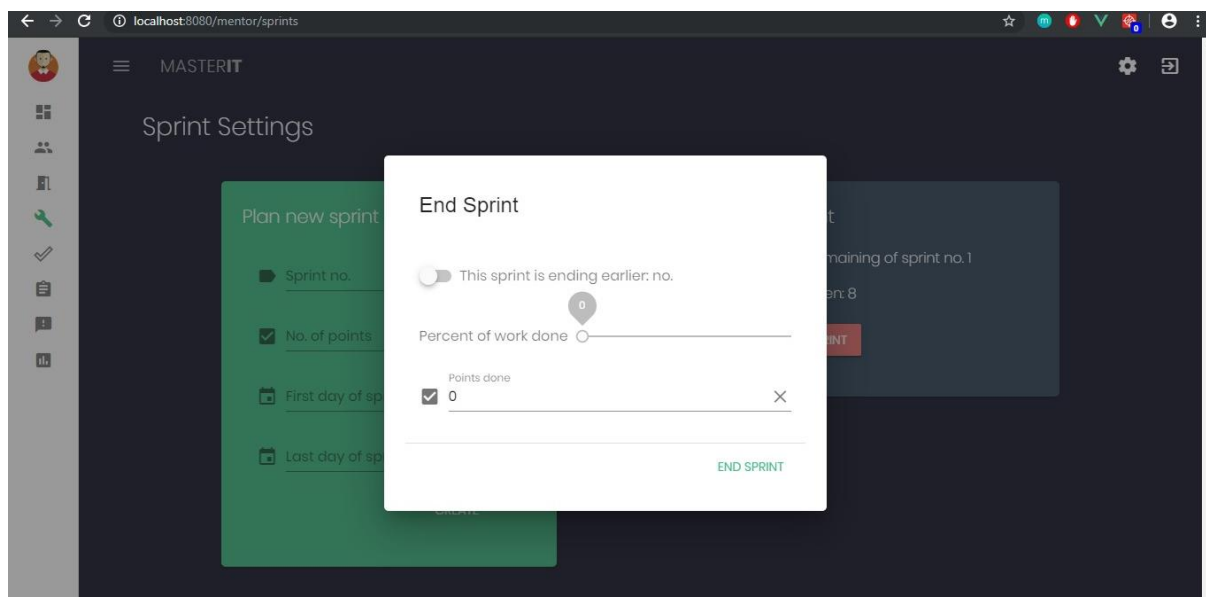


Figura 23 - Opțiune încheiere sprint

Questions

De pe această pagină mentorul poate crea și modifica întrebări grilă pentru examenul tehnic pe care studenții îl vor susține la sfârșitul programului de internship. O întrebare poate avea mai multe răspunsuri corecte și textul acesteia poate conține și o porțiune de cod afișată corespunzător (Figura 24).

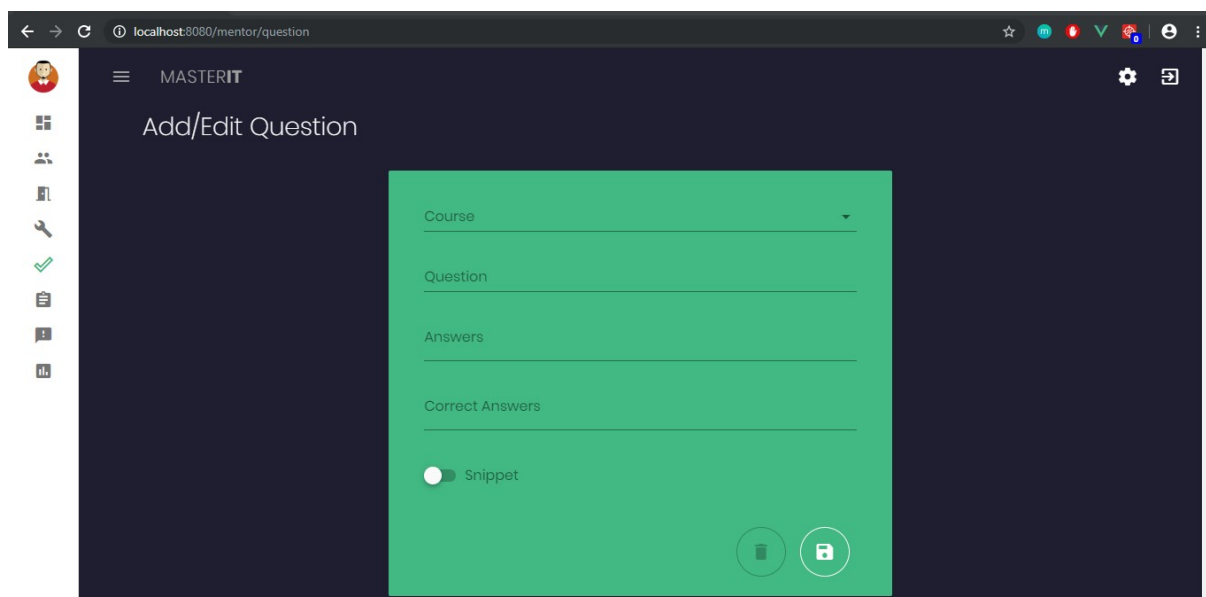


Figura 24 - Pagina Question

Evaluation

În pagina Evaluation (Figura 25) sunt sortate toate întrebările adăugate de mentori în funcție de cursul asignat. Un mentor poate adăuga sau edita o întrebare existentă, poate șterge întreg test sau îl poate face disponibil internilor. Întrebările nu sunt afișate studenților până când un mentor nu face evaluarea disponibilă. Doar studenții asigurați unui curs pot vizualiza testul acelui curs.

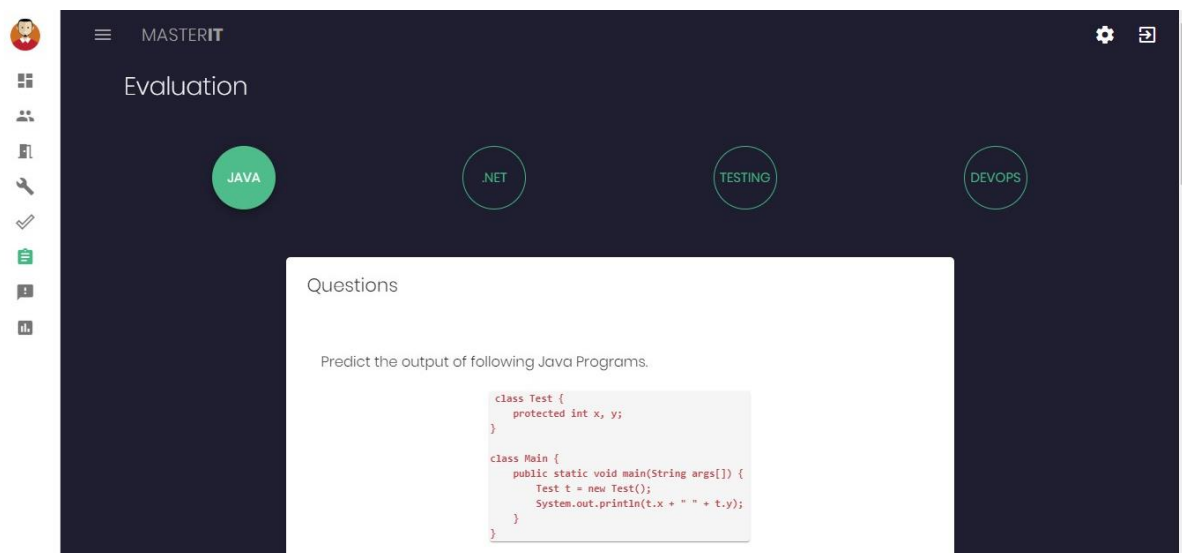


Figura 25 - Pagina Evaluation

Feedback Tests

Internii au posibilitatea de a-și exprima părerile la sfârșitul fiecărui sprint prin intermediul unui formular de feedback anonim pus la dispoziție de către mentorul lor. Acesta poate trimite studenților 3 întrebări de bază specifice metodologiei Scrum, sau le poate edita, în funcție de preferințe. De asemenea, mentorul poate vedea răspunsurile date de studenți la fiecare sprint și pe baza acestora poate crea reguli, sau good practices, pentru echipa sa (Figura 26).

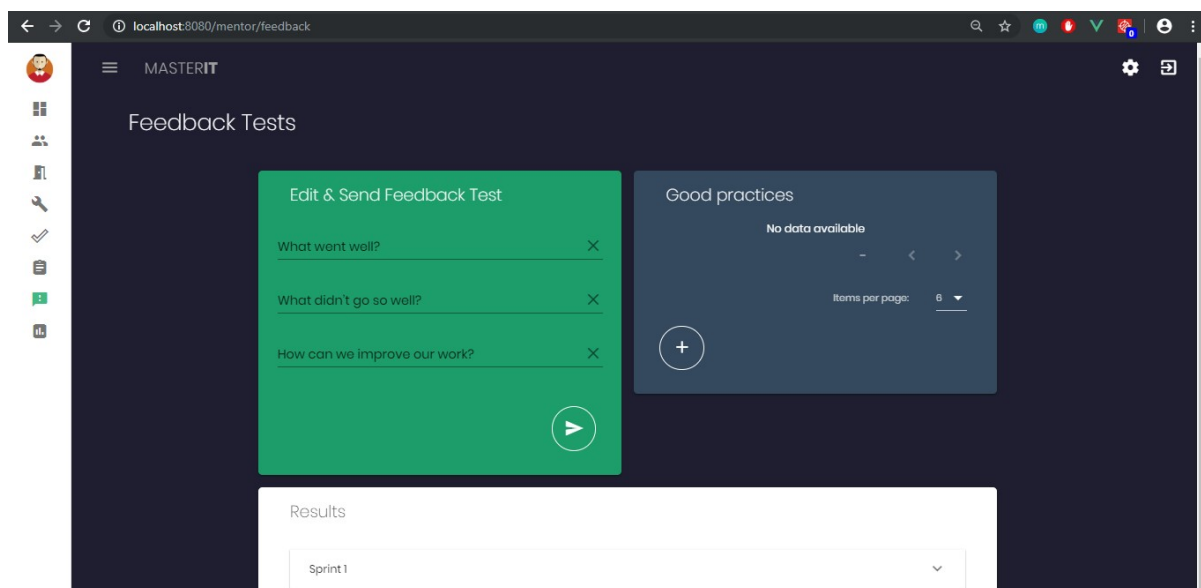


Figura 26 - Pagina Feedback Tests

Results

În această pagină (Figura 27), mentorii pot vedea rezultatele obținute de internii din echipa lor, rezultate sortate în funcție de tipul cursului.

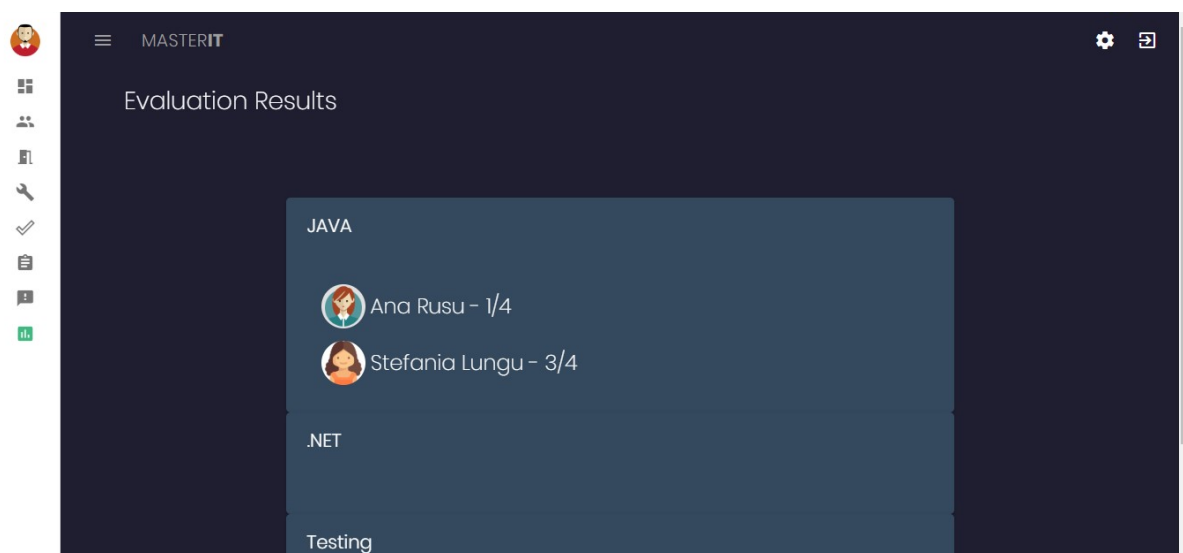


Figura 27 - Pagina Results

În continuare vor fi prezentate funcționalitățile din perspectiva unui intern.

Dashboard

Ca și în cazul mentorului său, în Dashboard (Figura 28) vor fi afișate detalii despre sprint-ul curent și următoarea ședință a echipei, cât și regulile de care trebuie să țină cont toți membrii (good practice-urile). Bineînțeles, internul nu are dreptul de a modifica aceste amănunte, deoarece au fost stabilite în prealabil împreună cu mentorul și restul echipei sale.

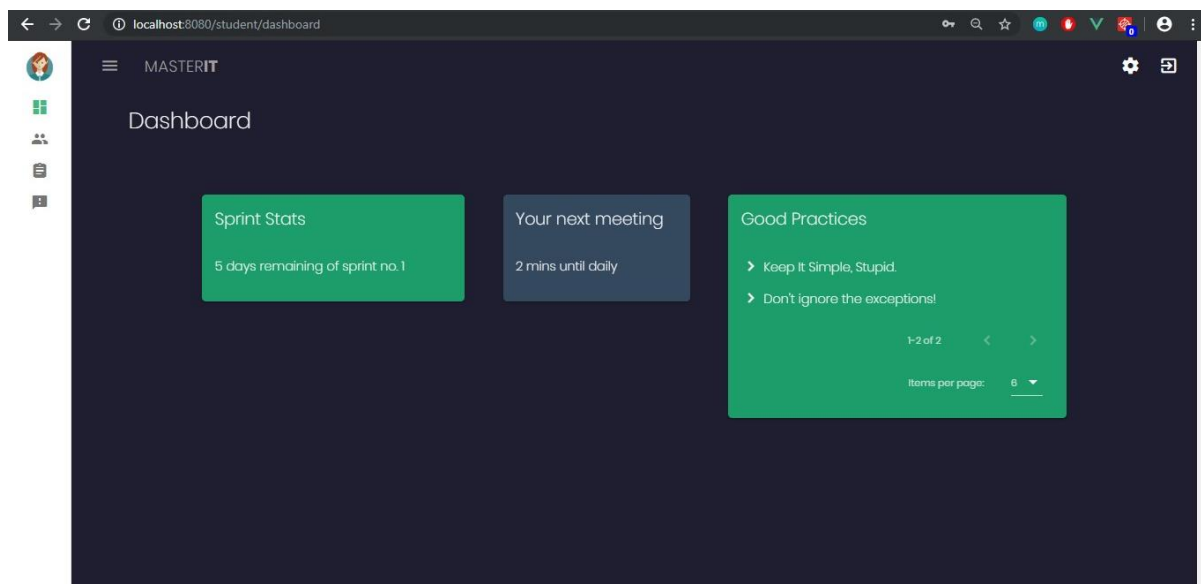


Figura 28 - Dashboard Intern

Team

De asemenea, un intern poate vizualiza echipa din care face parte. Fiecare membru al echipei are trecut și cursul pentru care s-a înscris la internship.

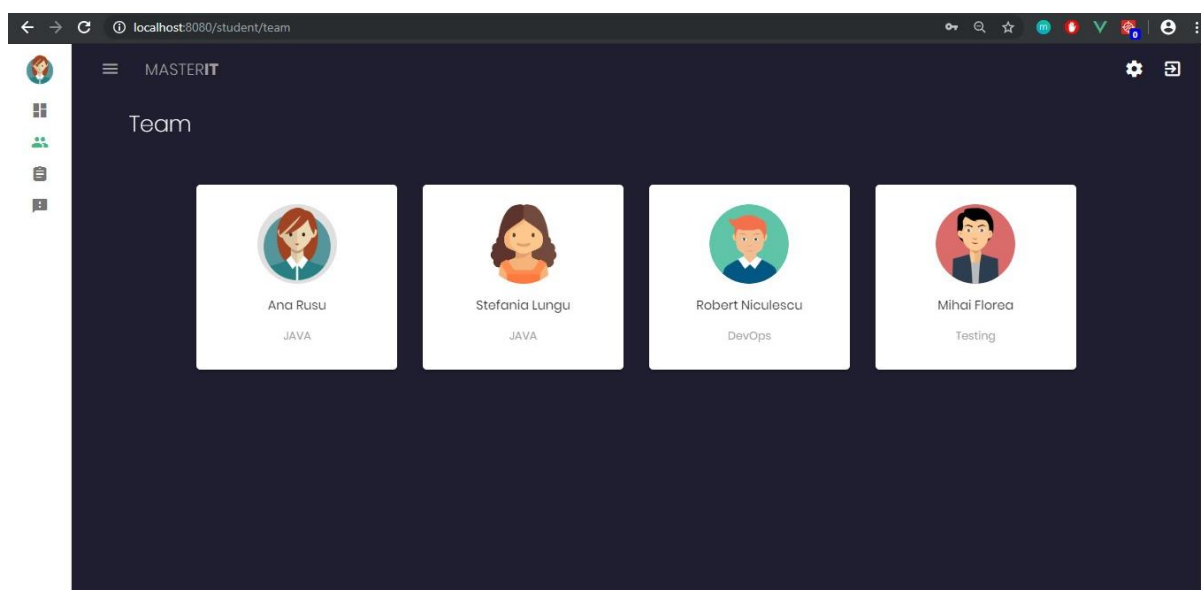
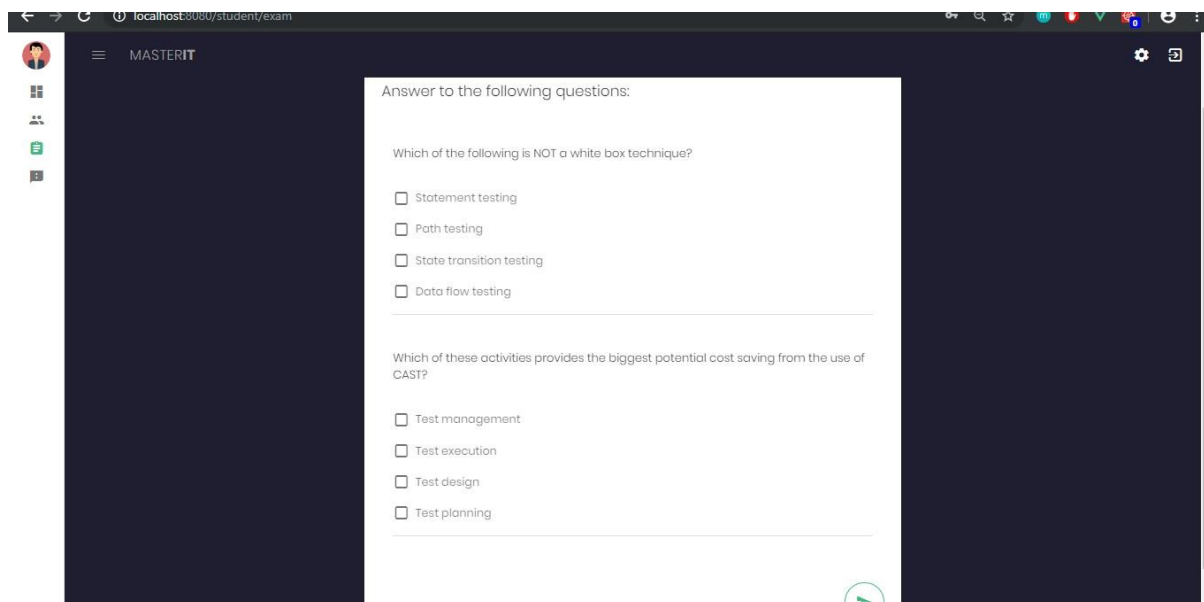


Figura 29 - Pagina Team

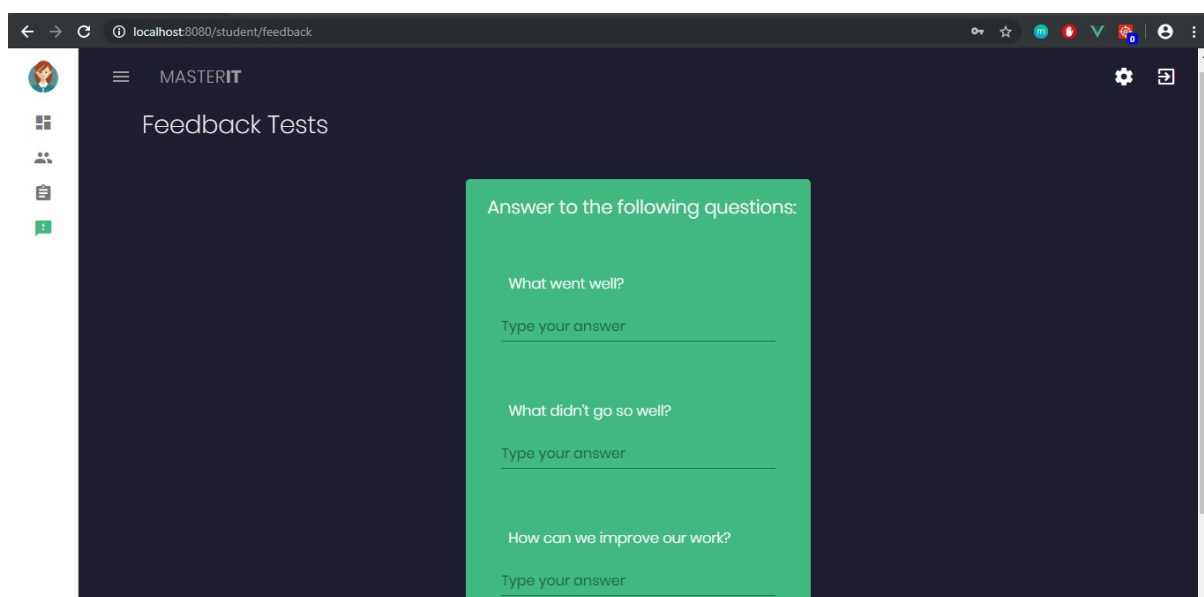
Evaluation (Figura 30) și Feedback tests (Figura 31)

Aceste două pagini vor afișa testul tehnic, respectiv cel de feedback de îndată ce vor fi disponibile.



The screenshot shows a web browser window with the URL `localhost:8080/student/exam`. The page is titled "MASTERIT" and contains a section for "Answer to the following questions:". There are two multiple-choice questions. The first question is "Which of the following is NOT a white box technique?" with four options: "Statement testing", "Path testing", "State transition testing", and "Data flow testing". The second question is "Which of these activities provides the biggest potential cost saving from the use of CAST?" with four options: "Test management", "Test execution", "Test design", and "Test planning". A green arrow icon is visible at the bottom right of the question area.

Figura 30 - Pagina Evaluation din contul internului



The screenshot shows a web browser window with the URL `localhost:8080/student/feedback`. The page is titled "MASTERIT" and contains a section for "Feedback Tests". There are three text input fields for feedback. The first field is labeled "What went well?" and the second field is labeled "What didn't go so well?". Both fields have a "Type your answer" label below them. The third field is labeled "How can we improve our work?" and also has a "Type your answer" label below it.

Figura 31 - Pagina Feedback Tests din contul internului

Concluziile lucrării

Pe de o parte, cu ajutorul aplicației MasterIT, internii au posibilitatea de a deprinde mai repede regulile metodologiilor Scrum și Agile, prezente în majoritatea firmelor de IT, se pot integra mai ușor atât în companie, cât și în echipă, își pot expune părerea în legătură cu fiecare sprint, ce a mers bine, ce nu a funcționat cum trebuie și ce ar trebui îmbunătățit pe viitor. Aceste lucruri le sporesc spiritul de echipă, simțul critic și prin sistemul de feedback și cel de testare tehnică descoperă în ce stadiu sunt cu adevărat, află ce trebuie îmbunătățit la ei înșiși, precum și la deprinderile necesare unui programator/tester de succes.

Pe de altă parte, aplicația oferă mentorilor un mod de a le organiza mai ușor timpul și sarcinile care vin odată cu această responsabilitate. Alegerea echipei, stabilirea notificărilor pentru ședințele Scrum, regulile good practice care sunt stabilite împreună cu membrii facilitează comunicarea dintre cele două părți și buna organizare a timpului. Știm bine că acel mentor, pe lângă toate acestea, este în primul rând un angajat al firmei, prin urmare, trebuie să ducă la bun sfârșit și sarcinile proprii asignate în cadrul echipei din care face parte.

Lucrând la această aplicație web am avut ocazia să implementez noțiunile învățate în cadrul facultății, am descoperit ce presupune arhitectura unui proiect, cum să îl construiesc astfel încât să fie scalabil, cu posibilitate de extindere în viitor. De asemenea, am învățat cât de importante sunt acele practici bune de care am tot pomenit în conținutul lucrării și cum poate o documentare foarte detaliată în prealabil să reducă volumul de muncă.

Datorită scalabilității, aplicația se poate extinde în viitor astfel:

- printr-un sistem de aproximare a numărului de puncte oferit următoarelor sprint-uri, astfel încât să fie măsurată viteza echipei;
- adăugarea unui serviciu terț de email prin care internii să primească ultimele noutăți de la mentorul lor;
- implementarea unui sistem prin care internii care se confruntă cu o problemă sau au o nelămurire să o poată adresa în cadrul aplicației și să primească răspuns de la unul dintre mentori sau chiar și colegi;
- implementarea unui Pomodoro timer cu ajutorul căruia internii ar putea să își mărească capacitatea de concentrare și productivitatea.

Bibliografie

- <https://agilemanifesto.org/>
- <https://www.todaysoftmag.ro/article/394/de-ce-ne-racim-gura-cu-agile>
- <https://www.pmoinnovations.com/blog/5-agile-characteristics>
- <https://www.scrumguides.org/>
- <https://www.scrum.org/resources/what-is-scrum>
- <https://www.sitepoint.com/developers-rest-api/>
- <https://www.mongodb.com/>
- <https://github.com/szokodiakos/typegoose>
- <https://hackernoon.com/what-is-vue-js-and-what-are-its-advantages-4071b7c7993d>
- <https://www.codeinwp.com/blog/angular-vs-vue-vs-react/>
- <https://devops.com/7-best-practices-new-mongodb-users-know/>
- <https://docs.hangfire.io/en/latest/>
- <https://c4model.com/#coreDiagrams>
- <https://medium.com/javascript-in-plain-english/implement-movie-app-with-vue-vuetify-axios-open-movie-database-api-d12290318cf9>
- <https://medium.com/@siddharthac6/json-web-token-jwt-the-right-way-of-implementing-with-node-js-65b8915d550e>
- <https://medium.com/@onejohi/building-a-simple-rest-api-with-nodejs-and-express-da6273ed7ca9>
- <https://www.toptal.com/nodejs/secure-rest-api-in-nodejs>
- <https://vuetifyjs.com/en/getting-started/quick-start>

- <https://www.bennadel.com/blog/2795-thinking-about-inversion-of-control-ioc-in-node-js.htm>
- <https://codeburst.io/understanding-solid-principles-dependency-injection-d570c15560ab>

Anexe

Anexa 1 – Agile și Scrum

Agile este un model de dezvoltare a produselor software bazat pe *Cele douăsprezece principii pentru software Agile*⁸ care, împreună cu cele 4 valori fundamentale, alcătuiesc *Manifestul Agile*⁹.

Principalele caracteristici ale metodologiei *Agile* sunt:

Abordare iterativă: dezvoltarea proiectelor *Agile* este structurată în perioade scurte de timp (1-4 săptămâni), numite *sprint*-uri. La sfârșitul unui număr stabilit de *sprint*-uri se prezintă progresul clientului, astfel că se pot face modificări și testări repetate și se re-prioritizează sarcinile.

Scalabilitate: metodologia permite scalarea proiectelor de dimensiuni mari, împărțirea acestuia în sarcini mai mici, permițând o mai bună organizare și coordonare a echipei și totodată creșterea productivității.

Adaptabilitate: clientul este implicat în procesul de producție astfel că echipa poate să răspundă schimbărilor în cerințe pe tot parcursul dezvoltării produsului.

Termen de livrare: *Agile* promite că fiecare *deadline* va fi respectat, astfel productivitatea nu va avea de suferit, iar progresul va fi vizibil după fiecare *sprint* încheiat.

Scrum este un framework construit peste modelul *Agile* și vine în completarea lui cu metode de eficientizare a modului de lucru și de creștere a colaborării dintre echipa de dezvoltare și client.

Rolurile într-o echipă *Scrum* sunt: *Product Owner*, *Development Team*, și *Scrum Master*.

Evenimentele *Scrum* (Figura 32) au un rol important în buna desfășurare a *sprint*-ului și monitorizarea progresului. Acestea sunt: *Sprint Planning*, *Daily Scrum*, *Sprint Review* și *Sprint Retrospective*.

⁸ *Twelve Principles of Agile Software* - <https://agilemanifesto.org/principles.html>

⁹ *Agile Manifesto* - <https://agilemanifesto.org/>

SCRUM FRAMEWORK

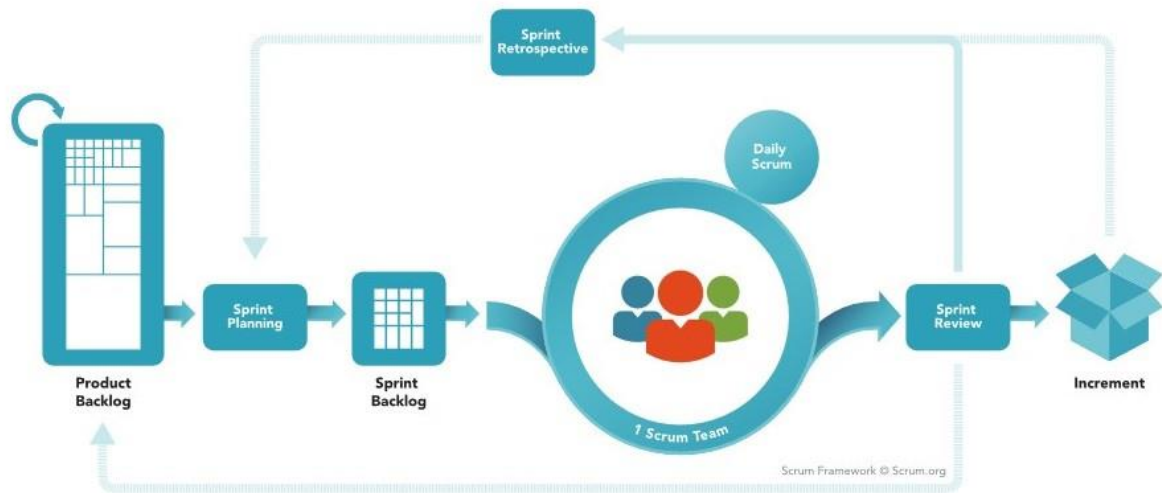


Figura 32 - Etapele unui sprint în manieră Scrum - <https://www.scrum.org/resources/what-is-scrum>

Anexa 2 – REST API

Un *API*¹⁰ reprezintă o colecție de funcții și metode de comunicare între diferite componente software.

*REST*¹¹ API este un set de funcții prin care dezvoltatorii de software pot face cereri HTTP pentru a primi și trimite date de tip XML sau JSON. Spre deosebire de SOAP, acesta este simplificat și ușor de implementat. Fiecare apel REST conține un verb HTTP (GET, POST, DELETE, PUT etc) care descrie modul cum se vor procesa de către partea de server a aplicației datele din URL-ul apelat cât și din corpul cererii HTTP.

Caracteristicile principale ale unui REST API:

Este bazat pe o arhitectură client-server

Este *stateless*: o cerere REST din partea clientului conține toate elementele necesare pentru ca serverul să trimită un răspuns valid. Odată cu îndeplinirea cererii, serverul nu păstrează informații despre acea interacțiune cu clientul.

Reprezentarea resurselor se face printr-un *Uniform Resource Identifier* (URI) care atribuie cererii tipul de resurse.

¹⁰ Application Programming Interface

¹¹ Representational State Transfer

Anexa 3 – Tehnologii utilizate pe partea de server

Pe partea de server, aplicația este scrisă în TypeScript, un limbaj de programare open-source dezvoltat de Microsoft. Este un superset sintactic al limbajului de programare JavaScript și permite scrierea de cod JavaScript într-o manieră potrivită pentru programarea orientată obiect. Spre deosebire de JavaScript unde variabilele nu au neapărat un tip anume (integer, string etc), în TypeScript datele sunt *strongly-typed*, evitându-se astfel erorile de compilare. De asemenea, fiind un limbaj de programare orientat-obiect, TypeScript oferă suport pentru implementarea claselor, interfețelor, moștenirilor.

În ceea ce privește reprezentarea datelor din baza de date pe server, am ales să folosesc, pe lângă TypeScript, Typegoose, o alternativă pentru *Mongoose*¹², adică un ODM (*Object Document Mapper*) specific atât pentru o bază de date NoSQL cât și pentru limbajul TypeScript. Acesta aduce nou faptul că modelul Mongoose corespunzător unui document din baza de date este sincronizat cu o clasă/interfață TypeScript cu aceleași proprietăți la care se adaugă decoratori speciali Typegoose, reducând substanțial codul și complexitatea acestuia.

NodeJS



Figura 33 - Logo NodeJS - <https://nodejs.org/en/>

NodeJS este un mediu de lucru pe server open-source, cross-platform, care folosește ca limbaj de programare JavaScript, implicit și TypeScript. Dezvoltatorii JavaScript au dorit să extindă acest limbaj de la unul care doar se executa în browser la unul care se poate comporta ca o aplicație reală, de sine stătătoare. Ca și un browser obișnuit, NodeJS are implementat

¹² <https://mongoosejs.com/>

motorul de rulare V8¹³ care interpretează codul JavaScript, îl transformă în limbaj mașină și îl execută mai rapid.

Cele mai importante caracteristici pentru care NodeJS este preferat de mulți dezvoltatori de software sunt:

- Este asincron și bazat pe evenimente: toate API-urile din librăriile NodeJS sunt asincrone, non-blocante.
- Foarte rapid: datorită motorului de rulare V8.
- Are un singur fir de execuție scalabil: cererile sunt servite de un singur *thread*, dar datorită faptului că mecanismul bazat pe evenimente este non-blocant, fiecare dintre cereri se execută cu succes (Figura 34).
- Are acces la pachete NPM (*Node Package Manager*): managerul de pachete conține o suită impresionantă de librării folosite pentru diverse implementări și disponibile prin download.

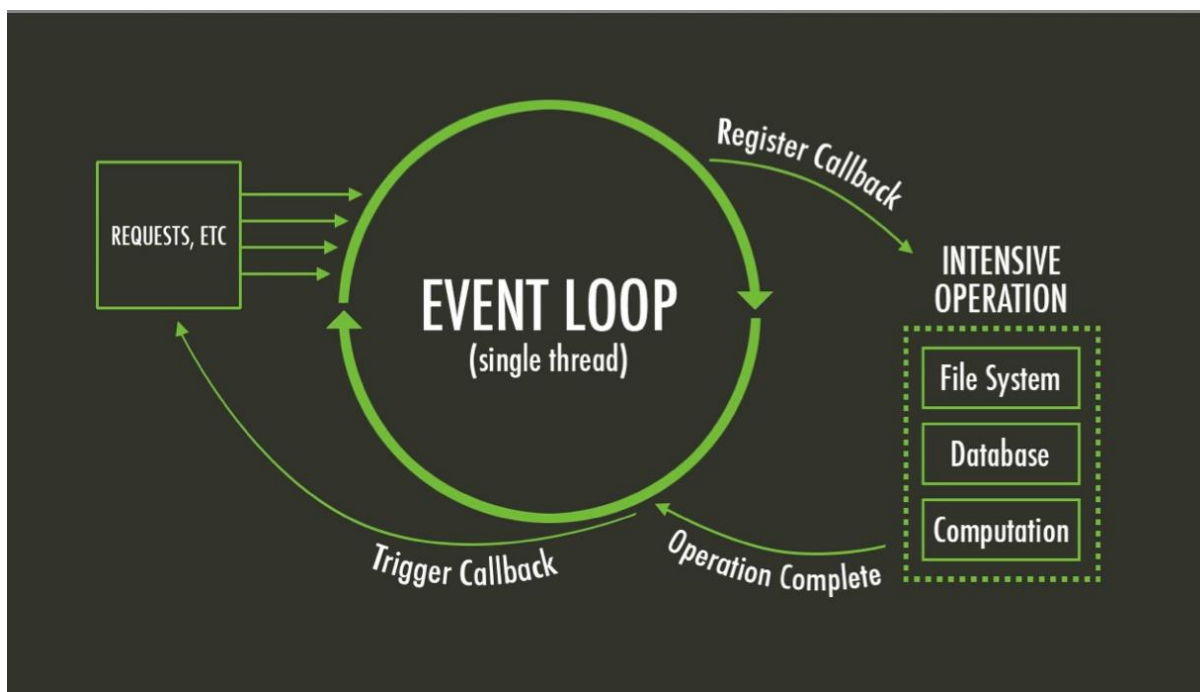


Figura 34 - Cum funcționează NodeJS - <https://codeburst.io/the-only-nodejs-introduction-youll-ever-need-d969a47ef219>

¹³ <https://v8.dev/>

NodeJS este utilizat de numeroase aplicații web foarte cunoscute, printre care se numără eBay, General Electric, GoDaddy, PayPal și Uber.

Framework-ul Express

Pentru a implementa REST API-ul într-o manieră cât mai elegantă am utilizat framework-ul Express, un framework minimalist, care simplifică implementarea unei arhitecturi de tip MVC¹⁴ și furnizează soluții pentru crearea și particularizarea rutelor, manipularea cererilor HTTP.

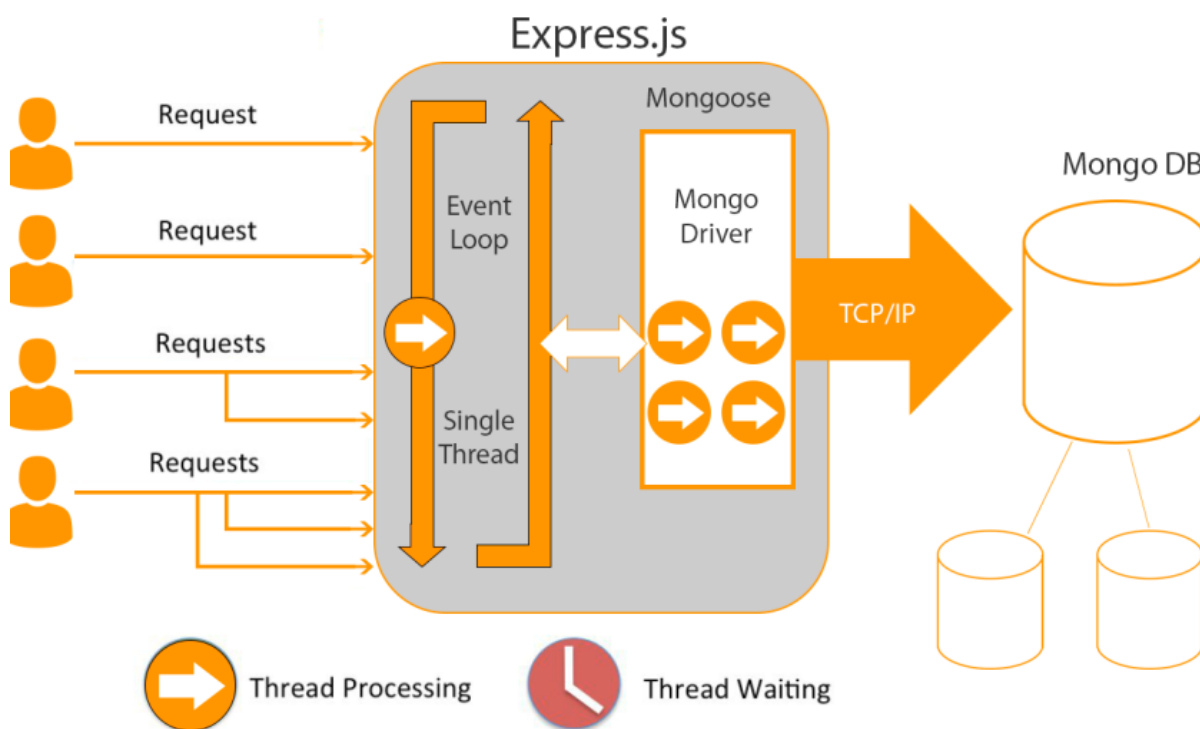


Figura 35 - Privire de ansamblu a serverului construit cu NodeJS, Express și conectat la o bază de date MongoDB - <https://binariks.com/blog/tools/express-js-mobile-app-development-pros-cons-developers/>

¹⁴ Model – View – Controller: <https://www.codecademy.com/articles/mvc>

Anexa 4 – Tehnologii utilizate pe partea de client

Vue.js



Figura 36 - Logo Vue.js - <https://vuejs.org/>

Pe partea de client a aplicației, am decis să folosesc un framework ce prinde tot mai mult teren în 2019 deoarece reușește să combine beneficiile aduse de framework-urile existente în producție (ReactJS și Angular), Vue.js. De asemenea, Vue.js are o dimensiune considerabil mai mică (18 – 21 kB), are o structură simplă care se pliază atât pe proiectele mici cât și pe cele complexe. Un alt plus ar fi documentația foarte detaliată de care dispune, documentație necesară celor care învață cum să construiască o aplicație web folosind Vue.js.

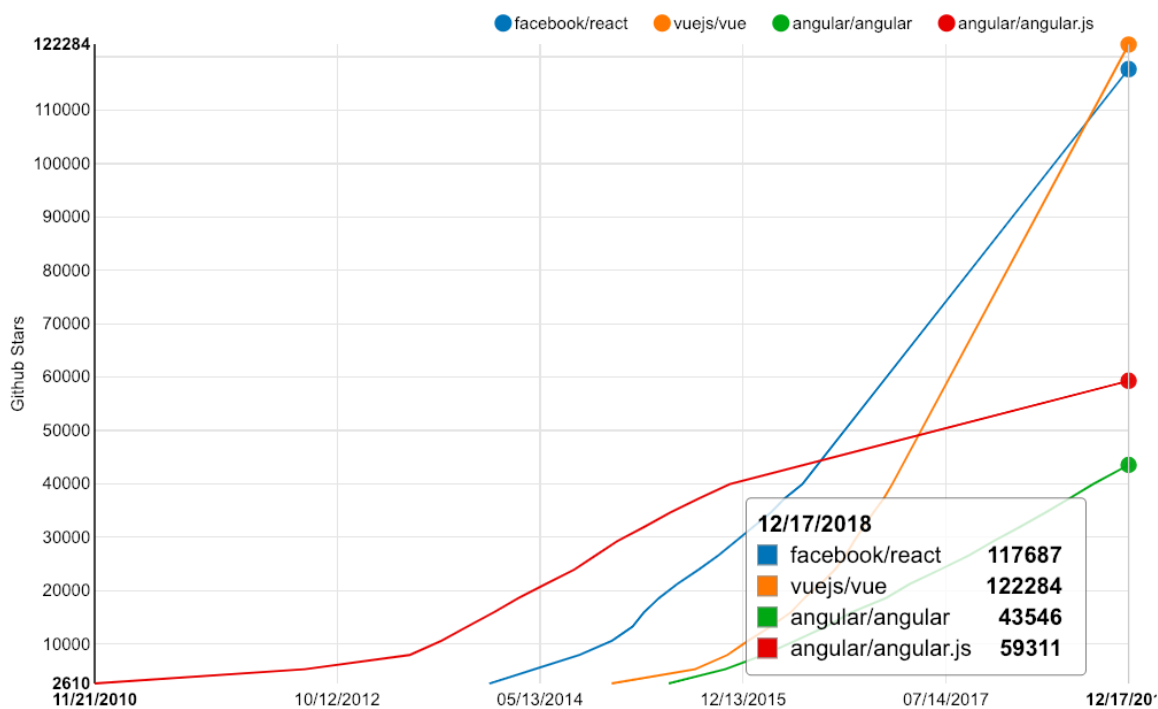


Figura 37 - Popularitatea framework-urilor pentru aplicații frontend - <https://www.codeinwp.com/blog/angular-vs-vue-vs-react/>

Așa cum se poate observa în Figura 37, Vue.js a crescut în popularitate considerabil, într-un interval de timp destul de scurt față de ceilalți competitori.

Vuetify



Vuetify

Material Component Framework

Figura 38 - Logo Vuetify - <https://vuetifyjs.com/>

Vuetify este un UI (*User Interface*) framework care îmbină specificațiile de la *Google Material Design*¹⁵ cu elementele definiției dintr-o aplicație creată cu Vue.js. Cu alte cuvinte, poate fi considerat un *wrapper* util pentru a structura și integra părți definiției într-o componentă Vue.js realizând totodată o interfață responsive și plăcută vizual.

¹⁵ <https://material.io/design/introduction/#principles>

Anexa 5 – MongoDB

Bazele de date relaționale au fost introduse în perioada anilor 1970 și permiteau stocarea informațiilor printr-un limbaj de interogare standard, SQL (*Structured Query Language*) ce urmărea 4 principii: ACID (Atomicitate, Consistență, Izolare, Durabilitate). Pentru a configura cerințele de stocare era nevoie de un sistem de management al bazelor de date relaționale (*RDBMS*¹⁶). La momentul respectiv, unitățile de stocare erau costisitoare iar aplicațiile nu necesitau scheme de date complicate.

Odată cu extinderea rapidă a aplicațiilor web și creșterea complexității acestora, volumul de date stocat s-a mărit enorm, la fel și complexitatea operațiilor efectuate în bazele de date. Mai mult decât atât, și numărul accesărilor acelor date a crescut considerabil, spre exemplu rețelele sociale, care au cunoscut un progres vizibil în ultima decadă au nevoie de o metodă eficientă de stocare a datelor astfel încât să ofere o experiență plăcută și fluidă utilizatorilor. Acesta este motivul pentru scăderea popularității bazelor de date relaționale și trecerea la o formă mai flexibilă de stocare a datelor: bazele de date nestructurate sau NoSQL.

Principalele beneficii aduse de bazele de date *Not Only SQL* sunt:

- Scalabilitate mare
- Calcul distribuit
- Costuri reduse
- Schemă flexibilă de date

Spre deosebire de SQL, NoSQL se bazează pe sistemul *BASE*¹⁷, sistem implementat de Erick Brewer, cel care a enunțat și teorema CAP¹⁸. Ca o consecință a teoremei, Brewer a descris faptul că se pot implementa 2 din cele 3 proprietăți, alegerea fiind făcută în funcție de cerințele sistemului. Sistemul BASE se concentrează pe îndeplinirea ultimelor două proprietăți, adică

¹⁶ Relational Database Management System

¹⁷ <https://www.3pillarglobal.com/insights/short-history-databases-rdbms-nosql-beyond>

¹⁸ Teorema CAP enunță faptul că orice sistem distribuit nu poate garanta următoarele 3 proprietăți în același timp:

1. Consistență (Consistency) – odată ce data este scrisă, fiecare cerere de citire va conține această dată
2. Disponibilitate (Availability) – baza de date este întotdeauna disponibilă
3. Toleranța partiționării (Partition Tolerance) – dacă o parte a bazei de date este indisponibilă, restul nu este afectat

renunță la consistența datelor (oferită de principiile ACID ale RDBMS), dar asigură o disponibilitate mai mare a datelor și toleranța la partiționarea acestora.

Tipuri de baze de date NoSQL:

- Cheie – Valoare (*key – value*): se folosește de o tabelă hash în care există id-uri unice și pointeri la datele propriu-zise.
- Orientate pe coloane (*column oriented*): au fost proiectate pentru a fi utilizare pe un volum mare de date partiționat pe mai multe mașini.
- Documente stocate (*document stored*): reprezintă colecții de obiecte de tip cheie – valoare. Documentele sunt date nestructurate salvate în JSON și permit imbricarea item-ilor, făcând interogările mai ușoare. Un exemplu de bază de date NoSQL document stored este MongoDB.
- Bazate pe grafuri (*graph based*): este o alternativă scalabilă a schemelor rigide SQL bazate pe tabele cu coloane și rânduri, permit partiționarea pe mai multe mașini.

Așa cum am menționat mai sus, MongoDB este o bază de date NoSQL bazată pe documente, deci fiecare înregistrare este de fapt un document JSON a cărui structură poate varia, indexarea și interogarea acesteia fiind realizate mai ușor, permițând accesul facil și rapid la date. Documentele sunt de fapt rândurile dintr-o tabelă SQL, iar cele din urmă sunt reprezentate de colecții în MongoDB.