**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

# CUPRINS

[CUPRINS 2](#_Toc138184721)

[1. INTRODUCERE 3](#_Toc138184722)

[1.1 CONTEXT 3](#_Toc138184723)

[1.2 TEMA LUCRĂRII DE LICENȚĂ 4](#_Toc138184724)

[2. STUDIU BIBLIOGRAFIC 5](#_Toc138184725)

[2.1 JIRA 5](#_Toc138184727)

[2.2 MONDAY.COM 6](#_Toc138184728)

[2.3 YOUTRACK 8](#_Toc138184729)

[2.4 TRELLO 9](#_Toc138184730)

[2.5 REDMINE 9](#_Toc138184731)

[2.6 COMPARAȚIE APLICAȚII ANALIZATE 11](#_Toc138184732)

[3. FUNDAMENTARE TEORETICĂ 12](#_Toc138184733)

[3.1 APLICAȚIA CLIENT 12](#_Toc138184735)

[3.2 APLICAȚIA SERVER ȘI API 15](#_Toc138184736)

[4. SPECIFICAȚIILE APLICAȚIEI 21](#_Toc138184737)

[4.1 DESCRIEREA APLICAȚIEI 21](#_Toc138184739)

[4.2 FUNCȚIILE APLICAȚIEI 21](#_Toc138184740)

[4.3 INTERFAȚA CU UTILIZATORUL 22](#_Toc138184741)

[4.4 BAZA DE DATE 22](#_Toc138184742)

[5. PROIECTAREA APLICAȚIEI 24](#_Toc138184743)

[5.1 COMPONENTELE APLICAȚIEI 24](#_Toc138184745)

[5.2 STOCAREA INFORMAȚIEI 29](#_Toc138184746)

[6. IMPLEMENTAREA APLICAȚIEI 32](#_Toc138184747)

[7. MANUAL DE UTILIZARE 36](#_Toc138184748)

[8. CONCLUZII ȘI DIRECȚII VIITOARE 42](#_Toc138184749)

[9. BIBLIOGRAFIE 43](#_Toc138184750)

# 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 astăzi este un mediu care se schimbă rapid și devine tot mai digitalizat, iar astfel necesitatea unui management eficient al proiectelor este mai presantă ca niciodată din cauza mărimii și complexității acestora. În industria IT, un proiect complex cu multe sarcini și dependințe necesită un efort mare în planificare, monitorizare, coordonare și alocare eficientă a resurselor vine și cu foarte multe provocări printre care gestionarea timpului, urmărirea sarcinilor, colaborarea între echipe și/sau între membrii echipelor, gestionarea bugetului etc.

Pentru a aborda aceste provocări, mai multe metodologii și practici de management ce si își propun să optimizeze procesele de dezvoltare, să îmbunătățească colaborarea între membrii echipei și să reducă riscul de eșec al proiectului au câștigat popularitate în industria IT. În continuare vom menționa și explica trei astfel de metodologii și procese des întâlnite în companiile din industrie: Agile, Scrum si Kanban.

În primul rând trebuie să facem o distincție între termenul „Agile” și „Agile Software Development” (Dezvoltare Agila de Software): a fi „Agile” reprezintă un mod de gândire referitor la adaptabilitate sau capabilitatea de a răspunde rapid la schimbări, în timp ce „Agile Software Development” este un termen umbrelă pentru o serie de practici, metodologii și 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-incrementală. Neintrând în detalii, menționez doar faptul că Scrum este un proces bazat pe anumite caracteristici și valori care fac referire la un stil de organizare al muncii, la componența unui echipe, artefactele pe care le produce și evenimentele în care aceasta (echipa) participă.

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 ce se mișcă de-a lungul tabloului (de pe o coloana pe alta) pe măsură ce progresează și îndeplinesc toate criteriile de mutare.

Astfel, datorită nevoilor de îmbunătățire a proceselor spre a răspunde mai repede nevoilor clienților împreună cu digitalizarea accelerată, sistemele de management a proiectelor și sarcinilor precum Jira, Monday.com, sau YouTrack au devenit instrumente esențiale.

## TEMA LUCRĂRII DE LICENȚĂ

Tema prezentei lucrări este reprezentată de proiectarea și implementarea unui sistem de management al proiectelor bazat pe metoda Kanban, numit Spider. Se va evidenția modul în care acest sistem va îmbunătăți eficiența și productivitatea echipelor prin facilitarea vizualizării fluxului de lucru, prioritizarea sarcinilor și îmbunătățirea colaborării în cadrul echipei de dezvoltare.

Din punctul de vedere al structurării, lucrarea va fi împărțită în 8 capitole, fiecare capitol urmând sa trateze un aspect specific din proiectarea, implementarea și evaluarea sistemului Spider, sau aspecte generale legate de managementul proiectelor sau sarcinilor. Spre exemplu, în primul capitol vom explora concepte teoretice de bază legate de managementul proiectelor și metoda Kanban. În al doilea capitol vom prezenta 5 aplicații din domeniul gestionării proiectelor IT ce utilizează Kanban sau alte metode asemenea. Începând cu capitolul trei ne vom concentra pe sistemul Spider, și vom începe prin explorarea tehnologiilor folosite în implementarea acestei aplicații. În capitolele patru, cinci și șase ne vom uita la specificațiile, proiectarea și respectiv implementarea aplicației, vom crea un manual de utilizare pentru sistem în capitolul șapte, și vom încheia lucrarea prin prezentarea unor concluzii și direcții viitoare pentru aplicație.

# STUDIU BIBLIOGRAFIC

În acest capitol vom analiza și compara câteva aplicații asemănătoare cu Spider.



## JIRA

Atlassian este o companie fondată în 2001 de către doi studenți din Australia. Prima lor aplicație lansată pe piață un an mai târziu a fost Jira[[1]](#footnote-1), un sistem utilizat de companiile din industria IT pentru urmărirea erorilor și problemelor din diferite aplicații software (bug tracker). De atunci, aplicația a fost mult îmbunătățită, primind multe funcționalități noi, iar în ziua de astăzi este una dintre cele mai cunoscute aplicații de management, utilizată atât în IT cât și în multe alte domenii[[2]](#footnote-2).

Jira oferă foarte multe funcționalități utilizatorului, iar aceste funcționalități pot fi extinse prin intermediul unor module (plătite sau gratuite, în funcție de modul). Componentele principale ale aplicației sunt: Roadmaps (diagrame Gantt), Boards (tablouri Scrum sau Kanban), Reports and insights (Rapoarte), Automation (automatizări de procese și/sau notificări) și Code and deployment status (vizionarea de repository-uri și deployment-uri în funcție de mediul de dezvoltare)[[3]](#footnote-3).

Una din componentele principale este reprezentată de „Roadmaps” (foi de parcurs) și reprezintă o imagine de ansamblu a planificării și progresului unui proiect. Unul din aspectele esențiale ale acestei componente este flexibilitatea, deoarece managerii pot adăuga, actualiza sau șterge elemente precum epice (un grup mare de sarcini cu elemente comune), stories (sarcini care formează un epic), date și elemente ce indică progresul (estimări de timp pentru sprinturi sau epicuri împreună cu elemente vizuale ce arată stadiul aplicației), dependencies (dependințe între epicuri sau stories), etc.

Figura 2.1.1 de mai jos reprezintă o altă componentă a Jira, și anume un tablou Kanban. Aici, echipele de dezvoltare au posibilitatea să își creeze propriile tablouri (în Jira, coloanele sunt personalizabile ca număr și titlu), și să adauge sarcini sau probleme în aceste tablouri. Acestea ajută la vizualizarea sarcinilor de lucru, limitează numărul task-urilor ce trebuie implementate (WIP – Work In Progress), maximizează eficiența echipei de dezvoltare și ajută la comunicarea stării sarcinilor în timp real.

A screenshot of a computer

Description automatically generated with medium confidence

Figura 2.1.1 – Tablou Kanban din Jira (sursa: pagina web [Jira Support](https://support.atlassian.com/jira-work-management/docs/what-can-i-do-on-a-board/))

Componenta responsabilă de rapoarte din Jira se numește „Reports and insights”. Oferă managerului (sau altor persoane interesate din echipa de dezvoltare) diverse rapoarte gata făcute despre statusul proiectului, sau capabilitatea de a crea rapoarte personalizate în funcție de nevoile acestuia.

Ultimele două componente, cea de automatizare și cea de cod și deployment oferă utilizatorilor oportunitatea de a integra Jira cu alte sisteme precum Github, Bitbucket, Microsoft Teams, Slack și altele. Prin aceste integrări, se pot trimite de exemplu notificări către utilizatori sau grupuri de Teams/Slack, se pot primi notificări legate de statusul deployment-urilor, se pot aloca sarcini în mod automat echipei de dezvoltare în funcție de diferite condiții, etc.

## MONDAY.COM

Monday.com este o companie fondată în 2012 sub numele „datapulse” în Tel Aviv, Israel, iar în 2014 și-a schimbat numele, devenind „Monday.com”[[4]](#footnote-4). Platforma oferă trei categorii de produse țintite către diferite domenii: Sales CRM pentru vânzări, „Monday work management” pentru managementul proiectelor și marketing, și „Monday dev”, o aplicație pentru managementul proiectelor din industria IT. Deși Monday.com deține aplicația „**monday work management”**, aceasta aplicație este (după cum am menționat mai sus), orientată către managementul oricărui tip de proiect, așa căpentru lucrarea de față ne vom îndrepta atenția către „**monday dev”** deoarece aceasta este creată în mod special pentru industria IT.

**Monday dev** oferă o gamă largă de servicii precum planificare (Roadmap – diagrame Gantt), management-ul sprinturilor, managementul proiectelor și bug tracking.

Roadmap este la bază o diagrama Gantt ce prezintă o imagine de ansamblu a planificării și progresului unui proiect. Este folosită deoarece acest tip de reprezentare ușurează mult determinarea statutului task-urilor, datele de începere și terminare a sarcinilor și planificarea acestora. Tot în Roadmap se găsește și tabloul Kanban, în care echipa de dezvoltare poate vedea mai ușor statusul sarcinilor din sprint-ul curent. În figura 2.2.1 de mai jos se poate vedea un Roadmap din Monday Dev, ce arată sarcinile de pe un proiect.

O imagine care conține text, captură de ecran, software, Pictogramă computer

Descriere generată automat

Figura 2.2.1 – Pagina principala de task-uri din Monday dev (Roadmap)

Un Sprint este una din componentele Scrum și reprezintă o perioadă fixă de timp (de obicei de 2 săptămâni, dar poate fi fixată între 1 și 4 săptămâni), în care o echipă dezvoltă, testează și livrează un set de funcționalități determinate la începutul sprintului. Monday Dev are un serviciu prin care managerul de proiect/program poate gestiona sprinturi: creare (setare date de început/final), adăugare task-uri, modificare, alocări de persoane, etc.

Partea de bug tracking (urmărirea problemelor) este de asemenea una dintre cele mai importante servicii oferite de Monday Dev. Este reprezentată de un tabel în care echipa de dezvoltare (în special persoanele responsabile de partea de testare) adaugă probleme (buguri) noi sau gestionează problemele deja găsite.

Toate aceste componente menționate anterior sunt gestionate la nivel de proiect – fiecare proiect are propriul sau Roadmap, propriile sarcini și buguri, etc. Astfel, echipele de dezvoltare vor vedea doar task-urile sau bugurile de pe proiectul la care lucrează, și nu pot vedea task-uri de pe alte proiecte decât dacă schimbă proiectul (cu condiția să aibă acces de la Administrator pentru un alt proiect).

## YOUTRACK

YouTrack este o aplicație de gestionare buguri si proiecte lansată de compania cehă JetBrains în 2009[[5]](#footnote-5). Ca și celelalte sisteme analizate mai sus, are o gamă largă de funcționalități, precum tablouri Kanban (sub numele Agile Boards), gestionarea sarcinilor, sau rapoarte, dar conține și elemente de gestionare a echipei de dezvoltare, având integrate module precum monitorizarea timpului petrecut pe proiect sau pe sarcină (Time Tracking).

Pe partea de planificare, YouTrack oferă diagrame Gantt și tablouri Kanban personalizabile – există tablouri preconfigurate (un exemplu de tablou Kanban poate fi văzut în figura 2.3.1 de mai jos), dar echipele de dezvoltare își pot crea propriile coloane. Funcționalitățile de gestionare a sarcinilor sunt făcute prin modulul Tasks, unde membrii echipei pot căuta diferite sarcini, pot adăuga task-uri noi, pot șterge sau modifica task-uri existente și pot aloca aceste sarcini altor membri din echipă.

Modulul Dashboard este modulul responsabil de rapoarte. Ca și la modulul Kanban, există rapoarte preconfigurate, dar utilizatorii pot folosi diferite funcționalități din Dashboard pentru a-și crea propriile rapoarte.

Time Tracking este o componenta a YouTrack prin care fiecare membru a echipei de dezvoltare poate adăuga timpul petrecut pentru o anumită sarcină. Astfel, managerul de proiect sau șeful de echipă poate crea diferite rapoarte sau poate monitoriza activitatea subordonaților săi.

La fel ca Jira, YouTrack poate fi integrat cu diferite aplicații precum Slack sau GitHub, dar în plus poate fi integrat cu alte aplicații JetBrains precum IntelliJ IDEA (un IDE foarte popular pentru Java) sau PhpStorm (IDE pentru PHP) sau cu aplicații de helpdesk precum Zendesk[[6]](#footnote-6).

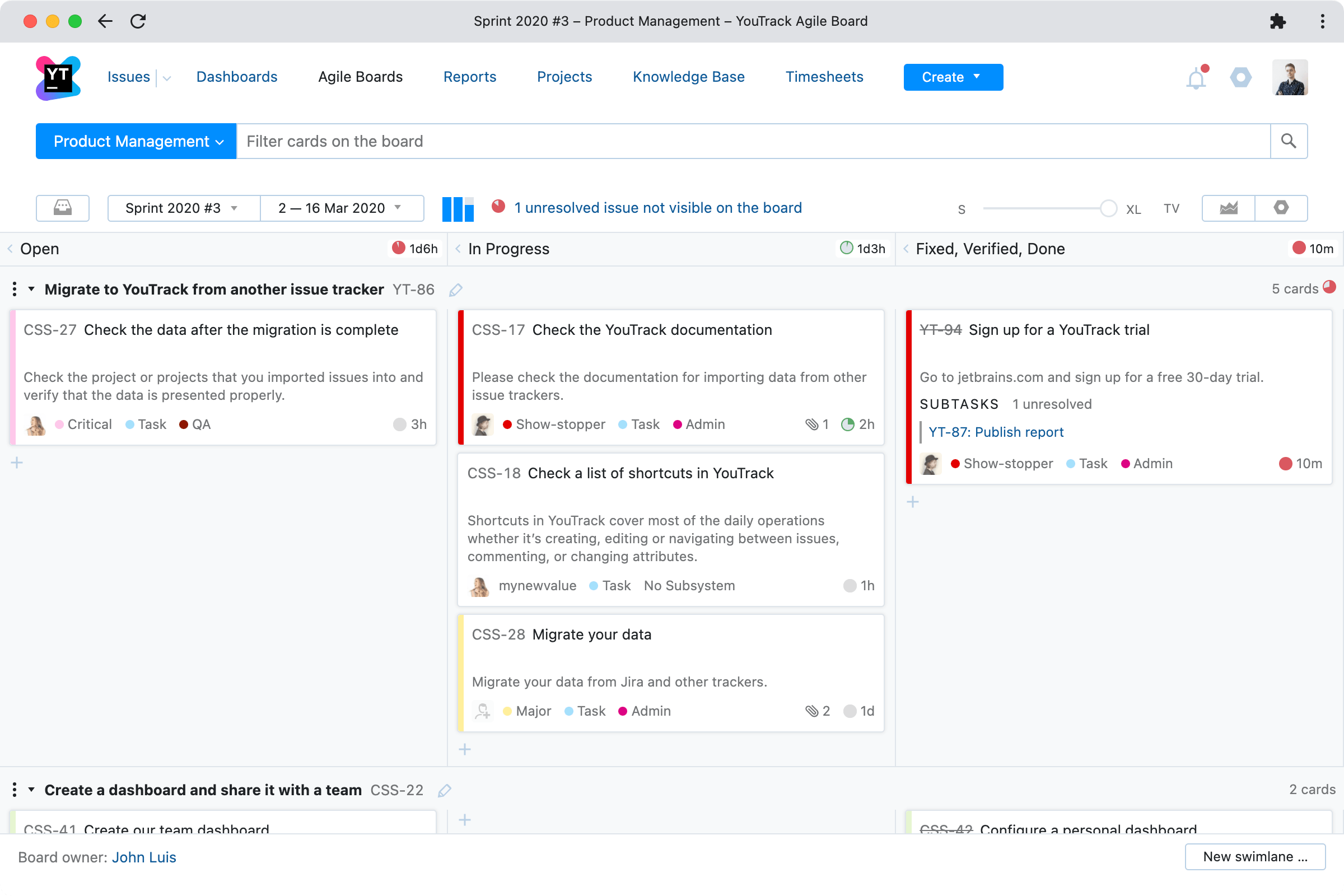


Figura 2.3.1 – Tablou Kanban din YouTrack (sursa: [YouTrack](https://www.jetbrains.com/youtrack/features/agile_project_management.html))

## TRELLO

Trello este o aplicație de management al proiectelor lansată în 2011 de către Fog Creek Software și achiziționată în 2017 de către Atlassian, firma ce deține și Jira[[7]](#footnote-7).

Spre deosebire de Jira, Trello nu oferă la fel de multe funcționalități (principala funcționalitate fiind cea de creare tablouri Kanban personalizabile), și astfel este folosit în general de către firme (sau echipe de dezvoltare) care nu au nevoie de o aplicație de gestionare foarte avansată. Un exemplu de tablou Kanban din Trello poate fi văzut în figura 2.4.1 de mai jos.

Avantajul major al Trello este că poate fi integrat cu foarte multe aplicații (în funcție de necesități), plaja de integrări fiind mult mai mare decât la aplicațiile analizate anterior. Pe lângă prețul redus, plaja largă de integrări face din Trello un sistem folosit nu doar în industria IT.

Un alt avantaj al Trello este reprezentat de către Hărți (Trello Maps). Aceasta este o funcționalitate prin care echipele pot adăuga hărți și locații în sarcini sau proiecte. Aceste hărți sunt de ajutor firmelor care lucrează de exemplu în industria de imobiliare sau construcții.

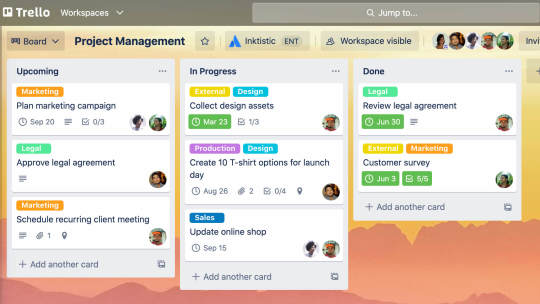


Figura 2.4.1 – Tablou Kanban din Trello (sursa: [Trello](https://trello.com/guide/trello-101))

## REDMINE

Redmine este o platformă de gestionare a proiectelor și de urmărire a problemelor (bug tracking) lansată în 2006 de către un dezvoltator pe nume Jean-Philippe Lang[[8]](#footnote-8). Aceasta este singura aplicație open-source (gratuită) analizată pentru prezenta lucrare.

Redmine a fost conceput pentru a gestiona și planifica diferite tipuri de proiecte și sarcini, pentru a monitoriza progresul și pentru a urmări și gestiona probleme (buguri). Oferă o gamă de funcționalități generale pentru facilitarea gestionării unei echipe precum gestionarea timpului (time-tracking) și gestionarea de documente (de la atașarea de capturi de ecran la buguri până la atașarea de documente în format Word sau Excel pentru crearea de baze de date cu informații despre proiect), dar de asemenea oferă și funcționalități de gestionare a proiectelor și sarcinilor precum crearea și administrarea de proiecte și alocarea de resurse, urmărirea problemelor și sarcinilor, și integrarea cu diferite instrumente de control a versiunilor precum Git.

Deoarece Redmine este un proiect de tip open-source, multe funcționalități nu sunt disponibile în mod direct (de la instalare), dar pot fi instalate cu ajutorul unor pluginuri, în acest stadiu fiind și tablourile Kanban – pentru a putea folosi tablouri Kanban în aplicație, este necesar ca administratorul Redmine să instaleze pluginul necesar.

O imagine care conține text, captură de ecran, software, Pagină web

Descriere generată automat

Figura 2.5.1 – Tablou Kanban din Redmine (sursa: [Redmine](https://www.redmine.org/plugins/redmine_agile))

## COMPARAȚIE APLICAȚII ANALIZATE

Următorul tabel compară funcționalitățile Spider cu aplicațiile analizate pentru a sublinia diferențele dintre ele.

Tabel 1. Compararea aplicațiilor

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Caracteristici | Jira | Monday.com | YouTrack | Trello | Redmine | Spider |
| Tablou Kanban | Da | Da | Da | Da | Da, prin plugin | Da |
| Gestionare proiecte | Da | Da | Da | Da | Da | Da |
| Gestionare sarcini | Da | Da | Da | Da | Da | Da |
| Integrare cu alte aplicații | Da | Da | Da | Da | Da, prin pluginuri | Nu |
| Time Tracking | Da, prin plugin | Nu | Da | Da, prin plugin | Da, prin plugin | Nu |
| Rapoarte | Da | Da | Da | Da | Da | Nu |
| Automatizare | Da | Da | Da | Da | Nu | Nu |
| Roadmaps | Da | Da | Da | Nu | Da | Nu |
| Hărți | Nu | Nu | Nu | Da | Nu | Nu |

# FUNDAMENTARE TEORETICĂ

Începând cu acest capitol ne vom uita la proiectarea și implementarea aplicației Spider. Astfel, acest prim 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 folosesc NodeJS împreună cu Express (din moment ce aplicația este bazată pe JavaScript în general și NodeJS în particular, trebuie menționat faptul că toate bibliotecile sunt instalate cu ajutorul NPM, acesta fiind tehnologia principală folosită de NodeJS pentru managementul modulelor). Pentru baza de date folosesc PostgreSQL, iar pentru cache și token management folosesc Redis. În continuare, vom detalia aceste tehnologii și biblioteci și vom motiva alegerea lor în crearea aplicației. Totuși, 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 către bibliotecile folosite în crearea aplicației client.



## APLICAȚIA CLIENT

Pentru aplicația client – frontend - au fost folosite următoarele tehnologii și biblioteci:

1. **Axios:** conform site-ului oficial, Axios este un client HTTP pentru browser și node.js[[9]](#footnote-9). 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 implică operații CRUD efectuate asupra datelor din aplicație sunt făcute folosind această bibliotecă: pentru Create-Creare se folosește axios.POST, pentru Read-Citire avem axios.GET, pentru Update-Modificare se folosește axios.PUT iar pentru Delete-Ștergere se folosește axios.DELETE). Am ales această bibliotecă în detrimentul altora precum Fetch deoarece este mai simplu de utilizat și mai sigur (având integrată protecție anti CSRF).
2. **React**: este o bibliotecă JavaScript dezvoltată de Meta, folosită în construirea interfețelor de utilizator (UI) pentru aplicații web și aplicații mobile[[10]](#footnote-10). 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[[11]](#footnote-11). 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[[12]](#footnote-12). 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 și în același timp este și foarte puternică, oferind o performanță ridicată și flexibilitate în personalizare.
4. **React-hook-form**: este o bibliotecă simplă, flexibilă și performantă, folosită pentru gestionarea formularelor în aplicațiile React[[13]](#footnote-13). Deși există și alte biblioteci care ajută la gestionarea formularelor (precum Formik), am ales această bibliotecă deoarece este integrată direct cu React, oferă o performanță ridicată comparativ cu celelalte biblioteci și este ușor de utilizat[[14]](#footnote-14).
5. **React-icons**: această bibliotecă oferă o soluție ușoară și convenabilă pentru utilizarea pictogramelor în aplicații React. Am ales această soluție deoarece oferă o gama largă de seturi de pictograme ce pot fi utilizate printr-un simplu import în aplicație (pictogramele sunt folosite de exemplu în meniul lateral sau în profilul utilizatorului). De asemenea, aceste pictograme pot fi ușor personalizate folosind CSS și au o performanță optimizată pentru a asigura o încărcare rapidă și o experiență plăcută pentru utilizatori.
6. **React-modal**: este o bibliotecă React utilizată pentru a crea și gestiona ferestre modale (pop-up) în aplicațiile web într-o manieră simplificată, oferind un control flexibil și complet asupra afișării și utilizării acestora. Am ales această bibliotecă deoarece este ușor de utilizat - oferă o componentă React ce poate fi personalizată folosind diferite proprietăți create în React și este ușor de integrat cu alte componente sau biblioteci UI precum TailwindCss pentru personalizare vizuală. De asemenea, nu este necesar să gestionăm suprafețe precum overlay-ul, evitând astfel bug-urile de genul în care click-urile efectuate pe elementele din spatele overlay-ului sunt înregistrate și activate înainte sau după închiderea modalului.
7. **React-router-dom**: o bibliotecă esențială folosită pentru gestionarea rutelor și navigării pe partea de client (client side routing) în aplicațiile web bazate pe React, prin care dezvoltatorii pot defini configurația rutelor într-un mod declarativ, utilizând componente React. Acest stil de rutare permite aplicației sa schimbe un URL-ul fără a cere un document nou de la server iar in acest fel, aplicația poate să afișeze imediat o componentă nouă.
8. **React-switch**: este o bibliotecă React utilizată pentru crearea și gestionarea comutatoarelor în aplicațiile web. Oferă o componentă <Switch> care poate fi personalizată vizual utilizând CSS (sau o biblioteca UI precum Tailwind). In Spider este folosita doar pentru schimbarea temei din zi în noapte (light și dark).
9. **React-toastify**: este utilizată pentru crearea și gestionarea notificărilor dintr-o aplicație React pe baza componentelor. Am preferat această soluție deoarece este ușor de utilizat, permite personalizarea aspectului și comportamentului notificărilor (precum culoare, poziționare, timp de afișare, etc). De asemenea, pot fi setate să fie interactive, dezvoltatorii putând adăuga butoane ce execută diferite acțiuni în funcție de tipul de notificare.
10. **Yup**: este o bibliotecă folosită pentru a aplica validări pe datele de intrare din formulare folosind o schemă definită de dezvoltator. De asemenea, dezvoltatorii pot crea mesaje de eroare personalizate care apoi sunt afișate în aplicație. Toate formularele din Spider folosesc yup pentru validare și afișare mesaje informative sau de eroare (în cazul în care cerințele din schema nu sunt îndeplinite).
11. **TailwindCSS**: conform website-ului oficial, Tailwind CSS este un framework CSS „utility-first” cu clase precum „flex”, „pt-4”, „rotate-90” și altele, ce pot fi utilizate pentru a construi orice design, direct in fișierul HTML[[15]](#footnote-15). Spre deosebire de alte framework-uri precum Bootstrap care oferă componente create în mod implicit, Tailwind CSS pune la dispoziție un set de extins de clase utilitare care pot fi combinate în diferite moduri direct în fișierul HTML (precum inline CSS) pentru a crea interfețe personalizate. De asemenea, pe lângă setul de clase predefinite, Tailwind oferă și posibilitatea de a adăuga propriile clase și a crea noi stiluri fără a fi nevoie de dependințe suplimentare sau de seturi predefinite de componente. Pentru Spider am ales Tailwind pentru a putea implementa mai ușor dark mode și a avea o flexibilitate mai mare asupra personalizării aplicației.

## APLICAȚIA SERVER ȘI API

Pentru aplicația server – backend – au fost folosite următoarele tehnologii și biblioteci:

1. **NodeJS**: este un mediu de execuție open-source care permite crearea de aplicații server-side, API-uri, aplicații de timp real sau alte aplicații de rețea folosind limbajul JavaScript[[16]](#footnote-16), în acest mod eliminându-se nevoia de a utiliza alte limbaje de programare și tehnologii precum Java sau Python. Totodată, NodeJS vine împreună cu propriul sau package manager (aplicație de gestionare a modulelor) numit npm prin intermediul cărora dezvoltatorii pot instala biblioteci JavaScript care pot fi folosite în crearea de aplicații. Exista și alte tehnologii utilizate în crearea de aplicații server-side precum Java, Python (menționate deja) sau .NET, dar am ales NodeJS deoarece este simplu de instalat și configurat, vine cu npm unde poți găsi o gamă largă de module ce pot fi folosite în mod direct fără a fi nevoie de configurații avansate și salvează timp în crearea aplicației, oferă o performanță ridicată și în același timp nu are cerințe de sistem mari.
2. **ExpressJS:** este un framework web minimalist și rapid pentru NodeJS[[17]](#footnote-17) folosit în crearea de aplicații web și API-uri. Deși NodeJS poate fi utilizat fără Express, acesta din urma oferă un set de funcții și metode simplificate pentru gestiunea rutărilor, cererilor și răspunsurilor HTML. De asemenea, poate fi integrat cu diferite module instalate prin npm, pentru a adăuga funcționalități suplimentare aplicației dezvoltate.
3. **Redis (ioredis)**: Redis este un sistem de stocare in-memory (adică datele sunt salvate in memoria RAM) de tip cheie-valoare folosit ca baza de date, cache, broker de mesaje sau motor de streaming și deoarece este un sistem in-memory, Redis oferă o performanță ridicată, permițând acces rapid la date. Totuși, trebuie menționat faptul că din moment ce memoria RAM este memorie volatilă, dacă serverul este oprit în vreun fel (restart, întreruperea alimentarii cu electricitate, etc), aceste date se vor pierde (bineînțeles, se pot folosi tehnici de salvare a datelor pe medii de stocare persistente pentru a minimiza pierderile). În aplicația propusă folosesc Redis pentru a salva token-urile neexpirate care nu mai au acces să folosească aplicația (de exemplu după Logout), iar pentru asta este nevoie de o biblioteca pentru NodeJS care ne permite sa interacționam cu serverul. Deși există mai multe biblioteci, am ales sa folosesc ioredis deoarece este o implementare completă care oferă suportul necesar, și în același timp este ușor de utilizat, nu necesită configurări avansate și are o performanță ridicată.
4. **Bcryptjs**: este o bibliotecă prin care funcția de criptare parole (password hash) BCRYPT este implementată în JavaScript. Stocarea parolelor sub forma de text (plain-text) în baza de date reprezintă o breșă de securitate majoră în industria IT (în cazul în care baza de date este compromisă, atacatorii pot utiliza parolele pentru a se conecta direct în contul utilizatorilor). Din acest motiv, utilizăm bcryptjs pentru a crea un hash care va fi salvat în baza de date și care va fi comparat cu un nou hash creat în momentul în care utilizatorul încearcă să se logheze în aplicație.
5. **Body-parser**: este o bibliotecă NodeJS folosită în analizarea datelor primite printr-o cerere HTTP. În momentul în care serverul primește date printr-o cerere HTTP (de exemplu o cerere de tip GET sau un POST trimisă de client), datele respective nu sunt ușor de manipulat. Body-parser preia aceste date (care pot fi in format text, JSON, etc), și le transformă în obiecte JavaScript accesibile dezvoltatorilor într-un mod facil. Deși există și alte biblioteci care fac aceasta transformare, am ales body-parser deoarece are o comunitate largă de utilizatori, este stabil și are multe funcții integrate folositoare în timpul dezvoltării aplicației.
6. **Cors**: este o bibliotecă prin care se gestionează mecanismul CORS. Pentru a înțelege acest mecanism, trebuie să menționăm faptul că browserele restricționează cererile HTTP dintre două origini din cauza unei politici numite „same-origin policy” - presupunând că avem 2 domenii diferite abc.com și xyz.com, această politică permite cererile dintre abc.com (client) și abc.com (server), dar nu permite cererile dintre abc.com și xyz.com (figura 3.2.1 de mai jos oferă o reprezentare vizuală a acestei politici). Din moment ce aplicația propusă este instalată pe 2 servere cu domenii diferite, avem nevoie de un sistem prin care să ridicăm această politică pentru cererile HTTP dintre aceste două domenii. Mecanismul CORS (Cross-Origin Resource Sharing) este un mecanism prin care i se dă voie unui server să indice orice origine prin care i se permite unui browser sa acceseze resurse de pe acel server[[18]](#footnote-18), creând astfel o „poartă” prin care se ocolește politica „same-origin”.

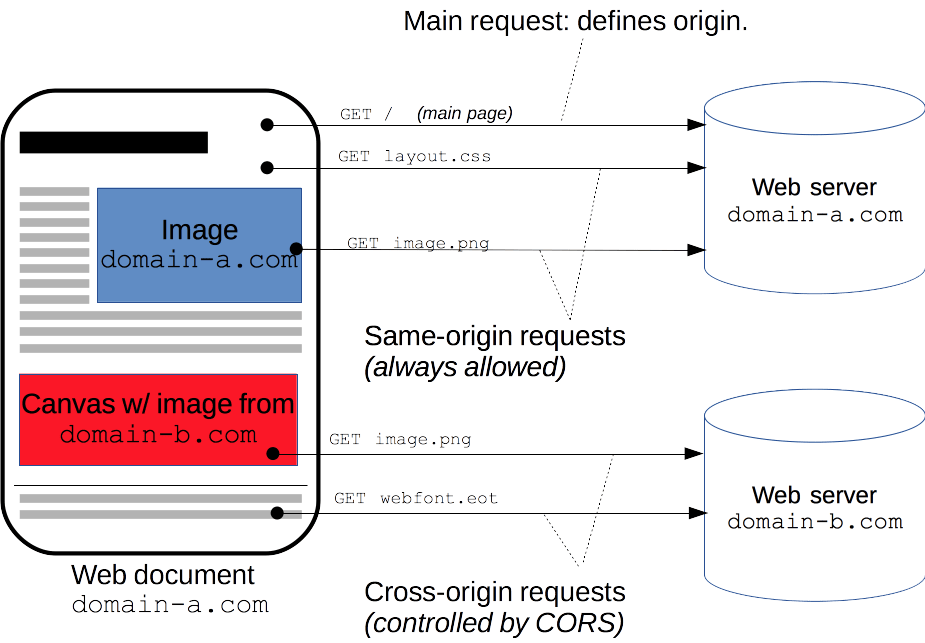


Figura 3.2.1 - Mecanismul CORS și politica same-origin (sursa: [Mozilla Web Docs](https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS))

1. **Dotenv**: este o bibliotecă prin care se permite folosirea variabilelor de mediu. Cu ajutorul acestora se stochează configurații specifice unei aplicații sau unui mediu (precum producție, test sau dezvoltare) într-un proiect NodeJS. Pentru Spider, aceasta bibliotecă este folosită pentru adresele către Redis și PostgreSQL, precum și pentru secretele prin care se creează json web tokens și parola pentru administrator. De asemenea, tot în variabilele de mediu avem și datele pentru sistemul de email (adresa și parola contului de email, precum și sistemul folosit).
2. **Generate-password**: acesta este o librărie care este utilizată pentru a genera parole aleatorii în funcție de configurația aleasă (de exemplu numărul de caractere utilizate, includerea de caractere speciale, litere majuscule sau minuscule, cifre, etc). Această bibliotecă este folosită pentru crearea de parole atunci când un utilizator nou este creat (utilizatorul primind un email prin care este notificat de crearea contului, email în care este trecuta și aceasta parolă generată aleator).
3. **Jsonwebtoken**: este un tip de reprezentare a unor revendicări (informații afirmate despre un anumit subiect) trimise între doi participanți[[19]](#footnote-19), și este format din 3 părți principale: header (care conține tipul de token și algoritmul de semnare utilizat), revendicările trimise (payload) și semnătura (o aplicare a algoritmului de semnare pe header și payload utilizând o cheie secretă cunoscută doar de server (în cazul nostru cheia este salvată în variabilele de mediu, bineînțeles în mod criptat). Biblioteca NodeJS care implementează această reprezentare se numește „jsonwebtoken” (JWT) și este folosită în aplicația propusă (Spider) pentru autentificare. La fiecare login, funcția de creare token-uri este apelată, iar token-ul rezultat este trimis către client ca răspuns la o autentificare efectuată cu succes (daca utilizatorul nu există sau parola este incorectă, nu se va crea nici un token), și este salvat în memoria locala a browserului (localStorage). Acest token va fi folosit în continuare la fiecare cerere făcută către server – va fi trimis în Authorization header – și va fi verificat pentru a demonstra (sau infirma) faptul că utilizatorul care a făcut cererea este autentificat, și are accesul necesar să facă acea cerere. În aplicația propusă, token-ul expira la 8 ore după creare (dar durata poate fi personalizată în funcție de cerințe).
4. **Nodemailer**: este biblioteca pentru aplicații NodeJS utilizată pentru trimiterea de emailuri în format text sau html. În Spider este folosit într-unul din următoarele 3 cazuri: atunci când este creat un cont nou (utilizatorul va primi un email de confirmare și parola pe adresa adăugată atunci când se creează contul), când utilizatorul își schimba parola sau atunci când utilizatorul și-a uitat parola și trebuie să o reseteze (va primi un email cu un link ce deschide pagina de resetare parola).
5. **Sequelize**: este o bibliotecă ORM (Object-Relational Mapping) NodeJS bazată pe promisiuni, folosită în interacțiunea cu diferite baze de date relaționale precum MySQL, MsSQL, PostgreSQL, etc. Cu ajutorul acestei biblioteci, în loc să folosească interogări SQL tradiționale, dezvoltatorii gestionează și manipulează datele dintr-o bază de date compatibilă folosind obiecte și metode (de exemplu, în loc de „SELECT \* FROM users” avem „Users.findAll()”), Sequelize transformând în mod automat obiectele în structuri de date compatibile cu baza de date aleasă (și invers). Am ales Sequelize deoarece simplifică operațiile CRUD (Create, Read, Update, Delete – Creare, Citire, Modificare, Ștergere) și interogările în baza de date, putem versiona schema bazei de date, și se integrează bine cu NodeJS precum și cu alte module populare precum ExpressJS pentru crearea de API-uri. Pentru ca Sequelize să funcționeze corect, este nevoie să instalăm anumite biblioteci în funcție de baza de date utilizată (de exemplu mysql2 pentru MySQL, mariadb pentru MariaDB, etc). Deoarece Spider utilizează PostgreSQL pentru stocarea datelor, Sequelize are nevoie de alte 2 biblioteci (dependințe): **pg** si **pg-hstore**. **Pg** este un driver PostgreSQL pentru NodeJS ce oferă funcționalitățile pentru interogări și pentru comunicarea cu baza de date, în timp ce **pg-hstore** este un modul suplimentar pentru **pg** ce permite transformarea datelor JSON în format hstore și invers.
6. **Nodemon**: în primul rând trebuie menționat faptul că deși această biblioteca nu este folosită în mediul de producție (poate fi folosita din punct de vedere tehnic, dar nu este recomandat), este folosită în mediul de dezvoltare. Pentru a înțelege nevoia de utilizare a acestei unelte, trebuie menționat faptul că atunci când un dezvoltator scrie cod server-side, acel cod nu este utilizat imediat în aplicație, dezvoltatorul fiind nevoit sa repornească manual serverul pentru a vedea modificările făcute. Resetarea manuala a serverului, deși nu este un proces dificil, încetinește totuși procesul de dezvoltare, lucru care trebuie evitat, și astfel, pentru a reduce timpul pierdut, a apărut nevoia de a avea o unealtă de automatizare a acestui proces de repornire. Soluția găsita pentru a rezolva această problema este biblioteca numită nodemon unde în loc să rulăm o aplicație NodeJS folosind comanda „node start” vom folosi „nodemon start” ceea ce înseamnă ca nodemon va porni serverul și va monitoriza toate fișierele proiectului iar atunci când acesta detectează o modificare într-un fișier (practic atunci când un fișier este salvat), va reporni automat serverul fără a mai fi nevoie de intervenție umană.

# SPECIFICAȚIILE APLICAȚIEI



## DESCRIEREA APLICAȚIEI

Spider este o aplicație de gestionare a proiectelor și sarcinilor utilizând metoda Kanban. Este o aplicație de tip client-server, creată integral cu JavaScript și CSS utilizând diferite framework-uri precum React și ExpressJS. Este formată din două aplicații: aplicația client (frontend), aplicația server (backend și API), o bază de date Redis pentru gestionarea token-urilor și o bază de date PostgreSQL pentru salvarea utilizatorilor, proiectelor și task-urilor.

## FUNCȚIILE APLICAȚIEI

În aplicația Spider există următoarele funcționalități:

1. Login
2. Resetare parolă uitată
3. Adăugare utilizator nou (doar pentru Administrator):
   1. Administrator
   2. Utilizator
4. Modificare date utilizator (doar pentru Administrator)
5. Adăugare proiect nou (doar pentru Administrator)
6. Modificare proiect (doar pentru Administrator)
7. Adăugare task nou
8. Modificare task
9. Ștergere task
10. Vizualizare tablou Kanban
11. Schimbare status task folosind tabloul Kanban (drag & drop)
12. Gestionare token-uri de acces
13. Afișare profil de utilizator
    1. Schimbare parolă utilizator
14. Notificări prin email:
    1. La crearea unui utilizator nou
    2. La resetare parolă
    3. La schimbare parolă

## INTERFAȚA CU UTILIZATORUL

Fiind o aplicație web, interfața este creată folosind HTML, JavaScript (React) și CSS (TailwindCSS). Este o interfață modernă, receptivă la schimbările de mărime a browser-ului dar nu este optimizată pentru telefoane mobile.

Există două teme în aplicație: tema de zi și tema de noapte (light și dark). Modul implicit este dat de către sistem (există o opțiune în sistemul de operare prin care utilizatorul poate selecta tema implicită ca fiind light sau dark), dar există și un buton în aplicație prin care tema poate fi schimbată de către utilizator. Gama de culori din tema de zi este bazată pe diferite nuanțe de gri deschis, cu excepția butoanelor, unde se folosește albastru (pentru butoanele de adăugare și selectare), verde (pentru modificarea datelor) și roșu (pentru ștergerea datelor din aplicație). În tema de noapte, folosim diferite nuanțe de gri sau albastru închis pentru a nu obosi ochii, dar butoanele nu se schimbă, rămân cu aceleași nuanțe ca în modul de zi.

Fiecare formular din aplicație are validări, iar în cazul în care datele introduse în câmpul respectiv nu sunt corecte, atunci când utilizatorul apasă pe butonul de salvare, sub câmpul completat incorect apare un mesaj de eroare în care se explică de ce câmpul respectiv nu este completat corect. De asemenea, de fiecare dată când un utilizator completează o acțiune cu succes, apare un popup (toast) de notificare. Defapt, fiecare acțiune din aplicație are un răspuns visual.

Sidebar-ul nu este fix, poate fi extins sau retras pentru a oferi un spațiu mai larg pentru restul aplicației. Atunci când este retras, textul dispare dar rămân pictogramele care pot fi utilizate (nu este necesar ca sidebar-ul să fie extins pentru a utiliza butoanele).

## BAZA DE DATE

Pentru moment Spider folosește cinci tabele într-o bază de date PostgreSQL:

1. Proiecte
2. Utilizatori (nume: Useri)
3. Rolul utilizatorilor (nume: UserRole)
4. Task-uri (nume: Taskuri)
5. Statusul task-urilor (nume: TaskStatus)

Figura de mai jos reprezintă tabelele din aplicație, numele și tipul coloanelor conținute și interconexiunile dintre ele (chei primare respectiv secundare).

O imagine care conține text, captură de ecran, diagramă, număr

Descriere generată automat

Figura 6.4.1 – Structura bazei de date

# PROIECTAREA APLICAȚIEI

Acest capitol face referire la proiectarea aplicației. Vor fi explicate componentele existente în aplicație și baza de date în detaliu.



## COMPONENTELE APLICAȚIEI

Sistemul Spider este format din patru componente majore: clientul (aplicația frontend), serverul/API-ul (backend), baza de date PostgreSQL pentru gestionarea informației și baza de date Redis pentru managementul token-urilor de login și resetare parolă.

În frontend avem următoarele componente:

1. **Login** – este un formular unde un utilizator poate adăuga adresa de email și parola. Dacă aceste două câmpuri sunt completate cu date valide, utilizatorul se poate conecta în aplicație – în caz contrar apare un mesaj de eroare (există un mesaj de eroare ce apare sub câmpul de introducere date pentru cazul în care utilizatorul adaugă o adresă de email cu un format invalid, sau în cazul in care adresa de email și/sau parola nu sunt corecte, apare un mesaj de eroare sub forma de toast).
2. **Cerere resetare parolă** – în cazul în care utilizatorul nu își aduce aminte parola, acesta o poate reseta cu ajutorul formularului de resetare parolă ce poate fi accesat din pagina de login. Atunci când utilizatorul apasă pe „Ai uitat parola?”, este redirecționat către pagina de cerere link pentru resetare parolă. În această pagină utilizatorul trebuie să își adauge adresa de email, iar dacă adresa de email există în baza de date, acesta va primi un email pe acea adresă, în care există link-ul necesar pentru resetare parolă (link-ul este valabil pentru 2 ore). Utilizatorul este notificat printr-o notificare de tip toast în momentul trimiterii cererii de resetare de faptul că în cazul în care adresa de email există, atunci va primi acel link de resetare indiferent dacă adresa de email există sau nu în baza de date.
3. **Resetare parolă** – pentru a putea accesa formularul de resetare parolă, utilizatorul trebuie să apese pe link-ul primit pe adresa sa de email. Dacă token-ul existent în link este valid, atunci utilizatorul va vedea un formular cu două câmpuri: parolă și confirmă parola. Ambele câmpuri au aceleași validări: sunt obligatorii, trebuie să fie între 8 și 32 de caractere și să conțină minim o literă mică, o literă mare, un simbol și o cifră. De asemenea, pentru a evita greșelile de scriere (typo-urile), ambele câmpuri trebuie să conțină aceleași date.
4. **Sidebar-ul (meniul lateral)** – poate fi extins sau retras, conține butoanele de navigare între pagini, butonul de schimbare temă (dark sau light) și numele aplicației împreună cu logo-ul (animat).
5. **Navbar-ul (bara de navigație)** – deși elementele de navigare sunt în meniul lateral, în bara de navigație avem numele și avatarul utilizatorului, împreună cu un mesaj de salutare. Dând click pe nume, se deschide profilul utilizatorului.
6. **Profilul de utilizator –** este deschis prin apăsarea pe numele utilizatorului din bara de navigație. Acest profil conține numele utilizatorului, avatarul, adresa de email și rolul din aplicație (Administrator sau Utilizator). Tot de aici utilizatorul își poate schimba parola sau se poate deloga din aplicație.
7. **Schimbare parolă** – această componentă poate fi găsită în profilul utilizatorului. Atunci când utilizatorul apasă pe acest buton, formularul de schimbare parolă este afișat. Acest formular are trei câmpuri: parola veche (pentru a evita problemele de securitate), parola nouă și confirmare parolă. Toate trei câmpuri sunt obligatorii, iar pentru câmpurile de parolă nouă și confirmare parolă valoarea adăugată trebuie să fie între 8 și 32 de caractere și să conțină minim o literă mică, o literă mare, un simbol și o cifră. De asemenea, pentru a evita greșelile de scriere (typo-urile), ambele câmpuri trebuie să conțină aceleași date. Dacă toate câmpurile sunt completate cu date valide, atunci parola este schimbată, utilizatorul primește un email de notificare și este delogat, datele din localStorage sunt șterse, și un toast pentru confirmare este afișat. Dacă unul sau mai multe câmpuri sunt incorecte, atunci utilizatorul poate vedea mesaje de eroare sub câmpul sau câmpurile incorecte.
8. **Logout** – această componentă se află în profilul utilizatorului. Atunci când acest buton este apăsat, utilizatorul este scos din aplicație, toate datele salvate în localStorage sunt șterse, iar token-ul este adăugat în Redis pentru perioada rămasă pentru a fi invalidat (nu mai poate fi folosit de altcineva).
9. **Proiecte** – în această pagină sunt afișate proiectele existente (nume proiect, descriere și persoana responsabilă, sortate după numele proiectului) și există butoanele „Adăugare”, „Modifică” și „Selectează” proiect, cu excepția cazului în care nu există proiecte în baza de date (atunci este afișat un mesaj de notificare, iar singurul buton existent este cel de Adăugare). La apăsarea butonului de Adăugare sau Modificare apare un formular modal, iar la apăsarea butonului Selectează, id-ul proiectului selectat este salvat în localStorage pentru a putea afișa date doar despre acel proiect în paginile Backlog sau Kanban. Restricții: doar conturile cu rol de Administrator au permisiunea de a adăuga sau modifica proiecte (conturile cu rol de Utilizator putând doar să selecteze proiecte pentru a vedea task-urile și tabloul Kanban din proiectul respectiv).
10. **Modal proiecte** – acest modal conține formularul de adăugare sau modificare proiecte. În cazul în care acest modal a fost deschis prin apăsarea butonului Adăugare, câmpurile din formular vor fi goale, dar în cazul în care modalul a fost deschis pentru a edita un proiect, atunci câmpurile vor fi precompletate cu datele existente. Formularul conține avatarul proiectului și următoarele câmpuri: Avatar, Nume Proiect, Descriere proiect și Responsabil proiect. Deși nu este nevoie ca utilizatorul să adauge un avatar, există o valoare implicită care va fi trimisă în baza de date. Avatarul implicit poate fi schimbat adăugând un link pentru un avatar de pe internet (noul avatar va putea fi văzut imediat).
11. **Backlog** – în această pagină sunt afișate toate sarcinile existente pe proiect (doar titlu și avatarul, numele și prenumele persoanei care a adăugat task-ul, sortate după numele task-ului). Tabelul de afișare va conține titlul sarcinii, numele și avatarul persoanei care a creat task-ul, și butoanele de modificare și ștergere iar în cazul în care nu există task-uri pe proiectul selectat, va fi afișat un mesaj de notificare. Tot în această pagină există și un buton de adăugare task-uri, vizibil pentru toate tipurile de utilizatori (Administratori sau Utilizatori) și un buton de ștergere task-uri prin care se șterge din baza de date task-ul dorit.
12. **Modal taskuri** – acest modal conține formularul de adăugare sau modificare taskuri. În cazul în care acest modal a fost deschis prin apăsarea butonului Adăugare, câmpurile din formular vor fi goale, dar în cazul în care modalul a fost deschis pentru a edita un task, atunci câmpurile vor fi precompletate cu datele existente. Formularul conține două câmpuri (input-uri) – titlul task-ului și descrierea (titlul este obligatoriu, nu și descrierea) și doi selectori (dropdown-uri): status task și persoana care trebuie să lucreze la task (assignee). Toți selectorii sunt obligatorii. De asemenea, acest formular conține și un câmp ce nu poate fi modificat, în care se află numele persoanei care a creat task-ul (pentru un task nou, acest câmp va conține tot timpul numele persoanei locată în acel moment).
13. **Kanban** – această pagină conține tabloul Kanban în care apar toate task-urile de pe proiect, grupate după status și este formată din trei componente: pagina principală ce conține tabloul, coloanele și cardurile de task-uri. În această pagină nu există butoane, dar cardurile pot fi trase și mutate în alte coloane – în acest mod se actualizează automat și statusul task-ului din card-ul respectiv și se salvează în baza de date. În cazul în care numărul de carduri dintr-o coloană devine mai mare decât spațiul disponibil, o bară de scroll va apărea în coloana respectivă. Titlul și numărul coloanelor depinde de numărul și titlul statusurilor
14. **Utilizatori** – în această pagină sunt afișați utilizatorii existenți (avatar, nume, prenume, adresa de email și rolul utilizatorului, sortați după nume și prenume) și butoanele de adăugare și modificare utilizatori. La apăsarea butonului de Adăugare sau Modificare apare un modal ce conține formularul de adăugare sau editare. Restricții: doar conturile cu rol de Administrator au permisiunea de a adăuga sau modifica utilizatori.
15. **Modal utilizatori** – acest modal conține formularul de adăugare sau modificare utilizatori. În cazul în care acest modal a fost deschis prin apăsarea butonului Adăugare, câmpurile din formular vor fi goale, dar în cazul în care modalul a fost deschis pentru a edita un utilizator, atunci câmpurile vor fi precompletate cu datele existente. Formularul conține avatarul utilizatorului și următoarele câmpuri: cale către un avatar, nume, prenume, adresa de email, rolul utilizatorului și un checkbox prin care administratorii pot forța utilizatorul să-și schimbe parola la primul login. Pe lângă aceste câmpuri, în formularul de modificare utilizator mai există un câmp pentru parolă (utilizat doar în cazul în care se dorește schimbarea parolei de către un administrator). Deși nu este nevoie ca utilizatorul să adauge un avatar, există o valoare implicită care va fi trimisă în baza de date. Avatarul implicit poate fi schimbat adăugând un link pentru un avatar de pe internet (noul avatar va putea fi văzut imediat în modal).
16. **Paginare** – aceasta este componenta responsabilă pentru paginare în întreaga aplicație.

Pe partea de backend, în aplicația server și API avem următoarele componente:

1. **Modele** – aici sunt modelele pentru crearea tabelelor din baza de date în format JavaScript obiect. Avem 5 modele, câte unul pentru fiecare tabelă descrisă în [Cap 5.1](#_STOCAREA_INFORMAȚIEI):
   1. **Proiect**
   2. **Task**
   3. **TaskStatus**
   4. **User**
   5. **UserRoles**
2. **Rute** – aici sunt rutele pentru API. În total sunt 9 rute și 21 de componente:
   1. **Rutele de autentificare** (login și logout)
   2. **Ruta pentru schimbarea parolei**
   3. **Rutele pentru gestionarea proiectelor** (afișare cu paginare, actualizare și adăugare)
   4. **Rutele pentru resetare parolă** (cererea pentru resetare parolă și actualizarea efectivă a parolei)
   5. **Rutele pentru gestionarea task-urilor** (preluare task-uri cu și fără paginare, adăugare task nou, actualizare și ștergere)
   6. **Ruta pentru afișarea statusului task-urilor**
   7. **Ruta pentru afișarea rolurilor utilizatorilor**
   8. **Rutele pentru gestionarea utilizatorilor** (preluare utilizatori cu și fără paginare, adăugare, actualizare și ștergere utilizatori)
   9. **Rute pentru validarea token-urilor**
3. **Controllere** – fiecărei componente din rute îi corespunde câte o componentă din controllere, ce conține codul de executat:
   1. **Pentru autentificare:** login și logout
   2. **Pentru schimbare parolă:** changePassword
   3. **Pentru gestionare proiecte:** getAllProjectsWithPagination (preluare proiecte cu paginare), addNewProject (adăugare proiect), updateProject (actualizare proiect)
   4. **Pentru resetare parolă:** resetPassReq(cererea pentru resetare parolă) și passUpdate (actualizarea efectivă a parolei)
   5. **Pentru gestionare task-uri:** getTasks (preluare task-uri cu sau fără paginare în funcție de parametrii trimiși în cerere), addTask (adăugare task), updateTask (actualizare task) și deleteTask (ștergere task)
   6. **Pentru afișarea statusului task-urilor:** getTaskStatus
   7. **Pentru afișarea rolurilor utilizatorilor:** getUserRoles
   8. **Pentru gestionare utilizatori:** getAllUsersWithPagination (preluare utilizatori cu paginare), getUsers (preluare utilizatori fără paginare), addUser (adăugare utilizator) și updateUser (actualizare utilizator)
   9. **Pentru validare token:** validateToken
4. **Middleware** – conține componentele middleware și utilitare:
   1. **Trimitere email:** sendEmail
   2. **Generare parolă aleatorie:** randomPass
   3. **Verificare token:** verifyToken
   4. **Verificare dacă utilizatorul trebuie să schimbe parola:** isRequiredPassChange

## STOCAREA INFORMAȚIEI

Pentru a stoca datele adăugate în aplicația propusă folosim o bază de date de tip PostgreSQL, iar pentru gestionarea tokenurilor folosim Redis, ambele create în Azure Cloud. Baza de date PostgreSQL conține pentru început cinci tabele, după cum urmează:

1. **Proiecte**
2. **Useri**
3. **UserRoles**
4. **Taskuri**
5. **TaskStatus**

În continuare vom detalia aceste tabele și vom explica din ce coloane sunt compuse.

Tabela **Proiecte** va conține proiectele existente în aplicație. Va avea obligatoriu un **id** auto-incremental ca și cheie primară, dat de către baza de date la adăugarea unui nou proiect. Următoarea coloană este reprezentată de **avatarul** proiectului (project icon), și este de tip VARCHAR (STRING). Apoi vom avea **numele** și **descrierea** proiectului, ambele de tip VARCHAR, numele este obligatoriu dar nu și descrierea (poate fi null). Ultima coloană din această tabelă este de tip INT și este reprezentată de id-ul utilizatorului răspunzător de proiect, sub numele de **responsabil**. Asfel, această coloană este este o cheie secundară ce referențiază coloana **id** din tabela Useri.

Tabela **Useri** conține datele utilizatorilor aplicației. Conține o cheie primară cu numele de **id** de tip INT, auto-incrementală, **numele** și **prenumele** utilizatorului (coloane de tip VARCHAR, obligatorii – nu pot fi lăsate necompletate), **emailul** (unic în baza de date, de tip VARCHAR), **parola** și **avatar**ul utilizatorului (ambele de tip VARCHAR și obligatorii), **rol**ul (de tip INT, cheie secundară ce referențiază câmpul id din tabela UserRoles) și o coloană de tip BOOLEAN cu numele **schimbaParola**, prin care se determină dacă utilizatorul trebuie să își schimbe parola la următorul login sau nu.

Următoarea tabelă este tabela **UserRoles** ce conține tipurile de utilizatori existente în aplicație (pentru moment doar Administrator și Utilizator). Este formată din două coloane: **id** – cheie primară de tip INT, și **nume** (VARCHAR).

Tabela a patra are numele **Taskuri** și conține datele adăugate pentru o sarcină din sistem. Conține o cheie primară de tip INT cu numele **id**, auto-incrementală, **titlul** și **descrierea** task-ului, ambele obligatorii, de tip VARCHAR și patru chei secundare de tip INT. Prima cheie secundară cu numele **idProiect** referențiază câmpul **id** din tabela Proiecte (cu ajutorul căruia se detectează ce task aparține cărui proiect), următoarele două chei secundare au numele **reporter** și **assignee** și referențiază câmpul **id** din tabela Useri. Prima dintre aceste două coloane are rolul de a determina utilizatorul care a adăugat task-ul în aplicație, iar a doua coloană are rolul de a determina persoana care trebuie să lucreze pe task-ul respectiv. Ultima cheie secundară din acest tabel este coloana **status** și referențiază câmpul **id** din tabela TaskStatus (este folosită pentru a determina status-ul task-ului).

Ultima tabelă existentă în aplicație este tabela **TaskStatus** și conține o cheie primară de tip INT cu numele **id** și **statusul** de tip VARCHAR. Această tabelă conține potențialele statusuri în care poate fi un task (momentan este precompletat cu șase valori: „Nerezolvate”, „De rezolvat”, „Rezolvate”, „Testate”, „Terminate” și „Livrate”).

Baza de date Redis este utilizată doar atunci când se dorește invalidarea unui token și conține doar token-ul și parametrul de expirare (token-ul va exista în Redis doar până când expiră).

# IMPLEMENTAREA APLICAȚIEI

Aplicația este implementată integral folosind limbajul de programare JavaScript. Pentru frontend (aplicația client) folosim o bibliotecă JavaScript numită React, pentru backend și API (aplicația server) folosim NodeJS și ExpressJS, iar pentru a face cereri din aplicația client către server se utilizează axios.

Toate cererile și răspunsurile făcute de către client către server folosesc antetul Content-Type cu valoarea **„application/json”**. De asemenea, pentru toate cererile (cu excepția Login și Resetare Parola) se trimite și token-ul primit la login folosind antetul Authorisation cu valoarea **„Bearer <token>”** (trebuie menționat faptul că token-ul este salvat în localStorage după login)**.** Spre exemplu, cererea de preluare a utilizatorilor utilizează acest token în antet:

const fetchUsers = async () => {

      try {

        const response = await axios.get(`${API\_BASE\_URL}/api/users`, {

          params: {

            page: currentPage,

            limit: 9,

            sortBy: "asc",

          },

          headers: {

            Authorization: `Bearer ${localStorage.getItem("accessToken")}`,

          },

        });

        setUsers(response.data.users);

        setTotalPages(response.data.totalPages);

      } catch (error) {

        console.error("Eroare Users -> useEffect() -> fetchUsers():", error);

        if (error.response && error.response.status === 403) {

          nav("/changepass");

          HandleToast("info", "Parola trebuie schimbată");

        }

      }

    };

Această cerere ajunge în server, la API, unde se verifică token-ul, se verifică dacă utilizatorul trebuie să schimbe parola, iar dacă token-ul este verificat cu succes și nu este necesar să se facă o schimbare de parolă, atunci se caută toți utilizatorii din aplicație și se trimit ca răspuns:

router

  .route("/")

  .get(verifyToken, isRequiredPassChange, getAllUsersWithPagination)

exports.verifyToken = async (req, res, next) => {

  const authHeader = req.headers.authorization;

  if (!authHeader)

    return res.status(400).send({ message: "Lipseste auth header" });

  const token = authHeader.split(" ")[1];

  if (!token) return res.status(400).send({ message: "Lipseste token-ul" });

  if (await redisClient.exists(token))

    return res.status(400).send({ message: "Token utilizat" });

  jwt.verify(token, process.env.ACCESS\_TOKEN\_SECRET, (err, payload) => {

    if (err) return res.status(401).send({ message: "Unauthorized" });

    req.user = payload;

    next();

  });

};

exports.isRequiredPassChange = (req, res, next) => {

  if (req.user.schimbaParola) {

    return res.status(403).json({ message: "Parola trebuie schimbata." });

  }

  next();

};

exports.getAllUsersWithPagination = async (req, res) => {

  try {

    const { page, limit, sortBy = "asc" } = req.query;

    const users = await User.findAndCountAll({

      include: [{ model: UserRole, as: "role" }],

      order: [

        ["nume", sortBy === "desc" ? "DESC" : "ASC"],

        ["prenume", sortBy === "desc" ? "DESC" : "ASC"],

      ],

      offset: (page - 1) \* limit,

      limit: parseInt(limit),

    });

    const totalPages = Math.ceil(users.count / limit);

    res.json({ users: users.rows, totalPages });

  } catch (error) {

    console.error("Eroare usersController - getAllUsersWithPagination:", error);

    res.status(500).json({ error: "Server error" });

  }

};

În aplicația client există un sistem de rute, prin care utilizatorul poate naviga în aplicație. Aceste rute sunt create și utilizate utilizând biblioteca **react-router-dom** și sunt de forma următoare:

<Routes>

            {isValidToken || <Route path="/login" element={<Login />} />}

            <Route path="/resetpass" element={<ResetPassword />} />

            <Route path="/reset/:token" element={<ChangePassUponReset />} />

            <Route element={<PrivateRoute />}>

              <Route path="\*" element={<Proiecte />} />

              <Route path="/" element={<Kanban />} />

              <Route path="/proiecte" element={<Proiecte />} exact />

              <Route path="/changepass" element={<ChangePassword />} />

              <Route path="/tasks" element={<Tasks />} />

              <Route path="/kanban" element={<Kanban />} />

              <Route path="/users" element={<Users />} />

            </Route>

După cum se poate vedea, toate rutele sunt protejate cu excepția rutelor de login, și resetare parolă. Protecția se face făcând o cerere către API, și cerând validarea token-ului. Dacă token-ul este valid, atunci utilizatorul poate vedea pagina dorită, dacă nu, utilizatorul este redirecționat spre pagina de login:

const PrivateRoute = () => {

  const { isValidToken } = useStateContext();

  if (!isValidToken) return <Navigate to="/login" />;

  return <Outlet />;

};

const validateToken = async () => {

      try {

        const res = await axios.post(

          `${API\_BASE\_URL}/api/validate-token`,

          localStorage.getItem("accessToken"),

          {

            headers: {

              Authorization: `Bearer ${localStorage.getItem("accessToken")}`,

            },

          }

        );

        res.status === 200 ? setIsValidToken(true) : setIsValidToken(false);

      } catch (err) {

        console.error(err);

      }

    };

Pentru trimiterea de emailuri se folosește nodemailer – spre exemplu la resetare parolă se trimite un email cu token-ul necesar:

resetareParola = (token) => {

  return {

    subiect: "Resetare parola pentru aplicatia Spider",

    text: `Bună,

    \nAi cerut resetarea parolei pentru contul tău. Te rugăm să dai click pe link-ul următor pentru a-ți reseta parola:

    \nhttps://nice-plant-08c8cad03.3.azurestaticapps.net/reset/${token}

    \nÎn cazul în care nu ai cerut tu resetarea parolei, te rugăm să ignori acest mesaj.

    \nO zi bună în continuare!`,

  };

};

const em = resetareParola(resetToken);

sendEmail(email, em.subiect, em.text);

const sendEmail = async (userMail, subiect, textEmail) => {

  const transporter = nodemailer.createTransport({

    service: process.env.EMAIL\_SERVICE,

    auth: {

      user: process.env.EMAIL\_ADDRESS,

      pass: process.env.EMAIL\_PASS,

    },

  });

  const mailOptions = {

    from: process.env.EMAIL\_ADDRESS,

    to: userMail,

    subject: subiect,

    text: textEmail,

  };

  try {

    await transporter.sendMail(mailOptions);

  } catch (error) {

    console.log(error);

  }

};

# MANUAL DE UTILIZARE

Aplicația Spider simplu de utilizat – singurul lucru de care e nevoie este de un cont valid. Pentru a crea un cont, unul dintre administratorii aplicației trebuie să navigheze către pagina **Utilizatori**, să apese butonul de Adăugare și să completeze formularul care apare (figura 7.1). După ce administratorul apasă butonul **Continuă**, dacă toate câmpurile au fost completate cu date valide, utilizatorul va primi un email pe adresa de email introdusă în formular cu o parolă generată aleator (în caz contrar, administratorul va primi un mesaj de eroare în care se explică de ce câmpuri conțin date invalide, sau ce câmpuri trebuie completate). De reținut faptul că doar persoanele cu rol de Administrator pot adăuga și edita utilizatori. Pentru a actualiza datele unui utilizator (inclusiv parola acestuia), administratorul navighează către pagina Utilizatori, și apasă pe butonul Modifică pentru utilizatorul la care se dorește să se actualizeze date.

O imagine care conține text, captură de ecran, software, Pictogramă computer

Descriere generată automat

Figura 7.1 – Formular adăugare utilizator

Dacă administratorul a selectat opțiunea de schimbare parolă la login, atunci utilizatorul va fi redirecționat către pagina de schimbare parolă la primul login.

Pentru a adăuga un proiect, la fel ca și la adăugare utilizatori, este nevoie de un administrator (doar administratorii au posibilitatea de a adăuga proiecte). Administratorul merge pe pagina Proiecte, apasă butonul Adaugă și completează formularul care apare afișat (figura 7.2). După ce administratorul apasă butonul Continuă, formularul dispare și proiectul apare în lista de proiecte (doar dacă toate câmpurile au fost completate cu date valide – în caz contrar, administratorul este notificat de greșelile care trebuie corectate). Aceeași pași se urmează și pentru actualizarea proiectelor, singurele diferențe fiind că formularul va fi precompletat cu datele existente, iar pentru a deschide formularul de editare se apasă pe butonul Modifică în loc de Adaugă.

O imagine care conține text, software, Pictogramă computer, Software multimedia

Descriere generată automat

Figura 7.2 – Formular adăugare proiect nou

Pentru a adăuga un task nou, utilizatorul trebuie întâi să selecteze un proiect (proiectul fiind ca un container pentru task-uri, acestea nu pot fi adăugate fără a se menționa într-un fel proiectul de care aparțin). Pentru a selecta un proiect, utilizatorul navighează la pagina Proiecte și apasă pe butonul **Selectează** pentru proiectul dorit (butonul se transformă într-un buton de culoare portocalie, cu textul **Selectat** pentru a notifica utilizatorul de proiectul utilizat, și apar butoanele de navigare pentru paginile Backlog și Kanban – figura 7.3).

A screenshot of a computer

Description automatically generated with medium confidence

Figura 7.3 – Selectare proiect

Apoi, după ce proiectul este selectat, utilizatorul merge în pagina Backlog (unde sunt afișate toate task-urile de pe proiectul respectiv) și apasă pe butonul Adaugă ceea ce va deschide formularul de adăugare task (figura 7.4). După ce utilizatorul completează formularul și apasă pe butonul Continuă, un mesaj de notificare este afișat, iar task-ul apare în listă.

A screenshot of a computer

Description automatically generated

Figura 7.4 – Formularul de adăugare task nou

Pentru a vedea toate task-urile de pe proiect grupate pe status și a modifica statusul unor task-uri fără a fi nevoie de a edita un task, utilizatorul poate naviga la pagina Kanban (figura 7.5). Acolo, acesta poate să tragă un task dintr-o coloană în alta (drag & drop), ceea ce va schimba și salva statusul task-ului în funcție de coloana în care este mutat.

A screenshot of a computer

Description automatically generated with medium confidence

Figura 7.5 – Tablou Kanban

În cazul în care utilizatorul dorește să-și schimbe parola, acesta trebuie să dea click pe numele său (aflat în colțul din dreapta-sus a aplicației) și să selecteze opțiunea **Schimbă parola.** Astfel, utilizatorul este redirecționat către pagina de schimbare parolă, unde este afișat formularul necesar (figura 7.6). Acolo trebuie adăugată parola veche, parola nouă și confirmarea parolei (aceeași parolă ca și în câmpul pentru parola nouă). În cazul în care valorile adăugate sunt valide, la apăsarea butonului Schimbă parola utilizatorul va primi un email de confirmare pe adresa sa de email și este delogat automat (din motive de securitate, la delogare token-ul este invalidat ceea ce înseamnă că orice sesiune de pe orice calculator pe care utilizatorul este logat este terminată). Dacă formularul nu este completat cu date valide (parola veche este incorectă, parola nouă nu este conformă cu standardele sau valoarea din câmpul Confirmă parola nu este identică cu valoarea din câmpul Introdu parola nouă), atunci un mesaj de eroare va fi afișat.

A screenshot of a computer

Description automatically generated with medium confidence

Figura 7.6 – Formular de schimbare parolă

Există posibilitatea ca utilizatorul să-și uite parola. În acest caz, există două modalități de rezolvare a acestei probleme: utilizatorul poate să-și reseteze singur parola dând click pe butonul **Ai uitat parola?** din pagina de Login, sau poate contacta un administrator. Administratorul are posibilitatea de a actualiza parola utilizatorului cu o parolă personalizată deschizând pagina Utilizatori și dând click pe butonul Modifică ce corespunde utilizatorului care a făcut cererea.

În cazul în care utilizatorul folosește funcția de resetare parola, la apăsarea butonului **Ai uitat parola?** acesta este redirecționat la pagina de cerere resetare parolă unde trebuie să introducă adresa sa de email. La apăsarea butonului Resetează parola, dacă adresa de email corespunde unui cont (există în baza de date), atunci un email cu un link personalizat este trimis utilizatorului (figura 7.7). La apăsarea link-ului trimis, utilizatorul este redirecționat spre pagina de resetare parolă (link-ul este valabil doar 2 ore, apoi nu va mai putea fi folosit), unde există un formular în care acesta poate adăuga noua parolă (figura 7.8). Dacă formularul este completat cu date valide, utilizatorul este notificat de faptul că parola a fost resetată (în caz contrar apare un mesaj de eroare) și este redirecționat către pagina de login, de unde se poate loga în aplicație cu noua parolă.

O imagine care conține text, electronice, captură de ecran, software

Descriere generată automat

Figura 7.7 - Emailul de resetare parolă

O imagine care conține text, captură de ecran, Font

Descriere generată automat

Figura 7.8 – Formularul pentru schimbare parolă după resetare

# CONCLUZII ȘI DIRECȚII VIITOARE

Sistemele de management bazate pe metoda Kanban sunt necesare în administrarea proiectelor complexe și deși nu este neapărat nevoie de un tablou Kanban pentru proiectele de dimensiuni mici și medii, este recomandat deoarece crește productivitatea echipei și facilitează colaborarea între membri. De aceea, este necesar să inovăm sistemele de acest tip, iar pentru aplicația Spider este necesar să implementăm următoarele funcționalități:

1. Adăugarea posibilității de personalizare a tabloului Kanban prin modificarea statusurilor de către administrator
2. Gruparea task-urilor pe epice
3. Gruparea epicelor pe sprint-uri
4. Posibilitatea de a afișa sprintul curent în tabloul Kanban
5. Posibilitatea de a adăuga și actualiza task-uri din tabloul Kanban
6. Limitarea accesului utilizatorilor doar pe proiectele pe care sunt alocați

După aceste funcționalități, putem extinde implementarea spre alte metode de management și funcționalități extra:

1. Diagrame Gantt
2. Rapoarte
3. Export de date în format Excel
4. Time Tracking

# BIBLIOGRAFIE

\*\*\*<https://www.atlassian.com/company>, iunie 2023

\*\*\*<https://www.atlassian.com/customers>, iunie 2023

1. \*\*\*<https://www.atlassian.com/software/jira/features>, iunie 2023
2. \*\*\*<https://en.wikipedia.org/wiki/Monday.com>, iunie 2023
3. \*\*\*<https://en.wikipedia.org/wiki/YouTrack>, iunie 2023
4. \*\*\*<https://www.jetbrains.com/youtrack/features/integrations.html>, iunie 2023
5. \*\*\*<https://en.wikipedia.org/wiki/Trello>, iunie 2023
6. \*\*\*<https://en.wikipedia.org/wiki/Redmine>, iunie 2023
7. \*\*\*<https://axios-http.com/docs/intro>, iunie 2023
8. \*\*\*<https://react.dev/>, iunie 2023
9. \*\*\*<https://en.wikipedia.org/wiki/JSX_(JavaScript)>, iunie 2023
10. \*\*\*<https://legacy.reactjs.org/docs/faq-internals.html>, iunie 2023
11. \*\*\*<https://www.react-hook-form.com/>, iunie 2023
12. \*\*\*<https://www.react-hook-form.com/faqs/#ReactHookFormFormikorReduxForm>, iunie 2023
13. \*\*\*<https://tailwindcss.com/>, iunie 2023
14. \*\*\*<https://nodejs.org/en/about>, iunie 2023
15. \*\*\*<https://expressjs.com/>, iunie 2023
16. \*\*\*<https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>, iunie 2023
17. M.Jones, J.Bradley, N.Sakimura <https://datatracker.ietf.org/doc/html/rfc7519>, versiunea Mai 2015

1. <https://www.atlassian.com/company>, iunie 2023 [↑](#footnote-ref-1)
2. <https://www.atlassian.com/customers>, iunie 2023 [↑](#footnote-ref-2)
3. <https://www.atlassian.com/software/jira/features>, iunie 2023 [↑](#footnote-ref-3)
4. <https://en.wikipedia.org/wiki/Monday.com>, iunie 2023 [↑](#footnote-ref-4)
5. <https://en.wikipedia.org/wiki/YouTrack>, iunie 2023 [↑](#footnote-ref-5)
6. <https://www.jetbrains.com/youtrack/features/integrations.html>, iunie 2023 [↑](#footnote-ref-6)
7. <https://en.wikipedia.org/wiki/Trello>, iunie 2023 [↑](#footnote-ref-7)
8. <https://en.wikipedia.org/wiki/Redmine>, iunie 2023 [↑](#footnote-ref-8)
9. <https://axios-http.com/docs/intro>, iunie 2023 [↑](#footnote-ref-9)
10. <https://react.dev/>, iunie 2023 [↑](#footnote-ref-10)
11. <https://en.wikipedia.org/wiki/JSX_(JavaScript)>, iunie 2023 [↑](#footnote-ref-11)
12. <https://legacy.reactjs.org/docs/faq-internals.html>, iunie 2023 [↑](#footnote-ref-12)
13. <https://www.react-hook-form.com/>, iunie 2023 [↑](#footnote-ref-13)
14. <https://www.react-hook-form.com/faqs/#ReactHookFormFormikorReduxForm>, iunie 2023 [↑](#footnote-ref-14)
15. <https://tailwindcss.com/>, iunie 2023 [↑](#footnote-ref-15)
16. <https://nodejs.org/en/about>, iunie 2023 [↑](#footnote-ref-16)
17. <https://expressjs.com/>, iunie 2023 [↑](#footnote-ref-17)
18. <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>, iunie 2023 [↑](#footnote-ref-18)
19. M.Jones, J.Bradley, N.Sakimura <https://datatracker.ietf.org/doc/html/rfc7519>, versiunea Mai 2015 [↑](#footnote-ref-19)