

Cuprins

Introducere	3
Informații generale	3
Tema	3
Motivație	4
Gradul de noutate	4
Obiectivele generale	5
Funcționalități propuse	5
Metodologia folosită	6
Descrierea soluției	6
Structura lucrării	6
Contribuții	7
Capitolul 1 – Descrierea problemei	8
Capitolul 2 – Abordări anterioare	8
harvestapp.com	8
oraclecloud.com/time	8
timeneye.com	9
Concluzie	9
Capitolul 3 – Descrierea soluției	9
Prezentarea aplicației	9
Arhitectura Sistemului	10
Controller	11
Service	12
Repository	12
Model	12
Arhitectura bazei de date	14
Modul de funcționare al aplicației	15
Algoritmi dezvoltați	25
Algoritmul de inserare al intervalelor	25
Algoritmul de spargere al intervalelor	26
Concluziile lucrării	27
Dezvoltare actuală	27
Dezvoltare ulterioară	28
Anexe	28
Anexa 1 - Java	28
Anexa 2 - Spring framework	28
Anexa 3 - Dependency Injection și Inversion of Control	29

Anexa 4 - Beneficiile utilizării framework-ului Spring	29
Anexa 5 - Spring Boot	30
Anexa 6 - Maven	30
Anexa 7 - Orm, JPA, Hibernate	31
Anexa 8 - Arhitectura ORM	31
Anexa 9 - Adnotari	32
Anexa 10 - JWT	32
Anexa 11 - Librăria React	35
Anexa 12 - Framework-ul Material-UI	35
Bibliografie	36

Introducere

Informații generale¹

O foaie de pontaj (sau o foaie de timp) este o metodă de înregistrare a valorii timpului petrecut al unui lucrător pentru fiecare lucrare. În mod tradițional, o foaie de hârtie cu datele aranjate în format tabelar, o foaie de pontaj este acum adesea un document digital sau foaie de calcul. Cardurile de timp ștampilate de ceasurile de timp pot servi ca o foaie de pontaj sau pot furniza datele pentru a umple una. Și acestea sunt adesea digitale. Foile de timp au intrat în uz în secolul al XIX-lea ca și cărți de timp.¹

Urmărirea timpului poate reduce costurile în trei moduri:

1. · prin efectuarea mai eficientă a procesării salarizării;
2. · prin creșterea vizibilității costurilor, astfel încât să le puteți reduce și prin automatizarea facturării;
3. · facturări.

Urmărirea timpului poate crește veniturile prin automatizarea facturării, ceea ce tinde să ușureze o companie și să obțină facturi corecte pentru toate orele lucrate de personalul consultant. Acest lucru accelerează plata și elimină dificultățile întâmpinate de de procesul manual de creare de facturi. Prin scăderea costurilor în trei moduri și creșterea veniturilor într-un singur mod, tehnologiile de gestionare a timpului de lucru care sunt bazate pe web pot îmbunătăți starea companiilor.

Tema

Tema aleasa este reprezentata de o aplicație web care automatizează procesul de pontare a angajaților dintr-o firmă, minimizând activitatea utilizatorului în vederea generării de rapoarte precum și reducerea spațiului folosit de către baza de date printr-o noua abordare a problemei legate de stocarea liniilor dintr-un tabel de pontaj, abordarea problemei fiind bazata pe stocarea pe intervale a liniilor din tabelul de pontaj. Atenția fiind împărțita în egala măsura între interacțiunea minima a utilizatorului pe aplicație și stocarea eficienta în baza de date a liniilor din tabelul de pontaj.

¹ <https://en.wikipedia.org/wiki/Timesheet>

Motivație

Fiindcă nevoia unei astfel de abordări în vederea automatizării procesului de pontare în interiorul companiilor este una reală și de actualitate, am ales această temă deoarece aplicația intitulată „Chronos” poate fi considerată un pas important către noile generații de aplicații de pontaj bazate pe o interacțiune minimă a utilizatorului în vederea generării de rapoarte.

Aplicația poate fi considerată un început de afacere care poate fi dezvoltată ulterior prin închirierea, actualizarea și mentenanța acestui produs. Prin faptul că se încearcă dezvoltarea unei aplicații cât mai simple din punct de vedere al utilizatorului consider că acest proiect poate să atragă potențiali clienți sau dezvoltatori.

În prezent, companiile în proporție de 80% - 90% folosesc aplicații web pentru gestionarea orelor lucrate și generarea de rapoarte provenite de la angajații acestora. Piața din industria IT aduce o mulțime de soluții în acest sens, dar foarte puține dintre acestea oferă un produs extrem de simplu de utilizat din perspectiva utilizatorului și care să minimizeze activitatea acestuia pe aplicație. Din punct de vedere al clientului, aplicația propune o soluție elegantă la această problemă prin conferirea unei interfețe grafice intuitive și prin salvarea implementarea anumitor mecanisme capabile să reducă efortul utilizatorului la minim.

Deoarece aplicația care își propune automatizarea procesului de pontaj precum și minimizarea efortului utilizatorului în vederea generării de rapoarte bazate pe datele completate de acesta nu necesită acces deosebit la resursele sistemului de operare și se vrea ca această aplicație să fie accesibilă de pe orice tip de dispozitiv, consider că folosirea unei aplicații web pentru a realiza tema aleasă este o abordare potrivită.

Gradul de noutate

Tema proiectului este una de actualitate, aplicațiile de pontaj fiind deja folosite de către companii de o perioadă îndelungată de timp. Majoritatea proiectelor au în spate o stocare în baza de date a liniilor din tabela de pontaj, fiecare înregistrare din baza de date reprezentând o linie din raport precum și o interfață dificil de înțeles de către utilizator, fiind nevoie de un manual complex de utilizare pentru a prezenta modul în care aplicațiile de acest fel funcționează.

Obiectivele generale

Asa cum am precizat mai sus, multe dintre aplicațiile de pontaj salvează un număr mare de înregistrări în baza de date ceea ce devine o problema pentru companiile mari care au un număr imens de angajați. Cu cat baza de date este mai încărcata cu atât răspunsurile primite de la aceasta vor fi mai lente rezultant o încetinire a sistemului. O alta problema majora întâlnită la acest tip de aplicație este gradul de complexitate în utilizarea acesteia.

Principalele obiective pe care aplicația trebuie sa la îndeplinească sunt :

1. reducerea numărului de înregistrări din baza de date;
2. crearea unei interfețe intuitive ușor de folosit pentru utilizator;
3. minimizarea activității utilizatorului pentru generarea de rapoarte.

Funcționalități propuse

Pe lângă faptul că aplicația facilitează automatizează procesul de pontare a orelor de carte angajații companiilor prin implementarea unei soluții de actualitate, îmi propun ca prin această lucrare să rezolv și problemele observate în timpul investigării aplicațiilor asemănătoare existente.

Printre problemele descoperite în timpul cercetării se numără:

- imposibilitatea setării unui proiect implicit pentru un utilizator – exemplu:
In general un angajat al unei companii lucrează la un singur proiect, iar dacă acel proiect se setează ca fiind implicit pentru acel utilizator aplicația va pontă automat orele pentru luna curentă bazându-se pe proiectul setat
- imposibilitatea pontării a doua sau mai multe proiecte în același timp – exemplu: Doresc printr-o singura operație de adăugare să pontez orele lucrate pe doua proiecte. In mod normal aplicațiile standard nu-ți vor permite să faci acest lucru fiind necesare doua operații de adăugare. Problema rezolvată prin implementarea unui mecanism de adăugare ce suportă adăugarea simultană a mai multor proiecte
- interfețe neintuitive – Rezolvată prin crearea unei interfețe ușor de utilizat
- numărul mare de înregistrări în baza de date – Rezolvată prin implementarea unui mecanism ce salvează zilele pontate sub forma de interval

Metodologia folosita

Aplicația a fost dezvoltată în decurs a un an, iar modul de lucru a fost unul segmentat în intervale de timp alocate pentru dezvoltarea serverului de backend respectiv serverului de frontend. Inițial am început implementarea serverului de backend, adăugând câteva funcționalități de baza, iar după ce finalizam de implementat funcționalitatea, am implementat partea de frontend corespunzătoare acelei funcționalități. Pentru majoritatea mecanismelor implementate pe partea de backend s-au creat și unit teste pentru a asigura o buna funcționalitate a algoritmilor.

Pe partea de cercetare s-a luat ca principal punct de reper aplicația de pontaj furnizată de către compania Oracle, studiind în detaliu modul prin care aceștia au implementat principalele lor mecanisme din acel program. Cea mai dificilă etapă fiind găsirea unei soluții pentru stocarea datelor din tabela de pontaj și crearea algoritmului de spargere al intervalelor în momentul în care se dorește a se face operații mai complexe pe intervalele din aplicație.

Descrierea soluției

Principale probleme ale aplicațiilor de pontaj constau în salvarea în baza de date a liniilor din tabela de pontaj pentru ca fiecare zi pontată avea asociată câte o înregistrare în baza de date și interfața acestora obliga utilizatorul să facă multe operații pentru a genera rapoarte rezultând astfel o irosire a timpului de către angajații companiilor. Pentru problema numărului excesiv de intrări în baza de date am hotărât stocarea zilelor pontate în intervale, spre exemplu un interval poate conține una sau mai multe zile ceea ce reduce semnificativ numărul de înregistrări. În cazul cel mai întâlnit la angajații companiilor pentru pontarea unei luni se vor folosi patru intrări în baza de date (asta dacă se vor stoca zilele sub forma de intervale) pentru stocarea datelor în comparație cu cele 22 de intrări utilizate pentru algoritmi care stochează o zi sub forma unei înregistrări. Iar pentru partea cu generare de rapoarte am creat o interfață sugestivă ce minimizează numărul de operații făcute de utilizator în vederea creării rapoartelor. De exemplu pentru cazul general când utilizatorul lucrează o luna întreagă la un sigur proiect este nevoie de o logare și o apăsare a unui buton pentru generarea de rapoarte, deoarece arhitectura proiectului permite stocarea unui proiect principal pentru fiecare persoană ce utilizează aplicația.

Structura lucrării

În Capitolul 1 este prezentat un studiu personal asupra aplicațiilor deja existente pe piață din întreaga lume. Sunt analizate elementele prezentate în fiecare aplicație, soluțiile curente și motivele pentru care am ales să construiesc o aplicație cu îmbunătățiri în acest domeniu.

Capitolul 2 prezintă modul de funcționare al aplicației precum și arhitectura acesteia
Capitolul 3 sunt descriși algoritmi dezvoltați în vederea lucrării cu intervale

Contribuții

Pentru o buna funcționare a aplicației precum și îndeplinirea scopului propus pentru aceasta au fost implementate diverse mecanisme :

- - Pe partea de server:
 - · mecanism de autentificare bazata pe JWT ce permite o autentificare securizata;
 - · mecanism de resetare de parola bazata pe JWT ce expira după o anumita perioada setata sau după ce s-a efectuat cu succes operație de schimbare de parola;
 - · mecanism de prelucrare a intervalelor folosit pentru ștergere, adăugare, modificare a înregistrărilor din baza de date;
 - · mecanism de separare a intervalelor în momentul în care doua sau mai multe intervale se intersectează și se observa ca este nevoie de o separare a acestora;
 - · mecanism de adăugare de imagini și de salvare a acestora;
 - · mecanism de generare a unui tabel de pontaj în format PDF;
 - · mecanism de calculare dinamica a anumitor date cum ar fi: zile de concediu ramase, zile de concediu luate, ore nelucrate din luna curenta, ore lucrate în luna curenta, ore lucrate pe anumite proiecte în luna curenta;
 - · mecanisme de validare a datelor primite de pe frontend;
 - · mecanism de blocare a utilizatorului după un număr prestabilit de încercări de autentificare eșuate.
- Pe partea de client:
 - · mecanism de verificare a inputului dat de utilizator înainte de a se face un apel la server
 - · animație pentru partea de logare
 - · mecanism de salvare a token-ului în cookie astfel încât userul sa rămână autentificat o perioada de timp chiar dacă a redeschis browser-ul
 - · mecanism de randare a pdf-ului primit în format blob de la server

Capitolul 1 – Descrierea problemei

Pe piața actuala exista multe aplicații de pontaj dar încă niciuna nu a reușit sa imbine un sistem prietenos cu utilizatorul și o modalitate eficienta de a stoca înrările în baza de date. Majoritatea celor existente au o interfață greu de utilizat și nu include o stocarea a datelor eficienta în baza de date precum și posibilitatea de pontare pe mai multe proiecte în interiorul aceleiași operațiuni, fiind necesare noi operațiuni de pontare specifice pentru fiecare proiect. O alta problema importanta care apare la acest tip de aplicație este faptul ca având o stocare nu foarte buna a înregistrărilor în baza de date, după un număr mare de înregistrări aplicațiile tind sa se miște foarte greu . Mai jos sunt prezentate câteva aplicații pe care le-am considerat relevante în momentul în care am studiat piața actuala.

Capitolul 2 – Abordări anterioare

După cum am specificat și în capitolul 1, pe piața actuala exista multe astfel de aplicații capabile sa genereze rapoarte de pontare ale orelor lucrate de angajați. Făcând cercetare pe aceasta tema am reușit sa analizez 3 dintre cele mai utilizate aplicații de pontaj din industrie reușind sa surprind câteva caracteristici bune și câteva slabe.

harvestapp.com

În opinia mea, din toate aplicațiile investigate, harvestapp.com este una care folosește stocarea pe intervale și are o interfață grafică relativ ușor de folosit dar care necesita un număr mare de operații în vederea generării de rapoarte. Caracteristicile pe care aceasta aplicație le oferă sunt:

- stocarea înregistrărilor pe intervale de o săptămână asociate pe cate un proiect
- generarea diverselor rapoarte, sau facturi
- calculul integral al orelor lucrate pe săptămâni pentru fiecare proiect în parte
- calculul total al orelor lucrate pe toate proiectele unui angajat
- interfața grafică relativ ușor de folosit

Aceasta aplicație, chiar dacă nu are cele mai multe funcționalități, atrage prin design-ul atractiv și prin modul de interacționare cu utilizatorul, aspect care nu este întâlnit în totalitate la aplicațiile prezentate mai jos

oraclecloud.com/time

Majoritatea aplicațiilor nu surprind printr-un design atractiv. În opinia mea aceasta aplicație este exemplul perfect ce surprinde un design neintuitiv din perspectiva utilizatorului precum și nivelul de dificultate pentru a crea un pontaj pe durata unei luni. OracleCloud.com

care oferă o gama larga de servicii printre care se număra și serviciu de pontare de carte angajații companiilor. Caracteristicile principale surprinse la aceasta aplicație sunt :

- generarea diverselor rapoarte, sau facturi
- calculul integral al orelor lucrate pe săptămâni pentru fiecare proiect în parte
- calculul total al orelor lucrate pe toate proiectele unui angajat
- centralizarea datelor

timeneye.com

Aceasta este o aplicație ce folosește o stocare ineficienta în baza de date prin faptul ca fiecare zi din săptămână are asociata cate o înregistrare. Interfața grafica este ușor de folosit oferind și rapoarte în timp real pentru anumite elemente cum ar fi: proiecte, ore lucrate, zile de concediu, concediu medical.

Caracteristicile care se regăsesc în aceasta aplicație sunt:

- generarea diverselor rapoarte
- afișarea în timp real a rapoartelor legate de proiecte, ore lucrate, zile de concediu, concediu medical
- interfață grafica intuitive

Concluzie

Pe parcursul investigării pieței în acest domeniu nu am descoperit o aplicație care să îmbine un sistem de pontaj cu o interfață ușor de folosit de carte utilizatori care sa se focuseze pe o activitate minima a utilizatorului pentru a ajunge la generarea de rapoarte, cu axare precisa pe numărul minim de intrări în baza de date și cu posibilitatea de a pontă pe mai multe proiecte simultan în decursul unei sigure operații.

Capitolul 3 – Descrierea soluției

Prezentarea aplicației

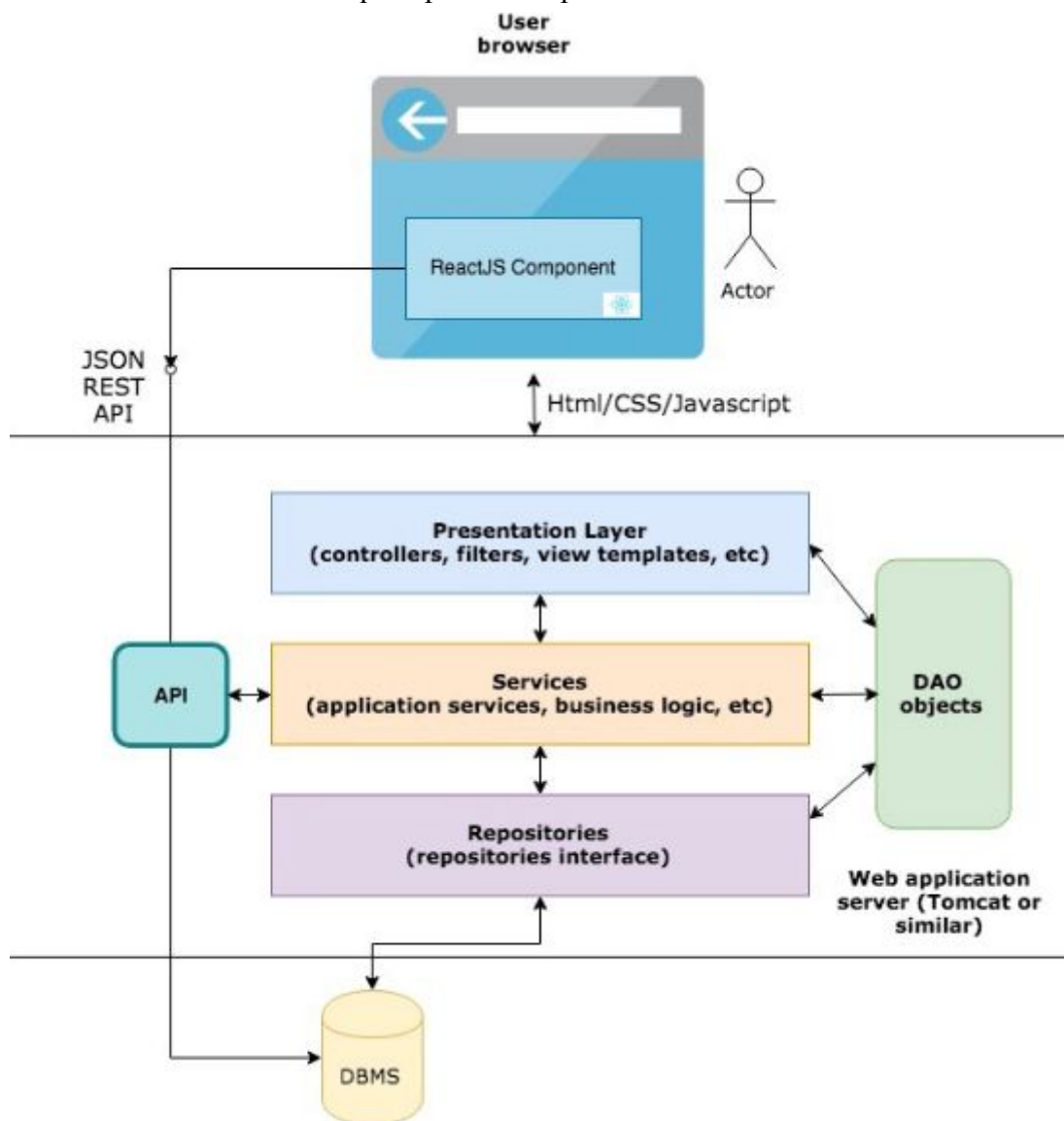
Această lucrare are ca obiectiv crearea unei aplicații web cu numele “CHRONOS” pentru gestionarea orelor de pontaj ale angajatorilor dintr-o companie. Ca scopuri secundare aplicația își propune să automatizeze procesul de pontare astfel încât utilizatorii sa depună un efort minim în vederea generării de rapoarte. Pe tot parcursul implementării programului s-a urmărit și s-au analizat diferite moduri prin care sa se elimine cat mai mult activitatea userului pe aplicație. In stadiul actual aplicația poate genera un raport ce cuprinde durata unei luni de munca în format PDF doar printr-o autentificare și printr-o apăsare a unui buton, nemaiaivând nevoie de alte operații.

Implementarea obiectivului a fost efectuata prin integrarea conceptelor și tehnologiilor prezentate la sfârșitul documentației în partea de Anexe. Astfel, pe partea de

server aplicația folosește Spring ca framework peste Java, Spring boot pentru a rezolva configurările ușor și rapid și pentru ascunderea detaliilor de care nu a fost nevoie, JWT(Anexa11) pentru autentificare și autorizarea interacțiunilor și Maven(Anexa 7) pentru rezolvarea dependentelor de librăriile externe. Pe partea de client, aplicația folosește React(Anexa 13) ca și librărie de JavaScript și Material-UI(Anexa 14) ca framework peste React (Anexa 12). Ca sistem de management al bazei de date am ales H2.

Arhitectura Sistemului

Aplicația este constituită din doua servere, unul pentru frontend altul pentru backend. Serverul de backend este unul de tip REST expunând endpoint-uri care sunt folosite de serverul de nodeJS specific proiectului de frontend. In diagram de mai jos este descris flow-ul prin care trebuie sa treacă un request pentru a fi procesat de serverul de backend.



Mai exact, procesarea unui request primit de la browser se face în felul următor:

1. Requestul este trimis la un endpoint apoi este preluat de un controller
2. Controller-ul apelează unul sau mai multe servicii cu scopul de a procesa informația primită
3. Serviciul apelează Repository-urile necesare pentru a aduce informații din baza de date pentru a executa logica business-ului. Un serviciu poate apela la rândul sau alt serviciu
4. Repositoryul este folosit pentru a aduce informațiile stocate în Modele din baza de date

Schimbul de date între browser și server se face prin intermediul DTO³-utililor (DATA TRANSFER OBJECT) care sunt serializate și deserializate.

Controller

Un controller primește datele introduse de utilizator printr-un apel ajax către un anumit endpoint, manipulează elemente din categoria “Model” pentru a obține datele cerute de utilizator și întoarce un răspuns către utilizator în format JSON. Controller-ul decide ce fel de cerere HTTP accepta (GET, POST, DELETE, PUT) și ce pași trebuie să se execute.

Un controller este definit prin adnotarea `@Controller`. Orice clasă a cărei definiție este precedată de această adnotare este considerată de Spring un controller. Metodele dintr-o clasă controller sunt numite metode handler.

Adnotarea `@RequestMapping` face corespondența între o clasă controller sau o metodă handler și un URL. De obicei, adnotarea aceasta este folosită la nivelul clasei și are rolul de a mapa întreaga clasă controller la un path unde vor fi direcționate apelurile. Metodele handler vor restrânge maparea controller-ului specificând tipul cererii HTTP acceptate⁴. Error! Reference source not found. Tabelul 1 prezintă un exemplu de mapare extras din implementarea aplicației. Exemplul folosește adnotarea `@PostMapping` care este echivalentul adnotării `@RequestMapping(method = RequestMethod.Post)`

```
@PostMapping(AdminController.API_NAME)
public ResponseEntity<ApiResponse<String>> createUser(@Valid @RequestBody UserDetailsRequestDTO userDetailsRequestDTO)
    adminService.createUser(userDetailsRequestDTO);
    return new ResponseEntity<>(new ApiResponse<>(result: "User created successfully"), HttpStatus.CREATED);
}
```

Tabel 1

a=X&ved=0ahUKEwjYstKktoDjAhXkwYsKHSPUD0QQ_AUIECgB&biw=2560&bih=888#imgsrc=UPWSyGG0tIUWDM:

³ https://en.wikipedia.org/wiki/Data_transfer_object

⁴

<https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/mvc.html>

Clasa ResponseEntity este folosită ca un Wrapper pentru răspunsul dat de endpoint, HttpStatus reprezentând codul întors pe acel răspuns.

Service

Un service primește datele de la controller și le procesează, urmând ca mai apoi rezultatul procesării să fie întors spre controller. Ca și funcționalitate, service-ul are rolul de a aduce informații din baza de date prin intermediul unui repository și de a aplica logica de business pe datele primite. Pentru implementarea unui service în Spring este nevoie de adnotarea @Service ce indică faptul că orice clasă adnotată astfel va fi folosită pentru scopul descris mai sus. O implementare concretă a unei astfel de clase este:

```
@Service
@AllArgsConstructor(onConstructor = @__(@Autowired))
public class ActivityTypeServiceImpl implements ActivityTypeService {
    private ActivityTypeRepository activityTypeRepository;

    @Override
    public List<ActivityType> getActivityTypes() { return activityTypeRepository.findAll(); }
}
```

Repository

Un repository este o interfață adnotată cu @Repository. Aceasta clasă servește ca scop de mediator între cod și baza de date, mai precis, prin intermediul interfeței se pot face interogări la baza de date și maparea acestora în modele. Implementarea concretă a unui astfel de repository arată astfel:

```
@Repository
public interface HolidayRepository extends JpaRepository<Holiday, Long> {
    @Query("from Holiday as h where h.date >= :startDate and h.date <= :endDate")
    List<Holiday> getAnnualHolidays(@Param("startDate") Instant startDate, @Param("endDate") Instant endDate);

    @Query("from Holiday h where year(h.date) = year(:date)")
    List<Holiday> getHolidaysForInputYear(@Param("date") Instant date);

    @Query("select new Holiday(h.date, h.description) from Holiday h where month(h.date) = month(:date) and year(h.date) = year(:date)")
    List<Holiday> getHolidaysPerMonthForDate(@Param("date") Instant date);
}
```

De regulă repository-urile construite în Spring extind JPA repository, JPA fiind un modul din Spring Data și cu ajutorul acestui modul operațiile de CRUD pentru fiecare repository sunt implementate automat.

Model

Un model este o reprezentare a unei stări de care avem nevoie în aplicația noastră. Este un șablon pentru ceva ce trebuie reprezentat în cod, precum o carte, un utilizator sau altele. Modelul trebuie să includă și este responsabil de atributele lucrului reprezentat, cât și de funcțiile care operează pe aceste variabile.

```

@Entity
@NoArgsConstructor
@AllArgsConstructor
@Table
public class ActivityType {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @NotBlank
    private String name;
    private Boolean isProjectRelated;

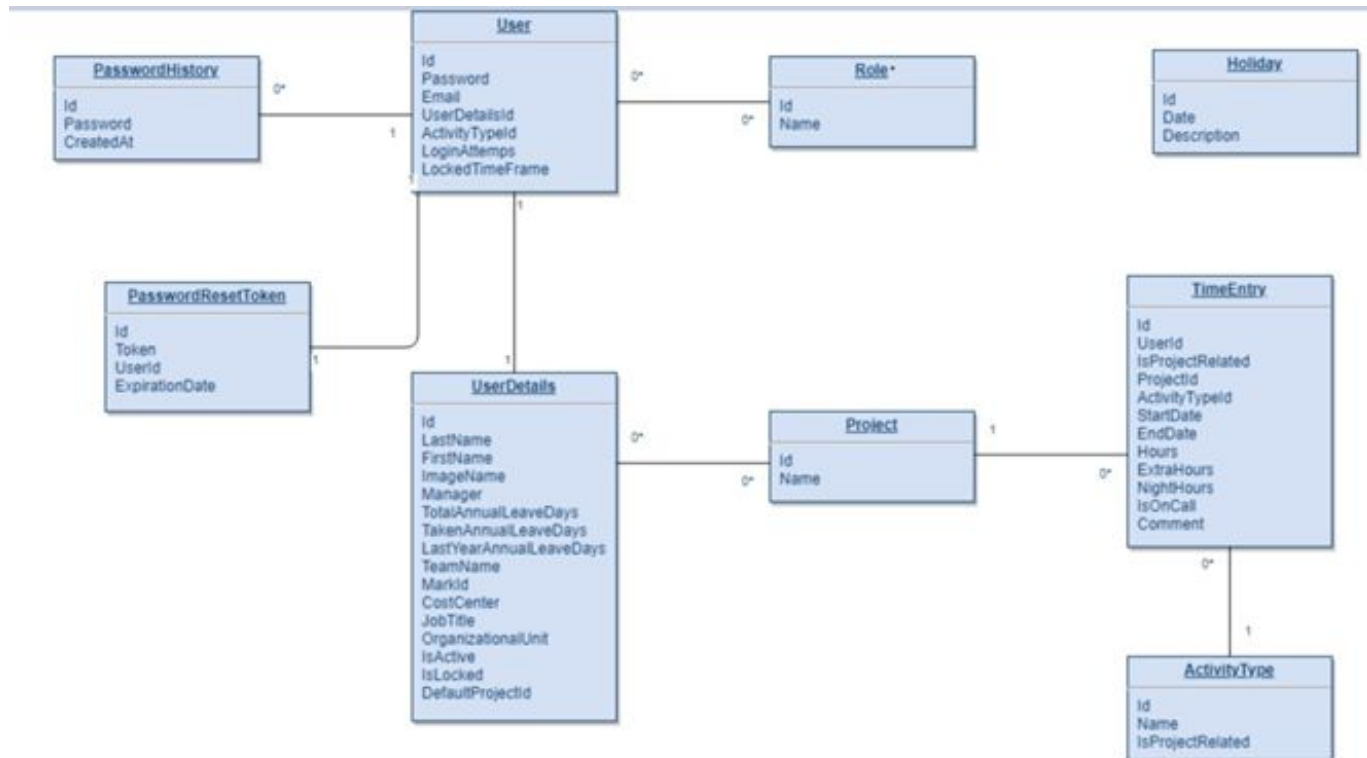
    public ActivityType(@NotBlank String name) {
        this.name = name;
        this.isProjectRelated = false;
    }
}

```

Adnotarea `@Entity` specifică faptul că obiectul `ActivityType` din Java trebuie tratat ca o entitate în procesul de mapare în baza de date, ceea ce înseamnă că un tabel cu numele “`ActivityType`” ce are ca și coloane atributele adnotate cu `@Column` (nu este necesară această adnotare) din obiectul respectiv va trebui creat. Adnotarea `@Table` poate fi omisă, dar este de folos atunci când se dorește adăugarea unor constrângeri asupra tabelului asociat.

Adnotarea `@Id` specifică faptul că atributul `id` de tipul `long` este considerat cheie primară în tabelul ce urmează a fi creat, dar include și funcționalitatea oferită de `@Column`. `@GeneratedValue` se aplică atributului ce va fi considerat cheie primară a tabelului și specifică strategia folosită în generarea acesteia. În cazul de față strategia `Identity` indică faptul că generarea cheii primare intra în sarcinile Hibernate, sarcina ce este implementată default de către tipul bazei de date. În cazul aplicației “`Identity`” semnifică incrementarea cu o unitate a id-ului

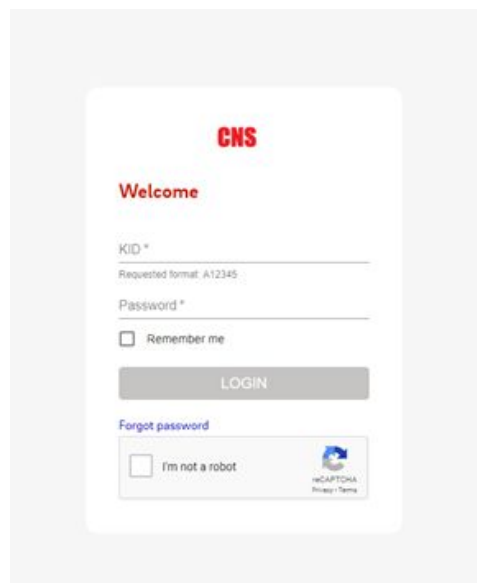
Arhitectura bazei de date



- Tabela “User” conține informații legate des credențialele acestuia precum și diferite relații între tabele
- Tabela “Role” definește rolurile pe care un user le poate avea
- Tabela “PasswordHistory” are înregistrări legate de istoria parolilor puse de user, ea este folosită în momentul în care un utilizator vrea să-și schimbe parola constrângându-l ca parola pusă să fie diferită de ultimele 5 parole.
- Tabela “PasswordResetToken” conține anumite informații legate de token-ul utilizatorului în vederea procesului de resetare de parola
- Tabela “UserDetails” conține informații personale despre utilizator precum și o relație “many to many” cu tabela “Project”
- Tabela “Project” conține Id-ul proiectului concatenat printr-o cratimă cu numele proiectului
- Tabela “TimeEntry” conține informațiile legate de intervale, datele pe care utilizatorul vrea să le ponteze și o relație “many to one” cu tabela “Activity Type”
- Tabela “ActivityType” conține tipul de activitate pontată de utilizator. Ex: concediu, ore lucrătoare, concediu medical, absență etc
- Tabela “Holiday” conține informații despre concediile de la stat. Ex: Crăciunul, Paștele etc

Modul de funcționare al aplicației

În prima fază, pentru utilizarea aplicației este nevoie de un cont care va fi creat de către persoanele de la HR al companiei sau de utilizatorul suprem al aplicației pentru fiecare angajat în parte. Odată ce contul a fost creat utilizatorul are nevoie să se autentifice în aplicație. Pagina de autentificare este prezentată mai jos.

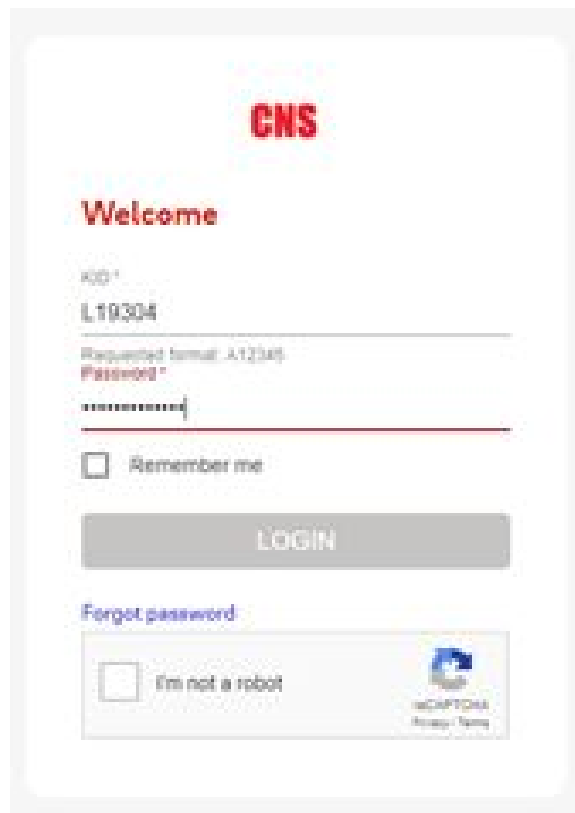


The screenshot shows the login interface for the CNS application. At the top, the 'CNS' logo is displayed in red. Below it, the word 'Welcome' is written in red. The login form consists of two input fields: 'KID *' and 'Password *'. The 'KID *' field has a red border and a red error message 'Requested format: A12345' below it. Below the password field is a checkbox labeled 'Remember me'. A grey 'LOGIN' button is positioned below the checkbox. A blue link 'Forgot password' is located below the login button. At the bottom of the form, there is a checkbox labeled 'I'm not a robot' and a reCAPTCHA logo with links for 'Privacy' and 'Terms'.

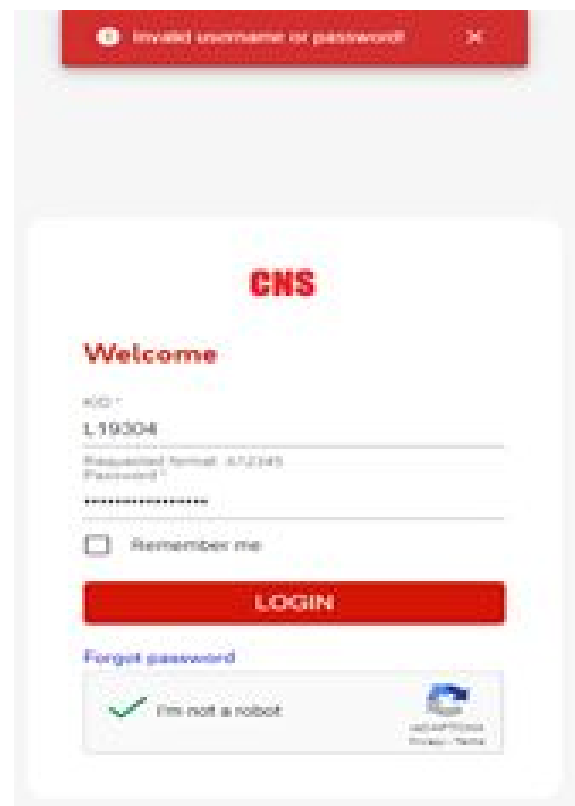
Pentru fiecare input în parte sunt făcute diverse verificări atât pe partea de frontend cât și pe partea de backend. Validările de pe partea de frontend au fost făcute pentru a nu face apel la server dacă datele nu respectă anumite formate, iar validările de pe backend au fost create pentru a asigura o securitate sporită a informațiilor primite.



This screenshot shows the same login interface as the previous one, but with a validation error. The 'KID *' input field is highlighted with a thick red border, and a red error message 'KID is not valid' is displayed below it. The 'Password *' field, 'Remember me' checkbox, 'LOGIN' button, 'Forgot password' link, and reCAPTCHA section remain unchanged.



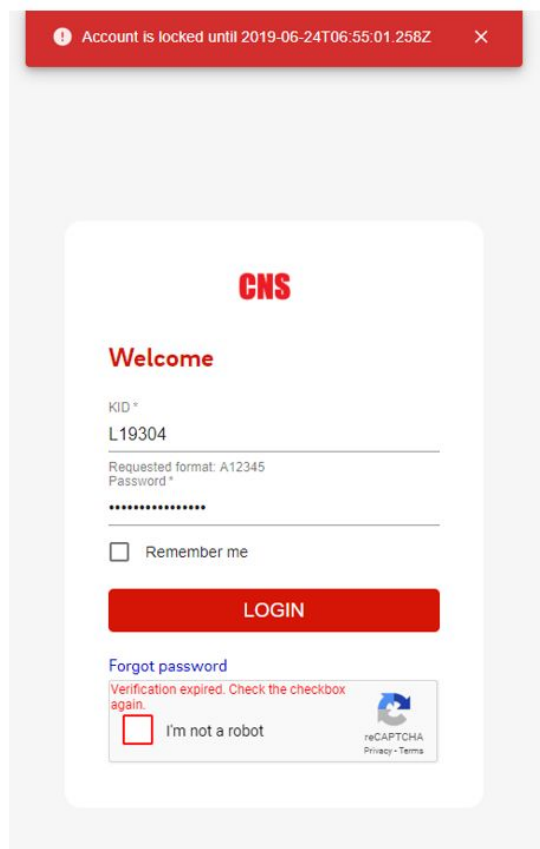
The image shows a login interface for a system named "CNS". At the top, the text "CNS" is displayed in red. Below it, the word "Welcome" is also in red. There are two input fields: the first is labeled "ID:" and contains the text "L19304"; the second is labeled "Password:" and contains a series of dots. Above the password field, there is a small text "Requested format: A12345". Below the password field, there is a checkbox labeled "Remember me". A grey button with the text "LOGIN" is positioned below the checkbox. Below the login button, there is a link "Forgot password" in blue. At the bottom, there is a checkbox labeled "I'm not a robot" and a reCAPTCHA logo.



The image shows the same login interface as above, but with an error message at the top: "Invalid username or password" in white text on a red background. The "LOGIN" button is now red. The "I'm not a robot" checkbox is now checked, and there is a green checkmark icon to its left. The reCAPTCHA logo is still present.

Aplicația mai înglobează și alte mecanisme de securitate, unul dintre ele se afla chiar pe partea de autentificare. Dacă se încearcă autentificarea de mai mult de cinci ori a unui user

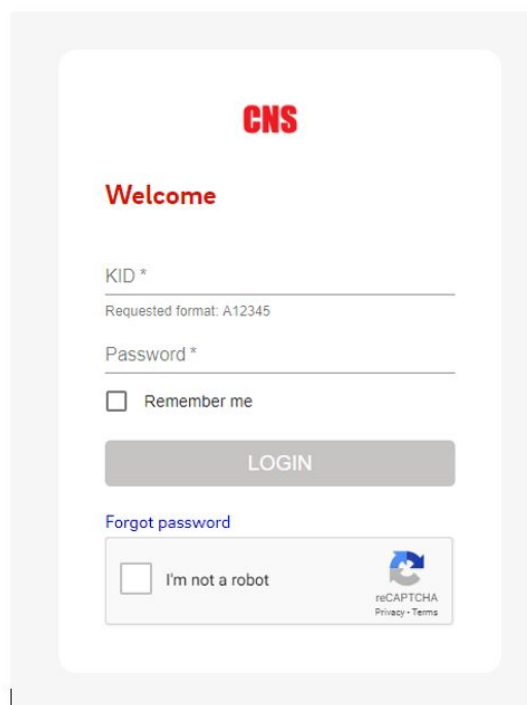
cu credentiale greșite userul va fi blocat pe o perioadă de o zi (Imaginea 1). Pentru a preveni suprasolicitarea aplicației de către mecanisme automate (programe de calculator) s-a adăugat și un CAPCHA⁵, iar dacă aceasta nu este completat butonul de “logare” este dezactivat (Imaginea 2).



The image shows a web interface for the CNS system. At the top, a red banner displays the message: "Account is locked until 2019-06-24T06:55:01.258Z". Below this, the CNS logo is visible. The main section is titled "Welcome" and contains a login form. The form includes a "KID" field with the value "L19304" and a "Password" field with masked characters. A "Remember me" checkbox is present. A red "LOGIN" button is located below the password field. At the bottom of the form, there is a "Forgot password" link and a CAPTCHA challenge. The CAPTCHA message states: "Verification expired. Check the checkbox again." and includes a checkbox labeled "I'm not a robot". To the right of the checkbox is a reCAPTCHA logo and links for "Privacy" and "Terms".

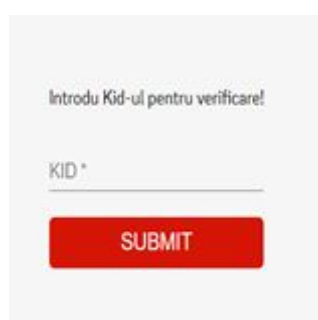
Imagine 1

⁵ <https://ro.wikipedia.org/wiki/CAPTCHA>



Imagine 2

Funcționalitatea de “Forgot password” este implementată cu ajutorul JWT-ului. Odată ce este apăsat link pentru această funcționalitate utilizatorul va fi direcționat către un formular(Imagine3) în care acesta va trebui să-și introducă KID⁶. După ce se va completa inputul pentru KID se va trimite un mail utilizatorului(Imagine 4) ce conține un link către un formular de resetare de parolă. Formularul va fi valid 24 de ore sau odată completat și trimis cu succes către server acesta nu mai va fi valid(Imagine5), iar dacă se încearcă accesarea link-ului trimis de către un alt utilizator operația de resetare nu se va executa.



⁶ KID = id unic asignat unui angajat dintr-o companie ce este vizibil și în afara acesteia

Hi Lucian Cochior,

You recently requested to reset your password for your Chronos Timesheet account. Use the button below to reset it. **This password reset link is only valid for the next 1 hour.**

[Click here to change your password.](#)

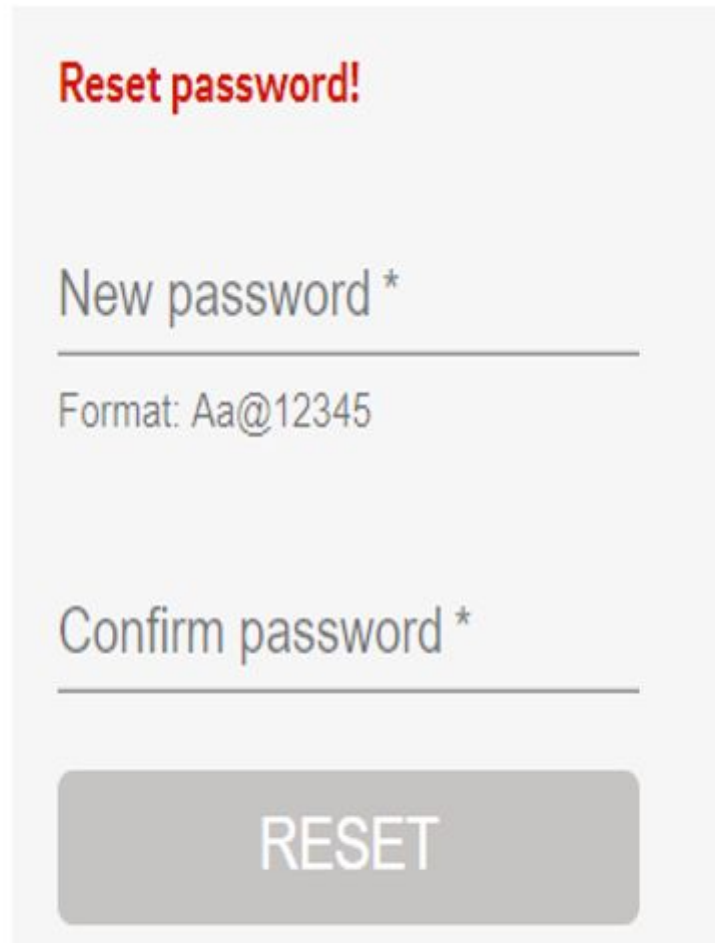
If you did not request a password reset, please ignore this email.

Thanks,
The CNS Team

If you're having trouble with the button above, copy and paste the URL below into your web browser.

<http://localhost:3000/reset-password?token=6399e62e-c685-4a39-b30c-3e886130bff2>

Imagine 4

A light gray rectangular form with a red title 'Reset password!' at the top. Below the title is a text input field labeled 'New password *' with a horizontal line underneath. Below the input field is the text 'Format: Aa@12345'. Further down is another text input field labeled 'Confirm password *' with a horizontal line underneath. At the bottom of the form is a large, rounded gray button with the word 'RESET' in white capital letters.

Reset password!

New password *

Format: Aa@12345

Confirm password *

RESET

Imagine 5

Odată ce utilizatoru a efectuat autentificarea cu succes, aplicația îi va genera un token și îl va salva în cookie, acesta urmând să fie pus în headerul fiecărui request către server, apoi utilizatorul va fi redirecționat pe pagina principală (dashboard, Imagine 6). Prin ajutorul token-ului un utilizator odată autentificat va rămâne logat pe aplicație timp de 24 de ore, iar dacă acesta deschide un alt tab în browser nu va mai trebui să se logheze, logarea făcându-se pe tokenul salvat în cookie.

Imagine6

Exact după header se afla un container ce conține o lista de informații actualizate dinamic în funcție de datele inserate de utilizator iar în partea stânga este un buton de generare a tabelului de Pontaj în format PDF. Butonul de generare face un apel la server, iar serverul se va ocupa de tot procesul de creare al PDF-ului iar ca răspuns va întoarce un sir de biți ce vor fi randati mai apoi pe partea de client.

21

Componenta de calendar este adusa din framework-ul Material-UI, iar fiecare celula din aceasta este completata dinamic cu informațiile legate de numele proiectului și numărul de ore normale sau suplimentare lucrate pe acel proiect.

În partea dreapta al calendarului se afla un container prin care utilizatorul își poate completa informațiile legate de pontaj. Fiecare câmp din acest container are validări și diverse mecanisme pentru a minimiza numărul de click-uri al utilizatorului

Pagina de profil al utilizatorului (Imagine 7) conține toate datele confidențiale ale acestuia precum și o poza de profil care o să fie utilizata în rapoarte, tot în aceasta este înglobată și funcționalitatea de schimbare de parola. Un utilizator dacă dorește să-și schimbe parola este obligat ca parola pusă să fie diferită de ultimele 5 parole avute pe acel cont. Toate câmpurile necesare pentru schimbarea de parola au validări, iar butonul de “Change Password” este invalid până când cele două câmpuri vor trece peste validările făcute.



KID	Mark ID	
L19304	200888	Current password *
First Name	Organizational Unit	
Lucian	CNS Iasi	New password *
		At least 8 characters
Last Name	Job Title	
Cochior	Junior Java Developer	Confirm password *
Email	Team	
lucian.cochiorheghi@gmail.com	Junior	Change Password

Self Service Password Change

- The new password must contain at least:
- 1 lowercase and 1 uppercase letter
 - 1 digit
 - 1 special character
 - minimum 8 characters

Imagine 7

Daca pe aplicatie se autentifica un utilizator cu rolul de admin atunci meniul reprezentat in partea stanga a headerului va contine si optiune de adaugare a unui user nou. Odata aleasa aceasta optiune se va redirectiona utilizatorul(adminul) catre un nou formular unde. Toate campurile din formular contin validari atat pe partea de frontend cat si pe partea de backend. După ce se va trimite formularul către server utilizatorul nou creat va primi pe mail un link către un formular de setare a parolei, acest link va fi valid 24 de ore de la data trimiterii mail-ului sau invalidat în momentul în care noul user îs va schimba parola.

Add user

Projects ▾

Roles ▾

KID *

Email *

Organizational unit *

Team name *

First name *

Last name *

Manager *

☐ is active

Job title *

Cost center *

0

Mark id *

0

Last year annual leave days *

0

Taken annual leave days *

0

Total annual leave days *

25

Add user

Imagine 8

Hi test test,

Your account has been created successfully! Use the following link to setup your password. **This password reset link is only valid for the next 24 hours.**

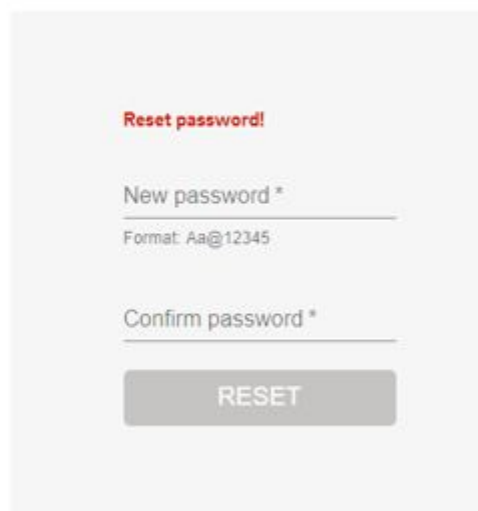
[Click here to change your password.](#)

Thanks,
The CNS Team

If you're having trouble with the button above, copy and paste the URL below into your web browser.

<http://localhost:3000/reset-password?token=480cf226-20b7-480f-b7ee-be3b9f8cac3a>

Imagine 9



The screenshot shows a web form for resetting a password. At the top, the title "Reset password!" is displayed in red. Below the title, there are two input fields. The first is labeled "New password *" and has a small text hint below it that says "Format: Aa@12345". The second is labeled "Confirm password *". At the bottom of the form is a grey button with the text "RESET" in white capital letters.

Imagine 10

Algoritmi dezvoltati

Algoritmul de inserare al intervalelor

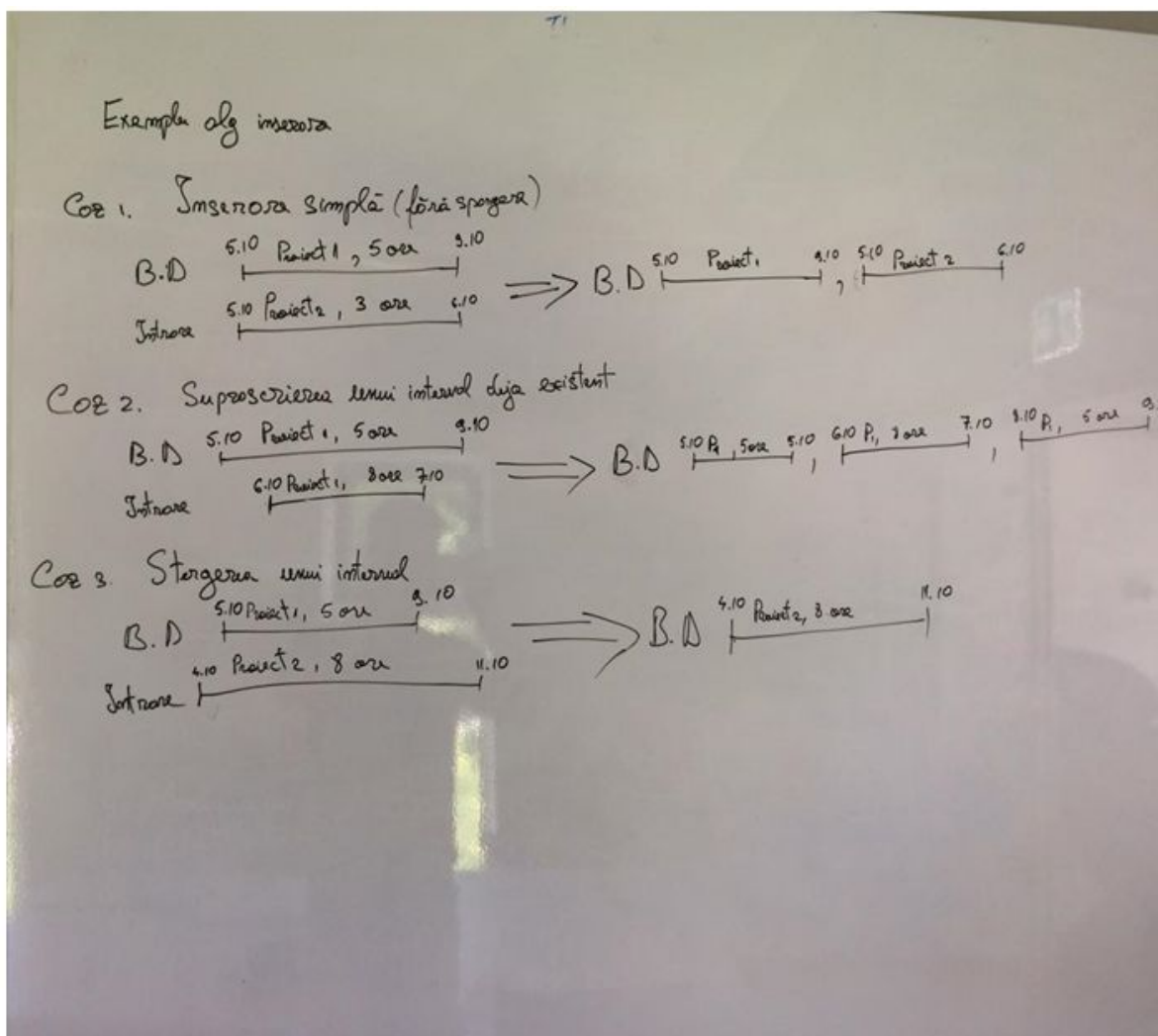
Dupa cum am specificat pe tot parcursul acestei lucrări, aplicația salvează zilele pontate sub forma de intervale pentru a reduce numărul de înregistrări. Problema principală la această abordare a fost găsirea unui algoritm care să poată să însereze, să modifice și să șteargă astfel de interval fără a se specifica tipul operației.

Algoritmul primește o listă de intervale și lucrează în felul următor:

1. Se uita în lista de intervale primită și salvează cea mai mică margine inferioară și cea mai mare margine superioară găsită în listă.
2. Pe baza marginilor calculate la punctul 1 se aduc intervalele din baza de date care se intersectează cu intervalul creat pe baza marginilor de mai sus
3. Se parcurge lista de intervale de intrare și pentru fiecare element se caută o potrivire⁷ în lista de intervale din baza de date
 - 3.1.1 Dacă există o potrivire atunci se ia intervalul din baza de date și cel de intrare și se separă intervalele prin-un algoritm de separare ce va fi explicat mai jos în funcție de cum se intersectează acestea, apoi intervalele obținute vor fi adăugate în lista de rezultate.
 - 3.1.2 Dacă nu există nicio potrivire atunci se adaugă intervalul de intrare în lista de rezultate
4. Se reface lista de interval primite din baza de date cu cele rezultate obținute la pasul 3
5. Se șterg vechile intrări din baza de date și se adaugă intrările noi rezultate după terminarea elementelor din lista de intrare primită

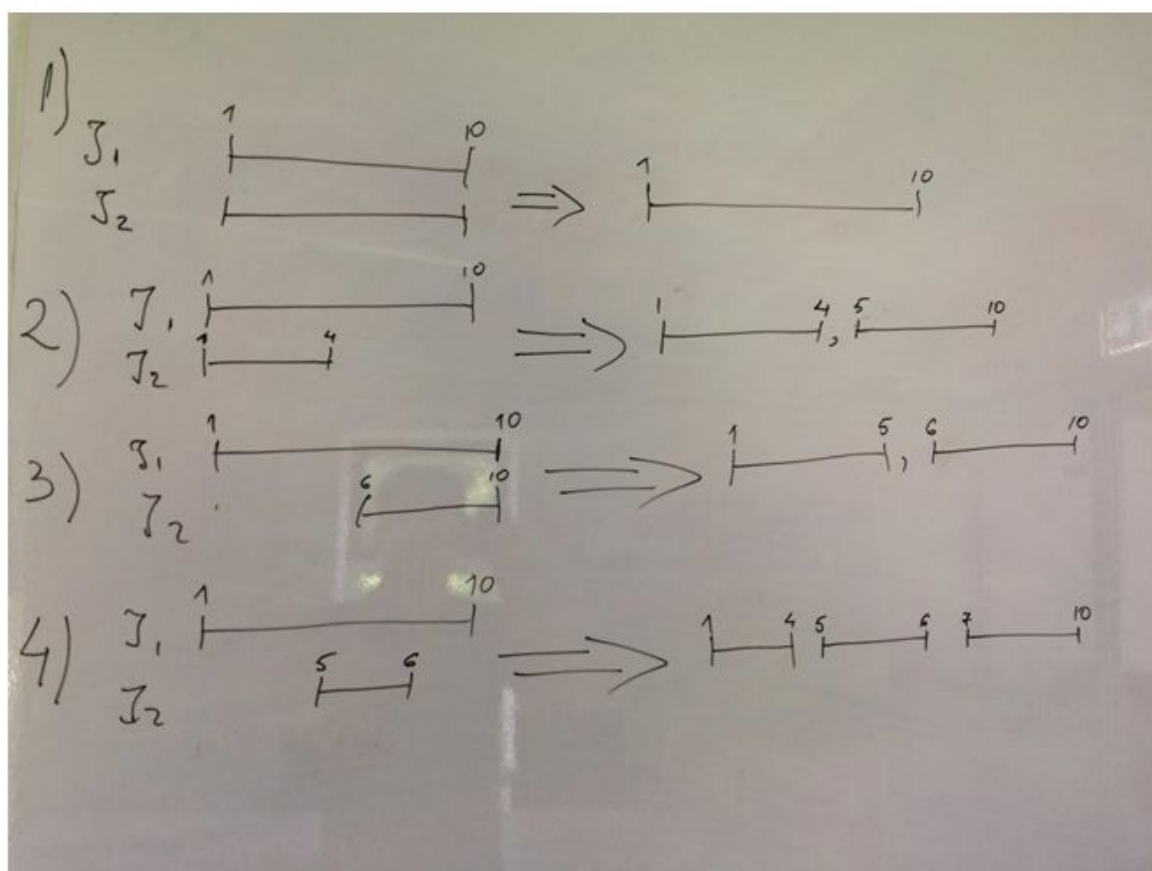
Aveți atașat mai jos o imagine care exemplifică modul de funcționare al algoritmului de inserare.

⁷ potrivire = două intervale conțin aceleași date (ore lucrate, ore suplimentare, ore de noapte)



Algoritmul de spargere al intervalelor

Algoritmul acesta joaca un rol esențial în funcționalitatea de adăugare a intervalelor în baza de date. Ideea principal la acesta este ca el se uita la doua interval și în funcție de cum se intersectează cele doua interval acestea vor fi sparte. Pentru o mai buna înțelegere a modului de funcționare al acestui algoritm aveți atașat mai jos o poza cu exemple de spargeri.



Concluziile lucrării

Aplicația CHRONOS este o aplicație end-to-end care oferă utilizatorului final, trei tipuri de roluri (utilizator, HR, admin), o experiență plăcută indiferent de dispozitivul de pe care este accesată aplicația.

Lucrarea menționată mai sus încearcă să automatizeze procesul de pontaj al angajaților unei companii și minimizarea timpului petrecut de utilizator pe aplicație.

Pornită de la funcționalitățile de bază întâlnite în alte aplicații asemănătoare și rezolvând unele lipsuri evidente ale acestora, aplicația în discuție oferă diferite posibilități de dezvoltări ulterioare.

Dezvoltare actuală

Pe lângă funcționalitățile de bază asigurate de aplicațiile existente deja, proiectul oferă:

- lucrul pe intervale pentru stocarea zilelor pontate pe proiecte în baza de date
- generarea unui raport PDF care este descărcat în browser
- o interfață foarte intuitivă și ușor de utilizat care permite pontarea în aceeași operație a mai multor proiecte
- un algoritm eficient de inserare și spargere de interval

Dezvoltare ulterioară

În viitor aplicația va avea îmbunătățiri majore:

1. noi mecanisme de generare a rapoartelor (mai multe formate)
2. Automatizarea completa a procesului de pontare. Din momentul în care se generează raportul acesta să fie automat trimis către persoanele care se ocupă cu pontarea
3. Adăugare de noi funcționalități pentru Admin și pentru userii cu rol de HR

Anexe

Anexa 1 - Java

Java este un limbaj de programare orientat-obiect de nivel înalt dezvoltat inițial de compania Sun Microsystems lansat pentru prima oară în anul 1995. Ultima versiune de Java Standard Edition este Java SE 10 lansat în 2018, cea mai populară.

Un program ce este format din cod Java poate fi rulat pe orice tip de mașină indiferent de sistemul de operare. Cele mai importante caracteristici ale acestui limbaj sunt⁸:

- orientat-obiect – orice este scris în Java este un obiect
- poate rula pe orice mașină indiferent de sistemul de operare
- sigur – are o bună documentație și asigură diverse mecanisme de securitate
- se poate lucra pe mai multe fire de execuție

Anexa 2 - Spring framework

Un framework este o structură conceptuală destinată să fie folosită ca și suport sau ghid pentru construirea unor materiale care extind structura respectivă în ceva folositor.

În domeniul IT un framework este, de obicei, o structură stratificată ce indică tipul programelor care pot fi construite și cum ar trebui să acestea să interacționeze⁹.

Spring este cel mai popular framework pentru dezvoltarea aplicațiilor în Java. Milioane de dezvoltatori folosesc Spring Framework pentru a crea cod reutilizabil, ușor de testat și cu o performanță ridicată. Un element cheie al Spring-ului este suportul infrastructural la nivelul aplicației ceea ce înseamnă că framework-ul se concentrează pe “sanitizarea” aplicațiilor, astfel încât echipele să se poată concentra pe logică afacerii la nivelul respective, fără legături inutile cu medii specifice de livrare.

Spring framework este o platformă Java care a apărut pentru prima dată sub licența Apache 2.0 în anul 2003, dezvoltat de către Rob Johnson. Versiunea de bază a frameworkului are o dimensiune scăzută de doar 2MB. Funcționalitățile de bază ale acestui

⁸ https://www.tutorialspoint.com/java/java_overview.htm

⁹ <https://whatistechtarget.com/definition/framework>

framework pot fi folosite în dezvoltarea tuturor aplicațiilor Java, dar există și extensii ale acestuia construite special pentru dezvoltarea aplicațiilor web în platformă Java EE¹⁰. Fiind open-source¹¹, Spring se bucură de o comunitate vastă și activă care oferă suport bazat pe o varietate de cazuri reale, ceea ce a ajutat framework-ul să evolueze cu succes într-o perioadă lungă de timp.

Framework-ul este împărțit pe module. Aplicațiile pot alege doar modulele de care au nevoie. La baza stau modulele containerului principal (core) ce include un model de configurație și un mecanism de injectare a dependentelor. Pe lângă acestea, Spring oferă suport pentru diferite arhitecturi de aplicații printre cum ar fi mesagerie, date tranzacționale și persistarea datelor și web. Include de asemenea un framework web bazat pe servlet-uri numit Spring MVC.

Spring continuă să inoveze și să evolueze. În afară de Spring Framework există și alte proiecte precum Spring Boot, Spring Security, Spring Data, Spring Cloud, Spring Batch.

Anexa 3 - Dependency Injection și Inversion of Control

Injectarea dependentelor, ce face parte din conceptul mai vast de inversiune a controlului, sta la baza Spring-ului. Principiul de design numit “inversiune a controlului” este folosit pentru a inversa diferite tipuri de control într-o proiectare orientata-obiect pentru a obține un cuplaj slab între elemente. În contextul curent, controlul înseamnă orice responsabilitate adițională pe care o clasă o are, pe lângă responsabilitatea de bază a acesteia, cum ar fi controlul asupra pașilor de execuție a unei aplicații sau controlul asupra pașilor de execuție urmați când se creează un obiect.^[1]¹² Injectarea dependentelor nu este nimic altceva decât un caz concret al inversiunii controlului.

Când o aplicație Java complexă este scrisă, clasele aplicației trebuie să fie cât mai independente pentru a crește posibilitatea refolosirii acestora și pentru a fi posibilă testarea pe unități. Injectarea dependentelor ajută la îmbinarea claselor, acestea rămânând totuși independente.

Anexa 4 - Beneficiile utilizării framework-ului Spring

Cele mai importante beneficii pe care le aduce framework-ul Spring și motivele pentru care acesta ar trebui ales pentru implementarea unei aplicații sunt:

- Spring asigură diverse mecanisme ce ajută la implementarea aplicațiilor complexe într-un mod simplu și elegant
- Spring este organizat în stil modular, există o imensitate de clase și pachete în acest framework dar noi putem să le importăm doar pe cele de care avem nevoie
- Codul este ușor de testat
- În Spring putem asocia ușor erorile sau excepțiile provenite de la alte tehnologii

¹⁰ https://www.tutorialspoint.com/spring/spring_overview.htm

¹¹ open-source = codul sursă original este accesibil oricui și poate fi redistribuit sau modificat

¹² <https://docs.spring.io/spring/docs/current/spring-framework-reference/overview.html>

- Dependency Injection și Inversion of Control
- Oferă o interfață pentru managementul tranzacțiilor de date¹³

Anexa 5 - Spring Boot

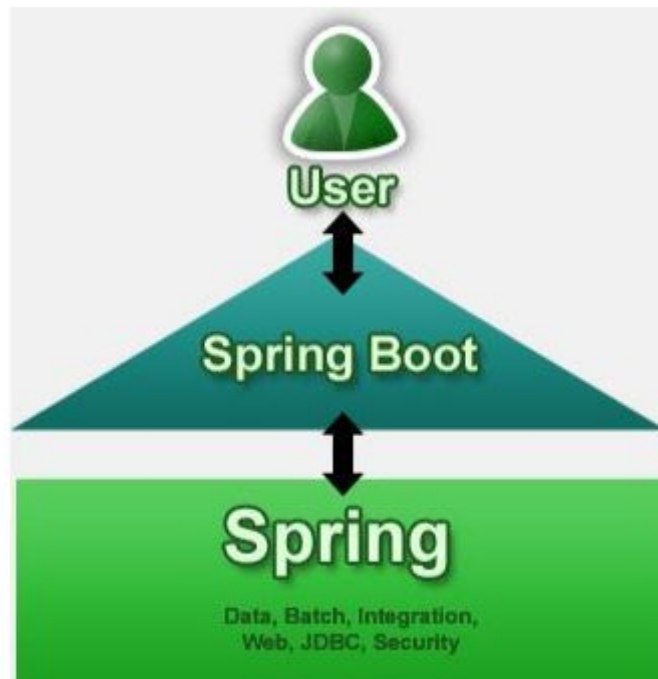


Fig 3 Spring boot

Spring Boot este o versiune superioara a Spring-ului care vine cu noi mecanisme care sa ajute la configurarea mai ușoara a unui sistem. Asa cum se vede și în Fig 3 , Spring Boot nu este decât o versiune superioara Spring care are ca obiective principale următoarele¹⁴:

- Să asigure o ușoara dezvoltare a proiectelor abia începute, neavând nevoie de toate cunoștințele necesare din Spring, Spring Boot-ul făcând multa magie în spate pentru programator.
- Sa elimine partea greoaie de configurări
- Să aducă o varietate de mecanisme care sa ușureze procesul de development

Anexa 6 - Maven

“Maven” este considerat ca fiind un instrument pentru gestionarea proiectelor software (“Project management tool”) și include facilități necesare dezvoltării de aplicații mari, la care contribuie mai multe persoane, care refolosesc părți din alte proiecte software, care pot genera rapoarte și site-uri Web. Versiunile Maven folosite în prezent sunt 2.2 și 3.0 și folosesc tot fișiere XML pentru descrierea operațiilor și dependentelor.

¹³ https://www.tutorialspoint.com/spring/spring_overview.htm

¹⁴ <https://spring.io/blog/2013/08/06/spring-boot-simplifying-spring-for-everyone>

Anexa 7 - Orm, JPA, Hibernate

ORM sau Object Relational Mapping este un procedeu de transformare a obiectelor în tipuri relaționale și viceversa.. Cea mai importanta problema pe care ORM-ul o rezolva este aceea de a realiza o corespondență între un obiect și datele din baza de date.

Avantajele unei implementări ORM iau această formă în Hibernate¹⁵:

- Relația obiect java – tabel în baza de date se face pe bază de fișiere XML
- O interfață simplă pentru stocarea și preluarea datelor în/din baza de date
- Dacă se produce o schimbare în tabele, o simplă modificare în fișierele XML este tot ce trebuie · Hibernate nu are nevoie de un server al aplicației pentru a funcționa
- Minimizează accesul la baza de date prin strategii de preluare a datelor JPA, sau Java Persistence API, este o colecție de clase și metode care au ca scop stocarea și preluarea de date dintr-o bază de date. Deoarece JPA este open-source, diferite companii au furnizat produse folosindu-se de funcționalitățile oferite de JPA, una dintre ele fiind Hibernate.

Anexa 8 - Arhitectura ORM

Asa cum se vede în Fig 5, salvarea datelor în baze de date relaționale se face prin 3 pași¹⁶, acești pași sunt:

Pasul 1: numit și Object data, folosește clase POJO, interfețe și alte clase.

Faza 2: numit și pasul de Mapping care realizează asocierea între obiecte Java și tabelele din baza de date.

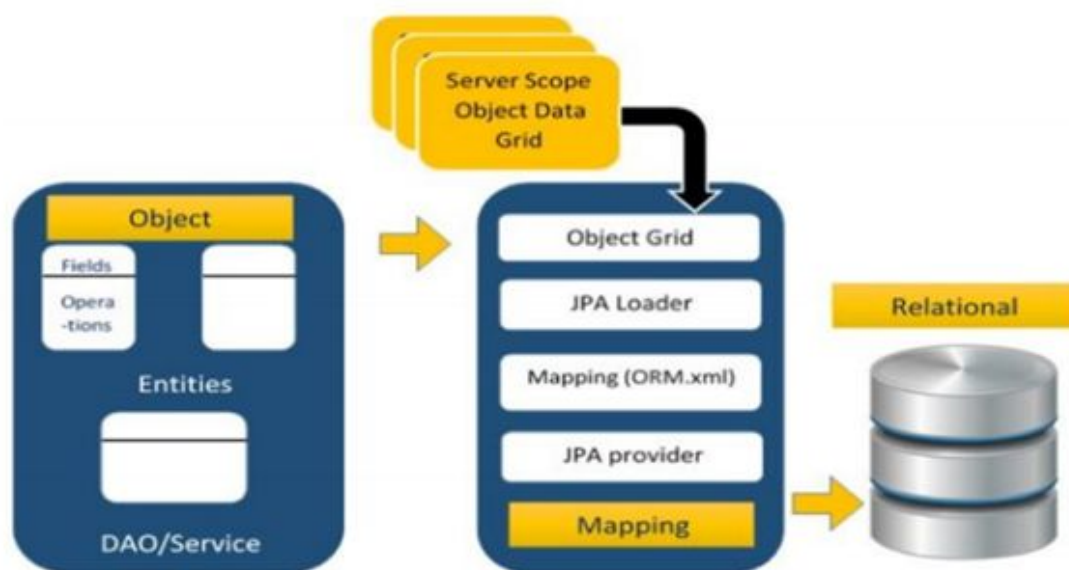


Fig 5 Arhitectura ORM

¹⁵ https://www.tutorialspoint.com/hibernate/hibernate_overview.htm

¹⁶ https://www.tutorialspoint.com/jpa/jpa_orm_components.htm

Pasul 3: Conține datele relaționale care sunt conectate în mod logic de business. Datele sunt stocate fizic în baza atunci când se face commit pe partea de business.

Anexa 9 - Adnotari

În general, fișierele XML sunt folosite pentru a configura o componentă specifică său. În cazul arhitecturii ORM, trebuie să păstrăm fișierul ORM.xml, ce definește relațiile între obiectele Java și obiectele relaționale, undeva în framework. Asta înseamnă că atunci când acest fișier xml este scris, trebuie mereu comparate numele atributelor din clasele POJO cu etichetele scrise în xml pentru a se asigura corespondența între cele două.

Adnotările sunt soluția pentru problema descrisă mai sus. În definiția clasei se poate scrie partea de configurare folosind doar adnotări. Adnotările sunt folosite pentru clase, proprietăți și metode, acestea începând cu simbolul “@”. Toate adnotările sunt definite în pachetul javax.persistence¹⁷.

Exemple de adnotări:

- @Entity – specifica ca o clasă este o entitate (tabel)
- @Id – apare atunci când se dorește ca un atribut din entitate sa fie cheia primara pentru tabelul din baza de date
- @GeneratedValue – este folosit pentru a specifica cum trebuie sa se genereze valorile pentru cheile primare ce sunt adnotate cu @Id
- @Column – indica ca un anumit atribut al entității este o coloana în baza de date
- @UniqueConstraint – este folosit pentru a declara o constrângere
- @OneToOne – este folosit pentru a declara o relație între doua tabele

Anexa 10 - JWT

JWT sau JSON Web Token este un standard ce definește o cale compactă și independentă pentru a transmite, în siguranță, informații între diferite componente sub forma unui obiect JSON. Aceste informații sunt de încredere deoarece pot fi verificate datorită semnăturii digitale. În aplicația în discuție, JWT sunt semnate cu chei publice și private utilizant algoritmul de criptare RSA.

Cele mai importante scenarii în care JSON Web Token este folosit sunt:

- Autorizare – Este cel mai comun scenariu de folosire a JWT. Din moment ce utilizatorul este conectat, orice cerere ulterioară va conține un JWT, astfel permițând utilizatorului să acceseze diferite rute, servicii și resurse care sunt disponibile cu acel token.
- Schimb de informații – JSON Web Tokens sunt o modalitate bună de a transmite informații în siguranță între diferite componente. Deoarece JWT poate fi semnat, putem fi siguri de identitatea utilizatorului. Adicional, datorită faptului că semnătura

¹⁷ https://www.tutorialspoint.com/jpa/jpa_orm_components.htm

este calculată folosind antetul și sarcină utilă (payload), se poate verifica faptul că nu a fost manipulat conținutul informației.

Structura unui JSON Web Token implică 3 componente separate printr-un punct.

Acestea sunt: antetul (header), sarcină utilă (payload) și semnătura (signature). După cum se poate observa și în Fig 6 de mai jos, antetul este format din două părți: tipul token-ului (adică JWT) și algoritmul de hashing folosit (în cazul nostru RSA).

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

Fig 6 JWT Header

Payload-ul conține solicitările (claims). Acestea sunt despre o entitate (de obicei utilizatorul) și datele adiționale. Există trei tipuri de solicitări:

- · Înregistrate – Sunt un set de solicitări predefinite care nu sunt obligatorii, dar sunt recomandate deoarece oferă funcționalități de identificare a emitentului, data de expirare a tokenului și multe altele
 - · Publice – pot fi definite liber, dar trebuie avut în vedere faptul că există posibilitatea apariției coliziunilor între acestea.
 - · Private – solicitări personalizate create pentru a împărtăși informații între componentele care își exprima acordul pentru folosirea lor. Fig 6 JWT Header 17
- Structura unui payload este descrisă în Fig 7 de mai jos.

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "admin": true
}
```

Fig 7 structura unui payload

Pentru a genera semnătura avem nevoie de antetul criptat, payload-ul criptat, un secret și de algoritmul specificat în header. Semnătură este folosită să pentru a verifica faptul că mesajul nu a fost schimbat în timpul schimbului de informații și, în cazul în care token-urile sunt semnate cu chei private, pentru a valida identitatea expeditorului.

Mod de funcționare:

Când un utilizator se autentifica folosind credentialele sale, un JSON Web Token va fi returnat. Oricând utilizatorul va dori accesarea unei rute sau unei resurse protejate, el va trebuie să trimită în cerere (request) JWT-ul returnat anterior. De obicei, JWT-ul este trimis în antetul de autorizare folosind schema “Bearer”, cum se poate observa și în Fig 8.

```
Authorization: Bearer <token>
```

Fig 8 schema "Bearer" în antetul de autorizare

Dacă se încearcă accesarea unei rute sau resurse protejate, serverul verifica existent unui JWT valid în antetul de autorizare. Dacă acesta există și este valid, utilizatorului i se permite accesarea resursei protejate.

După cum se observă și în Fig 9 pașii de execuție pentru obținerea și utilizarea JWT-urilor sunt:

- 1. Aplicația cere autorizare de la serverul competent
- 2. Când autorizarea este acordată, serverul de autorizare întoarce un token de acces
- 3. Aplicația folosește token-ul de acces pentru a accesa resurse protejate



Fig 9 flow pentru obtinerea si folosirea JWT

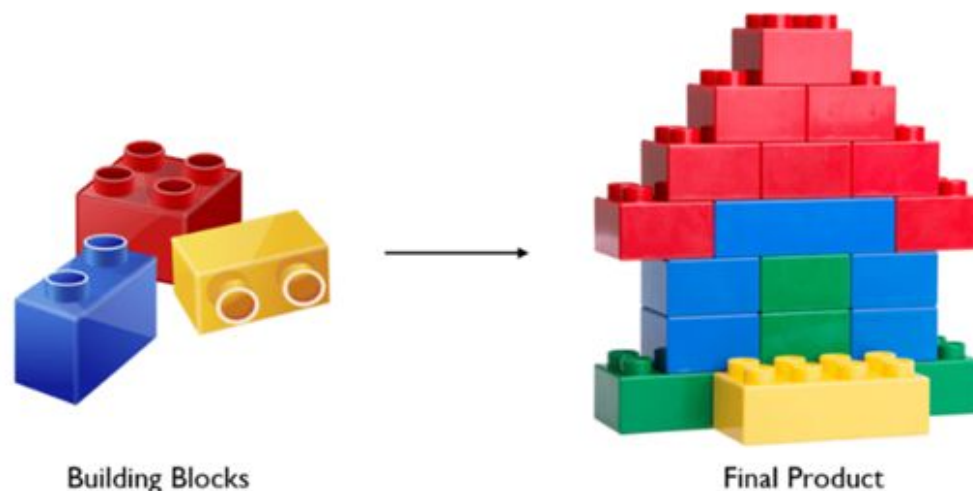
Avantajele utilizării JWT-urilor în detrimentul altor tehnologii asemănătoare precum SWT (Simple Web Tokens) sau SAML (Security Assertion Markup Language Tokens) sunt¹⁸ :

- JWT este o alegere mai bună decât SAML în contextul folosirii mediilor HTML și HTTP deoarece este mai compact. JSON folosit de JWT este mai puțin prolix decât XML folosit de SAML, iar când este criptat dimensiunea lui scade.
- Din punct de vedere al securității, SWT pot fi semnate simetric de un secret împărtășit de mai multe componente folosind algoritmul HMAC. JWT și SAML pot folosi perechi de chei publice-private conform certificatului X.509 pentru a semnatura. Deoarece semnarea fișierelor XML folosite de SAML fără a introduce breșe de securitate este foarte dificilă comparată cu semnarea obiectelor JSON folosite de JWT, cel din urmă este clar o opțiune mai rapidă, sigură și mai ușor de perceput.
- Tehnici de parsare ale unui obiect JSON sunt prezente în majoritatea limbajelor de programare deoarece se mapează direct la un obiect. XML nu beneficiază de asemenea funcționalitate.
- Folosit la scară mare pe internet ceea ce evidențiază dificultatea redusă de procesare a JWT pe platforme multiple, în special pe dispozitivele mobile.
- Folosit la scară mare pe internet ceea ce evidențiază dificultatea redusă de procesare a JWT pe platforme multiple, în special pe dispozitivele mobile.

¹⁸ <https://jwt.io/introduction/>

Anexa 11 - Librăria React

Reactul este o librărie de bazată pe componente ce este utilizată pentru crearea de interfețe interactive. La momentul actual este unul dintre cele mai folosite librării de javascript ce are o fundație stabilă și o comunitate enormă care îi oferă suport.¹⁹



Pentru o mai bună înțelegere al conceptului de bază al acestei librării vom lua ca și exemplu o casă construită din piese de LEGO. În același mod lucrează și Reactul, se construiesc piese mici numite componente iar apoi se pun toate împreună pentru a crea produsul dorit, componentele create fiind ușor de reutilizat.

Noutățile pe care această librărie le aduce sunt:

- Fișierele JSX ce sunt o combinație între fișierele de JavaScript și fișierele HTML. Practic într-un fișier JSX pot exista atât cod JS cât și tag-uri HTML
- DOM-ul virtual ce reprezintă un DOM secundar după care se randează DOM-ul principal. Acest DOM virtual vine cu funcționalitatea randării DOM-ului principal doar în cazul în care se observă o schimbare între DOM-ul virtual și cel principal
- Tot procesul de randare a paginii web se execută pe partea de client
- Simplitate – fișierele JSX fac ca aplicațiile scrise în React să fie ușor de înțeles

Anexa 12 - Framework-ul Material-UI

Material-UI este un framework dezvoltat de Google ce cuprinde o gamă variată de componente de React gata făcute și foarte bine documentate prin care se creează foarte ușor aplicații profesionale de front. Fiecare componentă furnizată de framework este responsive și poate fi configurată de utilizator prin diferite stiluri de CSS și funcții lambda.

¹⁹ [ps://www.edureka.co/blog/what-is-react/](https://www.edureka.co/blog/what-is-react/)

Bibliografie

- <https://www.maxcdn.com/one/visual-glossary/web-application/>
- https://www.tutorialspoint.com/java/java_overview.htm
- <https://whatis.techtarget.com/definition/framework>
- https://www.tutorialspoint.com/spring/spring_overview.htm
- <https://docs.spring.io/spring/docs/current/spring-framework-reference/overview.html>
- <http://www.tutorialsteacher.com/ioc/inversion-of-control>
- <https://spring.io/blog/2013/08/06/spring-boot-simplifying-spring-for-everyone>
- https://www.tutorialspoint.com/jpa/jpa_orm_components.htm
- https://www.tutorialspoint.com/hibernate/orm_overview.htm
- https://www.tutorialspoint.com/hibernate/hibernate_overview.htm
- <https://jwt.io/introduction/>
- <https://en.wikipedia.org/wiki/HTML>
- https://en.wikipedia.org/wiki/Cascading_Style_Sheets
- <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction>
- https://www.tutorialspoint.com/javascript/javascript_overview.htm
- <https://reactjs.org/>
- https://www.w3schools.com/whatis/whatis_react.asp
- <https://material-ui.com/>
- <https://www.callicoder.com/spring-boot-spring-security-jwt-mysql-react-app-part-1/>