**PROIECTAREA ȘI IMPLEMENTAREA UNUI SISTEM DE MANAGEMENT BAZAT PE METODA KANBAN**

**Candidat: David-Marian BOTEZATU**

**Coordonator științific: Ș.l.dr.ing. Mihaela Marcella CRIȘAN-VIDA**

Sesiunea: Iunie 2023

# INTRODUCERE

Lucrarea de față își propune să prezinte proiectarea și implementarea unui sistem de management al proiectelor bazat pe metoda Kanban, numit Spider. Această aplicație de gestionare a proiectelor este concepută pentru a facilita colaborarea în echipă și pentru a urmări în mod eficient statusul task-urilor individuale din cadrul unor proiecte.

Metoda de cercetare folosită implică atât studiul literaturii de specialitate privind metodele de management al proiectelor, cu un accent deosebit pe metoda Kanban, cât și proiectarea și dezvoltarea practică a sistemului Spider. Această lucrare se va concentra, de asemenea, asupra provocărilor cu care se confruntă managerii de proiect și modul în care metoda Kanban, și implicit sistemul Spider, pot contribui la atenuarea sau minimizarea acestora, ducând astfel la eficientizarea și succesul proiectelor în domeniul IT.

## CONTEXT

Mediul de afaceri de astazi este un mediu care se schimba rapid si devine tot mai digitalizat, iar astfel necesitatea unui management eficient al proiectelor este mai presanta ca niciodata din cauza marimii si complexitatii acestora. In industria IT, un proiect complex, cu multe sarcini si dependinte necesita un efort mare in planificare, monitorizare, coordonare si alocare eficienta a resurselor vine si cu foarte multe provocari printre care gestionarea timpului, urmarirea sarcinilor, colaborarea intre echipe si/sau intre membrii echipelor, gestionarea bugetului etc.

Pentru a aborda aceste provocari, mai multe metodologii si practici de management ce si isi propun sa optimizeze procesele de dezvoltare, sa imbunatateasca colaborarea intre membrii echipei si sa reduca riscul de esec al proiectului au castigat popularitate in industria IT. In continuare vom mentiona si explica trei astfel de metodologii si procese des intalnite in companiile din industrie: Agile, Scrum si Kanban.

In primul rand trebuie sa facem o distinctie intre termenul „Agile” si „Agile Software Development” (Dezvoltare Agila de Software): a fi „Agile” reprezinta un mod de gandire referitor la adaptabilitate, sau capabilitatea de a raspunde rapid la schimbari, in timp ce „Agile Software Development” este un termen umbrela pentru o serie de practici, metodologii si framework-uri ce pot oferi unei echipe sau companii oportunitatea de a deveni „Agile”. Scrum este unul dintre cele mai cunoscute framework-uri Agile, cu o abordare iterativ-incrementala. Neintrand in detalii, mentionez doar faptul ca Scrum este un proces bazat pe anumite caracteristici si valori care face referire la un stil de organizare al muncii, la componenta unui echipe, artefactele pe care le produce si evenimentele in care aceasta participa.

Kanban este o metodă pentru gestionarea sarcinilor (tasks) și fluxurilor de lucru (workflows), sub forma unui sistem vizual. Principiul fundamental al Kanban spune că oamenii trebuie să își limiteze activitatea "în desfășurare" la un număr rezonabil de sarcini, pentru a se asigura că munca este dusă la bun sfârșit într-un mod eficient și că nu există o suprasolicitare a resurselor.

În contextul industriei IT, Kanban permite membrilor unei echipe de dezvoltare software să vizualizeze starea de desfășurare a muncii, de la inițiere până la finalizare. Este reprezentat ca un tablou împărțit în coloane, fiecare coloană reprezentând o etapă diferită a fluxului de lucru, iar sarcinile sunt reprezentate sub forma de carduri care se mișcă de-a lungul tabloului (de pe o coloana pe alta) pe măsură ce progresează si indeplinesc criteriile de mutare.

Astfel, datorita nevoilor de imbunatatire a proceselor spre a raspunde mai repede nevoilor clientilor impreuna cu digitalizarea accelerata, sistemele de management a proiectelor si sarcinilor precum Jira, WorkOS (Monday.com) au devenit instrumente esentiale.

## TEMA LUCRĂRII DE LICENȚĂ

Tema prezentei lucrari este reprezentata de proiectarea si implementarea unui sistem de management al proiectelor bazat pe metoda Kanban, numit Spider. Se va evidentia modul in care acest sistem va imbunatati eficienta si productivitatea echipelor prin facilitarea vizualizarii fluxului de lucru, prioritizarea sarcinilor si imbunatatirea colaborarii in cadrul echipei de dezvoltare.

Ca structura, lucrarea va fi impartita in 8 capitole, fiecare capitol urmand sa trateze un aspect specific din proiectarea, implementarea si evaluarea sistemului Spider. In primul capitol vom explora concepte teoretice de baza legate de managementul proiectelor si metoda Kanban. In al doilea capitol vom detalia mai pe larg conceptele mentionate in introducere, pentru a intelege mai bine de ce este nevoie de un sistem de acest fel. Incepand cu capitolul trei ne vom concentra pe sistemul Spider, si vom incepe prin explorarea tehnologiilor folosite in implementarea acestei aplicatiei. In capitolele patru, cinci si sase ne vom uita la specificatiile, proiectarea si respectiv implementarea aplicatiei, vom crea un manual de utilizare pentru sistem in capitolul sapte, si vom incheia lucrarea prin prezentarea unor concluzii si directii viitoare.

# STUDIU BIBLIOGRAFIC

In acest capitol vom discuta in detaliu despre managementul proiectelor cu accent pe metodologia Kanban si vom analiza cateva sisteme de management bazate pe Kanban pe care le vom compara mai apoi cu aplicatia propusa.

# FUNDAMENTARE TEORETICĂ

Acest capitol are ca scop prezentarea tehnologilor și bibliotecilor folosite în crearea aplicației descrise în [Tema lucrării de licență](#_TEMA_LUCRĂRII_DE). Componentele principale în crearea aplicației client sunt Axios și React, iar pentru server folosim NodeJS împreună cu Express. Pentru baza de date folosim PostgreSQL, iar pentru cache și token management folosim Redis. În continuare, vom detalia aceste tehnologii și biblioteci si vom motiva alegerea lor în crearea aplicației. Totusi, deoarece Spider este o aplicație de tip client-server, și tehnologiile și bibliotecile folosite vor fi împărțite în aceleași categorii, prima categorie fiind reprezentata de catre bibliotecile folosite in crearea aplicatiei client.

## Aplicatia client (frontend)

1. **Axios:** conform site-ului oficial, Axios este un client HTTP pentru browser și node.js[[1]](#footnote-1). Prin urmare, Axios este o bibliotecă JavaScript open-source, folosită pentru a realiza cereri HTTP din browser către un server NodeJS, sau mai exact, către un API care rulează într-un mediu Node.js. Spider folosește Axios doar în aplicația client pentru a face cereri și a trimite date către API (defapt toate cererile care implica operatii CRUD efectuate asupra datelor din aplicatie sunt facute folosind aceasta biblioteca: pentru Create-Creare se foloseste axios.POST, pentru Read-Citire avem axios.GET, pentru Update-Modificare se foloseste axios.PUT iar pentru Delete-Stergere se foloseste axios.DELETE). Am ales această bibliotecă în detrimentul altora precum Fetch deoarece este mai simplu de utilizat si mai sigur (avand integrata protectie anti CSRF).
2. **React**: este o bibliotecă JavaScript dezvoltată de Meta, folosită în construirea interfețelor de utilizator (UI) pentru aplicații web și aplicații mobile[[2]](#footnote-2). React folosește bucăți individuale de cod în construcția acestor interfețe, numite **componente**. Acestea pot fi scrise de către un programator, sau pot exista în alte librării, și pot fi apelate doar atunci când este nevoie de ele și reprezintă una din principalele caracteristici ale React. Astfel, aceste componente sunt reutilizabile, ceea ce reprezintă un avantaj major deoarece facilitează organizarea și structurarea codului și duc spre crearea de aplicații modulare și flexibile. În crearea acestor componente React permite dezvoltatorilor să folosească o extensie a JavaScript numită JSX (JavaScript Syntax Extension), prin care aceștia pot adăuga cod HTML (sau alte componente/elemente React) în variabile sau metode JavaScript[[3]](#footnote-3). O altă caracteristică este reprezentată de procesul de randare: React folosește o tehnică numită **Virtual DOM** pentru actualizarea și reprezentarea aplicațiilor. Prin această tehnică React actualizează doar acele părți din aplicație care necesită schimbate în loc să actualizeze toată pagina web la fiecare modificare, acest proces permițând o actualizare mai rapidă și eficientă a interfeței[[4]](#footnote-4). Bineînțeles, există mai multe librării care permit dezvoltarea de aplicații web precum Vue.js, Angular sau Ember.js, dar am ales datorită caracteristicilor descrise mai sus, plus simplitatea acestuia și comunitatea și ecosistemul bogat in biblioteci și instrumente care ușurează și îmbunătățesc procesul de dezvoltare al aplicațiilor.
3. **React-beautiful-dnd**: este una din bibliotecile React folosite pentru gestionarea funcțiilor de tip drag-and-drop. În aplicația Spider este folosită în gestionarea tabelei Kanban prin crearea a două componente principale: Kanban Column și Kanban Task, în care Kanban Column este componenta statică, cea care nu poate fi mutată, iar Kanban Task este componenta care poate fi mutată și plasată în interiorul altor coloane. Deși există mai multe biblioteci care oferă acest tip de funcționalitate, am ales această bibliotecă deoarece este simplu de folosit si in acelasi timp este si foarte puternica, oferind o performanta ridicata si flexibilitate in personalizare.
4. **React-hook-form**: este o biblioteca simpla, flexibila si performanta, folosita pentru gestionarea formularelor in aplicatiile React[[5]](#footnote-5). Desi exista si alte biblioteci care ajuta la gestionarea formularelor (precum Formik), am ales aceasta biblioteca doarece este integrata direct cu React, ofera o performanta ridicata comparativ cu celelalte biblioteci si este usor de utilizat[[6]](#footnote-6).
5. **React-icons**: aceasta biblioteca ofera o solutie usoara si convenabila pentru utilizarea pictogramelor in aplicatii React. Am ales aceasta solutie deoarece ofera o gama larga de seturi de pictograme care pot fi utilizate printr-un simplu import in aplicatie (pictogramele sunt folosite de exemplu in meniul lateral sau in profilul utilizatorului). De asemenea, aceste pictograme pot fi usor personalizate folosind CSS si au o performanta optimizata pentru a asigura o incarcare rapida si o experienta placuta pentru utilizatori.
6. **React-modal**: este o bibliotecă React utilizată pentru a crea și gestiona ferestre modale (pop-up) în aplicațiile web intr-o maniera simplificata, oferind un control flexibil si complet asupra afisarii si utilizarii acestora. Am ales aceasta biblioteca deoarece este usor de utilizat - ofera o componenta React care poate fi personalizata folosind diferite proprietati create in React si este usor de integrat cu alte componente sau biblioteci UI precum Tailwind Css pentru personalizare vizuala. De asemenea, nu este necesar să gestionăm suprafete precum overlay-ul, evitând astfel bug-urile de genul în care click-urile efectuate pe elementele din spatele overlay-ului sunt înregistrate și activate inainte sau după închiderea modalului.
7. **React-router-dom**: o bibliotecă esențială folosita pentru gestionarea rutelor și navigării pe partea de client (client side routing) în aplicațiile web bazate pe React, prin care dezvoltatorii pot defini configuratia rutelor intr-un mod declarativ utilizand componente React. Acest stil de rutare permite aplicatiei sa schimbe un URL fara a cere un document nou de la server iar in acest fel, aplicatia poate sa afiseze imediat o componenta noua.
8. **React-switch**: este o bibliotecă React utilizată pentru crearea și gestionarea comutatoarelor în aplicațiile web. Ofera o componenta <Switch> care poate fi customizata utilizand CSS (sau o biblioteca UI precum Tailwind).
9. **React-toastify**: este utilizata pentru crearea si gestionarea notificarilor dintr-o aplicatie React pe baza componentelor. Am preferat aceasta solutie deoarece este usor de utilizat, permite personalizarea aspectului si comportamentului notificarilor (precum culoare, pozitionare, timp de afisare, etc). De asemenea, pot fi setate sa fie interactive, dezvoltatorii putand adauga butoane care executa diferite actiuni in functie de tipul de notificare.
10. **Yup**: este o biblioteca folosita pentru a aplica validari pe datele de intrare din formulare folosind o schema definita de dezvoltator. De asemenea, dezvoltatorii pot crea mesaje de eroare personalizate care apoi sunt afisate in aplicatie. Toate formularele din Spider folosesc yup pentru validare si afisare mesaje informative sau de eroare (in cazul in care cerintele din schema nu sunt indeplinite).
11. **TailwindCSS**: conform website-ului oficial, Tailwind CSS este un framework CSS „utility-first” cu clase precum „flex”, „pt-4”, „rotate-90” si altele, care put fi utilizate pentru a construi orice design, direct in fisierul HTML[[7]](#footnote-7). Spre deosebire de alte framework-uri precum Bootstrap care ofera componente create in mod implicit, Tailwind CSS pune la dispozitie un set de extins de clase utilitare care pot fi combinate in diferite moduri direct in fisierul HTML (precum inline CSS) pentru a crea interfete personalizate. De asemenea, pe langa setul de clase predefinite, Tailwind ofera si posibilitatea de a adauga propriile clase si a crea noi stiluri fara a fi nevoie de dependinte suplimentare sau de seturi predefinite de componente. Pentru Spider am ales Tailwind pentru a putea implementa mai usor dark mode si a avea o flexibilitate mai mare asupra personalizarii aplicatiei.

## Aplicatia server si API (backend)

1. **NodeJS**: este un mediu de executie open-source care permite crearea de aplicatii server-side, API-uri, aplicatii de timp real sau alte aplicatii de retea folosind limbajul JavaScript[[8]](#footnote-8), in acest mod eliminand nevoia de a utiliza alte limbaje de programare si tehnologii precum Java sau Python. Totodata, NodeJS vine impreuna cu propriul sau package manager numit npm prin intermediul carora dezvoltatorii pot instala biblioteci JavaScript care pot fi folosite in crearea de aplicatii. Exista si alte tehnologii utilizate in crearea de aplicatii server-side precum Java, Python (mentionate deja) sau .NET, dar am ales NodeJS deoarece este simplu de instalat si configurat, vine cu npm unde poti gasi o gama larga de module care pot fi folosite si salveaza timp in crearea aplicatiei, ofera performanta ridicata si in acelasi timp nu are cerinte de sistem mari.
2. **ExpressJS:** este un framework web minimalist si rapid pentru NodeJS[[9]](#footnote-9) folosit in crearea de aplicatii web si API-uri. Desi NodeJS poate fi utilizat fara Express, acesta din urma ofera un set de functii si metode simplificate pentru gestiunea rutarilor, cererilor si raspunsurilor HTML. De asemenea, poate fi integrat cu diferite module instalate prin npm, pentru a adauga functionalitati suplimentare aplicatiei dezvoltate.
3. **Redis (ioredis)**: Redis este un sistem de stocare in-memory (adica datele sunt salvate in memoria RAM) de tip cheie-valoare folosit ca baza de date, cache boker de mesaje sau motor de streamings si deoarece este un sistem in-memory, Redis ofera o performanta ridicata, permitand acces rapid la date. Totusi, trebuie mentionat faptul ca din moment ce memoria RAM este memorie volatila, daca serverul este oprit in vreun fel (restart, intreruperea alimentarii cu electricitate, etc), aceste date se vor pierde (bineinteles, se pot folosi tehnici de salvare a datelor pe medii de stocare persistente pentru a minimiza pierderile). In aplicatia propusa folosim Redis pentru a salva token-urile neexpirate care nu mai au acces sa foloseasca aplicatia (de exemplu dupa Logout), iar pentru asta este nevoie de o biblioteca pentru NodeJS care ne permite sa interactionam cu serverul. Desi exista mai multe biblioteci, am ales sa folosim ioredis deoarece este o implementare completa care ofera suportul necesar, si in acelasi timp este usor de utilizat, nu necesita configurari avansate si are o performanta ridicata.
4. **Bcryptjs**: este o biblioteca prin care functia de criptare parole (password hash) BCRYPT este implementata in JavaScript. Stocarea parolelor sub forma de text (plain-text) in baza de date reprezinta o bresa de securitate majora in industria IT (in cazul in care baza de date este compromisa, atacatorii pot utiliza parolele pentru a se conecta direct in contul utilizatorilor). Din acest motiv, utilizam bcryptjs pentru a crea un hash care va fi salvat in baza de date si care va fi comparat cu un nou hash creat in momentul in care utilizatorul incearca sa se logheze in aplicatie.
5. **Body-parser**: este o biblioteca NodeJS folosita in analizarea datelor primite printr-o cerere HTTP. In momentul in care serverul primeste date printr-o cerere HTTP (de exemplu un GET sau un POST request trimis de client), datele respective nu sunt usor de manipulat. Body-parser preia aceste date (care pot fi in format text, JSON, etc), si le transforma in obiecte JavaScript accesibile dezvoltatorilor intr-un mod facil. Desi exista si alte biblioteci care fac aceasta transformare, am ales body-parser deoarece are o comunitate larga de utilizatori, este stabil si are multe functii integrate folositoare in timpul dezvoltarii aplicatiei.
6. **Cors**: este o biblioteca prin care se gestioneaza mecanismul CORS. Pentru a intelege acest mecanism, trebuie sa mentionam faptul ca browserele restrictioneaza cererile HTTP dintre doua origini din cauza unei politici numite „same-origin policy” - presupunand ca avem 2 domenii diferite abc.com si xyz.com, aceasta politica permite cererile dintre abc.com (client) si abc.com (server), dar nu permite cererile dintre abc.com si xyz.com. Din moment ce aplicatia propusa este instalata pe 2 servere cu domenii diferite, avem nevoie de un sistem prin care sa ridicam aceasta politica pentru cererile HTTP dintre aceste doua domenii. Mecanismul CORS (Cross-Origin Resource Sharing) este un mecanism prin care i se da voie unui server sa indice orice origine prin care i se permite unui browser sa accesese resurse de pe acel server[[10]](#footnote-10), creand astfel o „poarta” prin care se ocoleste politica „same-origin”.
7. **Dotenv**: este o biblioteca prin care se permite folosirea variabilelor de mediu prin care se stocheaza configuratii specifice unei aplicatii sau unui mediu (precum productie, test sau dezvoltare) intr-un proiect NodeJS. Pentru Spider, aceasta biblioteca este folosita pentru adresele catre Redis si PostgreSQL, precum si pentru secretele prin care se creaza json web tokens si parola pentru administrator. De asemenea, tot in variabilele de mediu avem si datele pentru sistemul de email (adresa si parola contului de email, precum si sistemul folosit).
8. **Generate-password**: acesta este o librarie care este utilizata pentru a genera parole aleatorii in functie de configuratia aleasa (de exemplu numarul de caractere utilizate, includerea de caractere speciale, litere majuscule sau minuscule, cifre, etc). Aceasta biblioteca este folosita pentru crearea de parole atunci cand un utilizator nou este creat (utilizatorul primind un email prin care este notificat de crearea contului, email in care este trecuta si aceasta parola generata aleator).
9. **Jsonwebtoken**: este un tip de reprezentare a unor revendicari (informatii afirmate despre un anumit subiect) trimise intre doi participanti[[11]](#footnote-11), si este format din 3 parti principale: header (care contine tipul de token si algoritmul de semnare utilizat), revendicarile trimise (payload) si semnatura (o aplicare a algoritmului de semnare pe header si payload utilizand o cheie secreta cunoscuta doar de server (in cazul nostru cheia este salvata in variabilele de mediu, bineinteles in mod criptat). Biblioteca NodeJS care implementeaza aceasta reprezentare se numeste „jsonwebtoken” (JWT) si este folosita in aplicatia propusa (Spider) pentru autentificare. La fiecare login, functia de creare token-uri este apelata, iar token-ul rezultat este trimis catre client ca raspuns la o autentificare efectuata cu succes (daca utilizatorul nu exista sau parola este incorecta, nu se va crea nici un token), si este salvat in memoria locala a browserului (localStorage). Acest token va fi folosit in continuare la fiecare request facut catre server – va fi trimis in Authorization header – si va fi verificat pentru a demonstra (sau infirma) faptul ca utilizatorul care a facut cererea este autentificat, si are accesul necesar sa faca acea cerere. In aplicatia propusa token-ul expira la 8 ore dupa creare (dar durata poate fi personalizata in functie de cerinte).
10. **Nodemailer**: este biblioteca pentru aplicatii NodeJS utilizata pentru trimiterea de emailuri in format text sau html. In Spider este folosit intr-unul din urmatoarele 3 cazuri: atunci cand este creat un cont nou (utilizatorul va primi un email de confirmare si parola pe adresa adaugata atunci cand se creaza contul), cand utilizatorul isi schimba parola sau atunci cand utilizatorul si-a uitat parola si trebuie sa o reseteze (va primi un email cu un link ce deschide pagina de resetare parola).
11. **Sequelize**: este o biblioteca ORM (Object-Relational Mapping) NodeJS bazata pe promisiuni, folosita in interactiunea cu diferite baze de date relationale precum MySQL, MsSQL, PostgreSQL, etc. Cu ajutorul acestei biblioteci, in loc sa foloseasca interogari SQL traditionale, dezvoltatorii gestioneaza si manipuleaza datele dintr-o baza de date compatibila folosind obiecte si metode (de exemplu, in loc de „SELECT \* FROM users” avem „Users.findAll()”), Sequelize transformand in mod automat obiectele in structuri de date compatibile cu baza de date aleasa (si invers). Am ales Sequelize deoarece simplifica operatiile CRUD (Create, Read, Update, Delete – Creare, Citire, Modificare, Stergere) si interogarile in baza de date, putem versiona schema bazei de date, si se integreaza bine cu NodeJS precum si cu alte module populare precum ExpressJS pentru crearea de API-uri. Pentru ca Sequelize sa functioneze corect, este nevoie sa instalam anumite biblioteci in functie de baza de date utilizata (de exemplu mysql2 pentru MySQL, mariadb pentru MariaDB, etc). Deoarece Spider utilizeaza PostgreSQL pentru stocarea datelor, Sequelize are nevoie de alte 2 biblioteci (dependinte): **pg** si **pg-hstore**. **Pg** este un driver PostgreSQL pentru NodeJS care ofera functionalitatile pentru interogari si pentru comunicarea cu baza de date, in timp ce **pg-hstore** este un modul suplimentar pentru **pg** ce permite transformarea datelor JSON in format hstore si invers.
12. **Nodemon**: in primul rand trebuie mentionat faptul ca desi aceasta biblioteca nu este folosita in mediul de productie (poate fi folosita din punct de vedere tehnic, dar nu este recomandat), este folosita in mediul de dezvoltare. Pentru a intelege nevoia de utilizare a acestei unelte, trebuie mentionat faptul ca atunci cand un dezvoltator scrie cod server-side, acel cod nu este utilizat imediat in aplicatie, dezvoltatorul fiind nevoit sa reporneasca manual serverul pentru a vedea modificarile facute. Resetarea manuala a serverului, desi nu este un proces dificil, incetineste totusi procesul de dezvoltare lucru care trebuie evitat si astfel, pentru a reduce timpul pierdut, a aparut nevoia de a avea o unealta de automatizare a acestui proces de repornire. Solutia gasita pentru a rezolva aceasta problema este biblioteca numita nodemon unde in loc sa rulam o aplicatie NodeJS folosing comanda „node start” vom folosi „nodemon start” ceea ce inseamna ca nodemon va porni serverul si va monitoriza toate fisierele proiectului iar atunci cand acesta detecteaza o modificare intr-un fisier (practic atunci cand un fisier este salvat), va reporni automat serverul fara a mai fi nevoie de interventie umana.

# SPECIFICAȚIILE APLICAȚIEI

## DESCRIEREA APLICAȚIEI

## FUNCȚIILE APLICAȚIEI

## INTERFAȚA CU UTILIZATORUL

## BAZA DE DATE

## CONECTAREA CU ALTE APLICAȚII SAU SISTEME

# PROIECTAREA APLICAȚIEI

## COMPONENTELE APLICAȚIEI

## STOCAREA INFORMAȚIEI

# IMPLEMENTAREA APLICAȚIEI

# MANUAL DE UTILIZARE

# CONCLUZII ȘI DIRECȚII VIITOARE

Fiecare capitol trebuie să aibă o structură clară, va începe pe pagină nouă și va conține un titlu. Va fi urmat de două linii de 12 pt lăsate libere.

Fiecare secțiune a unui capitol (ex. 1.1 INFORMAȚII GENERALE) va fi poziționat la un rând liber sub text și va avea un rând liber de12 pt deasupra textului.

Textul lucrării va fi aliniat uniform (justify). Este de preferat ca textul să fie verificat pentru eventualele erori în limba de editare cu ajutorul facilității de verificare a ortografiei (speller) din programul word. Este recomandat ca lucrarea de finalizare a studiilor să nu depășească 100 de pagini, inclusiv anexele.

Reguli aplicate pentru textul lucrării:

a. Marginile paginii – se vor utiliza următoarele valori pentru marginile paginii (Page Setup -> Margins-> Mirror Margins):

 interior: 2 cm

 exterior: 2 cm

 sus: 2,5 cm (inclusiv header)

 jos: 2 cm

b. Spațiere între rânduri - textul va respecta o spațiere între rânduri de 1,15 linii (Format>Paragraph->Line spacing-> 1,15 lines);

c. Alinierea textului în cadrul paragrafelor - textul din cadrul paragrafelor normale va fi aliniat între marginile din stânga şi dreapta (justified). Primul rând al fiecărui paragraf va avea o aliniere de 1,5 cm (Format-> Paragraph-> Indentation-> Left). Excepție fac titlurile capitolelor, care vor fi aliniate la stânga, precum și etichetele tabelelor și ale figurilor (conform explicațiilor de mai jos);

d. Font – fontul utilizat pentru redactare va fi Arial, cu dimensiunea de 12 puncte, utilizând diacriticele specifice limbii în care este redactată lucrarea (ex: ă, ş, ţ, î, â - pentru limba română);

e. Numerotarea paginilor - numerotarea paginilor se face începând cu pagina de titlu, până la ultima pagină a lucrării, dar numărul paginii apare doar începând cu Introducerea. Numărul de pagină se inserează în subsolul paginii, centrat.

f. Antetul paginii – apare începând cu introducerea și va conține pe rânduri succesive un text cu înălțimea de 8, aliniat la stânga: (i) textul Universitatea Politehnica Timișoara ; (ii) denumirea programului de studii și anul susținerii ; (iii) numele candidatului (în stânga) și titlul lucrării. În partea dreaptă a antetului poate fi integrată sigla UPT;

## FIGURI ȘI FOTOGRAFII (12 pt, Bold, Uppercase, Left)

Figurile (incluzând imagini, grafice, capturi de ecran) se numerotează în ordinea apariției în lucrare. Alternativ, figurile pot fi numerotate în ordine în fiecare capitol, integrând în numerotare și numărul capitolului. Fiecare figură are număr și titlu, care se menționează sub figură, centrat. Dacă este cazul, sursa figurii se indică între paranteze după titlul figurii. Toate figurile şi fotografiile prezentate în lucrare trebuie să fie referite în textul lucrării, trebuie să fie numerotate şi însoţite de titlul figurii.

Se va lăsa câte o linie liberă (Arial 12 pt) între figură şi text. Figurile vor fi centrate pe pagină.

INT_EXTjoint

Figura 1 – Exemplu de figură (sursa: Buletinul ştiinţific al UPT seria construcţii-arhitectură nr.2 /2010)

## TABELE (12 pt, Bold, Uppercase, Left)

Tabele – tabelele se numerotează în ordinea apariției în lucrare. Alternativ, tabelele pot fi numerotate în ordine în fiecare capitol, integrând în numerotare și numărul capitolului. Fiecare tabel are număr și titlu, care se menționează deasupra tabelului, aliniat centrat. Dacă este cazul, sursa datelor se precizează între paranteze după titlul tabelului;

Toate tabelele prezentate în lucrare trebuie să fie referite în textul lucrării, trebuie să fie numerotate şi însoţite de un titlu (vezi exemplul de mai jos). Dacă se utilizează figuri copiate atunci se va indica sursa fotografiei în paranteză. Pe cât posibil, în tabel se va păstra fontul uzual (arial 12 pt) dar sunt acceptate şi modalităţi de a scoate în evidenţă rezultatele importante (bold, italic etc.)

Se va lăsa câte o linie liberă (arial 12 pt) între text şi tabel. Tabelele vor fi centrate pe pagină.

Tabelul 1. Exemplu de tabel

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Yield stress, fy [N/mm2] | | Tensile strength, fu [N/mm2] | |
| Element | Mill certificate | Coupon tests | Mill certificate | Coupon tests |
| Beam IPE360 | 285.0 | 329.8 flange  348.4 web | 427.0 | 463.2 flange  464.0 web |
| Column HEB300 | 311.3 | 313.0 flange  341.8 web | 446.0 | 449.8 flange  464.4 web |
| End plate | 281.0 | 248.3 | 424.7 | 416.0 |
| Cover plate | 296.0 | 273.2 | 443.0 | 436.7 |

## FORMULELE (12 pt, Bold, Uppercase, Left)

Formulele utilizate în text se vor numerota în ordinea apariției în lucrare. Alternativ, formulele pot fi numerotate în ordine în fiecare capitol, integrând în numerotare și numărul capitolului. Numerotarea formulelor se face în paranteze rotunde. Se va lăsa câte o linie liberă (arial 12 pt) între text şi formulă. Formulele vor fi aliniate la dreapta.

(1)

# N. CONCLUZII (14 pt, Bold, Uppercase, Center)

Lucrarea se va încheia cu un capitol de concluzii. Acesta va conţine principalele rezultate ale lucrării şi implicaţiile practice ale acestora. În cazul proiectelor de diplomă, se vor menționa principalele date sintetice obținute din procesul de proiectare

# BIBLIOGRAFIE

La sfârşitul lucrării va fi dată o listă de referinţe pentru textele ştiinţifice consultate pe parcursul realizării lucrării. Vor fi trecute toate sursele, inclusiv cele de pe internet. Acestea vor fi referite în text şi trecute în lista de referinţe în ordine alfabetică, după exemplele de mai jos.

Bibliografia trebuie să cuprindă toate titlurile din literatura de specialitate care au servit ca bază de documentare, respectiv autorii care au fost citați în text, la toate capitolele lucrării.

În cadrul Facultății de Automatică și Calculatoare se cere folosirea stilului de citare IEEE (detalii [IEEE Citation Guidelines2.doc (ieee-dataport.org)](https://ieee-dataport.org/sites/default/files/analysis/27/IEEE%20Citation%20Guidelines.pdf)), folosit cu precădere în publicațiile științifice din domeniul IT. Cele trei părți importante ale referinței sunt:

* Numele autorului indicat ca prima inițială a prenumelui, apoi numele complet.
* Titlul articolului, brevetul, lucrarea de conferință etc., între ghilimele.
* Titlul revistei sau cărții cu caractere cursive

Modul de redactare a referinței depinde de tipul publicației, vă rugăm să urmăriți cu atenție indicațiile de la link-ul de mai sus.

Fiecare citare trebuie notă în text prin utilizarea unor numere secvențiale simple. Un număr cuprins între paranteze drepte, plasat în textul raportului, indică referința specifică. Citările sunt numerotate în ordinea în care apar. Odată ce o sursă a fost citată, același număr este folosit în toate referințele ulterioare din text. Nu se face distincție între sursele electronice și cele tipărite, cu excepția detaliilor referințelor citate.

Fiecare număr de referință trebuie să fie cuprins între paranteze drepte pe aceeași linie cu textul, înaintea oricărei semne de punctuație, cu un spațiu înaintea parantezei.

Exemple

„. . .finalul cercetării mele [13].”

„Teoria a fost prezentată pentru prima dată în 1987 [1].

Lista de referințe din bibliografie este compusă din toate sursele folosite pentru documentarea lucrării și se realizează în ordinea numerică a citării în text și nu în ordine alfabetică a autorilor.

Preluarea identică a unei fraze sau paragraf va fi citată prin indicarea inclusiv a paginii din sursa utilizată, dar și prin ghilimele şi forma italică a literelor; pentru sursele preluate de pe internet, vor fi notate adresele de pagină web; în lista bibliografică finală lucrările se trec în ordinea alfabetică a numelor autorilor. La lucrările colective, regula referitoare la ordinea alfabetică este valabilă pentru primul autor.

Dacă se citează site-uri web, reviste sau articole, înainte de acestea se vor trece trei asteriscuri, informații referitoare la volum, număr, pagini consultate, adresa web exactă a articolului respectiv, data vizitării site-ului și a descărcării materialului, data accesării. Adresele de pagini web se regăsesc la finalul listei.

Sursele bibliografice la care nu se poate menționa autorul se vor specifica astfel: „\*\*\*”urmat de denumirea articolului și/sau a cărții, editura și locul apariției (pentru cărți), volumul, numărul acestuia, prima și ultima pagină a lucrării citate, anul apariției.

\*\*\* https://ro.wikipedia.org/wiki/Motor\_cu\_reac%C8%9Bie accesare februarie 2022

Ultima pagină a lucrării de licență/diplomă/disertație trebuie să conțină „Declarația de originalitate a lucrării de finalizare a studiilor”, completată olograf, în conformitate cu cerinţele UPT. Declaraţia se descarcă de pe adresa de web:

[DECLARAŢIE DE AUTENTICITATE A (upt.ro)](http://www.upt.ro/img/files/Regulamente_UPT/2020/Declaratie_de_autenticitate_UPT_2020.pdf)

1. <https://axios-http.com/docs/intro>, iunie 2023 [↑](#footnote-ref-1)
2. <https://react.dev/>, iunie 2023 [↑](#footnote-ref-2)
3. <https://en.wikipedia.org/wiki/JSX_(JavaScript)>, iunie 2023 [↑](#footnote-ref-3)
4. <https://legacy.reactjs.org/docs/faq-internals.html>, iunie 2023 [↑](#footnote-ref-4)
5. <https://www.react-hook-form.com/>, iunie 2023 [↑](#footnote-ref-5)
6. <https://www.react-hook-form.com/faqs/#ReactHookFormFormikorReduxForm>, iunie 2023 [↑](#footnote-ref-6)
7. <https://tailwindcss.com/>, iunie 2023 [↑](#footnote-ref-7)
8. <https://nodejs.org/en/about>, iunie 2023 [↑](#footnote-ref-8)
9. <https://expressjs.com/>, iunie 2023 [↑](#footnote-ref-9)
10. <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>, iunie 2023 [↑](#footnote-ref-10)
11. <https://datatracker.ietf.org/doc/html/rfc7519>, versiunea Mai 2015 [↑](#footnote-ref-11)