



UNIVERSITATEA DIN
BUCUREȘTI

FACULTATEA DE
MATEMATICĂ ȘI
INFORMATICĂ



SPECIALIZAREA INFORMATICĂ

Lucrare de licență

APLICAȚIE WEB PENTRU GESTIONAREA EFICIENTĂ A ANGAJAȚILOR ȘI A SARCINILOR DE LUCRU

Absolvent

Alexandra-Diana Ciocan

Coordonator științific

Lect.dr. Natalia Gabriela Ozunu

București, iunie 2024

Rezumat

Un instrument de ajutor pentru orice echipa ce dorește să lucreze organizat este o aplicație de management a sarcinilor de lucru. Aceasta permite centralizarea datelor necesare îndeplinirii muncii de către fiecare membru al echipei și analiza modului de muncă atât individual, cât și la nivel de grup. Totuși, gestiunea datelor dintr-o astfel de aplicație trebuie să fie cât mai simplă pentru a nu distrage atenția de la proiect către lucruri administrative.

Din acest motiv, această lucrare se concentrează pe crearea unei astfel de aplicații cu integrarea unei funcționalități specializate în asignarea automată, într-un mod eficient și echitabil, a atribuțiilor.

Abstract

A helpful tool for any team that strives to work in an organized manner is a task management application. It allows users to centralise the data needed for each team member to carry out their work and analyse how the team works both individually and as a group. However, data management in such an application should be as simple as possible so as not to distract attention from the project to administrative things.

For this reason, this paper focuses on the creation of such an application with the integration of functionality specialised in the automatic assignment of tasks in an efficient and fair way.

Cuprins

1	Introducere	4
1.1	Motivație	4
1.2	Domenii abordate	4
1.3	Analiza competitorilor	4
1.4	Nevoi îndeplinite de aplicație	5
2	Preliminarii	6
2.1	Funcționalități principale	6
2.1.1	Actori	6
2.1.2	Cazuri uzuale de utilizare	6
2.2	Utilizarea tehnologiilor	7
2.2.1	Typescript&React	7
2.2.2	Java&Spring	8
2.2.3	PostgreSQL	8
2.3	Structura aplicației	8
3	Concluzii	9

Capitolul 1

Introducere

1.1 Motivație

Prin contactul meu de pana la momentual acutal cu mediul corporate și, prin extensie, cu metodologia Agile, am observat că o piedică în maximizarea performanței unei echipe o constituie asignarea haotică a sarcinilor de lucru. Aceasta rezultă într-o lipsă de echilibru în volumul de muncă, împreună cu posibilitatea neîncheierii sarcinilor până la data limită a livrării.

Astfel, aplicația centrală acestei lucrări își propune simplificarea gestionării proiectelor din cadrul unei companii și creșterea productivității prin automatizarea procesului de atribuire a sarcinilor de lucru. Utilizatorii cheie sunt managerii, care pot crea panouri cu sarcini de lucru și distribui eficient volumul de muncă, și angajații obișnuiți, care își pot vizualiza sarcinile zilnice și oferi noutăți legat de statusul lor.

1.2 Domenii abordate

Această lucrare are în vedere, în principiu, domeniul dezvoltării aplicațiilor web și al inteligenței artificiale. În cadrul proiectării aplicației, am folosit tehnologii mature precum Java & Spring pentru back-end, Typescript & React pentru front-end, și baze de date relaționale, prin intermediul PostgreSQL. [de adaugat detalii dupa implementarea agentului inteligent]

1.3 Analiza competitorilor

Investigând cele mai populare aplicații web de gestiune a sarcinilor de lucru în cadrul unei echipe, competitorii principali identificați sunt: Jira, Monday.com și Azure DevOps.

Fiecare dintre acestea excelează în puncte diferite. Jira este foarte bine adaptat metodologiei Agile, constituind un sistem bun de organizare a unui workflow, Monday.com

oferă căi de comunicare pentru echipă, iar Azure DevOps include chiar si feature-uri de version control, facilitând CI/CD.

Atunci când vine vorba de automatizare, Jira și Monday.com pun la dispoziție un sistem bazat pe triggeri. Utilizatorul trebuie să configureze o regula de forma "atunci când condiția ... este îndeplinită, execută ...". Astfel, e nevoie de efortul de analiză și decizie al utilizatorului pentru a configura toate aceste condiții. Azure DevOps, pe de alta parte, își axează posibilitățile de automatizare spre sarcinile manuale repetitive, în zona de CI/CD.

1.4 Nevoi îndeplinite de aplicație

Aplicația care este subiectul tezei își propune să construiască un algoritm inteligent care să scadă implicarea utilizatorului în procesul de decizie și analiză a informațiilor adunate, pentru a simplifica munca managerilor sau a Scrum Master-ilor, a crește echitabilitatea împărțirii volumului de muncă, și a crește per total mulțumirea și productivitatea în cadrul echipei.

Mai mult, această procesare a informației poate detecta și nivele neobișnuite de productivitate pentru un anumit angajat, pentru a fi ușor atât pentru acesta, cât și pentru manager, să depisteze semnale de alarmă pentru sănătatea mintală a angajului și să acționeze corespunzător.

[de adăugat descriere a fiecarui capitol după finalizarea lucrării]

Capitolul 2

Preliminarii

2.1 Funcționalități principale

2.1.1 Actori

Principalii actori ai cazurilor de utilizare sunt:

- Manager
- Employee
- Administrator

2.1.2 Cazuri uzuale de utilizare

- Manager:

1. În calitate de manager, aş vrea să adaug sarcini de lucru pentru echipa mea, cu detalii despre criteriile de acceptanță, prioritate, dificultate și termen limită, pentru ca acestea să fie cât mai clare pentru membri echipei.
2. În calitate de manager, aş vrea să pot vizualiza factorii de performanță precum disponibilitatea, volumul de muncă, calificarea, eficiența și atribuțiile fiecărui membru al echipei mele, pentru a evalua corect atribuirea de sarcini de lucru noi.
3. În calitate de manager, aş vrea să pot modifica factorii de performanță precum disponibilitatea, volumul de muncă, calificarea, eficiența și atribuțiile fiecărui membru al echipei mele, pentru ca aceștia să fie mereu actuali.
4. În calitate de manager, aş vrea să atribui automat sarcini de lucru, bazat pe detaliile sarcinii de lucru și pe factorii de performanță angajatului, pentru a le împărți cât mai echitabil și a maximiza productivitatea și șansele ca sarcina să fie îndeplinită până la termen.
5. În calitate de manager, aş vrea să pot atribui manual sarcini de lucru, bazat pe analiza mea asupra situației, pentru a ajusta eventualele erori ale recomandă-

rilor, a adapta panoul la noi informații relative la disponibilitate, sau a încuraja creșterea/scăderea productivității pentru un anumit membru al echipei

- Employee:

1. În calitate de angajat obișnuit, aş vrea să îmi vizualizez sarcinile de lucru împreună cu detalii despre criteriile de acceptanță, prioritate, dificultate și termen limită, pentru a îmi putea organiza activitatea zilnică.
2. În calitate de angajat obișnuit, aş vrea să pot actualiza statusul sarcinilor de lucru pentru ca managerul meu să le cunoască progresul.
3. În calitate de angajat obișnuit, aş vrea să fiu notificat atunci când managerul îmi atribuie o sarcina de lucru nouă, pentru a percepe mai ușor schimbări ale panoului cu sarcinile mele.

- Administrator:

1. În calitate de administrator, aş vrea să pot crea, actualiza sau elimina utilizatori și echipe, pentru a putea gestiona accesul la platformă și organizarea internă.
2. În calitate de administrator, aş vrea să pot atribui manageri echipelor create, pentru ca fiecare echipă să fie gestionată de un manager.

2.2 Utilizarea tehnologiilor

2.2.1 Typescript&React

Pentru partea de front-end a aplicației web am ales să utilizez Typescript ca limbaj și React ca bibliotecă, împreună cu componente reutilizabile din biblioteca de componente UI Ant Design.

Utilizarea limbajului Typescript, în defavoarea limbajului JavaScript, este explicată prin minimizarea erorilor cauzate de inconsistența adusă de faptul ca JavaScript este un limbaj „loosely typed” („dinamicamente tipat”) și de incertitudinea adusă de acest fapt. Astfel, prezența interfețelor și tipurilor de date bine stabilite facilitează dezvoltarea rapidă și stabilă.

În plus, biblioteca React aduce o perspectivă structurată dezvoltării interfeței, prin definirea de componente reutilizabile în cadrul aplicației. O parte din componentele UI sunt utilizate din biblioteca Ant Design, folosite prin aspectul vizual unitar și formal, acompaniat de o documentație bine pusă la punct. Un dezavantaj al abordării prin componente este comunicarea între acestea, motiv pentru care am folosit state managerul MobX.

2.2.2 Java&Spring

Pentru partea de back-end a aplicației web am optat pentru limbajul Java și framework-ul Spring și ecosistemul acestuia, având în vedere opțiunile robuste pentru gestionarea logicii de afaceri, a securității și a interacțiunilor cu baza de date.

Versiunea de Java utilizată este Java 21, pentru a avea acces la cele mai noi update-uri ale limbajului, cu cele mai noi feature-uri, precum clase Record, fire virtuale de execuție, și Sequenced Collections.

Versiunea de Spring folosită este 6.1.3, ce vine împreună cu Spring Boot 3.2.2, alături de alte dependențe, printre cele mai notabile numărându-se Spring Security și Jakarta Persistence API (JPA).

Cele mai majore utilități puse la dispoziție de Spring și de ecosistemul de dependențe sunt:

- securitatea asigurată prin generarea de JSON Web Tokens pentru autentificarea și autorizarea utilizatorilor și a cererilor HTTP trimise;
- crearea tabelelor și conectarea la baza de date facilitată de ORM-ul JPA;
- injectarea dependențelor prin constructorii claselor, în scopul reducerii codului boilerplate.

2.2.3 PostgreSQL

Pentru baza de date, am ales PostgreSQL pentru potențialul de scalabilitate, și are în vedere integritatea datelor, mai presus de tot, fapt foarte important pentru antrenarea modelului de inteligență artificială folosit.

2.3 Structura aplicației

[de adăugat și descris structura fișierelor după ce e finalizată]

Capitolul 3

Concluzii