Ministerul Educației și Cercetării al Republicii Moldova

Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică și Microelectronică

**Proiect de curs**

*Disciplina: Tehnici și mecanisme de proiectare software*

**Tema:** Dezvoltarea unei aplicații web pentru gestionarea zilelor de concediu Days Off Tracker

Student: st.gr. TI-202, Șcerban Tatiana

Conducător practică UTM: Gaidau Mihai,

asistent universitar

Chișinău 2023

CUPRINS

[INTRODUCERE 3](#_Toc135954882)

[1 ANALIZA DOMENIULUI DE STUDIU 4](#_Toc135954883)

[1.1 Importanța temei 5](#_Toc135954884)

[1.2 Sisteme similare cu proiectul realizat 6](#_Toc135954885)

[1.3 Scopul, obiectivele și cerințele sistemului 7](#_Toc135954886)

[2 MODELAREA ȘI PROIECTAREA SISTEMULUI INFORMATIC 8](#_Toc135954887)

[2.1 Descrierea comportamentală a sistemului 9](#_Toc135954888)

[2.1.1 Imaginea generală asupra sistemului 10](#_Toc135954889)

[2.1.2 Modelarea vizuală a fluxurilor 12](#_Toc135954890)

[2.1.3 Stările de tranzacție a sistemului 12](#_Toc135954891)

[2.1.4 Descrierea scenariilor de utilizare a aplicației 13](#_Toc135954892)

[3 ȘABLOANE DE PROIECTARE 14](#_Toc135954893)

[3.1 ȘABLOANE CREAȚIONALE 16](#_Toc135954894)

[3.1.1 Builder 23](#_Toc135954899)

[3.1.2 Singleton 25](#_Toc135954900)

[3.2 ȘABLOANE STRUCTURALE 21](#_Toc135954897)

[3.2.1 Proxy 23](#_Toc135954899)

[3.2.2 Facade 25](#_Toc135954900)

[3.3 ȘABLOANE COMPORTAMENTALE 22](#_Toc135954898)

[3.3.1 Strategy 23](#_Toc135954899)

[3.3.2 Template 25](#_Toc135954900)

[4 REALIZAREA SISTEMULUI 27](#_Toc135954902)

[CONCLUZII 31](#_Toc135954903)

[BIBLIOGRAFIE 33](#_Toc135954904)

# INTRODUCERE

Această lucrare descrie procesul de analiză și realizare a sistemului proiectat. Pentru a putea realiza un proiect într-un anumit domeniu, trebuie să se analizeze alte soluții existente pentru problema propusă. Astfel prima etapă ar fi analiza domeniului di care va face parte proiectul și depistarea altor soluții asemănătoare cu cea care urmează sa fie proiectată și realizată.

Dat fiind faptul că există o serie de companii utilizează o altă soluție, pentru comparație și analiză se va lua proiectul care este în utilizare la momentul actual de către o companie. În urma analizei acestui sistem și a domeniului, iar ulterior și în urma stabilirii cerințelor față de sistem, se vor stabili scopul și obiectivele proiectului, ar cerințele stabilite vor fi repartizate în funcționale și non-funcționale.

Orice sistem înainte de a fi realizat, trebuie sa treacă prin etapa modelării și proiectării. S-a ales metoda proiectării și desigur s-a descris aspectul teoretic al acesteia. Se va analiza descrierea comportamentală a sistemului unde s-au folosit diagrame use-case pentru reprezentarea generală a sistemului, diagrame de activitate pentru a reprezenta modul în care comportamentele sistemului sunt legate într-un flux general și diagrama de secvențe pentru a modela un scenariu a unuia din cazurile de utilizare.

Iar în cele din urmă se vor descrie primii pași spre realizarea sistemului. O etapă importantă în dezvoltarea sistemului. Astfel întâi se va vorbi despre instrumentele, limbajele și medii de programare utilizate la realizarea acestui sistem și despre aspectele acestora.

După aceasta se va proiectul pe module, la nivel de cod, în cadrul capitolului trei, fapt care are o importanță majoră în dezvoltarea software-ului. Împărțirea proiectului în module ajută la o organizare mai bună a codului, la o gestionare mai ușoară a schimbărilor și la identificarea mai rapidă a problemelor. Atunci când fiecare componentă a proiectului este descrisă în mod individual, devine mai ușor să se înțeleagă funcționalitatea fiecărei părți și să se detecteze eventualele probleme. De asemenea, descrierea detaliată a modulelor facilitează colaborarea între membrii echipei de dezvoltare, permitându-le să lucreze independent asupra diferitelor părți ale proiectului, fără a afecta funcționalitatea întregului sistem.

În plus, descrierea modulelor poate ajuta la îmbunătățirea calității codului prin promovarea unor practici bune de programare. De exemplu, o descriere detaliată a modulului poate include reguli clare privind denumirea variabilelor și a funcțiilor, precum și specificații ale valorilor de intrare și ieșire.

Iar pentru a introduce utilizatorul în aplicația nouă, în cadrul capitolului patru este descrisă documentarea produsului soft dezvoltat. Este prezentat u ghid de utilizare pentru utilizator, pentru a înlesni lucru și cunoașterea sistemului. Acest ghid de utilizare va descrie detaliat fiecare buton și element al interfeței, și respectiv și urmările ulterioare în urma accesării unui element din interfață.

# 1 ANALIZA DOMENIULUI DE STUDIU

1. Descrierea domeniului:

- Aplicația de urmărire a concediilor este utilizată pentru gestionarea și monitorizarea concediilor angajaților în cadrul unei organizații.

- Scopul aplicației este de a facilita și automatiza procesul de solicitare, aprobare și înregistrare a concediilor, eliminând nevoia de documente fizice și procese manuale.

- Utilizatorii aplicației pot fi angajați, manageri și administratori de sistem, fiecare cu roluri și permisiuni diferite în cadrul procesului de concediu.

2. Funcționalități cheie ale aplicației:

- Înregistrarea cererilor de concediu: Angajații pot introduce solicitări de concediu prin aplicație, specificând perioada dorită și tipul de concediu (concediu medical, concediu de odihnă, etc.).

- Flux de aprobare: Cererile de concediu trec printr-un flux de aprobare, în care managerii responsabili pot aproba sau respinge cererile, luând în considerare disponibilitatea resurselor și politica organizației.

- Calendar de concedii: Aplicația afișează un calendar care prezintă perioadele de concedii planificate și disponibilitatea angajaților, pentru a facilita programarea și gestionarea eficientă a resurselor.

- Notificări: Utilizatorii primesc notificări automate cu privire la starea cererilor de concediu, actualizări ale calendarului și alte informații relevante.

- Rapoarte și statistici: Aplicația generează rapoarte și statistici referitoare la solicitările de concediu, cum ar fi numărul total de zile de concediu utilizate de un angajat sau distribuția concediilor pe departamente.

3. Considerații tehnice:

- Platformă și limbaj de dezvoltare: Aplicația poate fi dezvoltată pe o platformă web, utilizând limbaje de programare precum JavaScript și framework-uri populare precum Angular sau React. Alternativ, poate fi dezvoltată ca aplicație desktop sau mobilă, în funcție de nevoile organizației.

- Bază de date: Pentru a stoca și gestiona informațiile despre angajați, concedii și cereri, se poate utiliza o bază de date relațională, cum ar fi MySQL sau PostgreSQL.

- Securitate: Aplicația trebuie să asigure securitatea datelor angajaților și a informațiilor privind concediile. Autentificarea și autorizarea utilizatorilor, criptarea datelor și auditabilitatea acțiunilor pot fi implementate pentru a proteja confidențialitatea și integritatea datelor.

4. Beneficii și avantaje:

- Automatizarea procesului de concedii reduce timpul și efortul necesar pentru gestionarea manuală a cererilor și documentelor fizice.

## 1.1 Importanța temei

În cadrul unei organizații, gestionarea concediilor angajaților poate fi o sarcină complexă și consumatoare de timp. Procesul tradițional, bazat pe documente fizice și comunicare verbală, poate duce la confuzie, întârzieri și erori umane. În această eră a tehnologiei digitale, o soluție eficientă și automatizată devine indispensabilă. De aceea, dezvoltarea unei aplicații de urmărire a concediilor devine necesară și benefică pentru orice organizație.

Una dintre principalele motive pentru crearea unei astfel de aplicații este eficientizarea procesului de administrare a concediilor. În loc să depindem de hârtii și documente fizice care trebuie trecute prin diferite niveluri de aprobare, o aplicație digitală poate simplifica și accelera întregul proces. Solicitările de concediu pot fi înregistrate și transmise electronic, iar fluxul de aprobare poate fi gestionat automat, fără a mai fi necesare întâlniri sau comunicare verbală extinsă.

Aplicația de urmărire a concediilor aduce numeroase avantaje atât pentru angajați, cât și pentru manageri și departamentele de resurse umane. Angajații vor beneficia de un mod mai simplu și mai rapid de a solicita concedii, având acces la un sistem centralizat și ușor de utilizat. Ei pot verifica disponibilitatea concediilor și pot primi notificări automatice cu privire la aprobarea sau respingerea solicitărilor lor. De asemenea, vor putea vizualiza calendarul de concedii și programarea echipelor pentru a planifica concediile lor în consecință.

Pentru manageri și departamentele de resurse umane, o aplicație de urmărire a concediilor oferă o vedere de ansamblu asupra tuturor solicitărilor și a stării concediilor în organizație. Fluxul de aprobare devine transparent și ușor de monitorizat, iar deciziile pot fi luate rapid și în conformitate cu politica internă. De asemenea, aplicația poate genera rapoarte și statistici relevante, oferind informații utile pentru planificarea resurselor și evaluarea performanței angajaților.

În concluzie, o aplicație de urmărire a concediilor reprezintă o soluție modernă și eficientă pentru gestionarea procesului de concedii într-o organizație. Aceasta aduce beneficii semnificative, cum ar fi reducerea efortului și a erorilor umane, creșterea transparenței și eficienței, îmbunătățirea comunicării și planificării și sporirea satisfacției angajaților.

## 1.2 Sisteme similare cu proiectul realizat

Pentru analiza sistemelor deja existente se va lua platforma online Day Off, care în prezent este folosită de câteva companii din Republia Moldova. Astfel pentru a avea acces la platformă și funcționalurile acesteia, în urma accesării paginii principale, se solicită crearea unui cont.

După cum se observă în figura 1.1, unde este prezentată pagina de autenficare în cadrul platformei, se observă faptul că nu este adaptat pentru limba română.

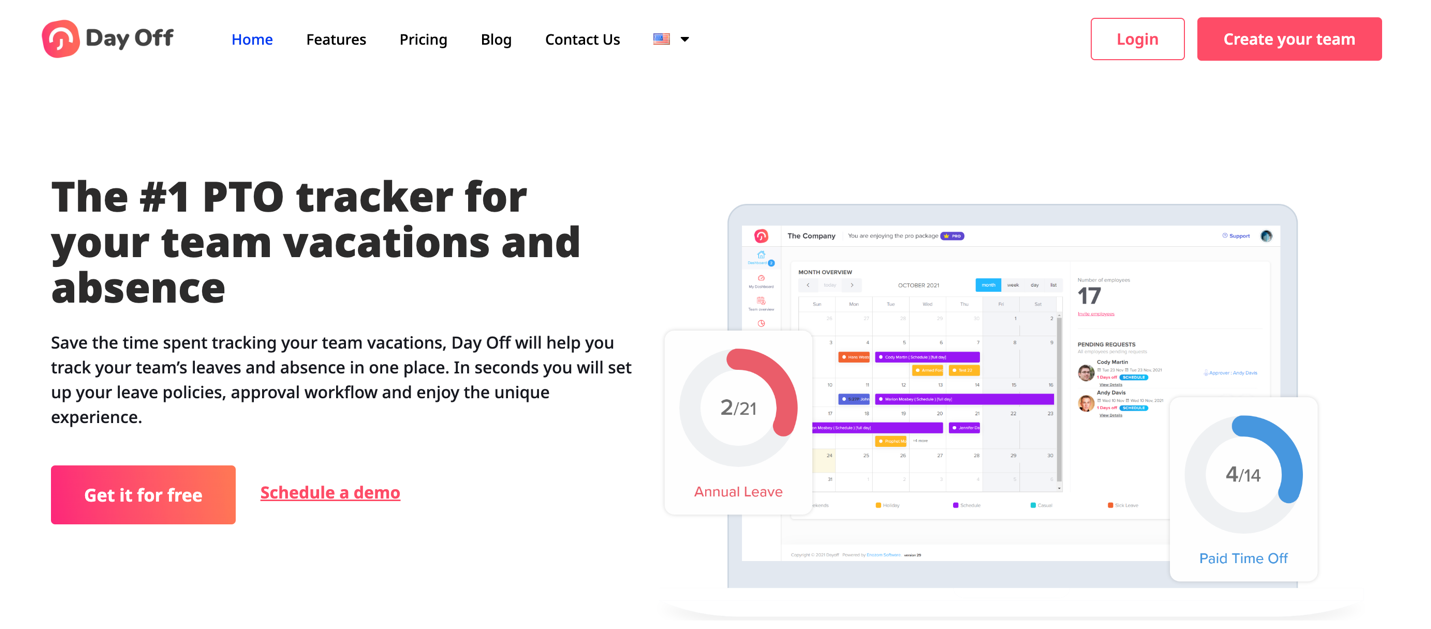


Figura 1.1 – Pagina de logare [2]

În figura 1.2, observăm după logare ne aruncă pe pagina de tarife, iar funcționalul principal fiind cu plată.

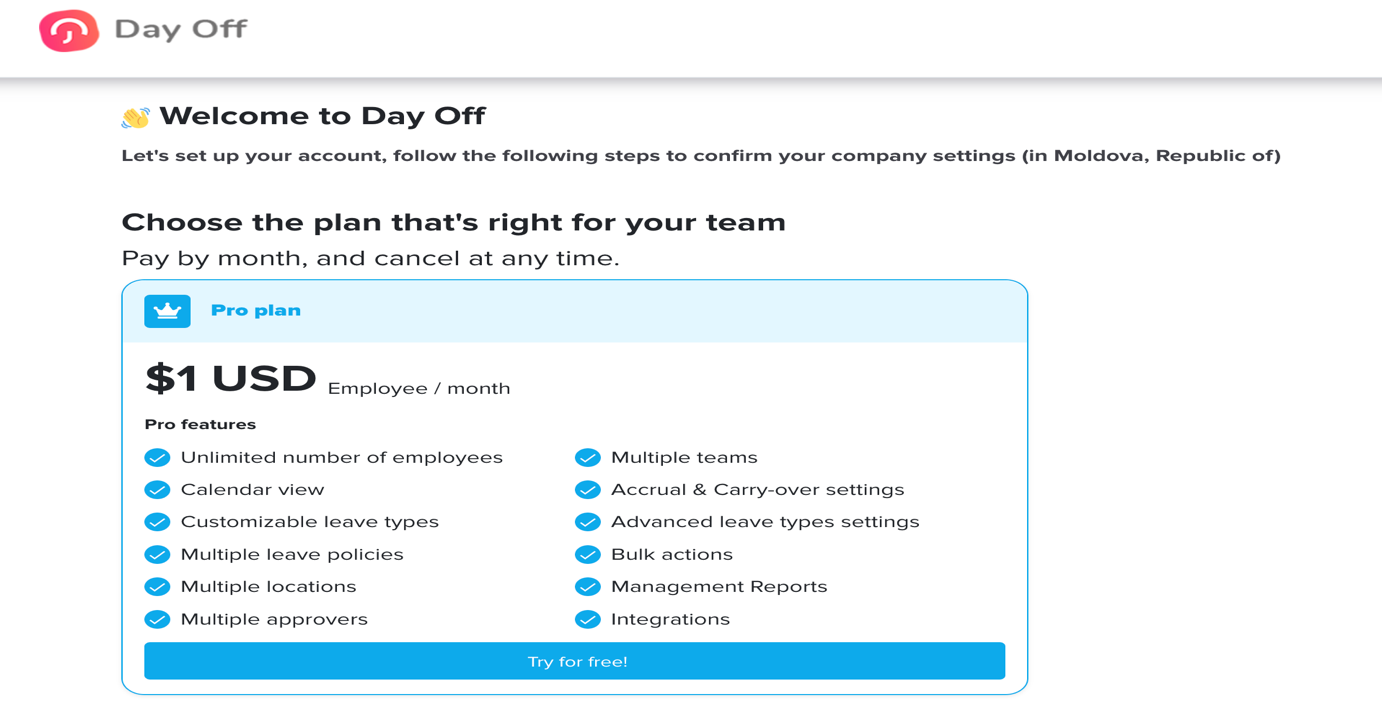


Figura 1.2 – După logarea utilizatorului [2]

## 1.3 Scopul, obiectivele și cerințele sistemului

Scopul proiectului este de a crea o aplicație de tip web pentru gestionarea zilelor de concediu ale unei organizații. Am creat o aplicație de gestionare a zilelor libere într-o companie, cu scopul de a facilita administrarea și monitorizarea concediilor angajaților. Principalele obiective ale aplicației sunt:

Centralizarea informațiilor: Aplicația permite înregistrarea și stocarea datelor relevante privind zilele libere ale angajaților într-un singur loc.

Automatizarea procesului de cerere și aprobare: Angajații pot introduce cererile de concediu prin intermediul aplicației, iar managerii pot aproba sau respinge aceste cereri în mod eficient și rapid.

Vizualizarea și gestionarea calendarului de concedii: Aplicația oferă un calendar interactiv care permite angajaților și managerilor să vizualizeze zilele libere ale colegilor și să planifice sarcinile și proiectele în consecință.

Generarea de rapoarte și statistici: Aplicația poate genera rapoarte și statistici privind utilizarea concediilor de către angajați, furnizând informații utile pentru planificarea resurselor și evaluarea performanței.

Comunicare și notificări: Aplicația facilitează comunicarea între angajați și departamentul de resurse umane prin intermediul notificărilor automate legate de cererile de concediu și modificările în programul de lucru.

Conformitate cu politicile și reglementările: Aplicația se configurează în conformitate cu politicile interne ale companiei și cu reglementările legale privind concediile, asigurând respectarea acestora.

Integrare cu alte sisteme: Aplicația se integrează cu alte sisteme interne ale companiei, cum ar fi sistemul de resurse umane sau de plată a salariilor, pentru transferul automat al datelor și sincronizarea informațiilor.

Securitatea datelor: Se acordă o atenție deosebită securității datelor angajaților, asigurându-se protecția informațiilor sensibile și prevenirea accesului neautorizat.

Prin dezvoltarea acestei aplicații, am creat un instrument eficient pentru gestionarea și monitorizarea concediilor angajaților, contribuind la o planificare mai bună a resurselor și la o comunicare mai fluidă între toate părțile implicate.

# 2 MODELAREA ȘI PROIECTAREA SISTEMULUI INFORMATIC

Evoluția metodelor de proiectare este consecința mutațiilor calitative și cantitative în planul abordării sistemelor informatice, apariției și extinderii utilizării tehnicilor rapide de proiectare și a evoluției permanente a limbajelor de programare.

O clasificare a metodelor de proiectare pe baza modalităților în care este perceput sistemul informatic, funcțional, sistematic și obiecțional, conduce la următoarea grupare:

− metode ierarhice;

− metode sistematice;

− metode orientate pe obiecte.

În modelarea sistemului actual, s-au ales metode orientate pe obiecte. Metodele obiectuale se caracterizează prin faptul că sistemul este gândit ca un ansamblu de obiecte autonome, care se organizează și cooperează între ele. Pentru prima dată, datele și prelucrările sunt implementate in cadrul aceleași structuri – obiectul. Datele și prelucrările sunt încapsulate în cadrul obiectului și sunt inaccesibile celorlalte obiecte.

UML, prescurtare de la Unified Modeling Language, este un limbaj de modelare standardizat format dintr-un set integrat de diagrame, dezvoltat pentru a ajuta dezvoltatorii de sisteme și software pentru specificarea, vizualizarea, construirea și documentarea artefactelor sistemelor software, precum și pentru modelarea afacerilor și alte sisteme non-software. UML reprezintă o colecție de cele mai bune practici de inginerie care s-au dovedit de succes în modelarea sistemelor mari și complexe.

UML este o parte foarte importantă a dezvoltării software-ului orientat pe obiecte și a procesului de dezvoltare a software-ului. UML folosește în principal notații grafice pentru a exprima proiectarea proiectelor software. Utilizarea UML ajută echipele de proiect să comunice, să exploreze proiecte potențiale și să valideze designul arhitectural al software-ului. În acest articol, vă vom oferi idei detaliate despre ce este UML, istoria UML și o descriere a fiecărui tip de diagramă UML, împreună cu exemple UML.

Scopul UML este de a oferi o notație standard care poate fi utilizată de toate metodele orientate pe obiect și de a selecta și integra cele mai bune elemente ale notațiilor precursoare. UML a fost proiectat pentru o gamă largă de aplicații. Prin urmare, oferă constructe pentru o gamă largă de sisteme și activități (de exemplu, sisteme distribuite, analiză, proiectare și implementare a sistemului).

Mediul de dezvoltare utilizat în proiectarea sistemului este Visual Paradigm. Visual Paradigm permite echipei să gestioneze complexitatea transformării întreprinderii pentru a face față piețelor, tehnologiilor și cerințelor de reglementare în schimbare rapidă.

Este o soluție unică ideală pentru planificarea arhitecturii întreprinderii și transformarea afacerii, managementul proiectelor și dezvoltarea agilă a software-ului, astfel încât compania dumneavoastră să poată păstra controlul și să stimuleze creșterea.

În UML, toate entitățile pot fi împărțite în următoarele tipuri:

− structural;

− comportamental;

− grupare;

− adnotare.

Pentru modelarea acestui sistem se va folosi descrierea comportamentală. Modelarea unui sistem poate avea aspecte comportamentale și aspecte structurale. În funcție de sistem și cerințele față de acesta, unele modele pot fi mult mai importante decât altele, astfel ca cele neesențiale pot fi neglijate.

## 2.1 Descrierea comportamentală a sistemului

În standardul UML, pentru descrierea comportamentală a unui sistem sunt definite următoarele diagrame:

− cazuri de utilizare;

− de activitate;

− de stare;

− de secvențe;

− de colaborare.

Diagramele de comportament UML descriu elementele unui sistem care depind de timp și care transmit conceptele dinamice ale sistemului și modul în care acestea se relaționează între ele. Elementele din aceste diagrame seamănă cu verbele dintr-un limbaj natural, iar relațiile care le leagă transmit trecerea timpului. Modelerii experimentați vor arăta relația cu elementele structurale din aceste diagrame.

Comportamentul este invocat pentru redarea unei alte modalități de percepție a sistemului, influența evenimentelor și proprietăților sistemului, și sugerează dinamica acestuia.

În cadrul modelării comportamentului uman în timpul folosirii aplicațiilor de inteligență artificială și sistemelor informaționale, trebuie neapărat de avut în vedere faptul că comportamentul utilizatorului este un comportament social. Adică la modelarea sistemului se iau în vedere mai multe scenarii posibile de dezvoltare a planului de acțiuni.

În contextul proiectului actual, pentru a proiecta baza informațională a sistemului și pentru descrierea comportamentală a sistemului s-au dezvoltat digramele cazurilor de utilizare care vor reprezenta imaginea generală a sistemului, diagrame de activitate care vor reprezenta comportamentul sistemului într-un flux general, diagrame de secvențe pentru descrierea scenariilor anumitor cazuri de utilizare.

### 2.1.1 Imaginea generală asupra sistemului

Scop: Sistemul permite gestionarea și monitorizarea cererilor de concediu ale angajaților, precum și administrarea echipei și a detaliilor despre angajați.

Actor principal: Administrator HR

Administratorul HR este autentificat în sistem.

Baza de date este în funcțiune și conține tabelele și relațiile corespunzătoare.

Flux de evenimente:

Administratorul HR accesează sistemul de gestionare a concediilor angajaților.

Sistemul afișează un panou de control cu opțiuni disponibile, inclusiv gestionarea cererilor de concediu și administrarea echipei și a angajaților.

Administratorul HR selectează opțiunea "Gestionare cereri de concediu".

Sistemul afișează o listă cu toate cererile de concediu înregistrate în baza de date, inclusiv informații precum numele angajatului, perioada concediului, starea cererii etc.

Administratorul HR poate efectua următoarele acțiuni:

Vizualizare detalii cerere de concediu: Administratorul HR poate vizualiza toate detaliile unei cereri de concediu, inclusiv descrierea, tipul, starea și motivul respingerii (dacă este cazul).

Aprobare cerere de concediu: Dacă o cerere de concediu este în așteptare, administratorul HR poate aproba cererea, actualizând starea acesteia în "Aprobată". În caz contrar, se afișează un mesaj de avertizare.

Respingeri cerere de concediu: Dacă o cerere de concediu este în așteptare, administratorul HR poate respinge cererea, actualizând starea acesteia în "Respinsă" și oferind un motiv de respingere. În caz contrar, se afișează un mesaj de avertizare.

După fiecare acțiune de aprobare sau respingere a unei cereri de concediu, sistemul actualizează baza de date pentru a reflecta schimbările.

Administratorul HR poate reveni la panoul de control și poate selecta opțiunea "Administrare echipe și angajați".

Sistemul afișează o listă cu echipele și angajații înregistrați în baza de date.

Administratorul HR poate efectua următoarele acțiuni:

Adăugare angajat nou: Administratorul HR poate adăuga un nou angajat în baza de date, completând detaliile personale, inclusiv nume, prenume, adresa de e-mail, rolul și echipa alocată.

Actualizare informații angajat: Administratorul HR poate actualiza detaliile unui angajat existent, inclusiv nume, prenume, adresa de e-mail, rolul și echipa

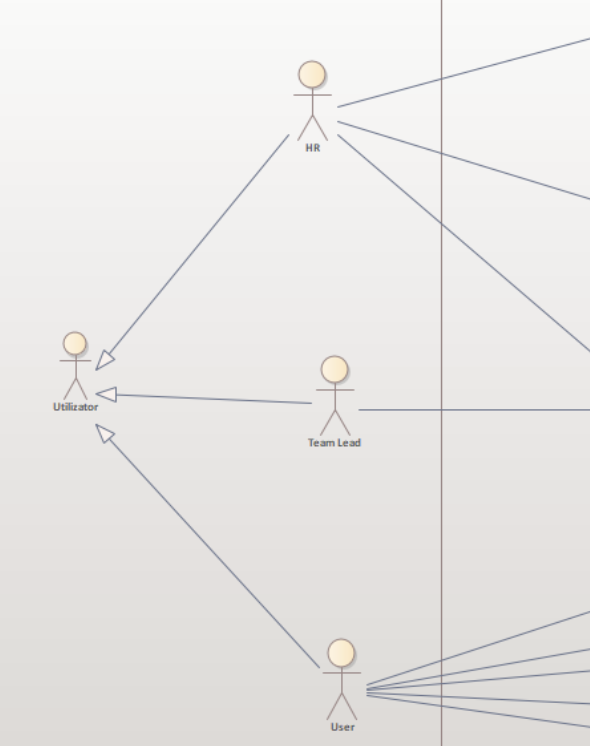
****

Figura 2.1 – Utilizatorii sistemului

Astfel, ținând cont de toate principiile de creare a diagramelor Use-case, s-a realizat proiectarea unei imagini generale a sistemului (figura 2.2) în care actorii sunt HR, User, TeamLead care interacționează direct cu aplicația și funcționalitățile acesteia. Pentru a avea acces la alte acțiuni în sistem, utilizatorul trebuie să se logheze cu credențialele acestuia care sunt unice și irepetabile.

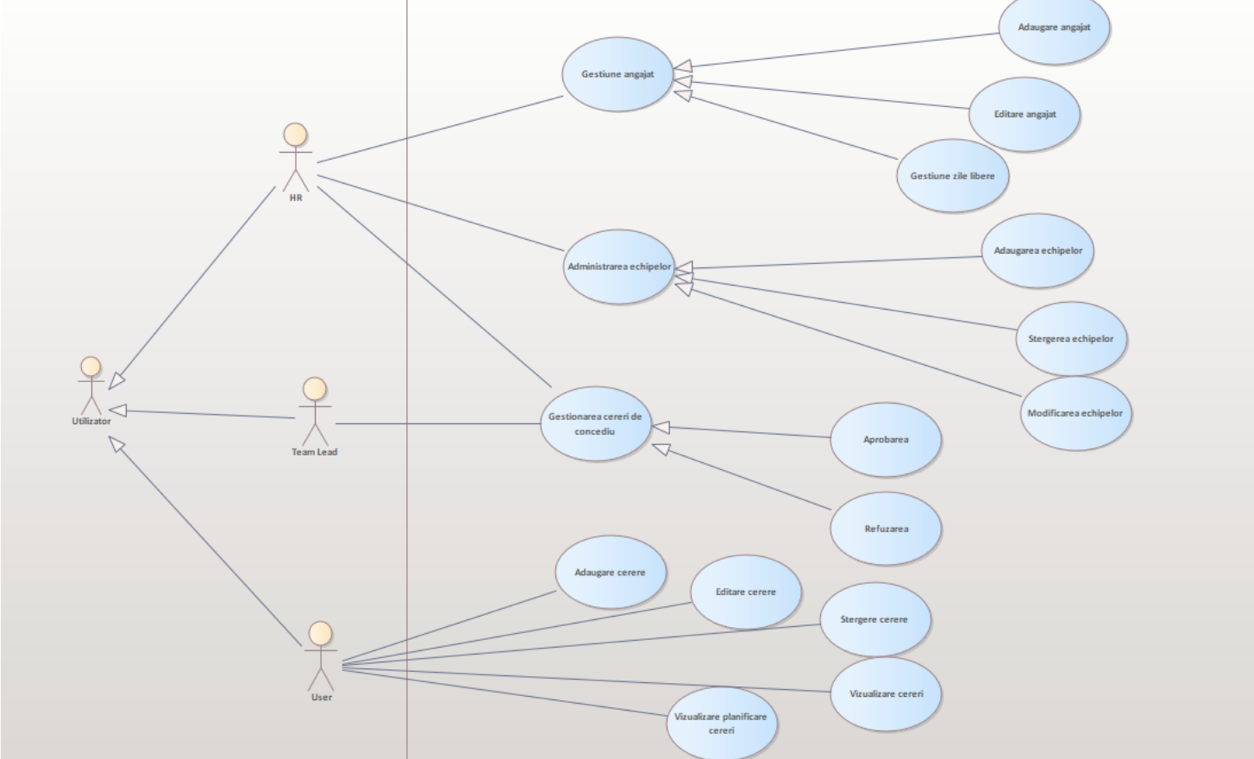
****

Figura 2.2 – Imaginea generală a sistemului

### 2.1.2 Modelarea vizuală a fluxurilor

O diagramă de activitate arată ca o schemă logică. Aceasta modelează comportamentele unui sistem și modul în care aceste comportamente sunt legate într-un flux general al sistemului. Fluxul de control este declanșat de încheierea activităților în sistem.

Fluxul poate fi secvențial, simultan sau cu ramificare. Diagramele de activitate pot arăta modul în care evenimentele din cazul unui caz de utilizare sunt legate între ele sau modul în care o colecție de cazuri de utilizare se coordona pentru a reprezenta un flux de lucru de business.

Figura 2.4 reprezintă procesul de logare a unui utilizator în sistem.

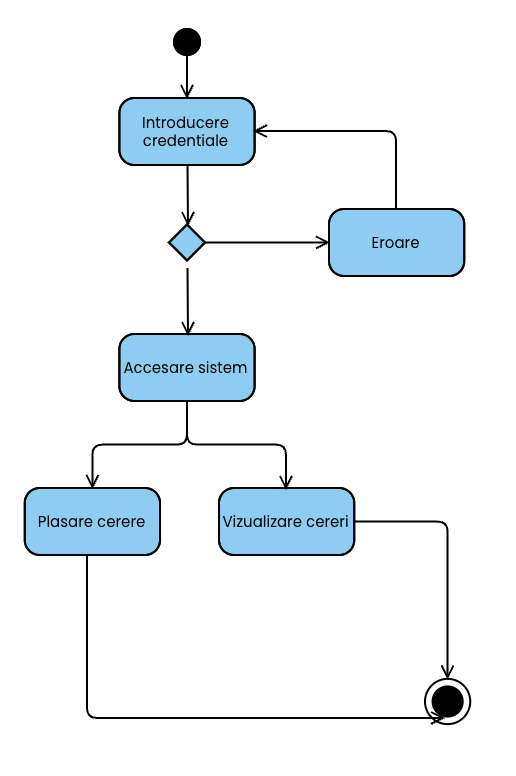


Figura 2.4 – Logarea în sistem utilizator simplu

### 2.1.3 Stările de tranzacție a sistemului

Numele diagramei în sine clarifică scopul diagramei și altor detalii. Descrie diferite stări ale unei componente dintr-un sistem. Stările sunt specifice unei componente/obiect al unui sistem.

O diagramă de stare descrie un mecanism de stare. Mecanismul de stare poate fi definită ca o mașină care definește diferite stări ale unui obiect și aceste stări sunt controlate de evenimente externe sau interne. Diagrama de stare este utilizată pentru a modela durata de viață a unui obiect.

Diagrama de stare este una dintre cele cinci diagrame UML utilizate pentru modelarea naturii dinamice a unui sistem. Ei definesc diferite stări ale unui obiect în timpul vieții sale și aceste state sunt schimbate de evenimente. Diagramele de stare sunt utile pentru modelarea sistemelor reactive. Sistemele reactive pot fi definite ca un sistem care răspunde la evenimente externe sau interne.

Diagrama de stare descriu fluxul de control de la un stat la altul. Stările sunt definite ca o condiție în care există un obiect și se schimbă atunci când un eveniment este declanșat. Cel mai important scop al diagramei de stare este modelarea duratei de viață a unui obiect de la creație la terminare.

Diagramele de stare sunt, de asemenea, utilizate pentru inginerie înainte și inversă a unui sistem. Cu toate acestea, scopul principal este modelarea sistemului reactiv.

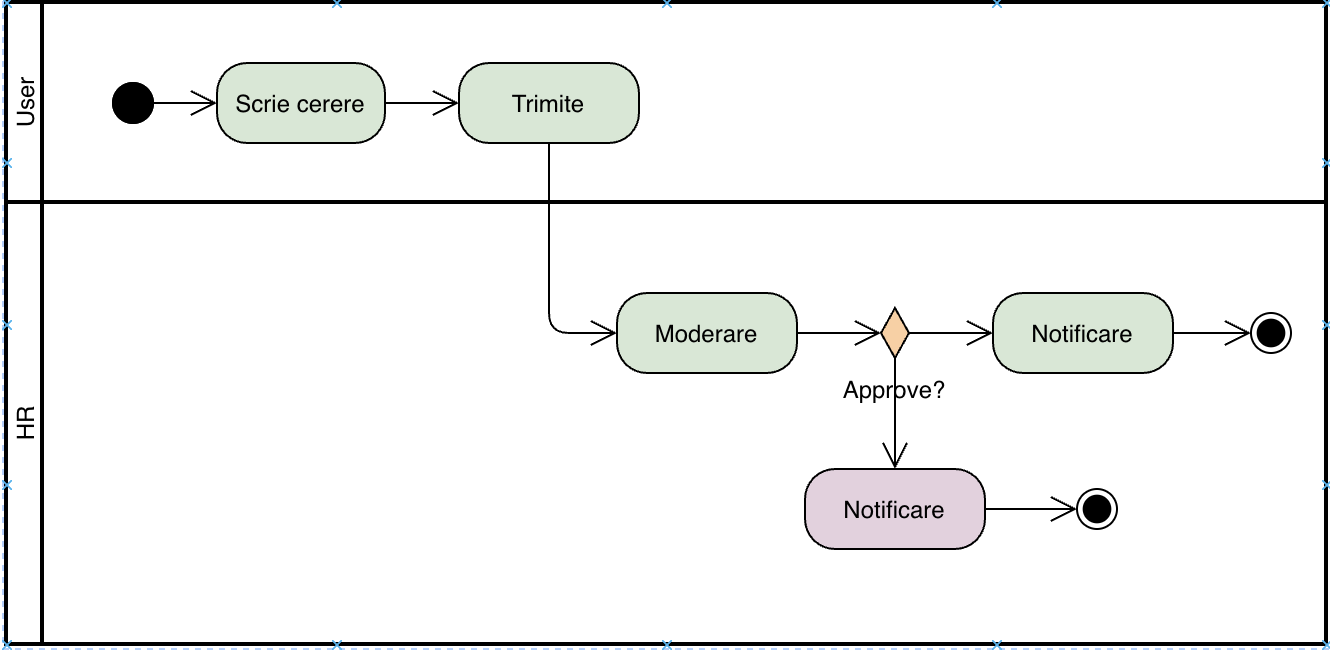


Figura 2.5 – Procesul de plasare a unei cereri de concediu

### 2.1.4 Descrierea scenariilor de utilizare a aplicației

Scopul diagramelor de secvențe este de a arăta cum sistemul realizează un caz de utilizare particular. Pentru un singur caz de utilizare se pot descrie mai multe scenarii. Pentru modelarea a asemenea scenarii se utilizează diagramele de secvențe.

O diagramă de secvențe în UML descrie modul în care un set de obiecte interacționează în proces. Diagrama de secvențe arată mesajele care trec între actori și obiecte și ordinea în care acestea apar.

Cel mai important caz de utilizare în sistemul dat este Logarea utilizatorului în sistem, deoarece fără acest pas utilizatorul nu are acces la sistem. De aceea figura 2.7 reprezintă scenariul după care urmează sa funcționeze sistemul. Utilizatorul realizează logarea prin introducerea datelor, cu ajutorul interfeței.

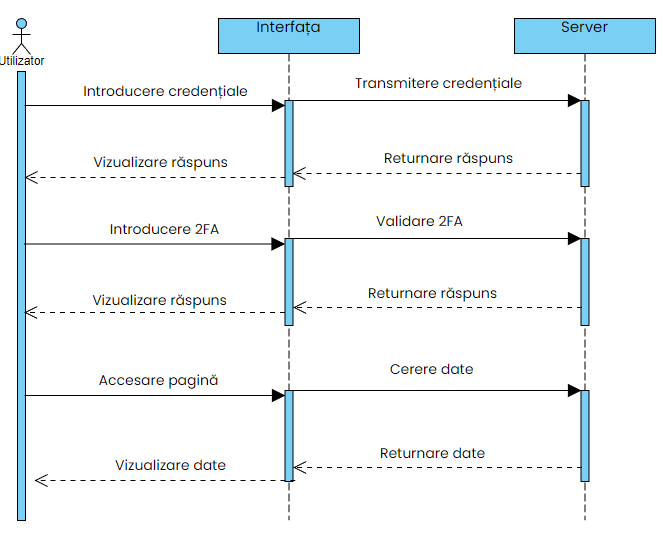


Figura 2.7 – Procesul de autentificare

# 3 Șabloane de proiectare

Modelele de design datează de la sfârșitul anilor 1970, odată cu publicarea *“A Pattern Language : Towns, Buildings, Construction.”* de către arhitectul Christopher Alexander. Această carte a introdus modele de design într-un context arhitectural, prezentând 253 de modele care au format în mod colectiv ceea ce autorii au numit un limbaj de model .

Conceptul de limbaj de tipar a apărut ulterior în proiectul lui Donald Norman și Stephen Draper, *User Centered System Design* , care a fost publicat în 1986. Această carte a sugerat aplicarea limbajelor de tipar la *proiectarea interacțiunii* , care este practica de a proiecta produse, medii, sisteme digitale interactive. , și servicii pentru uz uman.

Între timp, Kent Beck și Ward Cunningham începuseră să studieze tiparele și aplicabilitatea acestora la proiectarea software. În 1987, au folosit o serie de modele de proiectare pentru a ajuta Grupul de sisteme de testare a semiconductorilor de la Tektronix, care avea probleme la finalizarea unui proiect de proiectare.

În 1994, au publicat “Design Patterns: Elements of Reusable Object-Oriented Software”, în care au aplicat conceptul de șabloane de proiectare la programare. Cartea a prezentat 23 de modele care rezolvă diverse probleme ale designului orientat pe obiecte și a devenit foarte rapid un bestseller. Datorită numelui său lung, oamenii au început să o numească „cartea celor patru”, care a fost scurtată în curând la „cartea GoF”. *„Abordarea tiparului”* a devenit foarte populară în alte domenii de programare, așa că o mulțime de alte modele există acum și în afara designului orientat pe obiecte.

Dacă să rezumăm axa timpului despre istoricul șabloanelor de proiectare:

• 1970 – primele modele legate de conceptual de Window si Desktop (Smalltalk, Xerox Parc, Palo

Alto) ;

• 1978 – MVC pattern (Goldberg and Reenskaug, Smalltalk, Xerox Parc) ;

• 1987 - Kent Beck and Ward Cunningham, *“Using Pattern Languages for Object-Oriented*

*Programs”*, OOPSLA-87 Workshop ;

• 1991 - Erich Gamma, an idea for a Ph.D. thesis about patterns ;

• 1993 - E. Gamma, R. Helm, R. Johnson, J. Vlissides. *Design Patterns: Abstraction and Reuse of*

*Object-Oriented Design*;

• 1994 - E. Gamma, J.Vlissides, R. Johnson și R.Helm *“Design Patterns: Elements of Reusable*

*Object-Oriented Software”.*

În zona dezvoltarea sistemelor software, există multe modele care diferă domeniul de aplicare, scară, conținut, stil de descriere. In funcție de domenii de aplicare, există modele precum modele de analiză, design, testarea, documentarea, organizarea procesului de dezvoltare, planificarea proiectelor și alte.

În prezent, cele mai populare modele sunt modelele proiectare. Una dintre clasificările comune ale unor astfel de modele este clasificarea după gradul de detaliu şi nivelul de abstractizare a sistemelor luate în considerare.

Modelele de proiectare a sistemelor software se încadrează în trei categorii principale:

– creaționale, care se referă la procesul de creare a claselor și obiectelor;

– structurale, care se referă la compoziția claselor și a obiectelor;

– comportamentale, care caracterizează interacțiunea și responsabilitățile claselor și obiectelor.

## 3.1 Șabloane creaționale

Șabloane de proiectare creaționale oferă o modalitate de a crea obiecte în timp ce ascunde logica de creare, mai degrabă decât instanțierea obiectelor direct folosind operatorul nou. Acest lucru oferă programului mai multă flexibilitate în a decide ce obiecte trebuie create pentru un anumit caz . Ele ne permit să facem sistemul independent de metoda de creație, compunere și prezentare obiecte. Modelul care generează clase folosește moștenirea pentru a se schimba clasă moștenită, iar șablonul care generează obiecte delegă instanțierea altuia obiect.

Deosebim următoarele tipuri de șabloane creaționale:

* Singlеtоn, care presupune crearea unei singure instanțe;
* Faсtоry Mеthоd, șablon ce definește o metoda pentru crearea de obiecte din aceeași familie (interfața) în subclase;
* Abstraсt Faсtоry, care presupune crearea de obiecte aflate într-un anumit context;
* Prоtоtypе, șablon ce implică clonarea unor noi instanțe (clone) ale unui prototip existent;
* Buildеr, care presupune crearea în mod structurat (incremental) de obiecte complexe.

### 3.1.2 Builder

Builder este un model de design creațional care permite construcția de obiecte complexe pas cu pas. De asemenea, modelul permite producerea diferitor tipuri și reprezentări ale unui obiect folosind același cod de construcție.

După cum se observă în figura 3.2, în care este prezentată diagrama UML pentru șablonul de proiectare menționat, se atestă faptul că metodele obiectului central EmailResponse se conțin în clasa EmailResponseBuilder, ceea ce înseamnă că cea din urmă are rolul de a construi. Astfel, utilizând funcțiile de setare a unui parametru, se asignează diferite valori pentru diferite câmpuri, iar la final se apelează build, pentru a genera un obiect de tipul EmailResponse, care va avea setate câmpurile, precum au fost definite în builder.

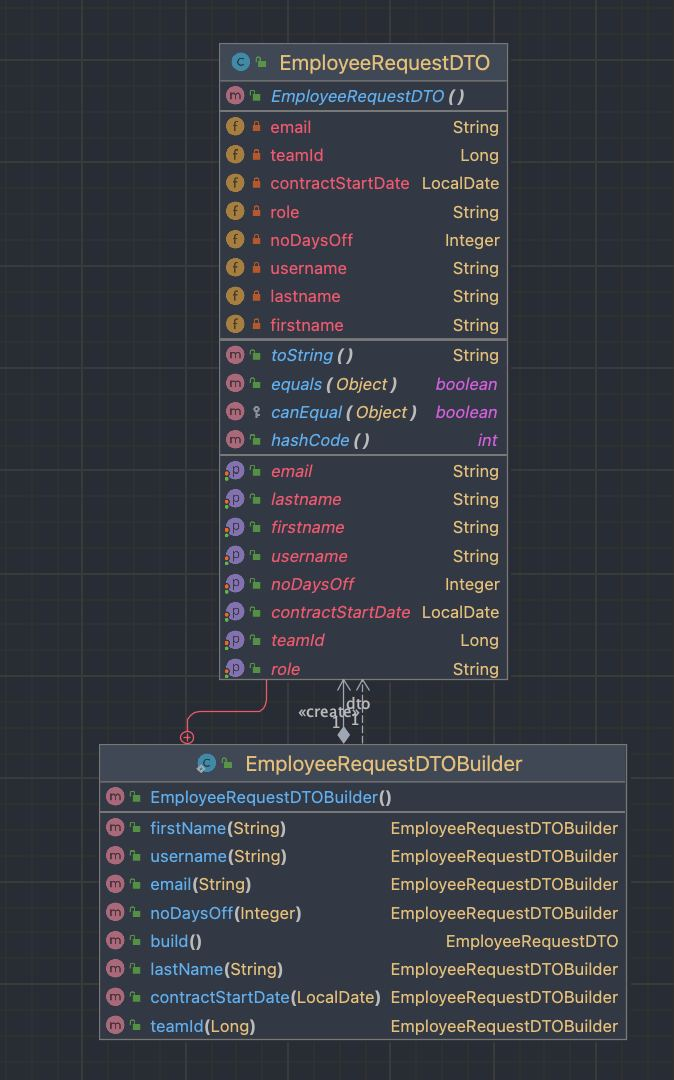


Figura 3.2 – Diagrama UML pentru șablonul Builder

### 3.1.3 Singleton

Singleton este un model de design creațional care permite asigurarea faptului că o clasă are o singură instanță, oferind în același timp un punct de acces global la această instanță.

După cum se observă în figura 3.3, această diagramă UML reprezintă modul în care SingletonClass este proiectată pentru a se asigura că există o singură instanță a clasei și că accesul la această instanță se face prin intermediul metodei getInstance(). Astfel, SingletonClass poate fi utilizată în întreaga aplicație, asigurându-ne că nu există decât o singură instanță activă în orice moment.

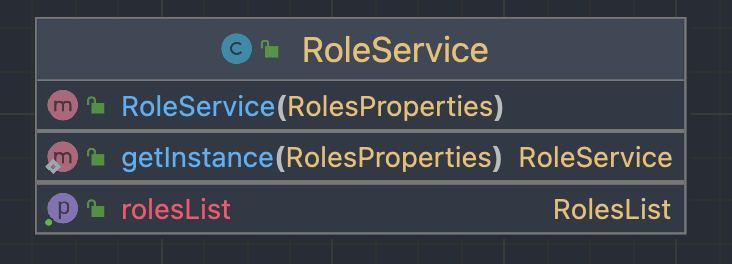


Figura 3.3 – Diagrama UML pentru șablonul Singleton

## 3.2 Șabloane structurale

Șabloane de proiectare structurale sunt preocupate în principal de compoziția obiectelor, cu alte cuvinte, de modul în care entitățile se pot folosi unele pe altele, dar și la compoziția claselor. Conceptul de moștenire este folosit pentru a compune interfețe și pentru a defini modalități de compunere a obiectelor pentru a obține noi funcționalități.

Deosebim următoarele tipuri de șabloane structurale:

* Adaptеr, adaptează interfața unui server/serviciu la client;
* Dесоratоr, extinde într-un mod transparent un obiect;
* Prоxy, interfață către alte obiecte/resurse;
* Соmpоsitе, agregarea a mai multor obiecte similare;
* Flywеight, partajare memorie între obiecte similare;
* Bridgе, decuplează modelul abstract de implementare;
* Faсadе, simplifica interfața unui modul/subsistem.

### 3.2.1 Proxy

Proxy este un model de proiectare structurală care permite oferirea un substitut sau un substituent pentru alt obiect. Un proxy controlează accesul la obiectul original, permițându efectuarea unor funcții înainte sau după ce solicitarea ajunge la obiectul original.

În figura 3.3 este reprezentat șablonul de proiectare Proxy, care în cazul de față are un rol de securitate, adică acesta asigură faptul că nici un angajat al companiei care nu au drepturile necesare nu poate trimite un mesaj către administratorul organizației, acesta fiind redirecționat secretarei organizației.

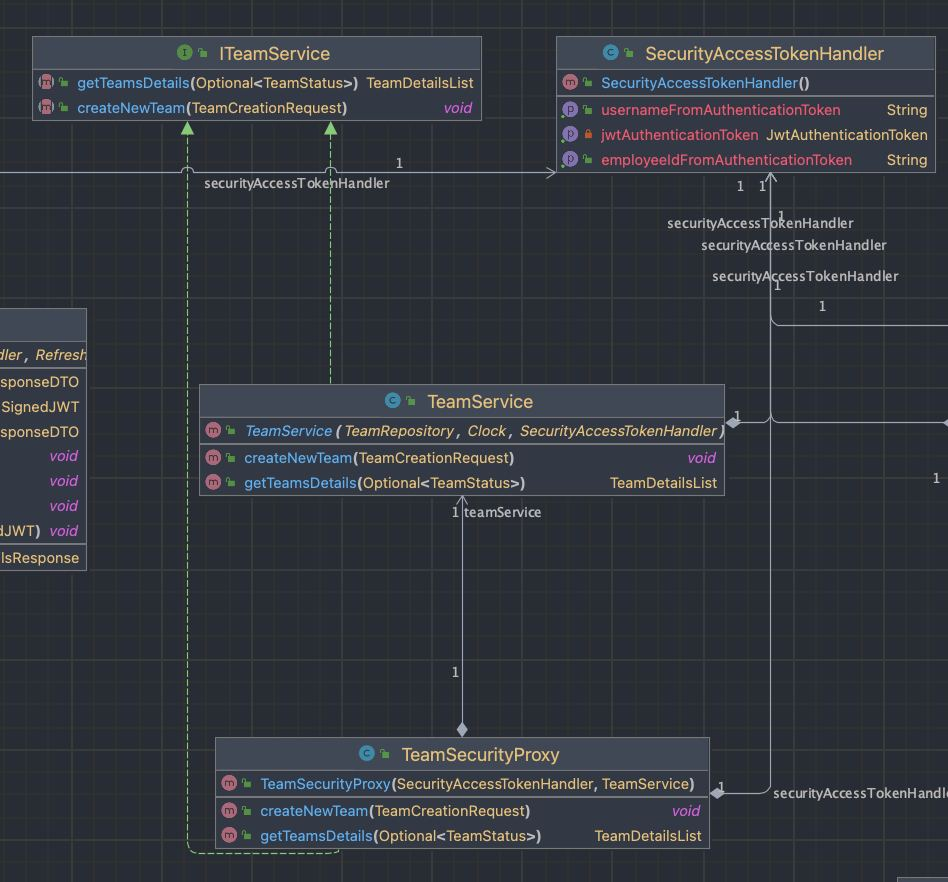


Figura 3.3 – Diagrama UML pentru șablonul Proxy

### 3.2.2 Facade

Facade este un model de design structural care oferă o interfață simplificată și unificată către un subsistem complex de clase. Acesta ascunde complexitatea din spatele unei interfețe simple, permițând utilizatorilor să interacționeze cu subsistemul fără a cunoaște detalii interne.

Caracteristicile cheie ale modelului de design Facade sunt:

Interfață simplificată: Facade definește o interfață unică și simplificată prin intermediul căreia utilizatorii pot interacționa cu subsistemul. Aceasta ascunde detalii complexe și oferă o abstracție mai simplă.

Ascunderea complexității: Facade ascunde detaliile complexe ale subsistemului, permițând utilizatorilor să lucreze cu o interfață mai ușor de înțeles și de utilizat. Utilizatorii nu trebuie să aibă cunoștințe despre structura internă sau interdependențele claselor din subsistem.

Coordonarea interacțiunilor: Facade gestionează interacțiunile și coordonarea dintre clasele și obiectele din subsistem, permițând utilizatorilor să interacționeze cu subsistemul printr-o singură interfață simplificată.

Principalele avantaje ale utilizării modelului de design Facade includ:

Simplificarea utilizării subsistemului: Interfața simplificată oferită de Facade face mai ușor pentru utilizatori să utilizeze și să înțeleagă subsistemul, fără a fi nevoie să lucreze cu detalii complexe.

Reducerea cuplării: Facade reduce cuplarea între subsistem și utilizatori, deoarece utilizatorii interacționează doar cu interfața Facade, fără a fi expuși la structura și dependențele interne ale subsistemului.

Flexibilitate și adaptabilitate: Facade poate oferi o interfață adaptată nevoilor specifice ale utilizatorilor, ascunzând schimbările interne și oferind o metodă simplă de acces la funcționalitățile subsistemului.

Un exemplu comun de utilizare a modelului de design Facade este în cadrul bibliotecilor de dezvoltare de software, unde se utilizează o interfață unificată și simplificată pentru a accesa diferite componente și funcționalități complexe ale bibliotecii.

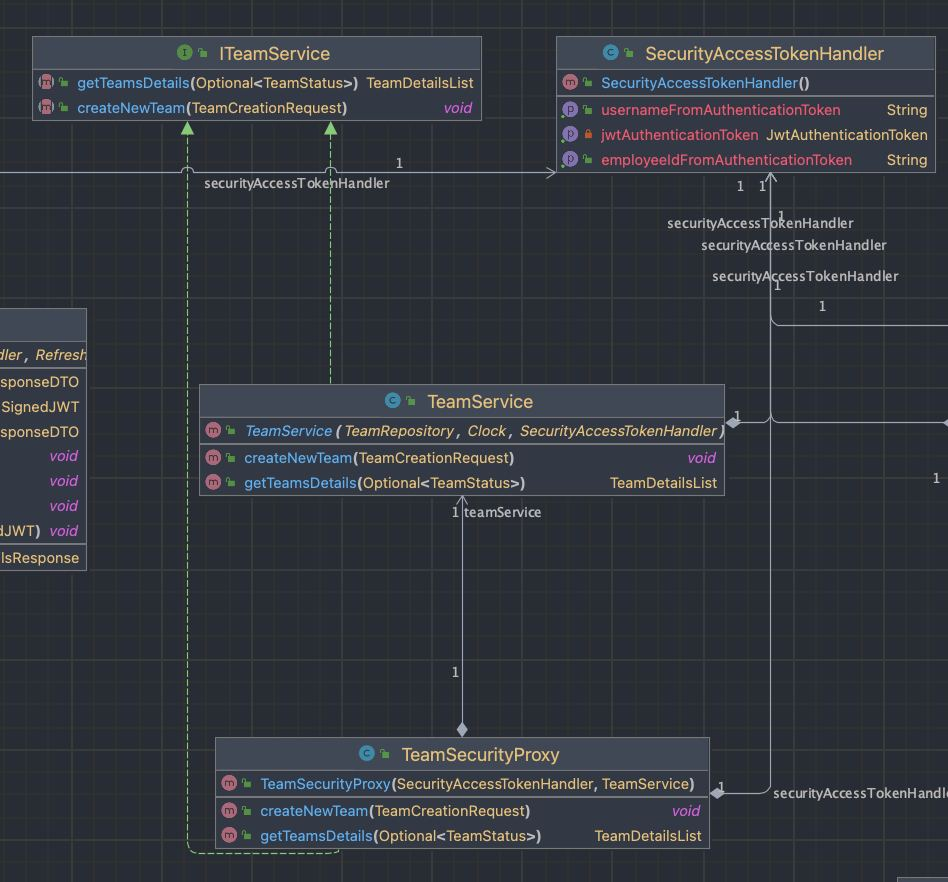
În concluzie, modelul de design Facade oferă o interfață simplificată și unificată către un subsistem complex, ascunzând complexitatea și permițând utilizatorilor să interacționeze cu subsistemul într-un mod mai ușor și mai convenabil. 

Figura 3.3 – Diagrama UML pentru șablonul Proxy

## 3.3 Șabloane comportamentale

Șabloane de proiectare comportamentale definesc algoritmi și modalități de implementare a interacțiunilor diferitelor obiecte și clase. Ele se ocupă în mod specific de comunicarea dintre obiecte. Aceasta este ultima categorie de modele design și diferă de primele două prin numărul mare de modele incluse în această categorie. Ele sunt responsabile de interacțiunea obiectelor între ele.

Deosebim următoarele tipuri de șabloane comportamentale:

* Stratеgy, încapsulează diferiți algoritmi;
* Cоmmand este o solicitare este încapsulată sub un obiect ca comandă și transmisă obiectului

invocator;

* Obsеrvеr, care definește un handler pentru diferite evenimente;
* Visitоr, care descrie metode ce pot fi aplicat pe o structură neomogenă;
* Mеmеntо, care salvează și restaurează starea unui obiect;
* Mеdiatоr, care definește un obiect care încapsulează modul în care un set de obiecte interacționează;
* Intеrprеtеr, un interpretor pentru un limbaj cu o gramatică simplă ;
* Itеratоr, gestionează parcurgerea unei colecții de elemente;
* Statе, gestionează obiecte al căror comportament depinde de starea lor;
* Tеmplatе mеthоd, încapsulează un algoritm ai cărui pași depind de o clasă derivată;
* Chain оf rеspоnsibility, gestionează tratarea unui eveniment de către mai mulți furnizori de soluții.

### 3.3.2 Strategy

Modelul de strategie, cunoscut și ca model de politică, este un [model de proiectare a software-ului](https://en.wikipedia.org/wiki/Design_pattern_(computer_science))[comportamental](https://en.wikipedia.org/wiki/Behavioral_design_pattern) care permite selectarea unui [algoritm](https://en.wikipedia.org/wiki/Algorithm) în timpul execuției. În loc să implementeze un singur algoritm direct, codul primește instrucțiuni de rulare cu privire la care dintr-o familie de algoritmi să folosească.

Șablonul de proiectare Strategy (figura 3.6), în cadrul proiectului actual permite schimbarea algoritmului de sortare a mesajelor din căsuța poștală în timpul execuției programului, astfel utilizatorul are posibilitatea de a sorta email-urile după dată sau în ordine alfabetică, fără a efectua oricare alte acțiuni, decât selectarea metodei de filtrare, datorită implementării șablonului.

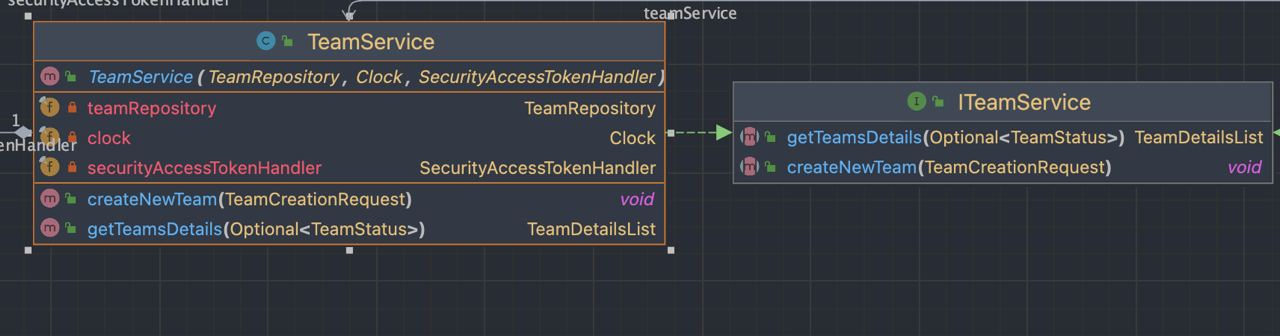


Figura 3.6 – Diagrama UML pentru șablonul Strategy

### 3.3.3 Template

Un model de design comportamental Template este un sablon care permite definirea unei structuri algoritmice generale într-o clasă de bază, lăsând detalii specifice și pașii de implementare la latitudinea subclaselor.

Caracteristicile cheie ale acestui model de design sunt:

1. Clasa de bază definește o metodă template care reprezintă scheletul algoritmului și definește ordinea generală a pașilor algoritmului.
2. Anumite părți ale algoritmului sunt definite în clasa de bază, dar pot fi suprascrise în subclase pentru a oferi implementări specifice.
3. Subclasele extind clasa de bază și implementează detaliile specifice ale pașilor suprascrisi.

Utilizarea modelului de design Template aduce următoarele beneficii:

1. Reutilizarea codului: Structura algoritmică generală este definită în clasa de bază, ceea ce permite reutilizarea codului comun în diferite subclase.
2. Extensibilitate: Subclasele pot extinde și personaliza pașii specifici ai algoritmului, fără a modifica structura generală.
3. Evitarea duplicării codului: Metodele comune sunt definite o singură dată în clasa de bază, evitând duplicarea codului în fiecare subclasă.

Un exemplu comun de utilizare a modelului de design Template este în crearea unui raport. Clasa de bază poate defini structura generală a raportului, cum ar fi deschiderea și închiderea fișierului, dar lăsând subclaselor să implementeze detaliile specifice, cum ar fi conținutul raportului și formatarea acestuia.

Este important de menționat că modelul de design Template face parte din categoria modelelor de design comportamental și este utilizat pentru a defini și organiza algoritmi cu pași variabili, dar structură fixă.

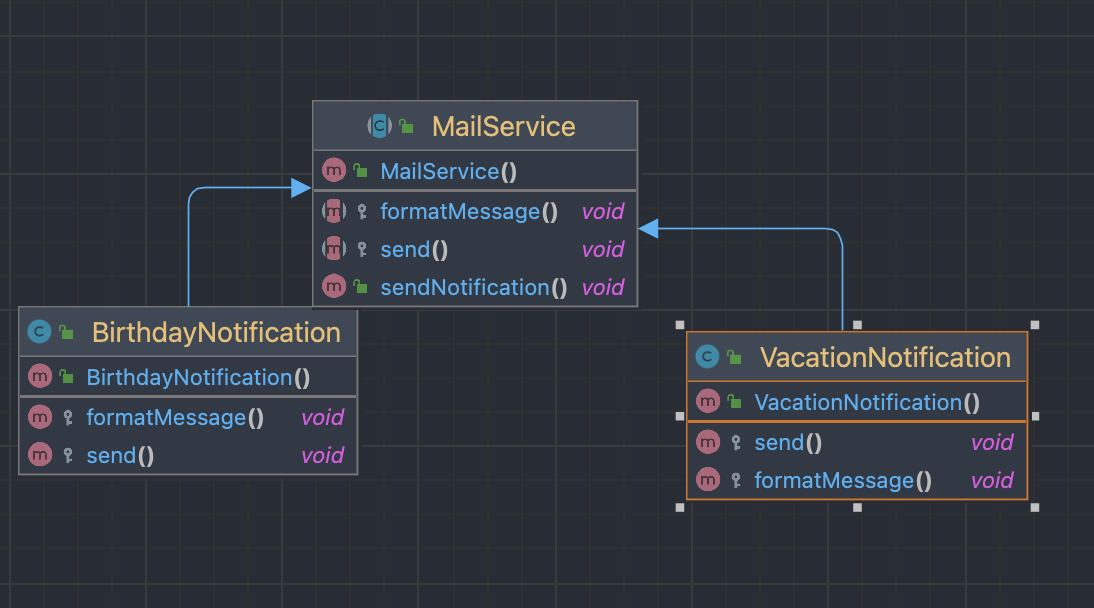


Figura 3.7 – Diagrama UML pentru șablonul Template

## 3.5 Proiectarea bazei de date

O bază de date permite aplicației să stocheze și să gestioneze cantități mari de date într-un mod structurat și eficient. Aceasta include informații despre utilizatori, produse, tranzacții, configurări etc. Baza de date oferă un mecanism de organizare și acces rapid la date, permițând aplicației să recupereze, să actualizeze și să șteargă datele într-un mod eficient.

Bază de date oferă un set bogat de funcționalități și comenzi care permit aplicației să efectueze operații complexe asupra datelor. Acestea includ interogări, filtrare, sortare, agregare și alte operații necesare pentru prelucrarea și analiza datelor

Persistența datelor: Baza de date asigură persistența datelor, ceea ce înseamnă că datele rămân stocate chiar și după închiderea și repornirea aplicației. Aceasta permite utilizatorilor să acceseze și să lucreze cu datele în orice moment, fără a le pierde.

Securitatea datelor: O bază de date poate oferi niveluri de securitate pentru a proteja datele sensibile și a controla accesul la acestea. Se pot implementa măsuri precum autentificarea utilizatorilor, permisiuni și restricții de acces pentru a asigura că doar utilizatorii autorizați pot accesa și modifica datele.

Scalabilitate și performanță: O bază de date bine proiectată poate oferi scalabilitate și performanță optimă pentru aplicație. Aceasta înseamnă că aplicația poate manipula eficient volume mari de date și un număr crescut de utilizatori, asigurând un timp de răspuns rapid și o experiență fluidă pentru utilizatori.

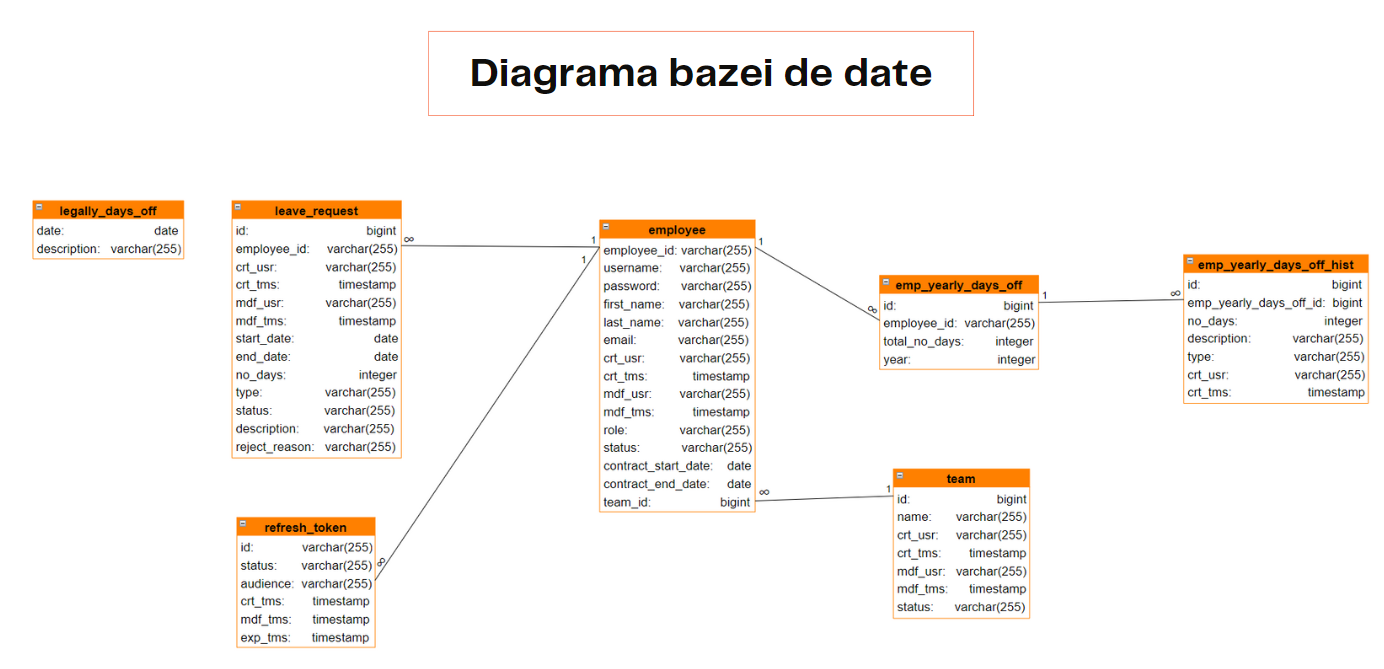


Figura 3.1 – Diagrama UML a bazei de date

Cum observă în figura 3.1 este reprezentată baza noastră de date iar mai jos este o scurtă descriere a fiecarui tabel.

Tabelul team:

Acest tabel stochează informații despre echipe.

Conține coloanele: id (cheie primară), name (numele echipei), crt\_usr (utilizatorul care a creat înregistrarea), crt\_tms (timestamp pentru momentul creării înregistrării), mdf\_usr (utilizatorul care a modificat ultima dată înregistrarea), mdf\_tms (timestamp pentru momentul ultimei modificări), status (starea echipei) și v (versiune tehnică a înregistrării).

Tabelul employee:

Acest tabel conține informații despre angajați.

Are coloanele: employee\_id (cheie primară), username (numele de utilizator al angajatului), password (parola angajatului), first\_name (prenumele angajatului), last\_name (numele angajatului), email (adresa de email a angajatului), crt\_usr (utilizatorul care a creat înregistrarea), crt\_tms (timestamp pentru momentul creării înregistrării), mdf\_usr (utilizatorul care a modificat ultima dată înregistrarea), mdf\_tms (timestamp pentru momentul ultimei modificări), role (rolul angajatului), status (starea angajatului), contract\_start\_date (data de început a contractului angajatului), contract\_end\_date (data de sfârșit a contractului angajatului), team\_id (id-ul echipei la care este alocat angajatul) și v (versiunea tehnică a înregistrării).

Tabelul leave\_request:

Acest tabel stochează informații despre cererile de concediu ale angajaților.

Conține coloanele: id (cheie primară), employee\_id (id-ul angajatului care solicită concediul), crt\_usr (utilizatorul care a creat înregistrarea), crt\_tms (timestamp pentru momentul creării înregistrării), mdf\_usr (utilizatorul care a modificat ultima dată înregistrarea), mdf\_tms (timestamp pentru momentul ultimei modificări), start\_date (data de început a concediului), end\_date (data de sfârșit a concediului), no\_days (numărul de zile de concediu), type (tipul de concediu), status (starea cererii de concediu), description (descrierea cererii de concediu) și reject\_reason (motivul respingerii cererii de concediu).

Alte tabele precum emp\_yearly\_days\_off, emp\_yearly\_days\_off\_hist, legally\_days\_off și refresh\_token păstrează informații legate de concediile anuale ale angajaților, istoricul concediilor, zilele libere legale și tokenurile de reîmprospătare pentru autentificare.

Acestea sunt doar câteva informații de bază despre fiecare tabel și coloanele sale. Este posibil să existe și alte restricții și legături între tabele, care ar putea completa imaginea generală a bazei de date.

**Autentificare**

Accesarea paginii de autentificare:

Utilizatorul accesează website-ul și navighează către pagina de autentificare.

Pagina de autentificare poate fi accesată prin intermediul unui link "Logare" sau prin intermediul unei secțiuni speciale din bara de navigare.

Introducerea credențialelor:

Pe pagina de autentificare, utilizatorul trebuie să introducă credențialele de autentificare în câmpurile corespunzătoare.

De obicei, se solicită introducerea numelui de utilizator și a parolei asociate contului.

Validarea credențialelor:

După ce utilizatorul a introdus credențialele, sistemul verifică dacă acestea sunt valide.

Verificarea poate include verificarea existenței contului asociat cu numele de utilizator specificat și potrivirea parolei introduse cu cea stocată în baza de date.

Tratarea erorilor de autentificare:

În cazul în care credențialele introduse sunt incorecte sau nu există un cont asociat cu numele de utilizator specificat, sistemul afișează un mesaj de eroare corespunzător.

Mesajul de eroare poate informa utilizatorul despre motivul exact al eșecului de autentificare (de exemplu, "Nume de utilizator sau parolă incorecte").

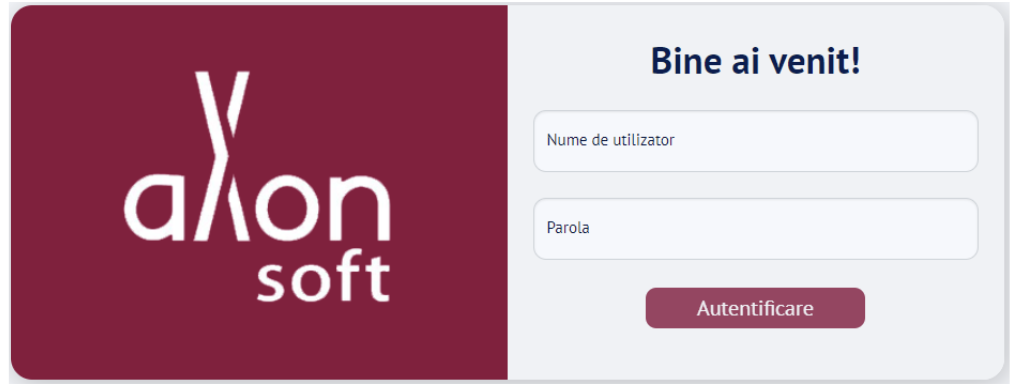


Figura 4.2 – Pagina de autentificare

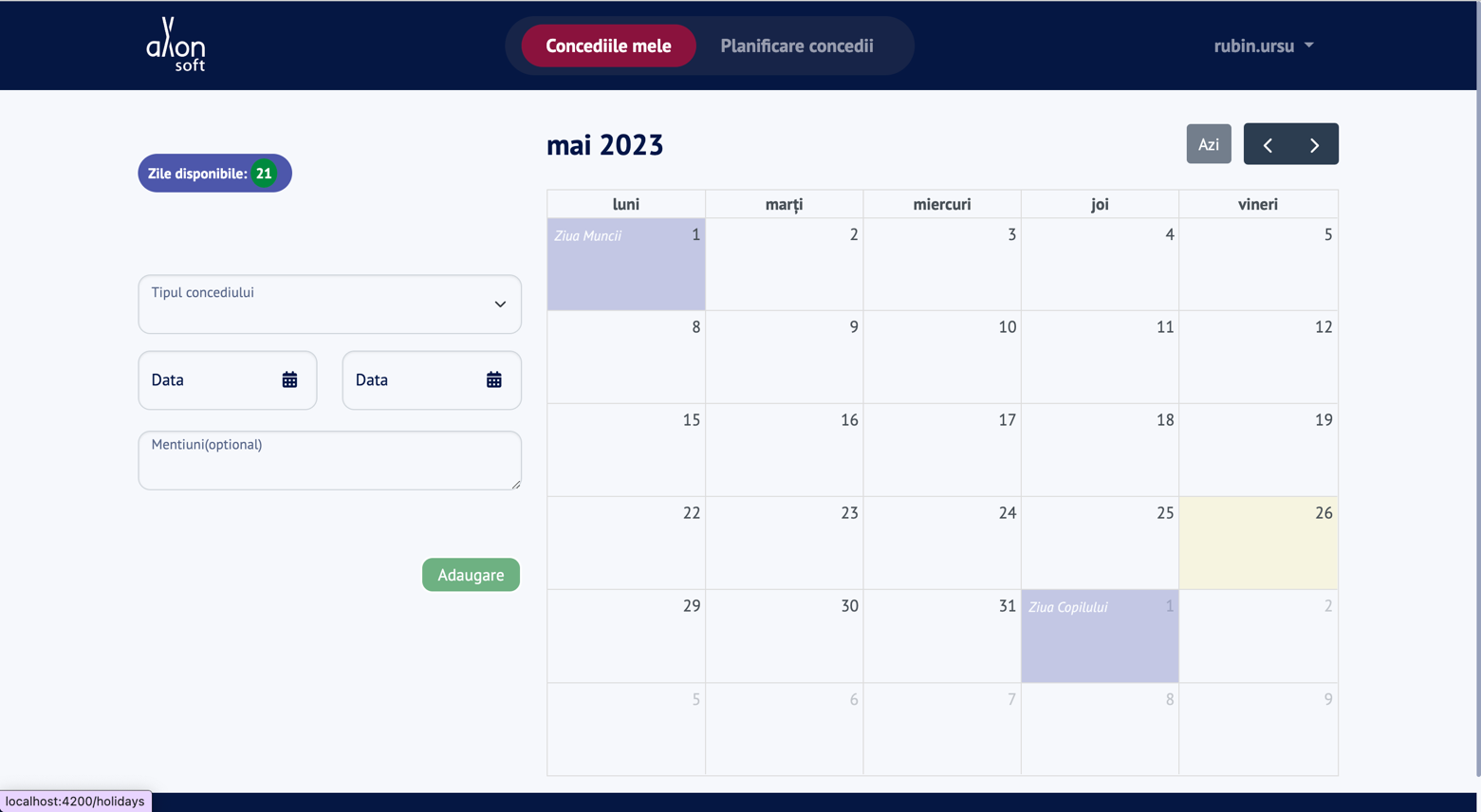
****

Figura 4.3 – Vizualizarea concediilor mele

Afișarea concediilor pe calendar:

Pe calendar, utilizatorul vede diferitele concedii programate de el.Fiecare concediu poate fi reprezentat printr-o casetă sau o marcă distinctă în calendar, care indică perioada în care angajatul este în concediu

Utilizatorul se autentifică în aplicația web prin intermediul unui browser și accesează contul său (dacă este necesar).

Navigarea către secțiunea "Planificare Concedii" sau similară:

După autentificare, utilizatorul navighează către secțiunea aplicației care se ocupă de gestionarea concediilor sau către pagina dedicată concediilor coechipierilor.

Afișarea tabelului concediilor coechipierilor:

Utilizatorul primește un tabel care prezintă informațiile despre concediile coechipierilor săi.

Fiecare rând al tabelului reprezintă un coechipier, iar coloanele reprezintă diferite detalii despre concediile lor.

# CONCLUZII

Obiectul de studiu *Tehnici și Mecanismele de Proiectare Software* au pus bazele gândirii critice, analizând și studiind bazele modelării și proiectării, permițând punerea în practică cunoștințele teoretice dobândite în timpul analizei și modelării diverselor sisteme, șabloane de proiectare care au permis modelarea și proiectarea propriul sistem informațional.

Șabloanele de proiectare, cunoscute și sub numele de design patterns, sunt soluții recurente la problemele comune de proiectare de software. Acestea reprezintă o abordare structurată și modulară în dezvoltarea de software, care ajută programatorii să dezvolte aplicații mai robuste și mai ușor de întreținut. UML, pe de altă parte, este o limbajă de modelare standard pentru a descrie, vizualiza și documenta sistemele software. Relația dintre șabloanele de proiectare și UML este strânsă, deoarece ambele abordează dezvoltarea de software dintr-o perspectivă orientată pe obiecte. UML este un instrument puternic pentru vizualizarea și documentarea sistemelor, iar șabloanele de proiectare sunt soluții practice pentru problemele comune de proiectare de software. Împreună, acestea oferă o modalitate eficientă de a construi sisteme software complexe, ușor de înțeles și de întreținut.

Importanța șabloanelor de proiectare în programare se datorează faptului că acestea ajută programatorii să abordeze problemele comune de proiectare de software într-un mod standardizat și modular. În loc să reinventeze roata pentru fiecare problemă, programatorii pot utiliza șabloanele de proiectare pentru a crea soluții testate și validate, care sunt cunoscute pentru a fi eficiente și fiabile.

De asemenea, șabloanele de proiectare facilitează comunicarea între membrii echipei de dezvoltare, deoarece acestea oferă un vocabular comun și un set de concepte standardizate pentru discuții. Aceasta face ca proiectul să fie mai ușor de înțeles pentru toți cei implicați și reduce riscul de neînțelegeri în ceea ce privește arhitectura și designul software-ului.

UML este, de asemenea, un instrument important în dezvoltarea de software, deoarece oferă o modalitate standardizată de a descrie sistemele software. Aceasta facilitează comunicarea între dezvoltatori, clienți și alte părți interesate, în timp ce oferă o modalitate eficientă de a documenta și vizualiza designul și arhitectura software-ului. Atunci când se utilizează șabloane de proiectare în UML, programatorii pot crea modele standardizate pentru sistemele software. Acestea sunt utile pentru a documenta designul și arhitectura sistemului, precum și pentru a comunica ideile și conceptele cheie între membrii echipei de dezvoltare.

În acest proiect de curs s-a construit un sistem de gestionare a zilelor libere in cadrul unei companii. Pe parcursul proiectării și implementării au fost întâlnite diverse blocaje, care ulterior au devenit rezolvabile, unele dintre ele fiind posibile de rezolvate utulizând șabloanele de proiectare, altele folosind clase suplimentare. Șabloanele conferă claritate și lizibilitate codului și ajută la simplificarea procesului de lucru, oferindu-ne posibilitatea de extindere a sistemului fără a fi nevoie de a modifica codul, acesta fiind punctul O din principiile SOLID (Open Closed Principle).

**BIBLIOGRAFIE**

1. Șabloane de proiectare, [citat 29.04.2023] Disponibil:

<https://www.youtube.com/@abcodemoldova1907>

1. em Client [citat 29.04.2023] Disponibil:

<http://www.emclient.com>

1. Library management system [citat 29.04.2023] Disponibil:

<https://www.skoolbeep.com/blog/library-management-system/>

1. Modele de proiectare [citat 29.04.2023] Disponibil:

<https://www.tutorialspoint.com/design_pattern/index.htm>

1. Tehnologia informației, [citat 29.04.2023] Disponibil:

<https://www.thc.ro/blog/tehnologia-informatiei-pe-intelesul-tuturor-definitie-utilitate-si-alte-informatii-interesante/>