Table of Contents

[Introducere](#_Toc10969_WPSOffice_Level1) [1](#_Toc10969_WPSOffice_Level1)

[Contribuții](#_Toc26510_WPSOffice_Level1) [3](#_Toc26510_WPSOffice_Level1)

[Motivație](#_Toc26539_WPSOffice_Level1) [4](#_Toc26539_WPSOffice_Level1)

[1. Arhitectură](#_Toc30422_WPSOffice_Level1) [5](#_Toc30422_WPSOffice_Level1)

[1.1 Baza de date](#_Toc26510_WPSOffice_Level2) [6](#_Toc26510_WPSOffice_Level2)

[Bibliografie](#_Toc7610_WPSOffice_Level1) [10](#_Toc7610_WPSOffice_Level1)

# Introducere

Kanban este un sistem de planificare a sarcinilor derivat din noțiunea de *lean software development**[[1]](#footnote-0)*. Această metodologie implementează un set de șapte principii: eliminarea timpului mort, punerea accentului pe specializarea individului(învățarea continuă), luarea de decizii cât mai târziu posibil, livrarea produsului cât mai rapid posibil, transferarea responsabilităților către echipa de dezvoltare, crearea unei imagini de ansamblu concrete și implementarea integrității și a intuiției.

Sistemul a fost propus pentru prima oară în cadrul firme producătoare de mașini Toyota, de către inginerul industrial Taiichi Ohno, luând inspirație din modul în care magazinele alimentare își reîmprospătau rafturile bazat pe cererea creată de clienți. Acest model a devenit popular abia în timpul recesiunii globale din 1970, în mod inițial pentru a reduce costurile și utilizarea de mașinării costisitoare. În prezent, compania Toyota folosește sistemul kanban pentru a detecta impedimentele din procesele de fabricație, dar și pentru a descoperi oportunități noi de optimizare.

David J. Anderson[5] a stabilit faptul că un sistem kanban poate fi împărțit în cinci componente: *semnale vizuale*, *coloane*, *limită de lucru*, *un punct de comitere* și *un punct de livrare*.

Semnalul vizual este reprezentat de către *tichet*. Echipele ce folosesc sistemul scriu toate obiectivele pe care trebuie sa le atingă pe astfel de tichete, cât mai concis și mai concret, de obicei, un singur obiectiv reprezentând un singur tichet.

Cea de a doua componentă sunt coloanele care au rolul de a găzdui tichetele și reprezintă un tip de activitate din cadrul procesului de lucru. După cum vom vedea în cadrul acestei lucrări de licența, ele pot fi denumite precum “În Progres” sau “Completate”, și în mod natural, vor conține tichetele la care se lucrează în mod curent, sau care sunt completate.

Limita de lucru reprezintă numărul maxim de tichete care pot exista în același timp într-o coloană. Când se atinge această limită, echipa trebuie să-și schimbe focusul asupra lor și să îndeplinească obiectivul descris de către acestea, pentru a fi mutate în următorul stadiu al fluxului de lucru. Aceasta limită este esențială pentru a detecta impedimentele din cadrul fluxului și a descoperi dacă munca depășește efortul pe care echipa îl poate depune. În mod ideal, limita propusă de metodologia kanban este de maxim trei tichete, dar mecanismul implementat în această lucrare de licență lasă la îndemâna utilizatorilor de a modifica această limita ori de cate ori este necesar pentru a acomoda nevoile unui proiect.

Punctul de comitere este locul unde clientul și echipa de dezvoltare își adaugă ideile și conceptele legate de proiect, de unde vor fi preluate mai departe de echipa și vor fi implementate.

Punctul de livrare este de obicei sfârșitul procesului de dezvoltare și etapa când produsul se află în mâna clientului.

Scopul echipei este să treacă tichetele din punctul de comitere, prin toate stadiile procesului de implementare și în final în punctul de livrare.

Peste toate elementele se află *tabela kanban*, este ustensila utilizată pentru a implementa kanban și care poate fi văzută ca și un recipient. Rolul ei este de a oferi o imagine de ansamblu celui care o accesează, pentru a înțelege cât mai ușor stadiul în care se află proiectul.

Tabele kanban pot fi implementate atât fizic cât și electronic, fiecare având avantajele și dezavantajele lor. Tabele fizice sunt ușor de construit, și există în mod perpetuu. Una sau mai multe echipe pot să contribuie la unul sau mai multe proiecte pe aceeași tabelă, și astfel membrii unei echipe pot observa modul de lucru al unei alte echipe, și astfel să poată utiliza aceleași principii pentru a optimiza procesele din cadrul proiectului lor. Dezavantajul major este cel al erorii umane și dacă aceasta nu este observată din timp, fluxul de lucru poate fi compromis. De exemplu, dacă o funcționalitate legată de securitatea unei baze de date este descrisă într-un tichet, și acesta este mutat în coloana finală a tabelei, toți participanții unui proiect vor considera acea funcționalitate completă. În lipsa unui mecanism care să ateste că acel tichet a fost mutat în mod corect, cei care vor analiza tabela vor fi induși în eroare.

Pe de cealaltă parte, o tabelă implementată în mod electronic poate avea un astfel de mecanism de monitorizare și pot fi accesate la distanță oricând și de oricine este înscris la ea. Din păcate, de-a lungul timpului se poate dezvolta o practică de izolaționism, participanții punând din ce în ce mai mult accentul pe propriile tichete, excluzând imaginea de ansamblu. Cel mai mare avantaj este că o astfel de tabelă poate fi modificată și customizată după preferințele celor ce lucrează pe proiect, se pot adăuga filtre, se pot controla atât limita de lucru, cât și numărul de coloane, se poate oferi și elimina accesul unui anumit cont înregistrat în sistem.

Toate aceste mecanisme prezentate mai sus sunt implementate în aceasta lucrare de licență, și se va putea observa cum aceste concepte pot fi utilizate nu numai de un număr nelimitat de persoane, dar se pot scala pentru a suporta un număr mare de proiecte, coloane și tichete.

# Contribuții

Primul pas în elaborarea acestei lucrări de licență a fost reprezentat de procesul de documentare și informare asupra metodologiei kanban și analizarea altor platforme care utilizează același sistem. În momentul elaborării aceste lucrări, exista șapte platforme care oferă o implementare a metodologiei kanban, dar este utilizată în concordanță cu metodologia *scrum*[[2]](#footnote-1), cea mai utilizată fiind JIRA, aplicație proprietară Atlassian.

Următoarea contribuție a fost conceperea unei arhitecturi care sa fie practică, ușor de utilizat și intuitivă. Rezultatul este o structură compusă din trei elemente, puse în pereche printr-o

relație de cardinalitate *unul-la-**mai-mulți[[3]](#footnote-2).*

O etapă importantă în dezvoltarea platformei descrise a fost cercetarea tehnologiilor necesare. Conceptul a fost gândit pentru a oferi utilizatorilor acces de pe stațiile de lucru personale(dispozitivele desktop), cat și de pe dispozitivele mobile pe care aceștia le au asupra lor. O mare parte din funcționalitațile pe care le oferă aplicația desktop sunt accesibile și din aplicația mobilă, cu excepția unora care nu sunt suportate de navigatorul dispozitivului.

Clientul și serverul sunt în totalitate dezvoltate în limbajul de programare JavaScript. Pentru partea de server, este utilizată biblioteca *NodeJS* în concordanta cu cea *ExpressJS*, pentru o implementare completa a stilului arhitectural REST.

Pe partea de front-end, *ReactJS* este folosit pentru capacitatea acestuia de a crea și defini componente, care au o mare capacitate de a fi reutilizate și scalate. Despre aceste tehnologii se va intra în detaliu în capitolele ce urmează.

Din punct de vedere al securității, se utilizează sistemul de *JSON web token*, mecanism ce oferă posibilitatea unui utilizator de a folosi mai multe aplicații prin metoda *single sign on*. În aceasta lucrare, JSON web token-ul se utilizează mai ales pentru a securiza toate rutele prin care serverul și clientul schimbă informații. Pentru criptarea parolelor utilizatorilor se utilizează biblioteca *bcrypt*, care dispune de o funcție de criptare unidirecțională.

Aplicația dispune de un administrator care poate crea și modifica orice resursa din sistem exceptând profilele celorlalți utilizatori. Restul utilizatorilor care nu au acest drept sunt constrânși a modifica doar resursele pentru care sunt participanți.

# Motivație

Ideea acestei aplicații a originat din experiența mea cu o platformă asemănătoare care implementează același principiu, pe care o utilizez la locul de muncă. Primul aspect pe care l-am observat a fost cât de eficient și de ușor era pentru fiecare angajat să urmărească propria muncă și pe cea a colegilor săi. Dar de asemenea, această platformă aduce cu sine și dezavantaje.

Primul aspect observat a fost complexitatea mare pe care această platformă o conferă. Cele menționate mai sus încearcă să suporte o categorie mare de companii, fiecare folosind propriul sistem de management și de aici rezultă complexitatea ridicată a aplicației. Angajații care se confruntă pentru prima oară cu o astfel de platformă nu pot apela doar la intuiție, principiu discutat în introducere, ci au nevoie de o pregătire pentru a înțelege conceptele implementate și mecanismele de care pot dispune.

Pe de alta parte, chiar dacă o aplicație de acest gen propune o structură simplă și ușor de utilizat, există posibilitatea mare ca o mare parte din funcționalitățile pe care le oferă să nu fie pe deplin gratuite. Acest lucru reprezintă o provocare pentru o singură persoană care doar dorește un mecanism prin care să-și planifice sarcinile și să-și înregistreze munca, sau chiar pentru o firma mică de antreprenoriat care are doar câțiva angajați.

Aplicația mea propune o modalitate prin care se oferă mult control pentru proiecte mici și persoane care nu sunt experimentate cu metodologia kanban și un start în lumea managementului.

# Arhitectură

Din punct de vedere arhitectural, exista mai multe aspecte care trebuie luate în considerare. Aplicația este alcătuită din trei elemente, un server, o baza de date și un client. La rândul lor, acestea au o arhitectură internă complexă.

## Baza de date

În următoarele pagini voi descrie structura bazei de date, relațiile dintre tabele acesteia și modul în care interacționează pentru a oferi utilizatorului posibilitatea de a utiliza platforma în mod cât mai sigur și mai intuitiv.

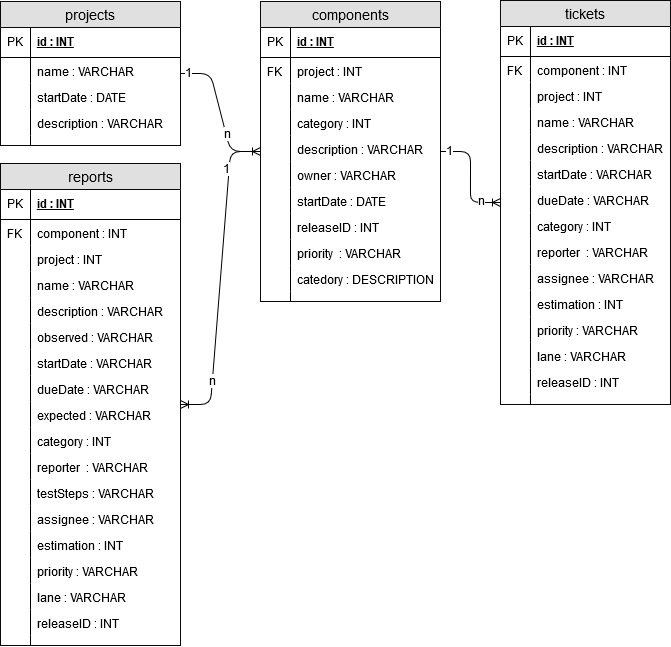


Fig. 1 - Relațiile *unul-la-mai-mulți3* dintre proiect, componentă, raport problemă și tichet

După cum se poate observa în Fig.1, un proiect joacă rolul de recipient pentru una sau mai multe componente.

La rândul ei, componenta este responsabilă să găzduiască unul sau mai multe tichete.

Relația aceasta reiese din utilizarea mecanismului de chei străine, care leagă tabele între ele printr-o constrângere pe operația de ștergere pentru coloana *id* din tabela *proiecte*. Comportamentul propus este ca atunci când o intrare din acestă coloană este eliminată, această operație să declanșeze ștergerea intrărilor din tabelele *componentelor* și cea a *tichetelor*. În acest mod se asigură că niciun element nu va rămâne în baza de date fără părinte, și nu va compromite modul în care aceste elemente vor fi afișate pe interfața grafică utilizatorului.

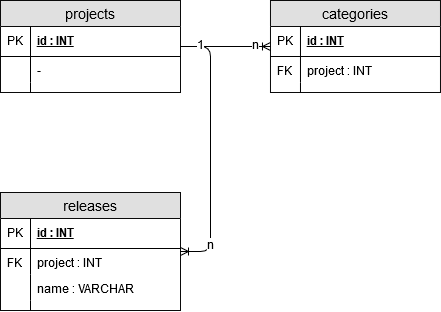


Fig. 2 - Relația dintre proiect, categorii și versiuni

Aceeași relație o putem descoperi între proiect, categorii(*categories*) și versiuni(*releases*). Categoria reprezintă denumirea tipului sau naturii unui tichet din proiectul respectiv, și are rol informativ pentru utilizator.

Versiunea reprezintă un punct izolat în timp a stagiului în care produsul se află, dar trebuie utilizat în concordanță cu un document de suport în care acestea sunt specificate, datele limită și alte aspecte.

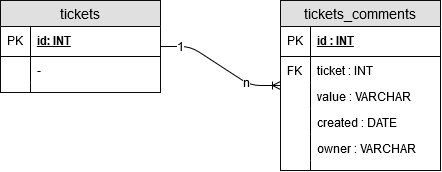


Fig. 3 - Relația dintre tichete și comentarii

Un tichet poate avea unul sau mai multe comentarii. Sunt folosite pentru a oferi utilizatorilor posibilitatea de a-și exprima opiniile în legătura cu orice aspect legat de sarcina descrisă în cadrul tichetului. Comentariul este identificat în baza de date folosind un identificator, care este unic. Valoarea(*value*) este textul introdus de către utilizator. Ziua de creare este generată în mod automat atunci când comentariul este introdus în baza de date, iar utilizatorul nu are acces la aceasta valoare. Proprietarul(*owner*) este completat în mod automat prin extragerea utilizatorului care este logat în mod curent pe partea de front-end, și de asemenea, acest atribut nu poate fi influențat de cel ce introduce acest comentariu.

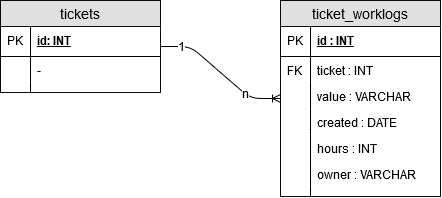


Fig. 4 - Relația dintre tichet și raport de muncă

Tabela rapoartelor de muncă(*ticket\_worklogs*) au structura asemănătoare cu cea a tabelei comentariilor. Singura diferență este reprezentată de coloana orelor înregistrate(*hours*), care reprezintă orele pe care o persoană le-a depus pentru a contribui la completarea sarcinii descrise de către tichet. Aceste ore sunt reprezentate grafic pe tichetele care se găsesc pe tabela kanban, și sunt puse în comparație cu orele estimate.

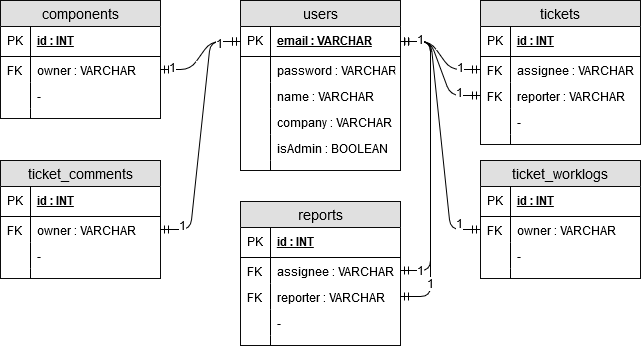


Fig. 5 - E-mail-ul ca și cheie străină

În figura anterioară, se prezintă toate tabele care sunt influențate de tabela utilizatorilor(*users*). În baza de date, un utilizator este descris de un *e-mail*, care este cheie primară și unică, o parolă(*password*), un șir de caractere ce semnifică parola utilizatorului criptată de o funcție unidirecțională, un nume(*name*), care în mod ideal este numele real al utilizatorului și numele companiei(*company*) la care acesta lucrează. Ultimul câmp(*isAdmin*) semnalează dacă acest utilizator are sau nu drepturi de administrator, drepturi care îi oferă posibilitatea să modifice majoritatea câmpurilor din interfața grafică.

Atunci când se creează o componentă sau un tichet, în mod opțional putem să specificăm, în cazul componentei, care este proprietarul acesteia, iar în cazul tichetului, care sunt participanții, unul din ei fiind supervizorul(*reporter*), iar celalalt fiind asignatul(*assignee*). În ambele situații, dacă un utilizator a fost specificat într-un câmp din aceste două elemente, dorim ca atunci când este eliminat din sistem, câmpul să rămână eliberat, ca altă persoană să îi preia locul la un moment dat. Din punctul de vedere al bazei de date, câmpul din aceste tabele va fi setat pe valoarea *NULL*.

În cazul tabelelor comentariilor și rapoartelor de muncă, intrările care au ca și cheie străină utilizatorul șters, vor fi și ele la rândul lor șterse.

# Bibliografie

1. Kanban Made simple: Demistifying and Applying Toyota’s Legendary Manufacturing Process, John M. Gross, Kenneth R.McInnis
2. Kanban: Successful Evolutionary Change for Your Technology Business, David J. Anderson
3. https://www.inflectra.com/methodologies/kanban.aspx
4. https://www.atlassian.com/agile/kanban
5. https://getbootstrap.com/docs/4.2/getting-started/introduction
6. https://reactjs.org/docs/getting-started.html
7. https://webpack.js.org/concepts
8. https://nodejs.org/en/docs
9. https://www.codecademy.com/articles/what-is-rest
10. https://jwt.io/introduction
11. http://www.capisci.ro/articole/Func%C5%A3ii\_hash

1. Set de principii și practici din domeniul dezvoltării de programe creat în anul 2003de către Marry și Tom Poppendieck [↑](#footnote-ref-0)
2. Metodologie asemănătoare kanban, bazată pe împărțirea muncii pe perioade de timp numite sprint-uri, care durează în medie doua săptămâni [↑](#footnote-ref-1)
3. Relație între două entități în care un element de tip A poate fi legat de mai multe elemente de tip B, dar elementul de tip B este legat la un singur element de tip A [↑](#footnote-ref-2)