Documentația proiectului

1. Scurtă descriere a proiectului

Acest proiect reprezintă o platformă web pentru managementul angajaților, al task-urilor și al cererilor de concediu într-o companie de software. Aplicația este gândită ca un sistem intern, utilizat de angajați, manageri și departamentul de resurse umane, fiecare având funcționalități diferite în funcție de rol.

Platforma permite autentificarea securizată a utilizatorilor, inclusiv autentificare cu doi factori (2FA) pentru managerul HR (admin), gestionarea task-urilor și a angajaților, trimiterea și aprobarea cererilor de concediu, notificări prin email și notificări interne în aplicație și o secțiune de statistici, menită să ofere o imagine de ansamblu asupra activității angajaților și a task-urilor.

Aplicația este dezvoltată modular, respectând principiile programării orientate pe obiecte și o arhitectură pe layere, ceea ce permite extinderea ușoară a funcționalităților în viitor.

2. Limbaje și tehnologii folosite

Backend-ul aplicației este realizat în Java, folosind framework-ul Spring Boot. Pentru securitate este utilizat Spring Security, împreună cu autentificare pe bază de JWT. Persistența datelor este realizată cu ajutorul JPA și Hibernate, iar baza de date utilizată este MySQL.

Pentru trimiterea de email-uri (coduri OTP pentru 2FA și notificări legate de cererile de concediu) este folosit Java Mail Sender. Aplicația backend expune un API REST consumat de frontend.

Frontend-ul este realizat folosind TypeScript, React și Next.js, iar pentru stilizare este utilizat Tailwind CSS. Interfața este gândită să fie modernă, intuitivă și responsive.

3. Cerințe funcționale

Aplicația permite înregistrarea și autentificarea utilizatorilor. Autentificarea se face pe bază de email și parolă, iar pentru managerul HR (admin) pentru un nivel suplimentar de securitate este implementată autentificarea cu doi factori (2FA), prin trimiterea unui cod OTP pe email.

Utilizatorii aplicației sunt gestionați pe bază de roluri bine definite, fiecare rol având acces la funcționalități specifice. Angajații pot vizualiza task-urile care le-au fost atribuite, pot actualiza statusul acestora și pot trimite cereri de concediu. Managerii au dreptul de a crea, modifica, șterge și atribui task-uri către angajați, însă nu au acces la informații detaliate despre echipa lor sau la date salariale. Managerul cu rol de HR are acces extins la datele angajaților, la informațiile salariale, precum și la gestionarea cererilor de concediu și a contractelor de muncă.

Aplicația oferă un modul complet de gestionare a task-urilor, care acoperă întregul flux, de la creare și atribuire, până la actualizarea statusului. Accesul la aceste operații este restricționat în funcție de rol, astfel încât doar managerii pot efectua acțiuni critice precum crearea, modificarea sau ștergerea task-urilor.

Un modul esențial al aplicației este cel de gestionare a cererilor de concediu. Angajații pot trimite cereri de concediu prin intermediul aplicației, iar acestea pot fi aprobate sau respinse exclusiv de către managerul HR. În momentul aprobării sau respingerii unei cereri, angajatul primește automat o notificare prin email. De asemenea, la trimiterea unei cereri de concediu, managerul HR este notificat direct în aplicație.

Aplicația include și un calendar colaborativ, care permite vizualizarea concediilor aprobate și a ocupării unui loc fizic la birou, contribuind la o mai bună organizare a activității la nivelul companiei.

Gestionarea contractelor de muncă este realizată exclusiv de către managerul HR. Acesta poate crea și modifica contracte, iar toate modificările efectuate sunt salvate în sistem, asigurând consistența și istoricul datelor contractuale.

4. Cerințe non-funcționale

Aplicația este securizată și respectă bunele practici în ceea ce privește autentificarea și autorizarea utilizatorilor. Codul este structurat clar, pe layere (controller, service, repository), și respectă principiile OOP.

Aplicația este ușor de întreținut și de extins, astfel încât să poată fi adăugate funcționalități noi fără modificări majore în codul existent. Interfața este intuitivă și ofere o experiență de utilizare plăcută.

Performanța aplicației este una acceptabilă, cu timpi de răspuns rezonabili pentru operațiile uzuale.

5. Design Patterns utilizate

Strategy

Problema identificată apare în momentul în care aplicația trebuie să calculeze salariul unui angajat în moduri diferite, în funcție de context, de exemplu salariu brut sau salariu net. Implementarea directă a acestei logici într-o singură clasă ar duce la un cod greu de întreținut și extins.

Pattern-ul Strategy rezolvă această problemă prin separarea algoritmilor de calcul al salariului în strategii diferite, care pot fi schimbate dinamic, fără a modifica logica principală a aplicației.

În implementare, există o interfață de strategie pentru calculul salariului, care definește o metodă de calcul. Pentru fiecare tip de salariu (brut, net) există o implementare concretă a acestei interfețe. În funcție de tipul de salariu care trebuie calculat, aplicația selectează și utilizează strategia corespunzătoare. Această abordare permite adăugarea unor noi tipuri de calcul al salariului fără modificări majore în codul existent.

Observer

Problema identificată este necesitatea notificării automate a unor componente atunci când apare o modificare importantă în sistem, cum ar fi trimiterea sau procesarea unei cereri de concediu sau actualizarea datelor contractuale ale unui angajat. Pattern-ul Observer permite notificarea automată a mai multor componente atunci când are loc un astfel de eveniment, fără a crea dependențe directe între ele.

În aplicație, acest pattern este implementat folosind mecanismul de evenimente din Spring și este utilizat pentru notificările interne din aplicație, trimiterea de email-uri la aprobarea sau respingerea cererilor de concediu, precum și în momentul salvării modificărilor aduse contractelor de muncă.

Singleton

Problema identificată este necesitatea existenței unei singure instanțe pentru anumite componente ale aplicației, precum service-urile sau clasele de configurare. Pattern-ul Singleton rezolvă această problemă prin asigurarea faptului că există o singură instanță a clasei pe durata rulării aplicației. În cadrul aplicației, acest pattern este implementat implicit prin intermediul Spring, deoarece bean-urile sunt Singleton by default.

6. Tutorial de rulare a aplicației

Pentru a rula aplicația, este necesar să fie instalate câteva componente de bază. În primul rând, este nevoie de Java JDK (versiunea 17 sau mai nouă). De asemenea, este necesar un server de baze de date MySQL.

După instalarea MySQL, trebuie creată o bază de date goală. Tabelele vor fi create automat de către Spring Boot, pe baza entităților JPA, dacă este setată corespunzător proprietatea `spring.jpa.hibernate.ddl-auto` (de exemplu, `update`). Nu este necesară scrierea manuală a scripturilor SQL pentru crearea tabelelor, decât dacă se dorește un control mai detaliat.

Pentru backend, proiectul se deschide într-un IDE precum IntelliJ IDEA, se configurează fișierul `application.properties` cu datele de conectare la baza de

date și cu datele contului de email folosit pentru trimiterea mesajelor. Aplicația se pornește ca un proiect Spring Boot.

Pentru frontend, este necesară instalarea Node.js. După instalare, se rulează comanda `npm install` pentru instalarea dependențelor, urmată de `npm run dev`. Aplicația va fi accesibilă din browser.

7. Scurt README

Această aplicație este o platformă web destinată managementului angajaților, task-urilor și cererilor de concediu într-o companie de software.

Aplicația oferă autentificare securizată, roluri bine definite, notificări prin email și în aplicație, precum și o interfață modernă.

Pentru rulare, este necesară configurarea backend-ului Spring Boot și a frontend-ului Next.js, conform pașilor descriși în documentație.

Proiectul este gândit astfel încât să poată fi extins cu ușurință, inclusiv cu funcționalități suplimentare de raportare și statistici.