UNIVERSITATEA BABEŞ-BOLYAI CLUJ-NAPOCA

FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ

Programul Postuniversitar de Pregătire şi Formare Profesională în Informatică

LUCRARE DE ABSOLVIRE

Managementul licențelor

Conducător ştiinţific
Asistent Universitar Dr. Șotropa Diana – Florina
Marchiș Cristian

Absolvent
Kovács Noémi

2020

Abstract of thesis

The following work presents a solution to the issue of managing software licenses. The main driving force for writing this thesis and software for me was the fact that in large, multinational companies, the negligent management (or lack thereof) of licenses is an arduous and considerable problem. Utilizing the knowledge gained through this post university course, I thought that I could develop a solution of my own.

The main benefits this solution brings to end-users in a corporate environment is lowering acquisition costs, increase productivity by keeping licenses of up-to-date software, limiting software license renewal issues and increased transparency in management of software assets by tracking the use and assignment of licenses.

The application was built from the ground up. It is a web application, with a back-end written in PHP and JavaScript, front-end predominantly written in HTML and CSS, with few JavaScript design elements scattered throughout. The backbone of the application lies within a MySQL database, which was designed using phpMyAdmin. The application consists of two main parts: user-side and administrator-side. The user-side mostly deals with the displaying of already inputted information, and allows for users to register requests for additional licenses (Tools). These requests are then processed by the system administrators. The administrators also have the ability to edit and delete existing information in the system, such as changing passwords, renewing licenses, adding new vendors to the Vendors list etc. The main aspects presented in this document are:

- Building a user-friendly, pleasant Graphical User Interface (GUI)
- Managing error messages client-side and server-side
- Developing the sign-up and sign-in pages
- Validating user-inputted information, on both front and back-end
- Implementing a user authorization system and management for system administrators
- Paginating, filtering, sorting displayed data in tables
- Displaying alerts
- Setting up hosting, web server and email capabilities.

Cuprins

Partea I – Introducere

Ideea de bază a proiectului de gestionare a licențelor (management-ul licențelor) este de a realiza un sistem în care utilizatorii își pot administra cât mai simplu și succint licențele software, datele legate de mentenanță software, datele de contact al vendor-ului (furnizorului) etc. Am ales acest subiect deoarece lucrez de mai mulți ani în domeniul IT și am vrut ca prin proiectul de absolvire a Programului postuniversitar de pregătire și formare profesională în informatică să leg o parte din munca mea din ultimii ani cu această aplicație ca și un instrument prin care să îmi pot ușura munca de zi cu zi.

O problemă importantă a marii majorități a organizaților din domeniul IT și nu numai este administrarea parcului de computere din infrastructura IT. În speță managementul licențelor software, cu cât compania are mai mulți angajați, cu atât este mai anevoioasă administrarea și actualizarea licențelor. Deși o porțiune substanțială a dezvoltatorilor de software oferă management pentru licențierea de volum (de obicei sub forma unui site), totuși în multe cazuri, clienții nu sunt conștienți de acest fapt, așadar managementul licențelor inadecvat rămâne o problemă curentă. De obicei problema cu soluția oferită este că nu prea putem să monitorizăm mai multe site-uri deodată de la vendori diferiți, și deseori se întâmplă că nici site-urile lor nu sunt actualizate sau nu ne oferă informații care se pot agrega sau detaliile necesare despre licențe, cum ar fi mai utile pentru noi.

Soluțiile de inventariere software devin așadar unelte indispensabile în viața oricărei organizații respectabile, în funcție de complexitatea infrastructurii IT, portofoliului de aplicații, disponibilitatea resurselor și obiectivele de business. Pentru multe firme, obiectivul principal al implementării unei soluții de management al activelor software este inventarierea licențelor achiziționate.

Gestionarea activelor software, sau management-ul licențelor, se referă la centralizare, monitorizare și transparență în alocarea licențelor, prelucrarea informațiilor contractuale și financiare legate de licențe soft, și urmărirea modului în care sunt utilizate programele din cadrul

organizației. Toate acestea oferă o trecere în revistă și o înțelegere despre ce softuri sunt disponibile, ce se folosește și cine le folosește.

Pilonii de baza în materia gestionării licențelor software sunt:

- centralizare
- monitorizare
- transparență

Programele de management al licențelor joacă un rol important în susținerea pilonilor de bază menționați, punând accentul pe optimizare, conformitate și controlul costurilor. Este un proces continuu care permite unei organizații să planifice și să gestioneze mai bine licențele software. Scopurile strategice ale programelor de management al licențelor sunt:

- Reducerea costurilor de achiziție software și a suportului prin negocierea de contracte de volum și realocarea licențelor neutilizate.
- Sporește predictibilitatea bugetelor.
- Creșterea productivității prin menținerea la zi a celor mai noi versiuni de software.
- Limitarea problemelor de actualizare software prin automatizarea proceselor IT.
- Asigurarea congruenței cu politicile de securitate ale producătorilor de software.
- Întocmirea unor proceduri bine stabilite pentru achiziția, instalarea, exploatarea, și retragerea din uz a diferitelor versiuni de software ce permite obținerea de beneficii în timp.

Aplicația mea este o soluție de tip management al licențelor, un subset de management care poate ajuta o organizație să-și atingă obiectivele menționate anterior. Oferă o interfață intuitivă și plăcută a gestionării.

Aplicația folosește în principal pentru back-end PHP cu JavaScript iar pentru front-end predominant HTML, CSS, cu mici elemente JavaScript. Baza de date a fost făcută în MySQL (MariaDB), gestionata prin utilitarul de linie de comanda si utilitarul web phpMyAdmin.

Aplicația este formată din doua părți, partea de utilizator și partea administrativa. Partea de utilizator ține mai mult de afișarea datelor deja înscrise în sistem, și permite utilizatorilor să facă cereri de licențe noi (denumite în aplicație tool-uri). Aceste cereri ajung la administratori care

apoi le procesează. Administratorii dețin privilegii mai înalte, și au capacitatea de a modifica și șterge informații din sistem, cum ar fi schimbarea unei parole sau ștergerea unei versiuni vechi de software, precum și adăugarea de informații noi, cum ar fi adăugarea unui furnizor (vendor) nou în listă, ș.a.m.d.

Principalele aspecte prezentate în această lucrare sunt următoarele:

- Setarea mediului de dezvoltare a aplicației.
- Construirea unei interfețe utilizator atractive folosind Boostrap (pagina index).
- Gestionarea mesajelor de eroare atât în partea de server cât și pe partea de client a aplicației.
- Construirea paginii de logare și înscriere.
- Validarea informațiilor/datelor introduse de utilizator pe front-end și pe back-end.
- Implementarea unei platforme (un sistem) de autorizare și management pentru administratori.
- Paginarea, sortarea și filtrarea datelor afișate.
- Afișarea alertelor.
- Setarea mediului de producție publicat (hostat) pe internet.

Toată aplicația, atât partea de server cât și partea de client a fost scrisă folosind Visual Studio Code, un editor gratis care poate fi folosit pe mai multe sisteme de operare. Pentru administrarea datelor în baze date am folosit phpMyAdmin în cadrul XAMPP, versiune 7.4.9, (utilizând în spate o bază de date MariaDB, versiune 10.4.13) care este un pachet software gratis, scris in PHP și se poate folosi pentru sarcini de administrare, inclusiv crearea unei baze de date, rularea interogărilor etc. Din motive de testare a interogărilor SQL, am folosit de asemenea și utilitarul de linie de comandă MySQL CLI. Pentru o mai bună vizibilitate a legăturilor între tabele, baza de date a fost proiectată în MySQL Workbench.

Pentru scopuri de dezvoltare am folosit un emulator de mail server, numit Papercut-SMTP (1), primind local mesajele de cerere din aplicație, fără să fie necesar accesul la internet.

La sfârșitul procesului de dezvoltare, am decis să găzduiesc aplicația mea pe internet, ca și prilej de a-mi dezvolta în plus abilitățile în sisteme Linux și tehnologii web.

Partea a II - a: License Management

Aplicațiile software ocupă un rol din ce în ce mai important în conducerea proceselor, din toate domeniile de activitate, începând de la zona de producție industrială la cea de servicii și divertisment. Utilizatorii caută constant soluții care să le aducă o valoare cât mai mare în procesele lor de zi cu zi, procese aflate sub o continuă presiune de a fi cât mai eficiente, mai competitive, mai măsurabile și mai adaptabile unui ritm de schimbare accelerat. Presiunea bugetelor de achiziție și scurtarea timpului de implementare a noilor aplicații sau funcționalități sunt factori ce influențează negativ ecuația alegerii optime a aplicației software pentru un proces și un utilizator date, iar problema creșterii numărului de atacuri cibernetice impune la rândul ei strategii dinamice de securizare a datelor care să protejeze utilizatorii într-un mediu aflat în permanentă schimbare.

2.1 Licență software

Legislația română, respectiv Art. 2 din Ordonanța de Guvern nr. 124/2000, cuprinde și o definiție a licenței: "Licența programului pentru calculator - acordul scris al titularului dreptului de autor asupra unui program pentru calculator pentru cesiunea unor drepturi către utilizatorul programului și care însoțește programul." Licența este, așadar, un contract - acord de voință - între titularul dreptului de autor și utilizator. Acesta însoțește programul pentru calculator și se prezintă în diverse forme. În practică, el poate fi întâlnit atât sub forma scrisă bine-cunoscută (pe hârtie), cât și sub forma unui document digital ce apare deseori la instalarea sau deschiderea programului. Denumirile folosite de cele mai multe ori pentru un astfel de document sunt întâlnite în engleză, *License Agreement*, iar în limba română *Acord de Licență* sau *Licență*, pentru software tradițional (program software), și în cazul paginilor web sunt regăsite în josul paginii cuvintele *Terms of Service*, în română *Termenii serviciului*, sau *Termeni și Condiții.*

2.2 Tipuri de licențe

Tipurile de licențe sunt de numeroase feluri, pe care programatorii și distribuitorii le pun la dispoziția utilizatorilor, unele dintre ele fiind mai răspândite în timp ce altele capătă un aspect particularizat, in funcție de termenii de licențiere. Sunt mai multe tipuri de licențe dar putem să împărțim în două categorii mari:

- Software proprietar (în engleză proprietary software)
- Software gratuit și cu sursa liberă/deschisă (engleză: free and open-source software - FOSS).

Diferența conceptuală dintre cele două constă în acordarea de drepturi de modificare și reutilizare a unui produs software obținut de un client. FOSS licențiază ambele drepturi asupra clientului și prin urmare îmbină codul sursă modificabil cu software (open-source), în timp ce software-ul proprietar nu autorizează de obicei drepturi și prin urmare ține acoperit codul sursă, adică sursa-închisă („closed source" în engleză).

Există numeroase tipuri de modele de licențiere, începând de la licențe simple perpetue, la licențe flotante, până la tipurile mai avansate cum ar fi licențe contorizate. Licențele variază în funcție de platformă, și sunt oarecum standardizate, cele mai uzuale tipuri utilizate în ziua de azi fiind:

- Licență Generală Publică (GPL – engl. General Public License) – acoperă software-ul gratuit și poate fi utilizat, copiat, distribuit și modificat fără cost (freeware).
- Per device sau CPU (Procesor) – menit pentru un singur server sau stație de lucru, întâlnit atât în domeniul datacenter-ului cât și pe stația desktop a unui utilizator obișnuit.
- Per rețea – acoperă toate dispozitivele (server și useri) dintr-o singură rețea specifică.
- Per subscripție – gestionată de un abonament pentru un user sau un dispozitiv, de obicei cu dată de expirare (pe luna, sau pe an).
- Per Bază de Date (DB) – de obicei asignat unui singur server, tipurile de licențiere a bazelor de date necesită o atentă considerare din cauza costurilor ridicate și construcțiilor complexe de licențiere, care diferă de la vendor la vendor, și se pot baza pe numărul de servere, pe câte nuclee într-un procesor, de câte socluri de procesoare dispune o placă de bază a unui server (CPU socket), per baze în producție utilizate în mod curent

(comparativ cu toate bazele de date instalate), per feature-uri utilizate sau instalate (un bun exemplu aici este baza de date Oracle, a cărei licență permite instalarea bazei cu toate caracteristicile, dar din momentul utilizării anumitor feature-uri intervin costuri suplimentare).

- Per un singur utilizator (single user, client sau nod) sau per utilizator, în volumul corespunzător nivelului de discount. Userii sunt obligați să se autentifice în aplicație pentru a-și confirma identitatea. Aceste licențe de volum deschis sunt în mod obișnuit numite Program de Licență Deschisă (OLP – engl. Open License Program), Program de Licență Tranzacțională (TLP – engl. Transactional License Program), Program de Licență de Volum (VLP – engl. Volume License Program) ș.a.m.d., Aceste programe de licență sunt într-o relație antagonică cu Programul de Licență Contractual (CLP – engl. Contractual License Program), unde clientul se angajează în achiziționarea unui număr anumit de licențe, pe o perioadă fixă (de obicei doi ani), iar acesta nu se reînnoiește, o licență nouă trebuie achiziționată după expirare.
- Per concurent/user flotant, se poate întâlni deseori în situațiile în care utilizatorii dintr-o rețea au acces la program, dar numai un număr limitat dintre ei se pot conecta în mod concomitent.

Licențierea software adeseori include și mentenanță. Aceasta, de obicei în termen de un an, este fie inclusă sau opțională, dar de regulă trebuie cumpărată împreună cu software-ul. Contractul de mentenanță, în mod normal conține o clauză care permite ca licența să primească update-uri minore (V1.1 => V1.2), dar uneori și revizii și update-uri majore (V1.2 => V2.0). Această opțiune este numită *asigurare de update*, sau *asigurare de upgrade*. Dacă în contract de mentenanță nu a fost inclus un update major, clientul trebuie să achiziționeze separat un upgrade. Pentru reînnoirea mentenanței, unii producători încasează o taxă de reinstalare retroactivă, pe lună, în cazul în care mentenanța curentă a expirat.

În mentenanță uneori este inclus și suportul tehnic. Atunci când este inclus, numele nivelelor ale pachetelor de suport tehnic de obicei sunt numite ca: aur (suport Premium), argint (suport Intermediar) sau bronz (suport Basic, limitat). Acestea sunt determinate prin mai multe caracteristici: modul de comunicație (de exemplu prin mail sau prin telefon), disponibilitate (de exemplu: 24/7 – 24 ore din 24, 5x8 - 5 zile pe săptămână, 8 ore pe zi) și timpul de reacție (de

exemplu în cel mult trei ore). Suportul în sine poate fi licențiat pe număr de incidente, ca și pachet de incidente (de exemplu 5 incidente de suport /an).

2.3 Gestionarea licențelor

Gestionarea licențelor software se referă la instrumentele sau procesele software utilizate de o organizație pentru a controla și documenta locul și modul în care produsele software ale companiei sunt capabile să ruleze pentru a impune și a asigura respectarea licențelor software (cunoscut și sub denumirea de Acord de licență pentru utilizator final, sau EULA – din eng. End User's License Agreement).

Instrumentele de gestionarea licențelor sunt folosite și de către furnizori pentru a monitoriza și a asigura respectarea licenței software pentru dezvoltatori. Aceste tool-uri ajută la prevenirea copierii, partajării sau utilizării ilegale (piratate) a software-ului de către alți utilizatori care nu dețin licență. Gestionarea licențelor de către furnizorul de software poate să implice crearea de chei de licență software de mai multe forme: forma de „încercare" (trial), personală, pentru mediul corporate și/sau soft pe bază de abonament. Aceste chei sunt create prin utilizarea unui generator de licențe.

Un avantaj mare pentru companii este de a deține toate licențele software sub control deoarece în acest fel se pot economisi bani. O licență nefolosită pe un desktop costă aproximativ 260 de dolari, irosind timp și bani din bugetul IT. Urmărind și monitorizând toate achizițiile se pot asigura că fiecare licență este contabilizată și utilizată.

Așadar, cel mai mare beneficiu al utilizării programelor de gestionare de licență este faptul că joaca un rol esențial în asigurarea ca toate produsele software să fie în congruență cu licențele software (cunoscut și sub numele de Contract de Licență pentru Utilizatorii Finali – eng. EULA End User's License Agreement).

Partea a - III a: Tehnologii

Aplicația mea este o aplicație web PHP. Pentru dezvoltarea unei astfel de aplicație necesită instalarea pachetului XAMPP (2). Componentele folosite din XAMPP (versiune 7.4.9) sunt: Apache, MariaDB și în cazul asta a fost și inclus PHP-ul. Dacă folosim un alt stack software, LAMP de exemplu, trebuie instalate separat cele menționate de mai sus. Serverul web Apache manipulează diferite tipuri de fișiere: HTML, imagini, fișiere Flash, mp3 etc.

HTTP sau HyperText Transfer Protocol este un protocol de comunicație de tip cerere-răspuns între un client web și un server. Clientul este reprezentat de către navigatorul web folosit de un utilizator pentru a accesa și afișa conținutul web dintr-o rețea sau de pe Internet. Clientul trimite o cerere către un server care găzduiește conținut web, care va răspunde clientului prin trimiterea conținutului spre afișare către client. Răspunsul server-ului reprezintă confirmarea că acesta a primit cu succes solicitarea clientului și îi poate răspunde conform cererii acestuia (fig.1). Așadar HTTP poate fi considerat o regulă pentru ca un utilizator să poată accesa site-uri web. De asemenea, nu există legătură între două comenzi HTTP consecutive diferite. În protocol există implementată și o listă de coduri pentru stare, folosite de un server pentru a notifica browser-ul în cazul în care sunt probleme. Cel mai des întimpinată problemă de exemplu *404 Not Found* este folosit pentru a informa un utilizator că pagina accesată nu există la adresa căutată.

fig.1 Funcționarea protocolului HTTP

O cerere HTTP conține una sau mai multe linii de text ASCII (American Standard Code for Information Interchange), precedate în mod necesar de denumirea metodei specificând operația ce se va realiza asupra conținutului respectiv:

METODA	DESCRIERE
GET	Cerere de citire a unei pagini Web
POST	Trimite date de intrare către server
ÎN CONTEXTUL SERVICIILOR WEB	
HEAD	Cerere de citire a antetului unei pagini Web
PUT	Cerere de depunere a unei pagini Web
DELETE	Ștergerea unei pagini de Web
TRACE	Transmite în ecou cererea care a sosit
CONNECT	Rezervat pentru o utilizare în viitor
OPTIONS	Interogarea anumitor opțiuni

Deși atât metoda GET cât și metoda POST pot fi utilizate pentru descărcarea conținutului unei pagini Internet, transmițând către serverul web valorile unor anumite atribute, între acestea există anumite diferențe:

- o cerere GET poate fi reținută în cache, fapt ce nu este valabil și pentru o cerere POST;
- o cerere GET rămâne în istoricul aplicației de navigare, fapt ce nu este valabil și pentru o cerere POST;
- o cerere GET poate fi reținută printre paginile Internet favorite din cadrul programului de navigare, fapt ce nu este valabil și pentru o cerere POST;
- o cerere GET impune unele restricții cu privire la lungimea (maxim 2048 caractere) și la tipul de date (doar caractere ASCII) transmise (prin URL), fapt ce nu este valabil și pentru o cerere POST;
- o cerere GET nu trebuie folosită atunci când sunt implicate informații critice (acestea fiind vizibile în URL), fapt ce nu este valabil și pentru o cerere POST;
- o cerere GET ar trebui să fie folosită doar pentru obținerea unei resurse, fapt ce nu este valabil și pentru o cerere POST (3).

Metode de cerere standard pentru HTTP POST se utilizează atunci când datele transmise serverului au dimensiuni mari sau sunt delicate. Creează o resursă, trimițând uzual entități (date, acțiuni) spre server. În sensul procesării datelor, datele vor fi preluate de la *intrarea standard*, lungimea în octeți a acestora fiind specificată de variabila *CONTENT_LENGTH* (4).

3.1. Limbajul PHP

PHP (acronim recursiv pentru PHP: Hypertext Preprocessor) este un limbaj de scripting de uz general, cu cod-sursă deschis (open source), utilizat pe scară largă, și care este potrivit în special pentru dezvoltarea aplicațiilor web și poate fi integrat în HTML . Există două tipuri de scripturi:

- Script Client Side: scriptul se rulează în browser (ex.: JavaScript).
- Script Server Side: scriptul se rulează pe server (ex.: PHP, ASP) (5).

Avantaje script Server Side:

- Acces la baze de date și filesystem (sistem de fișiere).
- Control asupra compatibilității platformelor.
- Acces la funcții și librării.
- Acces la rețea.
- Conținut dinamic.
- Securitate.

Ceea ce face PHP să difere de un JavaScript de partea clientului este că scriptul PHP este executat pe server, generând HTML care este apoi trimis către client. Clientul va primi rezultatele rulării acelui script, fără a putea cunoaște codul-sursă ce stă la bază (vezi fig.2). Se poate configura web serverul să proceseze toate fișierele HTML cu PHP, și atunci într-adevăr nu va fi nici o modalitate ca utilizatorii să afle ce este ascuns în mânecă.

Cel mai bun lucru la PHP este simplitatea extremă pentru un începător, dar totodată existența multor facilități avansate pentru un programator profesionist.

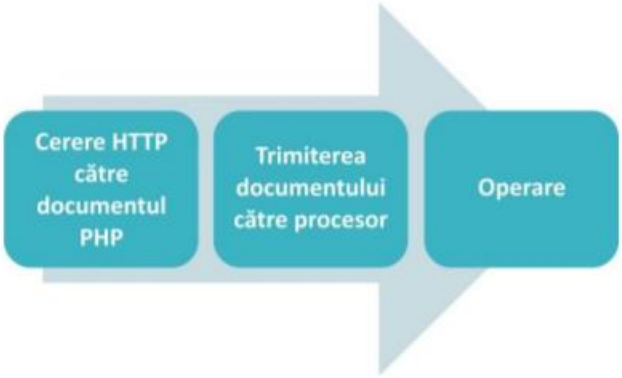


fig.2 Cum funcționează PHP

PHP este axat în principal pe scripting pe partea serverului, deci puteți realiza cu el orice poate realiza un alt program CGI (Common Gateway Interface), cum ar fi colectarea datelor din forme, generarea conținutului dinamic al paginilor sau trimiterea și primirea cookies. Dar PHP poate face mult mai multe. Sunt trei domenii principale, unde scripturile PHP sunt utilizate:

- Scripting pe partea serverului, acesta este cel mai tradițional și de bază domeniu al PHP.
- Scripting în linia de comandă.
- Scrierea aplicațiilor de birou.

Una dintre cele mai puternice și semnificative facilități ale PHP este susținerea unui larg domeniu de baze de date. Scrierea unei pagini web ce accesează o bază de date este foarte simplă utilizând una din extensiile de lucru cu baze de date (de ex. pentru MySQL), sau utilizând un nivel de abstractizare precum PDO, sau conectarea la orice bază de date ce susține standardul *Open Database Connection* cu ajutorul extensiei ODBC. Alte baze de date pot utiliza cURL sau socket-uri, cum ar fi CouchDB. Demonstrația vezi de mai jos (fig.3).

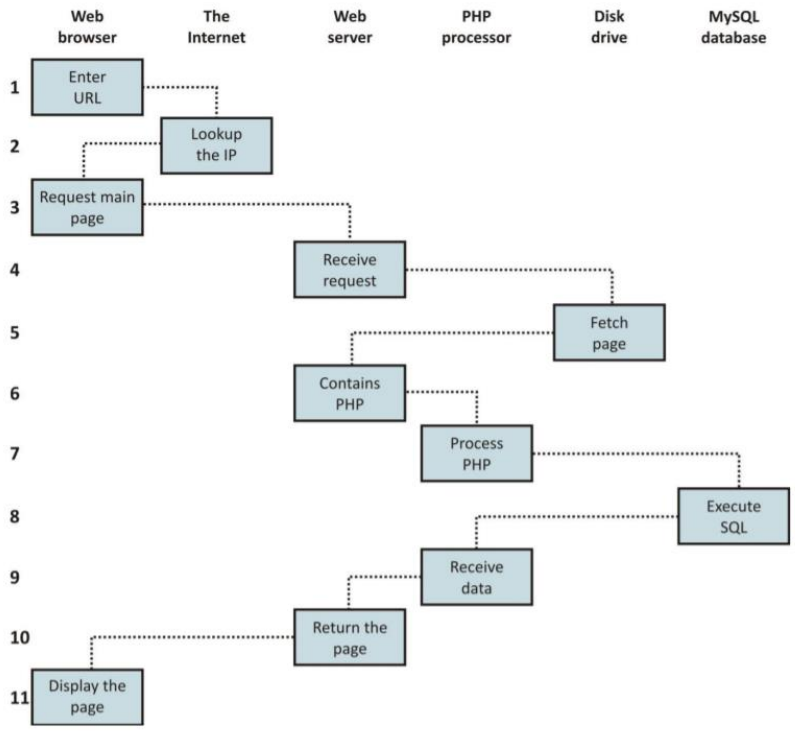


fig.3 Legătura între HTTP – PHP și bază de date

3.2. HTML și CSS

HyperText Markup Language, prescurtat HTML, definește conținutul unei pagini web. De exemplu conținutul poate fi reprezentat prin antete, paragrafe, imagini etc. Acesta este un limbaj descriptiv (engl. markup), pe care îl folosim pentru structurarea datelor și a diferitor elemente ale site-urilor noastre. Cu alte cuvinte, putem spune că limbajul HTML, împreună cu tag-urile sale, constituie structura de bază a fiecărui site modern. Nu se poate considera programare, deoarece îi lipsesc unele elemente de bază și regulile care sunt obligatorii în limbaje de programare precum PHP, JavaScript, C# și altele. Desigur, aceasta nu înseamnă că HTML este slab, imperfect sau orice altceva. HTML își are propriul rost și trebuie considerat ca atare. Cu ajutorul lui setăm elemente definite și determinate în mod specific, pe care le numim tag-uri (engl. tag), cu ajutorul cărora definim structura. Există diferite tag-uri pentru diverse scopuri. Dacă nu ar fi limbajul HTML, nu am putea să definim și să separăm elementele de ordine ale paginilor, să setăm determinante tipografice etc.

CSS (acronimul de la Cascading Style Sheets) este un limbaj de prezentare creat pentru a oferi conținutului aspectul și forma - folosind de exemplu fonturi, dimensiuni sau culori. Limbajul CSS a fost creat ulterior limbajului HTML și de aceea versiunile mai vechi de HTML conțin și elemente pentru specificarea aspectului paginilor care cu timpul vor fi eliminate în versiunile mai noi.

Cele două limbaje - HTML și CSS - sunt independente unul de altul. Acest lucru constituie un avantaj. Nu este recomandat să fie scris cod CSS în interiorul unui document HTML și nici viceversa. De regulă HTML va reprezenta structura și conținutul unei pagini web, în timp ce CSS controlează aspectul și forma acestui conținut. Ambele limbaje sunt, pe de o parte, limbaje foarte puternice cu care se pot obține multe, în timp ce, pe de altă parte, sunt foarte simple în propriile reguli de scriere și în structura pe care o urmează. Desigur, această simplitate și vizibilitate este unul dintre avantajele care a dus la popularitatea și acceptarea rapidă din partea experților, în perioada dezvoltării inițiale a web-ului (6). Prin înțelegerea elementelor simple și prin respectarea deplină a regulilor bine definite, se pot obține rezultate remarcabile în HTML și CSS. În același timp, aceste limbaje au propriile lor limite. Ca limbaje descriptive, mark-up (ci nu de programare), acestea nu ne pot ajuta în privința sarcinilor ce țin de logică, solicitate de programare. Din această cauză, majoritatea site-urilor din zilele noastre, pe lângă HTML și CSS, utilizează simultan JavaScript și PHP, devenind astfel norma acceptată. Ele nu concurează, ci se completează în armonie.

3.3 JavaScript

JavaScript a fost dezvoltat prima dată de către firma Netscape, cu numele de Live Script, un limbaj de scripting care extindea capacitățile HTML, oferă o alternativă parțială la utilizarea unui număr mare de scripturi CGI pentru prelucrarea informațiilor din formulare și care adaugă dinamism în paginile web. După lansarea limbajului Java, Netscape a început să lucreze cu firma Sun, cu scopul de a crea un limbaj de script cu o sintaxă și semantică asemănătoare cu a limbajului Java, și din motive de marketing, numele noului limbaj de script a fost schimbat în *JavaScript*. JavaScript a apărut din nevoia ca logica să fie și pe partea de client, nu doar pe partea de server. Dacă toată logica este pe partea de server, întreaga prelucrare este făcută la server,

chiar și pentru lucruri simple, așa cum este validarea datelor. Astfel, JavaScript îl înzestrează pe client și face ca relația să fie un adevărat sistem client-server.

JavaScript, care permite inserarea în paginile web a script-urilor care se execută în cadrul paginii web, mai exact în cadrul browser-ului utilizatorului, ușurând astfel și traficul dintre server și client, a fost un pas spre interactivizare. JavaScript conține o listă destul de amplă de funcții și comenzi menite să ajute la operații matematice, manipulări de șiruri, sunete, imagini, obiecte și ferestre ale browser-ului, link-urile URL și verificări de introduceri ale datelor în formulare. Codul necesar acestor acțiuni poate fi inserat în pagina web și executat pe calculatorul vizitatorului. După lansarea sa, în decembrie 1995, JavaScript și-a atras sprijinul principalilor distribuitori din domeniu, cum sunt Apple, Borland, Informix, Oracle, Sybase, HP sau IBM**.** S-a dezvoltat în continuare, obținând recunoaștere majorității browser-elor. Înțelegând importanta scripting-ului web**,** Microsoft a dorit să ofere suport și pentru JavaScript. Netscape a preferat să acorde licența de tehnologie companiei Microsoft în loc să o vândă, astfel Microsoft a analizat JavaScript, și bazând-se pe documentația publică a creat propria sa implementare, *Jscript*, care este recunoscută de Microsoft Internet Explorer. Jscript 1.0 este aproximativ compatibil cu JavaScript 1.1, care este recunoscut de Netscape Navigator. Pe lângă Jscript, Microsoft a introdus și un concurent pentru JavaScript**,** numit VBScript, realizat pentru a ușura pătrunderea pe web a programatorilor VB. VBScript este un subset al limbajului Visual Basic. Cu toate acestea JavaScript a devenit cunoscut ca limbajul de scripting standard pentru web (7).

Avantaje:

- Poate fi introdus in HTML – De obicei codul JavaScript este găzduit în documentele HTML și executat in interiorul lor. Majoritatea obiectelor JavaScript au tag-uri HTML pe care le reprezintă, astfel încât programul este inclus pe partea de client a limbajului. JavaScript folosește HTML pentru a intra in cadrul de lucru al aplicațiilor pentru web.
- Este un limbaj în totalitate interpretat – codul scriptului va fi interpretat de browser înainte de a fi executat. JavaScript nu necesită compilări sau preprocesări, ci rămâne parte integrantă a documentului HTML. Putem mult mai ușor să actualizăm codul sursă.
- Este un limbaj flexibil – in aceasta privința limbajul diferă radical de C++ sau Java. În JavaScript putem declara o variabilă de un anumit tip, sau putem lucra cu o variabilă deși nu-i cunoaștem tipul specificat înainte de rulare .

- Este multifuncțional – limbajul poate fi folosit într-o multitudine de contexte pentru a rezolva diferite probleme: grafice, matematice, și altele. Lucrează cu date calendaristice.
- Evoluează – limbajul evoluează, fapt pozitiv care însă poate genera și probleme, programatorii trebuind să verifice permanent ce versiune să folosească pentru ca aplicațiile să poată fi disponibile unui număr cât mai mare de utilizatori de browsere diferite.
- Acoperă contexte diverse – programarea cu acest limbaj este îndreptată mai ales către partea de client, dar putem folosi JavaScript și pentru partea de Server cu Node.js.

Dezavantaje:

- Este dependent de mediu – JavaScript este un limbaj de scriptare; software-ul care rulează de fapt programul este browser-ul web (Firefox, Opera, Netscape Navigator, Internet Explorer, Safari, etc.) Este important să luăm în considerare această dependență de browser atunci când utilizăm aplicații JavaScript.
- Rularea durează ceva mai mult deoarece comenzile JavaScript vor fi citite de navigatorul Web și procesate atunci când user-ul apelează la acele funcții (prin completare de formulare, apăsare de butoane, etc.).
- Este bazat pe obiecte – JavaScript nu este un limbaj de programare orientat-obiect, ca Java, ci mai degrabă este "bazat pe obiecte". Modelul de obiect JavaScript este bazat pe instanță și nu pe moștenire.
- Nu are acces la fișierele utilizatorului, cu excepția fișierelor cookies (8).

3.4 MySQL

MySQL este un sistem de gestionare open source a bazelor de date relaționale, foarte popular, utilizat în rândul aplicațiilor online.

Pentru a administra bazele de date MySQL se poate folosi modul linie de comandă (vezi fig.4) sau, prin descărcare de pe internet, o interfață grafică: *MySQL Administrator* și *MySQL Query Browser*. Un alt instrument de management al acestor baze de date este aplicația gratuită, scrisă în PHP, *phpMyAdmin*.

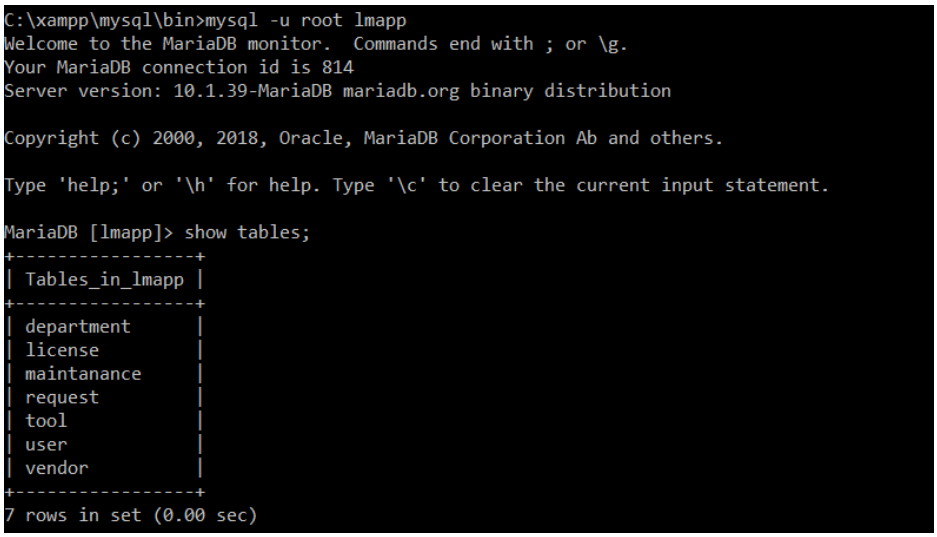


fig.4 MySQL din linia de comandă

MySQL este componentă integrată a platformelor LAMP sau WAMP (Linux/Windows-Apache-MySQL-PHP/Perl/Python). Popularitatea sa ca aplicație web este strâns legată de cea a PHP-ului, care este adesea combinat cu MySQL, combinație supranumită și "Duo-ul Dinamic". În multe cărți de specialitate este precizat faptul că MySQL este mult mai ușor de învățat și folosit decât multe din aplicațiile de gestiune a bazelor de date (9), ca exemplu comanda „datediff", care returnează numărul zilelor între două date.

Ca și motor de stocare, folosesc InnoDB. InnoDB oferă funcții de integritate/fiabilitate importante:

- Tranzacții ACID (Atomicitate, Consistență, Izolare, Durabilitate)
- Restricții cheie străine
- Recuperare robustă a accidentelor

Figura de mai jos (fig.5) prezintă accesarea MySQL din PHP.

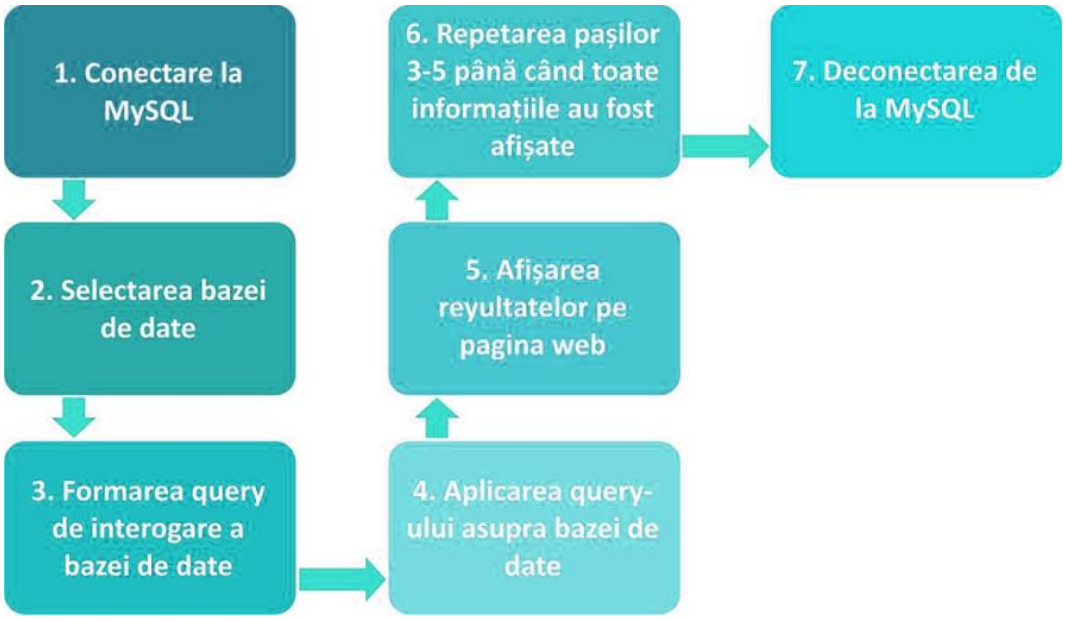


fig.5 Accesare MySQL din PHP

Partea a IV – a: Aplicația

4.1 Crearea bazei de date

Am creat baza de date cu ajutorul comenzii „CREATE DATABASE *lmapp”*; în command line, după asta am făcut arhitectura aplicației, și am plănuit entitățile (tabele) și atributele (coloane), după figura de mai jos (fig.6).

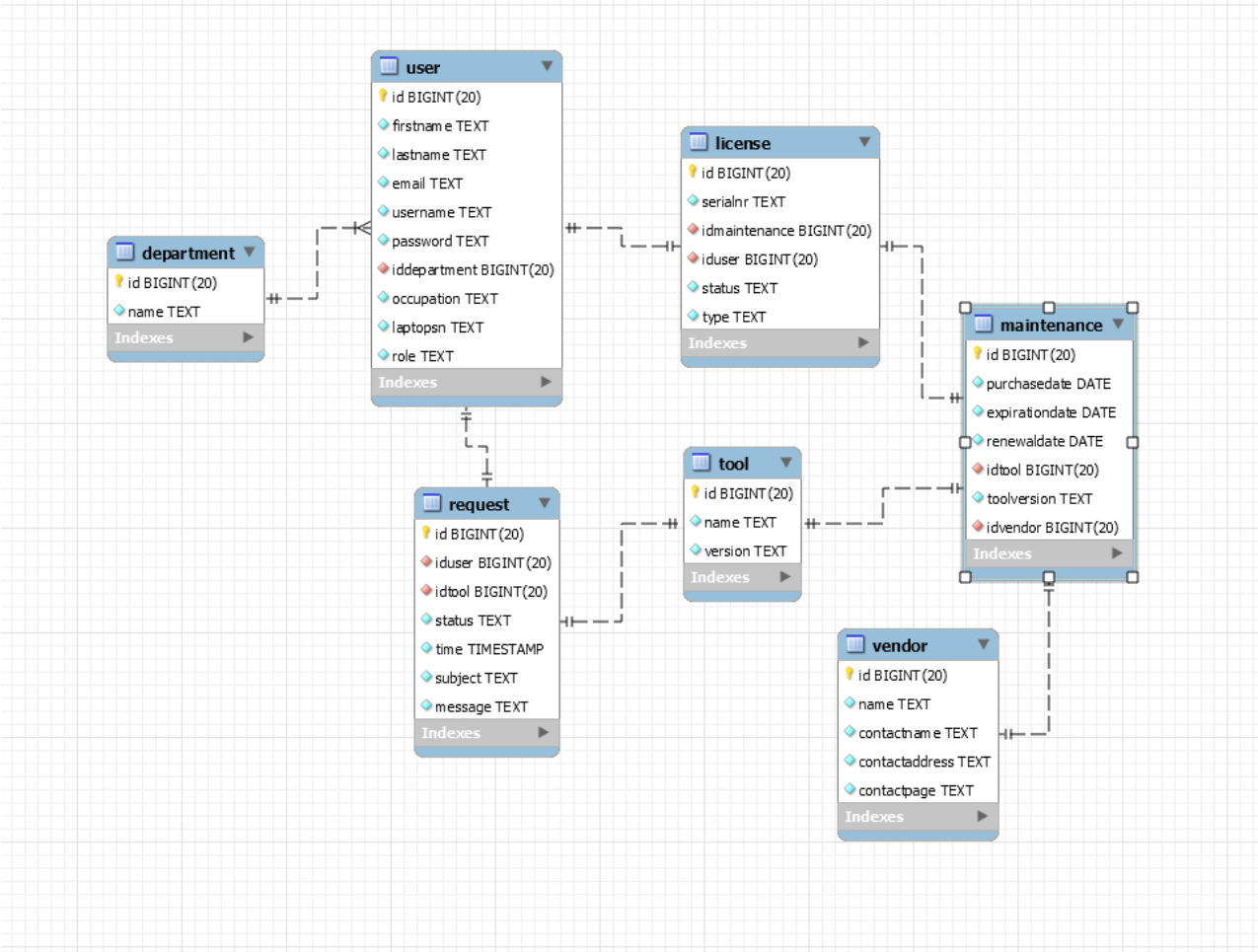


fig.6 Baza de date a aplicației

Un tabel păstrează informații despre anumite seturi de date. Acestea, de cele mai multe ori, trebuie să fie unice pentru că nimeni nu are interesul să păstreze aceeași informație de mai multe ori. Unicitatea este conferită printr-o constrângere. Astfel, datelor din tabel le sunt asociate

anumite valori (de regulă numerice) prin care este asigurată atât unicitatea cât și coerența datelor din tabela respectivă. Aceste valori se numesc în mod generic ID-uri și ele nu apar în fata utilizatorului aplicației care lucrează cu baza de date. Rolul lor este să păstreze consistența datelor. Toate ID-urile din bază de date au tip *bigint(20)* și cei care sunt cheie primare au fost setate ca *AUTO_INCREMENT*. Setul de caractere a fost definit ca *utf16_romanian_ci*. Restul variabilelor sunt de tip *text* și *date* (dată calendaristică).

Prima dată am creat tabela *user* unde am următoarele coloane: id (setat ca și cheia primară), date de utilizator: nume, prenume, adresă de e-mail, nume de utilizator, parolă, id-ul departamentului, ocupația, număr serial laptop al utilizatorului și ce rol are în bază de date (user, admin etc), vezi fig.7.

```
MariaDB [lmapp]> SHOW COLUMNS FROM user;
+--------------+------------+------+-----+---------+----------------+
| Field        | Type       | Null | Key | Default | Extra          |
+--------------+------------+------+-----+---------+----------------+
| id           | bigint(20) | NO   | PRI | NULL    | auto_increment |
| firstname    | text       | NO   |     | NULL    |                |
| lastname     | text       | NO   |     | NULL    |                |
| email        | text       | NO   |     | NULL    |                |
| username     | text       | NO   |     | NULL    |                |
| password     | text       | NO   |     | NULL    |                |
| iddepartment | bigint(20) | NO   |     | NULL    |                |
| occupation   | text       | NO   |     | NULL    |                |
| laptopsn     | text       | NO   |     | NULL    |                |
| role         | text       | NO   |     | NULL    |                |
+--------------+------------+------+-----+---------+----------------+
10 rows in set (0.01 sec)
```

fig.7 Tabela user

După care au urmat tabelele *department* și *tool*, care au puține coloane. La department avem id și name (adică numele departamentului), iar la tool avem id, name (numele tool-ului) și version (adică versiune actuală). Între timp am creat și tabela *vendor* pe care o folosim ca să stocăm date despre vendor (distribuitor/furnizor) de la care sunt cumpărate licențele. Tabela conține următoarele coloane: id, nume de contact, adresă de contact și pagina de contact.

Următoarea a fost tabela *maintenance* care are următoarele coloane: id, purchase date (adică data de cumpărare), expiration date (data expirării), renewal date (data reînnoirii), id tool, tool version (versiunea tool-ului) și id vendor.

După care am creat penultima tabela *license* care conține: id, un număr serial, id de maintenance, id user, status (dacă este folosită sau liberă) și type care se referă la tipul licenței – *multi*, adică este o licență pe care o pot folosi mai mulți useri sau *unique*, o licență unică pentru un singur calculator sau user.

Ultima tabela a fost tabela *request* care conține id, id user, id tool, status (se referă la statusul cererii, dacă a fost postat, sau este în progres, finalizat etc.), coloana time este un timestamp automat (care e setat prin parametrul ”ON UPDATE CURRENT_TIMESTAMP”, ceea ce înseamnă că la orice modificare efectuată se actualizează), subject (titlul cererii) și message (mesajul de la utilizator referitor la versiuni, detalii tehnice).

4.2 Conexiunea la baza de date

Pentru a folosi datele din bază cu ajutorul PHP-ului, avem nevoie să configurăm datele de logare pentru bază de date. În cazul meu acest file se numește *db_connection.php* (fig.8 de mai jos).

db_connection.php

db_connection.php > ...

```php
<?php
function OpenCon()
 {
 $dbhost = "localhost";
 $dbuser = "root";
 $dbpass = "parola_mea";
 $db = "lmapp";
 $conn = new mysqli($dbhost, $dbuser, $dbpass,$db) or die("Connect failed: %s\n". $conn -> error);

 return $conn;
 }

function CloseCon($conn)
 {
 $conn -> close();
 }

?>
```

fig.8 Datele conexiunii

Când folosim datele de conectare la un alt fișier, indicat să folosim funcțiile require sau require_once, în cazul în care vrem ca fișierul să fie invocat. Folosind funcțiile menționate, în cazul în care nu se conectează cu succes, funcția se oprește și returnează o eroare. După ce am invocat fișierul de config, cu ajutorul funcției OpenCon(), ne conectăm la baza de date (vezi fig.9 de mai jos). Dacă conectarea eșuează primim o eroare, returnată de clauza *die* din funcția OpenCon() (vezi linia 8, fig. 8).

```php
//make connection
require_once ("../db_connection.php");
$conn = OpenCon();
```

fig.9 Conectare la baze de date

4.3 Fișierul head.php

Fișierul *head.php* (vezi fig.10) are scopul limitării accesului nesecurizat:

- unul din metodele de injectare a bazei de date se bazează pe metoda GET. Însă metoda respectivă necesită o anumită lungime de caractere transmise prin GET.
- în mod normal URL-ul poate fi de maxim 2048 de caractere
- dacă în locul unui șir de caractere lung folosim la GET o altă metodă de identificare, de exemplul ID-ul înregistrării din baza de date, nu mai avem nevoie de multe caractere, și putem limita datele transmise în așa fel, încât metoda de injectare să nu mai funcționeze.

head.php > ...

```php
<?php
//limiting when using GET
$urlurl = $_SERVER['QUERY_STRING'];
$urlurlurl = strlen($urlurl);
if ($urlurlurl >= '20') echo "<meta http-equiv='refresh' content='0; URL=index.php'>";
?>
```

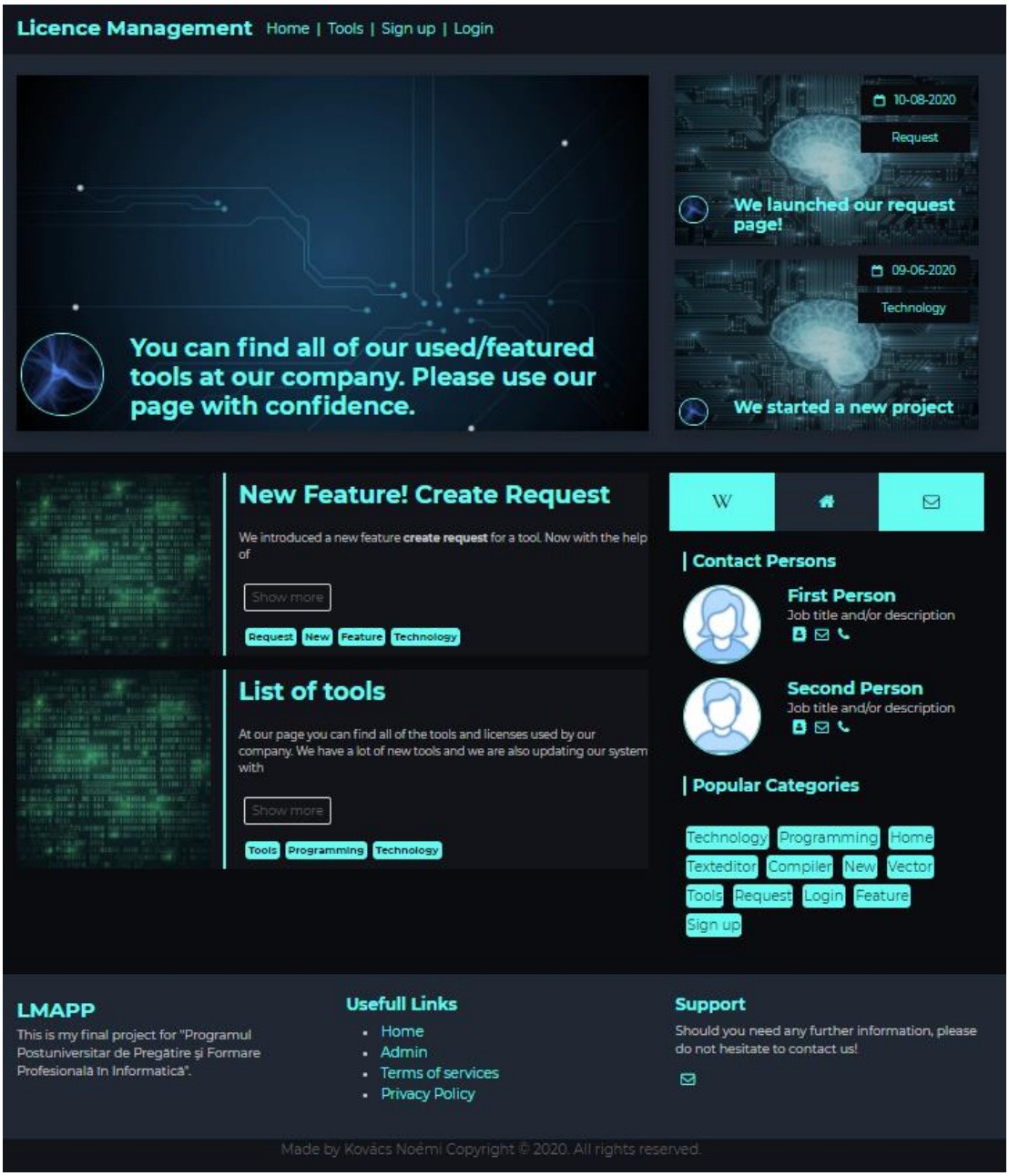
fig.10 Fișierul head.php

Acest fișier îl includem în toate celelalte fișiere ale noastre, formatul corect de a include este: *<?php include("head.php"); ?>*. Include inserează codul din fișierul specificat ori de câte ori

folosim funcția, mai există varianta inculde_once care prima dată verifică dacă deja a fost inserat codul, și dacă nu, numai atunci îl inserează.

4.4 Partea de user

Aș putea împărți codul în două părți: partea de user și partea de admin. În această parte descriu cum am scris partea de user, adică abordăm aplicația din perspectiva userului final. Când deschidem aplicația, avem o pagină de index (vezi fig.11 mai jos). Pe pagina de index, sau Home avem informații utile legat de licențe sau tool-uri noi, este o parte și pentru știri legate de noutăți (de ex. că am creat o interfață nouă pentru a crea cereri pentru tool-uri).



fig,11 Index.html (pagina de Home)

Sunt afișate datele de contact al suportului precum și o adresă de contact pentru întrebări generale și mai avem linkuri utile către site-uri interne. În navbar avem opțiuni de logare, înscriere și a vizualiza o parte din tool-urile folosite (sunt afișate doar numele tool-urilor). Aici ar apărea o lista întreagă, dar este limitat rezultatul, cu ajutorul interogării:

*$query="SELECT * FROM tool limit 3"*; și utilizatorul este anunțat să se logheze pentru mai multe detalii.

Pe fiecare pagină verificăm, dacă userul este deja logat, adică există o sesiune (vezi fig.12). În funcție de această verificare navbar-ul este afișat altfel.

```php
<?php
//verify if the user is already loged in - exist a session
if (isset($_SESSION['username'])) {
    echo "<a href='index.php' target='_self'>Home</a>";
    echo "  |  ";
    echo "<a href='request.php' target='_self'>Request</a>";
    echo "  |  ";
    echo "<a href='changepass.php' target='_self'>Change Password</a>";
    echo "  |  ";
    echo "<a href='logout.php' target='_self'>Logout</a>";

}
// if the user is not loged in
else {

    echo "<a href='index.php' target='_self'>Home</a>";
    echo "  |  ";
    echo "<a href='tool.php' target='_self'>Tools</a>";
    echo "  |  ";
    echo "<a href='signup.php' target='_self'>Sign up</a>";
    echo "  |  ";
    echo "<a href='login.php' target='_self' align='left'>Login</a>";
}
?>
```

fig.12 Navbar pe partea de user

Folosim o funcție pentru sesiune, aceasta trebuie folosită înainte de tagul <html> (*session_start();*), sesiunile se salvează în fișiere temporare pe server și sesiunile se distrug la închiderea browserului sau în cazul lipsei unei acțiuni pe o perioadă determinată din partea utilizatorului pe pagină. Există și *session_destroy()*, funcția cu care închidem o sesiune. Am folosit această funcție la log out, adică deconectare din aplicație.

4.4.1 Pagina de înscriere

Utilizatorii au oportunitate să se înscrie pe o pagină dedicată a înscrierilor. Această pagina înregistrează datele de user, și le stochează în tabela de user, dacă datele trec prin validările scrise în back-end, despre care utilizatorul este informat. Vezi fig.13 cum arată un formular necompletat în timpul validării după apăsarea butonului de submit.

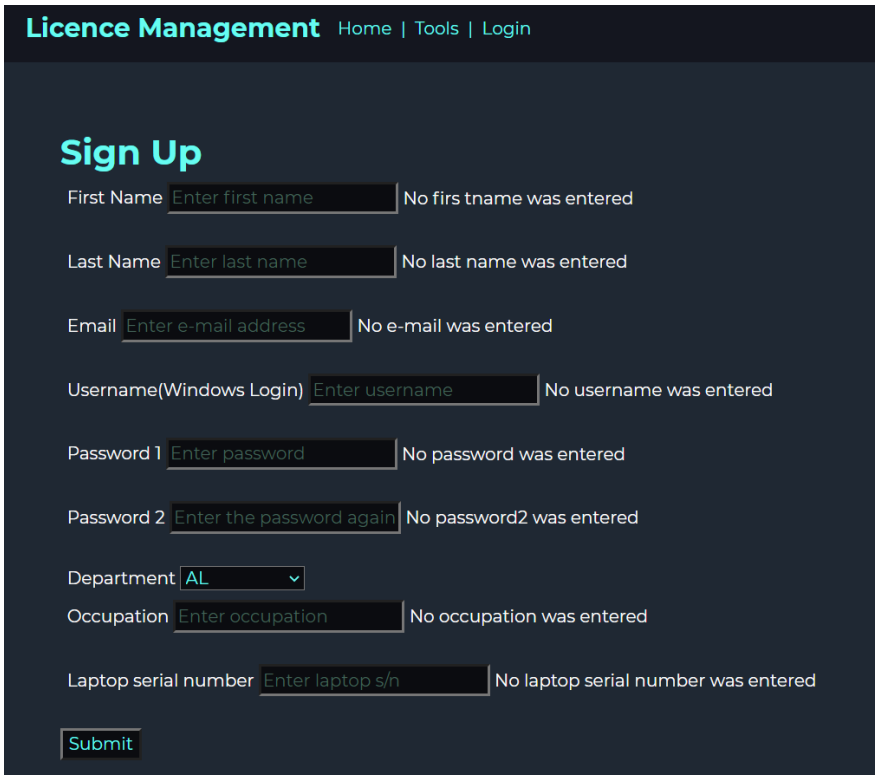


fig.13 Formular de înscriere – validări

Pentru validare am folosit o funcție de sanitizare:

```
function fix_string($string) {
 $string = stripslashes($string);
 return htmlentities($string);
}
```

unde *strisplashes()* șterge backslash-urile și *htmlentities()* ca și *htmlspecialchars()*, această funcție PHP convertește caracterele în entitățile HTML corespunzătoare. Marea diferență este aceea că toate caracterele care pot fi convertite, vor fi convertite. După sanitizare am validat toate câmpurile de intrare, de exemplu:

[…]

```
function validate_email($field) {
 if ($field == "")
 return "No Email was entered<br />";
 else if (!((strpos($field, ".") > 0) &&
(strpos($field, "@") > 0)) ||
preg_match("/[^a-zA-Z0-9.@_-]/", $field))
 return "The Email address is invalid<br />";
 return "";
}
function validate_username($field) {
 if ($field == "")
 return "No Username was entered<br />";
 else if (strlen($field) < 5)
 return "Username must be at least 5 characters<br/>";
 else if (preg_match("/[^a-zA-Z0-9_-]/", $field))
 return "Only letters, numbers, - and _ in username";
 return "";
}
```

[...]

Câmpurile validate au fost atribuite la câte o variabilă pentru a trata cazurile de eroare ca să afișăm ulterior pentru utilizatori (fig.13, de mai sus). În același timp am verificat dacă adresa de mail sau nume de utilizator există deja în baza de date ca să evităm duplicări. Dacă validarea rulează fără eroare se face inserare în baza de date, apare un pop-up cu succes (vezi codul din fig.14) și userul este redirecționat către pagina de logare.

Sigur ca ați observat că în baza de date este un câmp pentru rol, iar în formular nu se regăsește. Aceasta se datorează faptului că userii înregistrați din start au rol de user, pe care adminul îi poate modifica ulterior prin interfața de admin.

La câmpul department am folosit un meniu drop-down, care este conectat cu baza de date, și ne arată în timp real ce opțiuni sunt inserate, dintre care userul, dacă nu alege una, automat setează prima opțiune din listă.

signup.php > html

```
155     $eemail = "The mail is already exist! ";
156     $fail .= $eemail;
157   }
158
159   $fail=$fail.$efirstname.$elastname.$eemail.$eusername.$epassword1.$epassword2.$eiddepartment.$eoccupation.$elaptopsn;
160   if ($fail == "") {
161
162
163       //iddepartment in this case is the name of the department
164       //modified subsequently to dropdown, previously was with html select
165       $iddepquery= "SELECT id from department where name = '$iddepartment'";
166       $result1=mysqli_query($conn,$iddepquery);
167
168       $res1 = $result1->fetch_row();
169       $var1= $res1[0] ?? false;
170
171       //insert
172       $localvar = md5($_POST['password1']);
173       $sql="INSERT INTO user (firstname ,lastname, email, username, password, iddepartment, occupation, laptopsn, role)
          VALUES ('$_POST[firstname]','$_POST[lastname]','$_POST[email]','$_POST[username]','$localvar','$var1','$occupation',
          '$laptopsn','$role')";
174
175       echo "<br>";
176       if (mysqli_query($conn,$sql)){
177         echo '<script type="text/javascript">';
178         echo 'alert("The user was registered!");';
179         echo 'window.location.href = "login.php";';
180         echo '</script>';
181       }
182       else {
183         echo '<script type="text/javascript">';
184         echo 'alert("Error. The user was not registered!");';
185         echo 'window.location.href = "signup.php";';
186         echo '</script>';
187       }
188       exit;
189   }
```

fig.14 Inserare și pop-up

4.4.2 Pagina de logare

Pagina de logare este pentru useri existenți în bază de date. Acest formular conține două câmpuri, unul pentru username și celălalt pentru parolă (vezi fig.15).

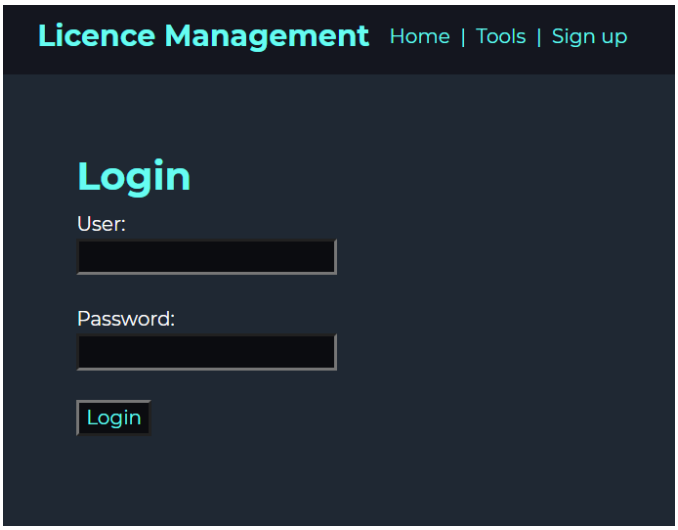


fig.15 Pagina de logare

În momentul logării, userul este verificat ce rol are. Dacă este admin, este redirecționat către pagina de admin cu ajutorul unei expresii condiționale *if*:

if (isset($_SESSION['role']) && $_SESSION['role'] == "admin") {header("Location: ./admin/admin.php");}.

Dacă credențialele nu sunt corecte, userul primește un mesaj general: „User and/or password incorrect.”. Dacă userul există în baza de date și datele sunt corecte, userul este redirecționat către pagina de index unde apar și alte tab-uri în afară de login și de sign up (vezi fig.12).

În procesul de verificare am folosit funcția *mysqli_real_escape_string()* pentru username și parolă, util pentru secvența de escape al caracterelor speciale dintr-un șir, pentru a fi utilizat într-o interogare SQL, ținând cont de setul curent al conexiunii (vezi fig.16 mai jos).

```php
$db = mysqli_connect($host,$db_username,$db_password,$dbname) or die ("could not connect");
$myusername=$_POST['username'];
$mypassword=$_POST['password'];
$myusername = stripslashes($myusername);
$mypassword = stripslashes($mypassword);
$myusername = mysqli_real_escape_string($db,$myusername);
$mypassword = mysqli_real_escape_string($db,$mypassword);
$mypassword = md5($mypassword);
$query="SELECT * FROM user WHERE username='$myusername' and password='$mypassword'";
$result=mysqli_query($db,$query);

while($r=mysqli_fetch_array($result))
  {
  $UserID=$r["id"];
  $UserFirstName=$r["firstname"];
  $UserLastName=$r["lastname"];
  $UserEmail=$r["email"];
  $UserName=$r["username"];
  $UserPassword=$r["password"];
  $UserIdDepartment =$r["iddepartment"];
  $UserOccupation=$r["occupation"];
  $UserRole=$r["role"];
  }
  $count=mysqli_num_rows($result);
  // if ok, signup user
  if($count==1){

    $_SESSION['id']= $UserID;
    $_SESSION['firstname'] = $UserFirstName;
    $_SESSION['lastname'] = $UserLastName;
    $_SESSION['email'] = $UserEmail;
    $_SESSION['username'] = $UserName;
    $_SESSION['password'] = $UserPassword;
    $_SESSION['iddepartment'] = $UserIdDepartment;
    $_SESSION['occupation'] = $UserOccupation;
    $_SESSION['role']= $UserRole;

    echo "<meta http-equiv='refresh' content='0;URL=index.php'>";
  }
// if the user and/or password is not correct
  else {
  echo "<div class=container>";
  echo "<div class=jumbotron>";
  echo "User and/or password incorrect. <br><br>";
  echo "</div></div>";
  }
```

fig.16 Verificare username și parola la logare

4.4.3 Pagina de Tools

După ce s-a logat userul și a dat click la Tools din navbar apare o tabelă cu toate tool-urile existente în baza de date (vezi fig.17 de mai jos). Userul are posibilitate de a filtra după ordine descrescătoare și crescătoare, are disponibil și un câmp pentru a căuta în tabela respectivă. Toate datele sunt afișate cu paginare, se poate seta câte intrări să apară, în cazul meu am setat la 5, 10, 20 sau toate intrările din tabela respectivă. Din stânga jos avem și un text care este asociat cu paginarea: *„Showing x to y of z entries”*, care se referă la intrări din baza de date, cum sunt afișate, în cazul acesta, pe pagina 1 sunt afișate 5 elemente din 8, care e totalul.

Pentru afișare am folosit dataTables (10), care este un plug-in pentru biblioteca (engl. library) jQuery JavaScript. Este un tool foarte flexibil, bazat pe principiile îmbunătățirii progresive, care poate adăuga toate aceste funcții avansate oricărui tabel HTML.

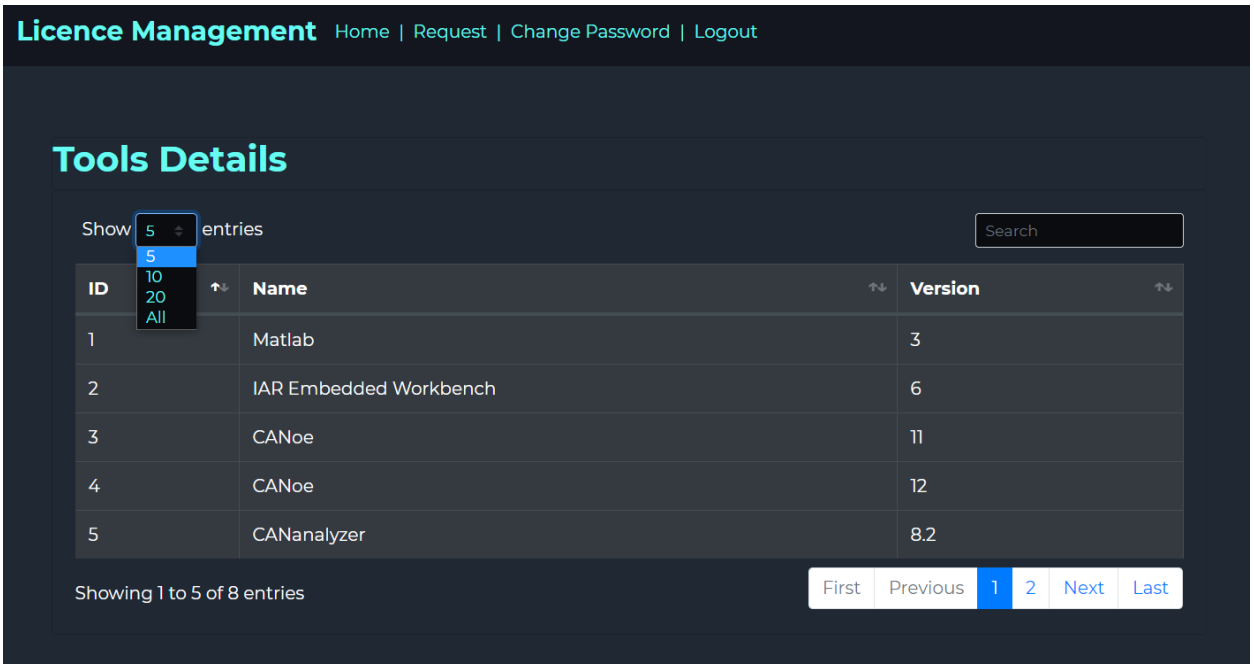


fig.17 Tabela tool parte de user

4.4.4 Pagina de Request

Pagina a fost creată special pentru useri, permițând crearea unei cereri pentru un tool. Pe pagină se găsește un formular, care este validat și datele sunt sanitizate, ca la înscriere.

În baza de date, în tabela request, apar ID de user și ID de tool, iar în front-end am scris astfel încât sa apară username și tool name (vezi fig.18 de mai jos). Am afișat numele tool-ului folosind o interogare SQL, prin JOIN cu tabela tool. Puteți să observați, că am folosit un drop-down pentru nume de tool, ca și la înscriere. La câmpul respectiv datele sunt luate din baza de date în timp real. Dacă userul nu alege nici un tool automat, i se asignează primul din listă.

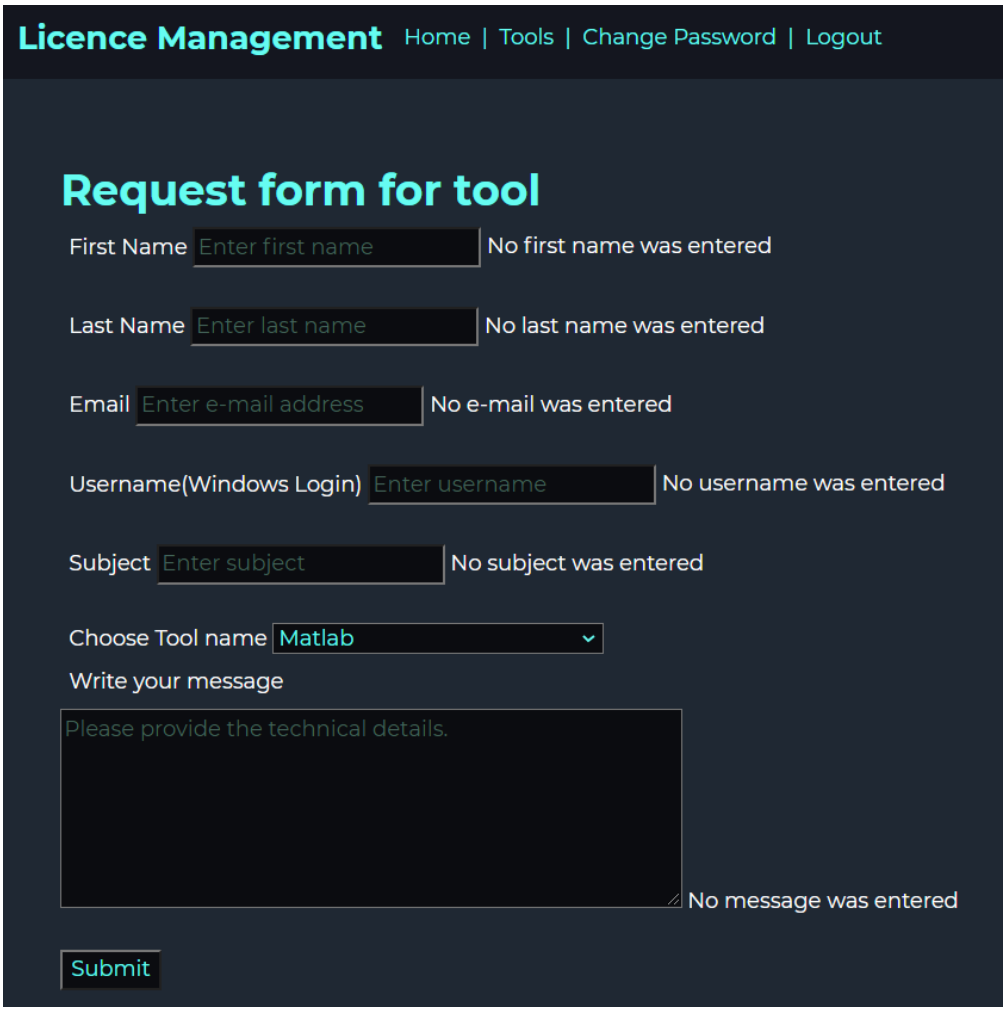


fig.18 Pagina de request cu validări afișate

Pentru fiecare cerere în momentul trimiterii apare în tabelă data trimiterii (data actuală). Are și un status, care iar, nu apare pentru useri, ci este vizibil doar pentru admin. La început statusul este *started* la fiecare cerere. Adminul poate modifica statusurile ulterior prin interfața de admin. Dar când adminul schimbă statusul, schimbă și timestamp-ul automat în baza de date ca să avem dovadă a modificării.

După ce s-a făcut înregistrarea în baza de date, administratorul primește un e-mail cu datele utilizatorului și cu mesajul scris. Pentru această funcționalitate am folosit pentru dezvoltare tool-ul Papercut, care are o interfață simplă care simulează un mail server în timp real (vezi fig.19 de mai jos). Ulterior am folosit un Mail Transfer Agent (MTA), *sendmail* (vezi capitolul 4.6).

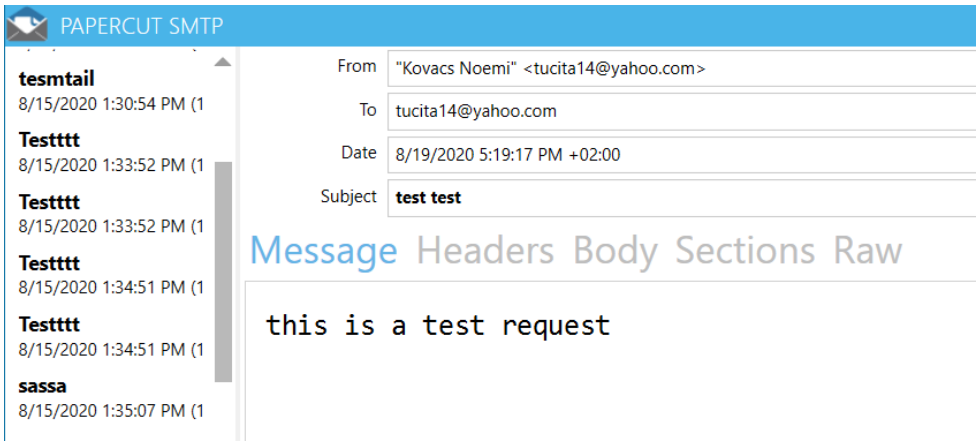


fig.19 Tool-ul Papercut SMTP

Dacă înregistrarea a fost facută, cu succes apare un pop-up (vezi fig.20) după care userul este trimis la pagina de index.

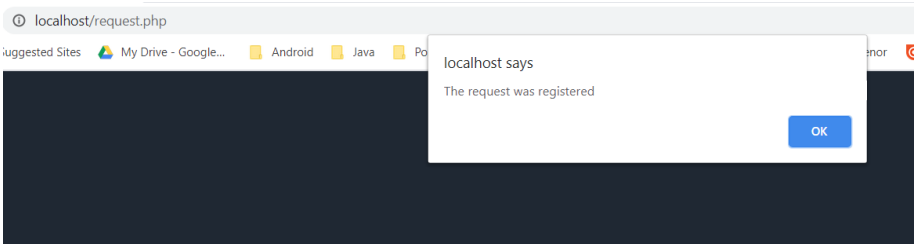


fig.20 Fereastră pop-up

4.4.5 Pagina de schimbare parolă

Utilizatorii au abilitatea de a își schimba parola. Prin acest formular (vezi fig.21 de mai jos) validăm câmpurile de parole, verificăm dacă parola veche este întocmai identică cu cea scrisă de utilizator, și parolele noi să fie la fel. Parola veche am comparat-o cu cea din sesiunea curentă, și este verificat să nu fie la fel parola nouă cu cea veche.

Dacă parolele trec prin toate validările, parola va fi schimbată și utilizatorul este redirecționat la pagina de index.

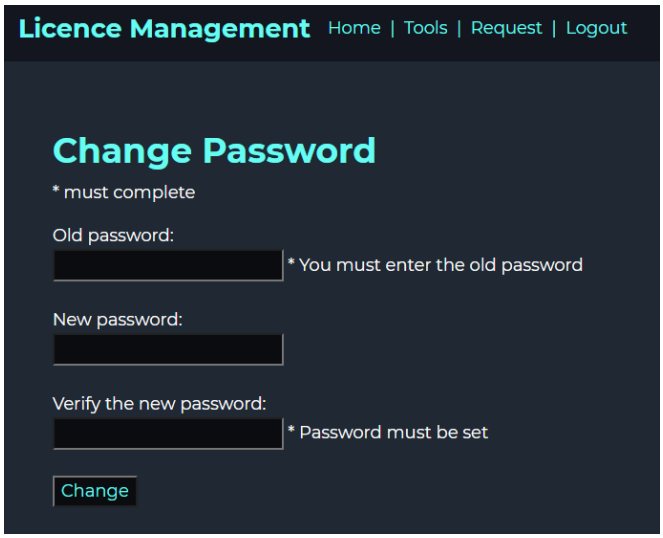


fig.21 Schimbare parolă –validări

4.5 Partea de admin

Partea a doua este pagina de administrare, unde utilizatorul cu rol admin are dreptul de a schimba, a adăuga și a șterge tool-uri, licențe, mentenanțe, vendori, useri, cereri lansate pentru tool-uri, departamente (vezi fig.22) și are opțiunea de schimbare parolă, care este identică cu a userului (vezi fig.21 de mai sus).

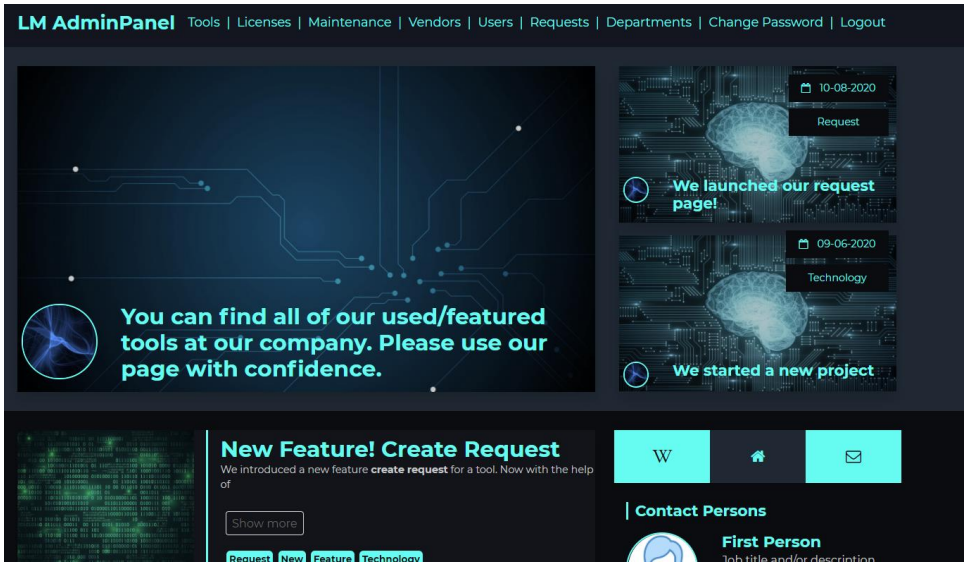


fig.22 Pagina de admin – index

Pe partea de admin, pe fiecare pagină facem o verificare dublă din motive de securitate, ca să evităm spargerea site-ului prin URL-ul dat (presupunem că user-ul știe adresa către o pagină de admin), când userul nu este logat, nu există nici o sesiune. Am verificat dacă există sesiune și dacă role-ul este setat, dacă nu, atunci apare un mesaj pop-up, cu un textul prin care user-ul este atenționat să se logheze (fig.23) și este redirecționat către pagina de login.

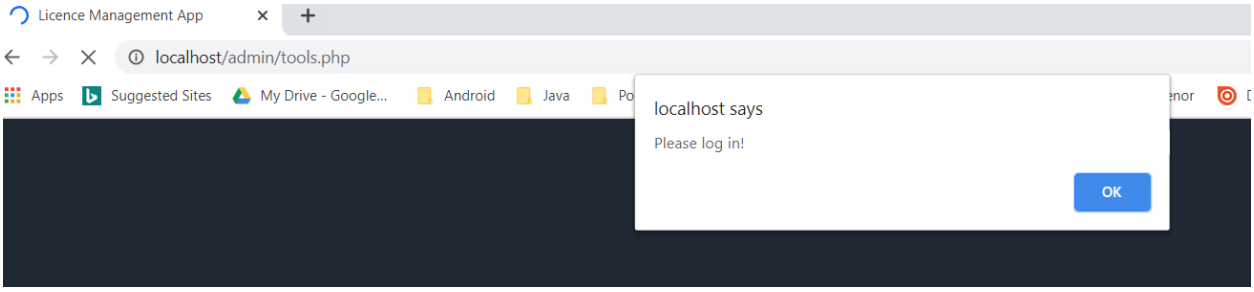


fig.23 Verificare dublă a sesiunii

În momentul logării este verificat dacă userul este administrator sau utilizator normal, în primul caz, design-ul navbar-ului se schimbă și apare LM AdminPanel (vezi codul din fig.24), altfel rămâne pagina de index neschimbată (partea de user).

```
if (isset($_SESSION['role']) && $_SESSION['role'] == "admin") {
  echo "<nav class='navbar navbar-expand-lg'>
    <div class='container'>
    <a class='navbar-brand' href='admin.php'>LM AdminPanel</a>
      <button class='navbar-toggler' type='button' data-toggle='collapse' data-target='#navbarSupportedContent'
      aria-controls='navbarSupportedContent' aria-expanded='false' aria-label='Toggle navigation'>
        <span class='navbar-toggler-icon'></span>
        </button>
```

fig.24 Verificare admin – LM AdminPanel

Când intrăm pe pagina de admin și dacă există licențe care expiră în mai puțin de 60 de zile, primim o notificare, vezi fig.25 de mai jos. Notificarea ne anunță că există un număr de licențe care vor expira în 60 de zile. Dacă dăm click pe OK, ne trimite la pagina de mentenanță, unde putem verifica licențele respective, iar la comanda Cancel rămânem pe pagina de admin.

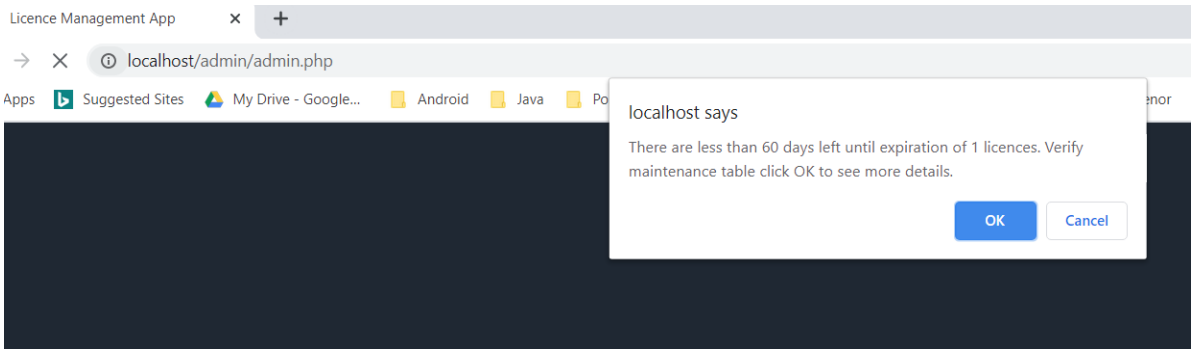


fig.25 Notificare expirare licențelor

Pentru crearea acestei notificări am folosit o interogare:

SELECT tool.name, maint.toolversion versiune, vend.name vendor, maint.expirationdate, maint.renewaldate, maint.purchasedate FROM tool tool, maintenance maint, vendor vend WHERE tool.id = maint.idtool AND maint.idvendor = vend.id AND expirationdate BETWEEN CURDATE() AND DATE_ADD(CURDATE(), interval 60 DAY);

unde verific data expirării între ziua actuală și următoarele 60 de zile.

4.5.1 Folosirea părții de admin

Partea de admin este esența aplicației. Aici putem gestiona toate datele noastre legate de tool-urile și licențele noastre. Componentele de administrare a tool-urilor, a licențelor, a vendorilor, a mentenanțelor, a cererilor, a utilizatorilor și a departamentelor oferă posibilitatea adăugării, editării (aproape orice câmp, detaliat mai departe). La efectuarea unui click pe butonul ADD, EDIT sau DELETE, o fereastră de tip „modal” este afișată pe ecran și oferă posibilitatea modificării datelor selectate, sau inserării datelor noi în baza de date. Pe fiecare pagină, pe lângă capacitatea de a pagina datele în tabelă, avem posibilitatea de a căuta în tabelă după orice tip de date. De asemenea avem și opțiunea de filtra datele în ordine crescătoare sau descrescătoare cu ajutorul săgeților de sus și jos, afișate lângă fiecare nume de coloană. De exemplu pe figura 26 rezultatele sunt ordonate descrescător, de aceea apar primele ID-uri pe pagina 2.

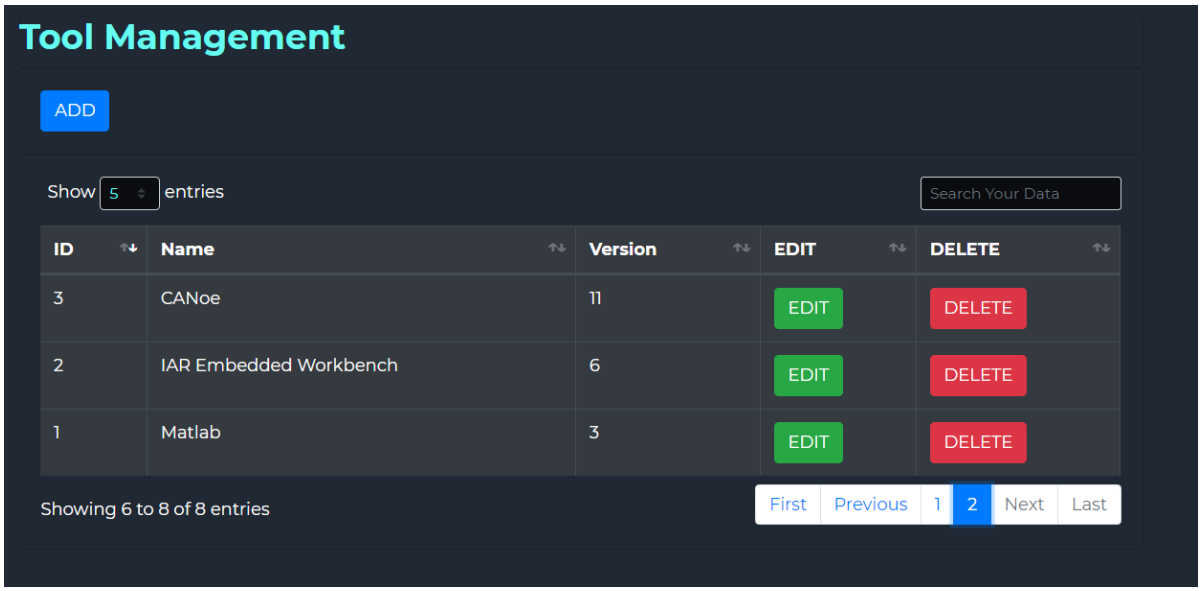


fig.26 Ordinare și paginare

4.5.2 Pagina de Tools

Pe pagina de Tools (Tool Management), sunt afișate cu paginare toate tool-urile folosite într-o tabelă (vezi fig.27 de mai jos). Aici putem administra. adică să adăugăm un tool nou sau să modificăm datele existente, eventual să retragem un tool din uz.

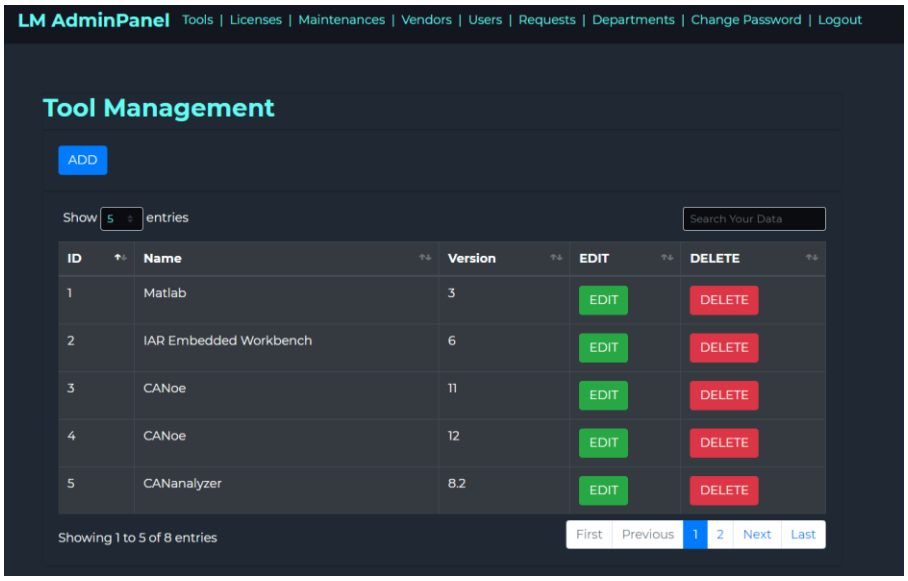


fig.27 Tool Management

Dacă dăm click pe buton ADD, cu ajutorul modal-ului apare un formular de tip pop-up, unde putem să completăm toate câmpurile și salvăm în baza de date (vezi fig.28 mai jos). Dacă dăm *Close*, rămânem pe pagina de Tool Management, fără să adăugăm ceva, iar la apăsarea butonului *Save Data*, datele se salvează în baza de date, cu un ID asociat automat, și deja va apărea în tabela noastră tool-ul recent adăugat.

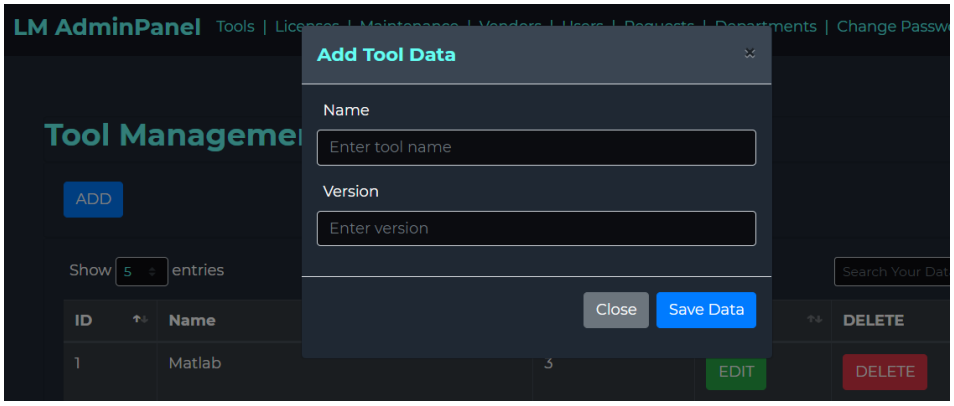


fig.28 Add pop-up modal Tool Management

La EDIT avem o fereastră nouă cu datele existente din baza de date, pe care putem să le editățm, în afară de ID, fiindcă ID-ul este implicit și nu are sens să fie modificat (vezi fig.29, mai jos). La apăsarea butonului *Update Data*, datele vor fi modificate în baza de date, și vom avea o listă deja actualizată, iar la apăsarea butonului *Close*, tabela și datele noastre rămân nemodificate.

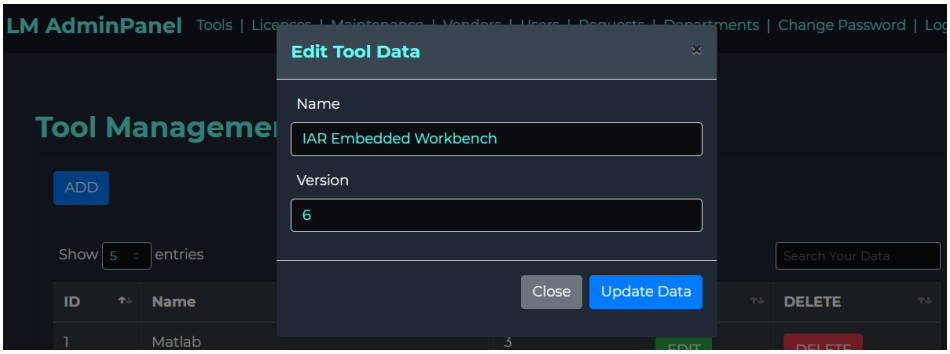


fig.29 Edit pop-up modal Tool Management

La efectuarea unui click pe butonul DELETE o fereastră de atenționare este afișată, solicitând confirmarea ștergerii, pentru a preîntâmpina ștergerea accidentală (vezi fig 30). Această fereastră este folosită pe fiecare pagină, fiindcă ștergerea este făcută după un ID, și ștergem un rând întreg din baza de date cu ID-ul selectat, doar numele tabelei se schimbă în funcția de pagină actuală.

Dacă apăsăm butonul *Yes, delete it* tool-ul selectat va fi șters din baza de date și vom avea o tabelă actualizată pe pagina noastră. Dacă alegem varianta *NO*, nu se întâmplă nimic, dispare fereastra și datele rămân așa cum au fost inițial.

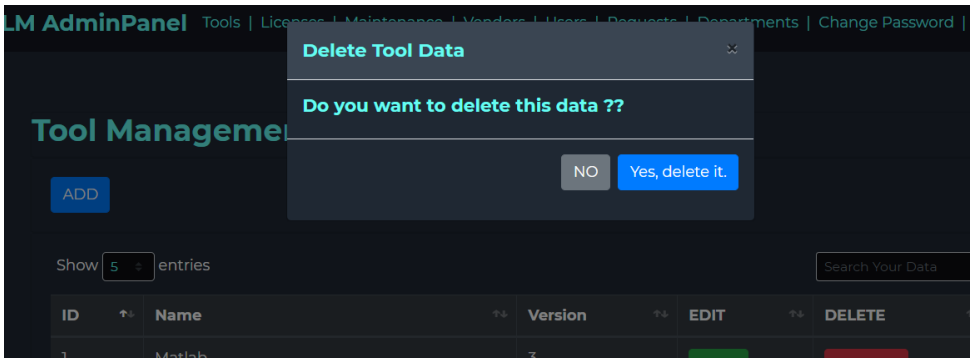


fig.30 Delete pop-up modal tool management

4.5.3 Pagina de Licenses

License Management are rol de gestionare a licențelor din baza de date. Datele sunt afișate după cum urmează: serial number, maintenance ID, username (în baza de date stocăm user ID, dar ca să avem o interfață mai plăcută și „user friendly” la afișare folosesc username-ul dobândit printr-un query SQL) adică cine folosește licența respectivă, status și type. Pagina de License Management ca și structură seamănă cu Tool Management. Și aici avem o tabelă unde datele sunt actualizate din baza de date (vezi fig.31).

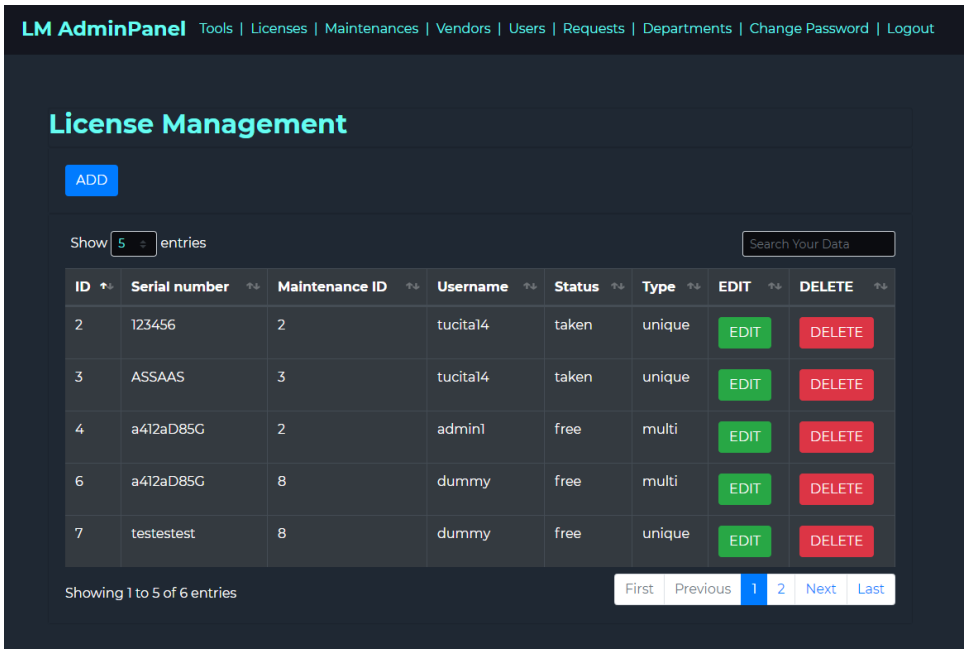


fig.31 License Management

Avem un buton de adăugare, editare și ștergere. Prin intermediul tabelei afișate, coloanele pot fi ordonate crescător și descrescător, iar din dreapta sus a tabelei avem o căsuță pentru căutare. Căutarea se poate face după orice tip de date. La adăugare (vezi fig.32) username-ul și maintenance ID-ul trebuie completate, dacă nu completăm aceste câmpuri, avem o eroare de tip pop-up (vezi fig.33).

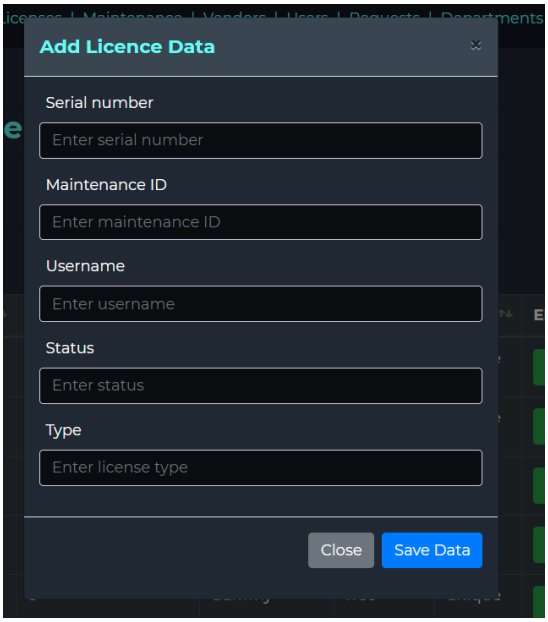


fig.32 Add pop-up modal license management

Verificările pe aceste ID-uri, în speță maintenance ID, au fost setate fiindcă implicit vom avea mentenanță pe orice licență, datorită faptului că licența a fost cumpărată la o anumită dată în trecut (vezi coloana Purchase Date din fig.35). Pentru a trece de verificarea de username, trecem userul actual, sau dacă licența este liberă se completează cu user-ul „dummy”.

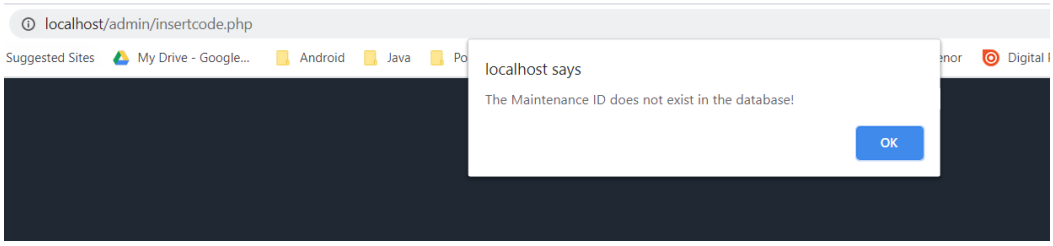


fig.33 Eroare la inserare

La editarea datelor avem un pop-up modal, unde câmpurile sunt deja completate cu datele vechi. Putem să modificăm orice câmp în afară de ID (vezi fig.34 de mai jos). La apăsarea butonului *Update Data*, datele vor fi modificate în baza de date, vom avea o listă actualizată, iar la acțiunea *Close*, tabela și datele noastre rămân nemodificate.

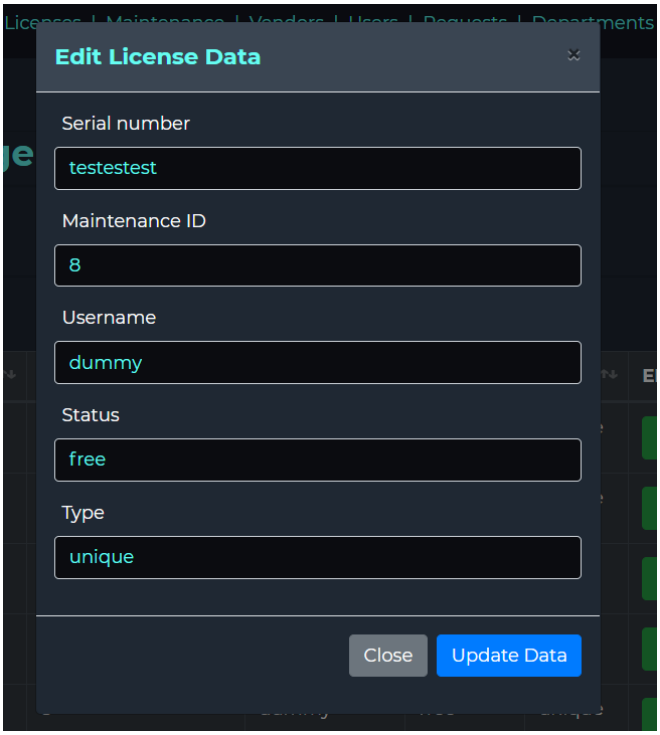


fig.34 Edit pop-up modal license management

Ștergerea datelor arată și funcționeze identic cu pop-up modal tool management, evidențiat în fig.30. Trebuie confirmat dacă sigur dorim să ștergem data respectivă. După ștergerea datei vom vedea tabela actualizată.

4.5.4 Pagina de Maintenances

Pe pagina de mentenanță sunt date legate de licențe, când au fost cumpărate (purchase date), când expiră (expiration date) și când (sau dacă) au fost reînnoite, cărui tool aparțin (în tabelă sunt trecute ID-urile, la fel și în cazul vendor-ului, dar din scopul creării unui design mai accesibil pentru user, folosim numele, din interogări SQL), versiune la care avem mentenanța respectivă și ultima coloana este numele vendor-ului de la care am achiziționat licența respectivă (vezi fig. 35).

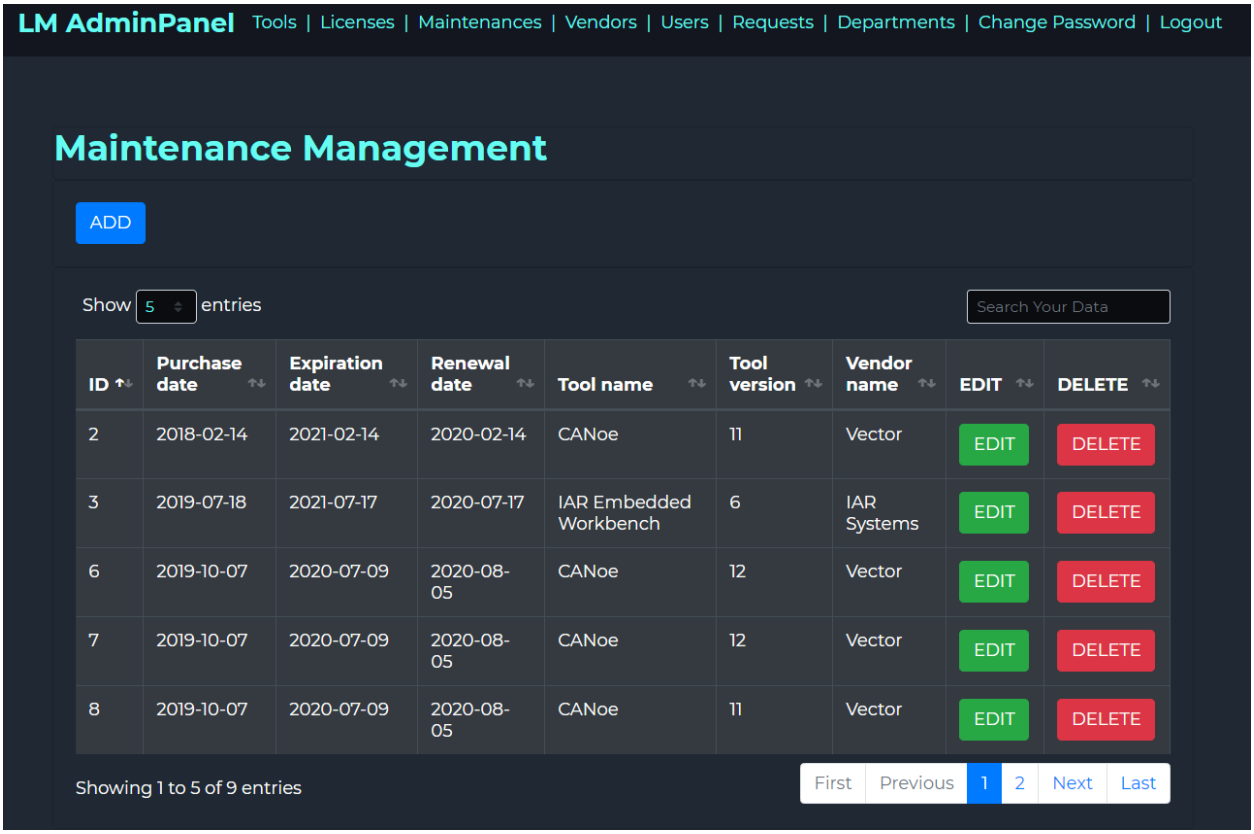


fig.35 Maintenance Management

Structura e la fel ca la orice pagină de admin. Avem următoarele butoane: adăugare, modificare și ștergere. Dispune de un search bar din dreapta sus a tabelei, în dreapta jos se află butoanele de navigare pentru paginare, în stânga jos elementele paginate și totalul, iar în stânga sus putem alege între paginare cu 5-10-20 sau toate intrările. Fiecare coloană are și opțiunea de a ordina date.

La adăugare și la editare apare un pop-up modal (vezi fig. 36) unde putem să adăugăm date sau putem să modificăm datele deja introduse, în afară de ID. Putem să salvăm datele, în cazul adăugării avem un buton *Save Data*, iar la modificare, *Update Data*. În ambele cazuri avem buton de *Close* pentru a închide fereastră de pop-up.

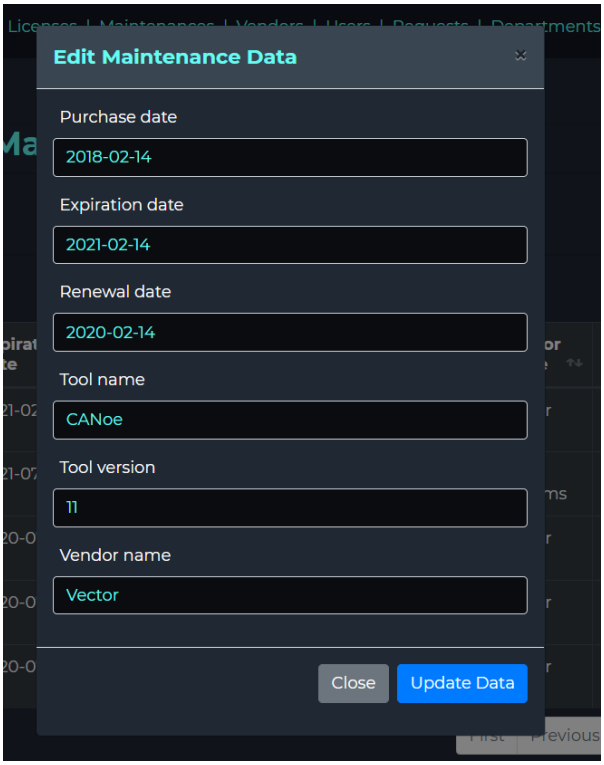


fig.36 Maintenance pop-up modal

După cum ne-am obișnuit deja, la apăsarea butonului *Delete* vom avea o fereastră de confirmare (vezi fig.30), prin care ștergem o entitate din baza de date pe baza ID-ului selectat.

4.5.5 Pagina de Vendors

Pagina Vendor Management, sugerat și de numele acesteia, servește la gestionarea datelor furnizorilor/distribuitorilor. Pe această pagină avem date referitoare la numele companiei (furnizorului), numele persoanei de contact, adresa de e-mail de contact și eventual alte de contact (telefon, pagină) (vezi fig.37). Această pagină este utilă când avem nevoie de suport de la firma respectivă, de exemplu dacă avem probleme cu o licență recent actualizată (de exemplu la o nouă versiune), sau când avem o eroare (software-ul nu recunoaște licența nouă). În această situație putem să căutăm prin această tabelă, și să contactăm vendor-ul pentru suport adițional.

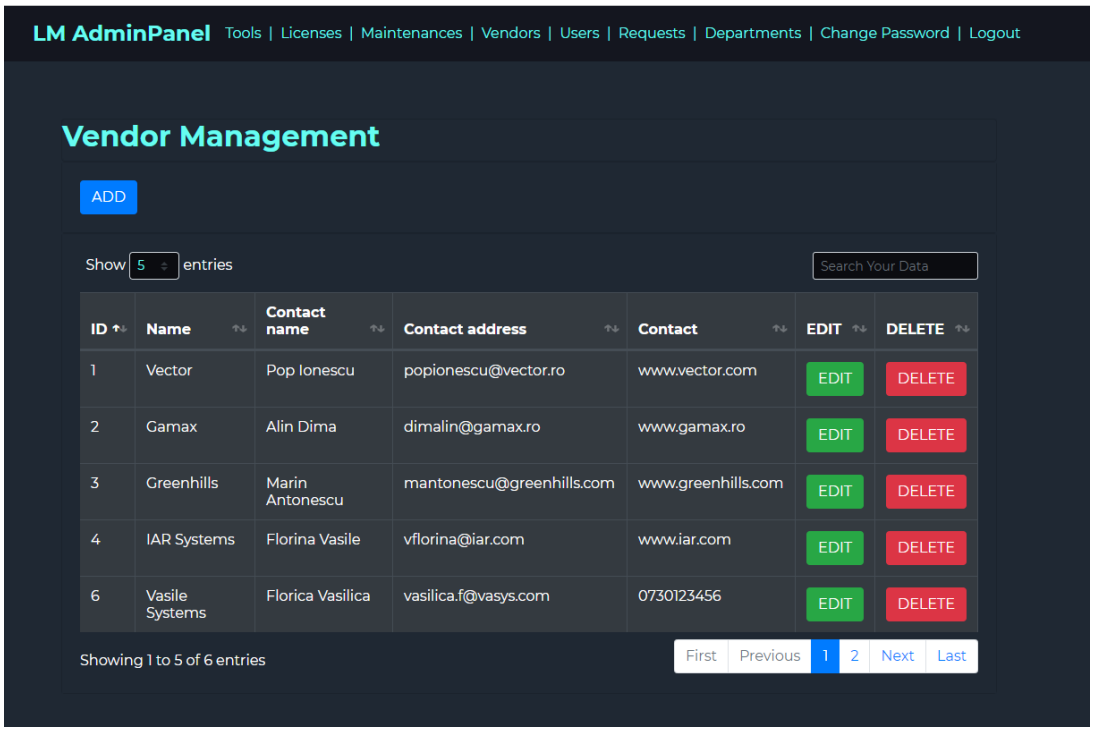


fig.37 Vendor Management

Structura paginii este deja cunoscută, butoanele și funcțiile fiind la fel ca la tabele anterioare. Adăugarea sau la modificarea se face prin o fereastră de pop-up modal (un exemplu de mai jos pentru adăugare vezi fig.38).

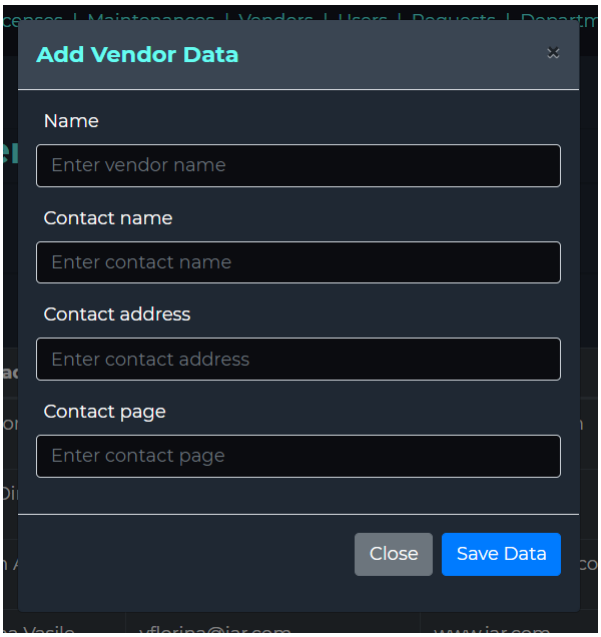


fig.38 Pop-up modal Vendor Management

Dacă dăm click pe butonul *EDIT* din dreapta rândului, vom avea o fereastră similară ca cea la adăugare, diferența va fi că datele vor fi deja completate cu date actuale din baza de date și în loc de butonul *Save Data* o să avem *Update Data*. Butoanele respective funcționează în mod așteptat, ori salvează ori modifică datele din bază. Butonul de ștergere este la fel ca la tabelele anterioare.

4.5.6 Pagina de Users

Pagina de User Management a fost creată pentru a gestiona utilizatorii aplicației License Management. După cum se vede pe poza de mai jos (fig.39) parolele sunt encriptate cu ajutorul funcției *md5()*, care returnează un șir de caracter bazate pe un algoritm (MD5 Message-Digest Algorithm).

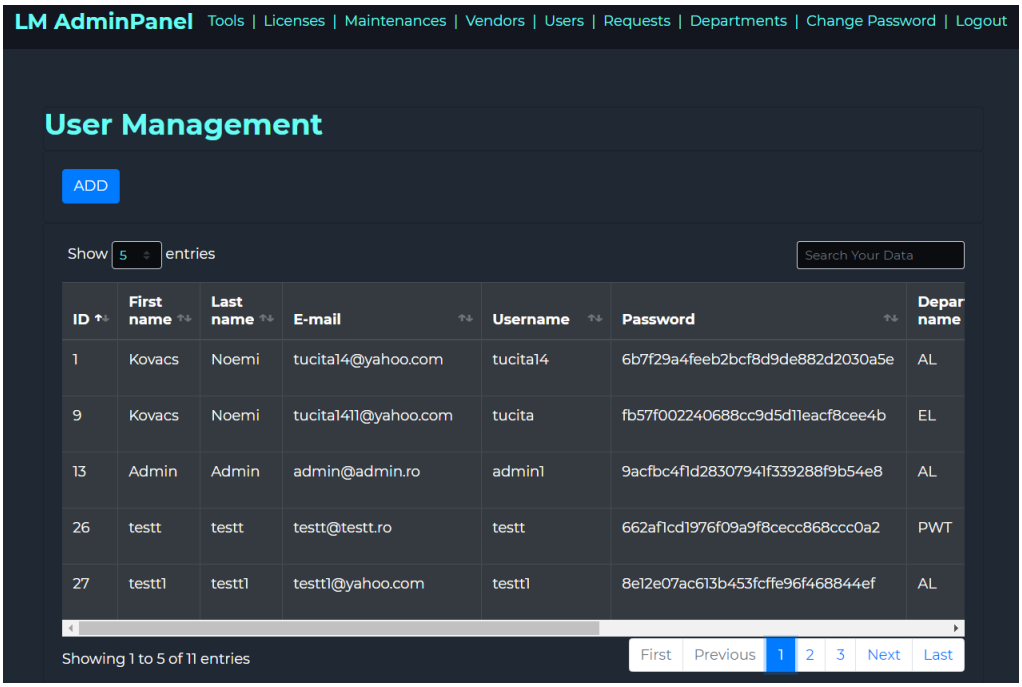


fig.39 User Management part 1

La User Management avem toate datele de înscriere după cum urmează sunt afișate: ID, prenume, nume, adresa de e-mail, username, parola, numele departamentului, ocupația, și număr de serie a laptopului (continuarea fig.39 de mai jos, fig.40).

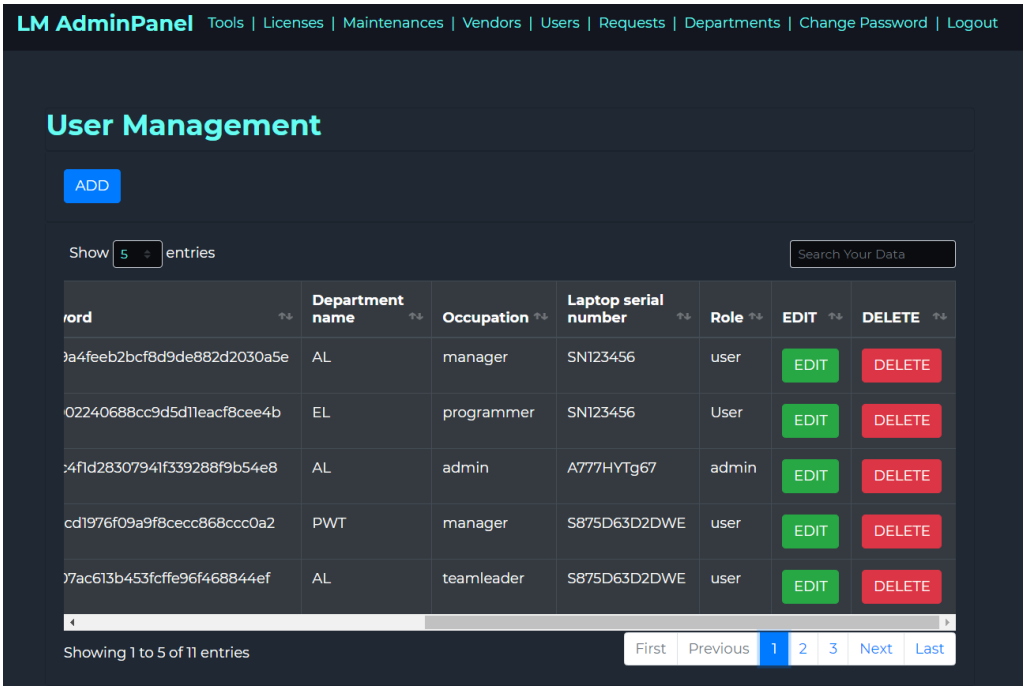


Fig.40 User Management part 2

Tabela a fost construită cu ajutorul librăriei dataTables, din care am folosit o funcție built-in (implicită), ca să avem bara de scroll orizontală, datorită faptului că avem prea multe coloane n-ar arăta frumos din punct de vedere al designului să fie afișate netrunchiat.

Ca orice tabela pe partea de admin, conține butoanele de adăugare, modificare și ștergere, iar pe partea de paginare, avem în aceleași locuri opțiunile de afișare, căutare și butoanele de navigare înainte și înapoi.

Când vrem să modificăm datele, dăm click pe butonul *EDIT* și prin pop-up modal (vezi fig. 41) putem edita datele. După cum se observă avem și un câmp pentru parola, fiind câmpul encriptat nu se vede parola. La schimbare de parolă, vom scrie parola modificată (la cererea userului) și în momentul modificării inserăm iar hash-ul parolei în bază. Aici la modificare putem să schimbăm și rolul userilor. La început la înscriere am menționat că userii în momentul înregistrării au implicit rolul *user*. Prin mediul de administrare putem să setăm orice câmp în afară de ID.

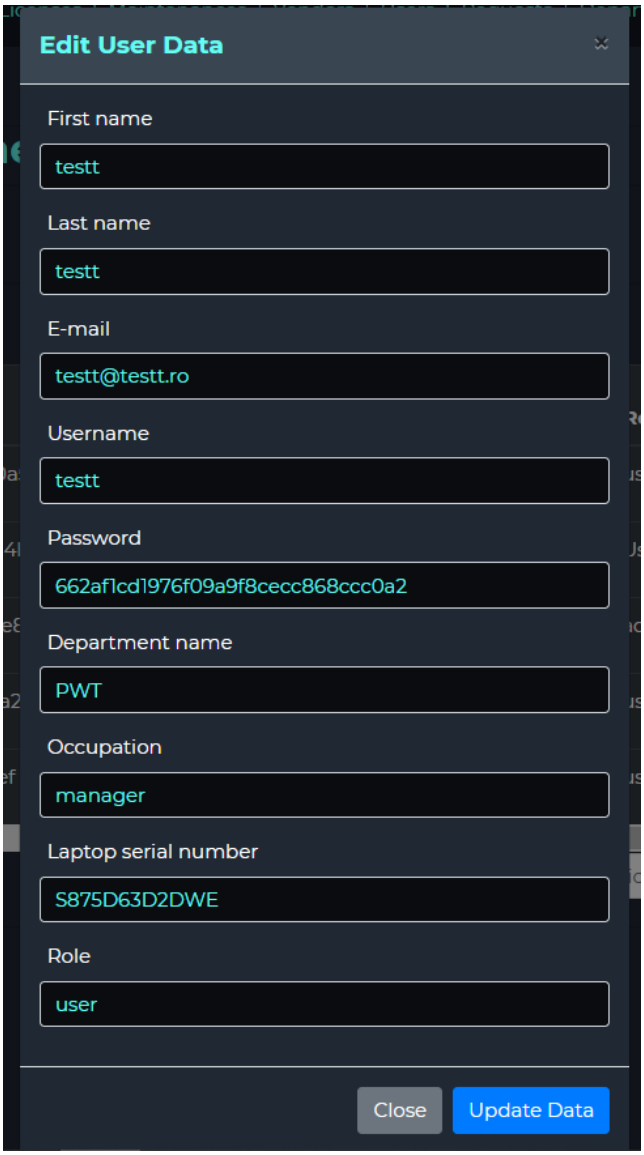


fig.41 Pop-up modal User Management

La câmpul *department name* afișăm numele departamentului, dar în spatele aplicației este stocat doar ID-ul de department, numele fiind obținut printr-o interogare SQL, deoarece astfel interfața devine mai plăcută și este mult mai ușor de reținut numele decât un ID numeric. În momentul adăugării sau modificării, verificăm department ID-ul, dacă nu există în baza de date apare o eroare pop-up (vezi fig.42).

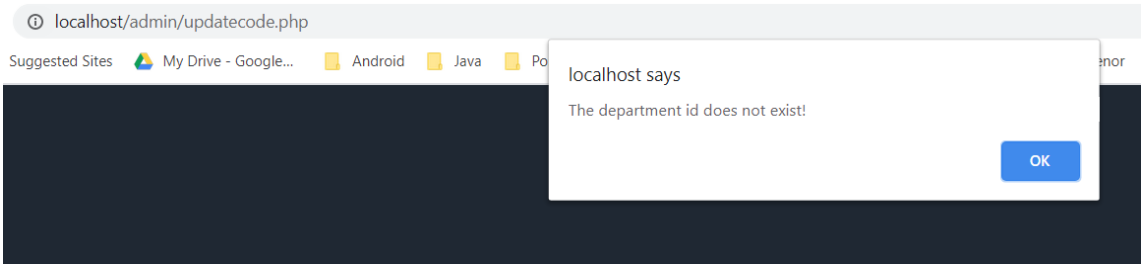


fig.42 Pop-up eroare User Management

4.5.7 Pagina de Requests

Pagina de Request Management are toate funcționalitate ca și paginile anterioare. Putem să adăugăm, să modificăm, să ștergem, să căutăm, ș.a.m.d. (vezi fig 43).

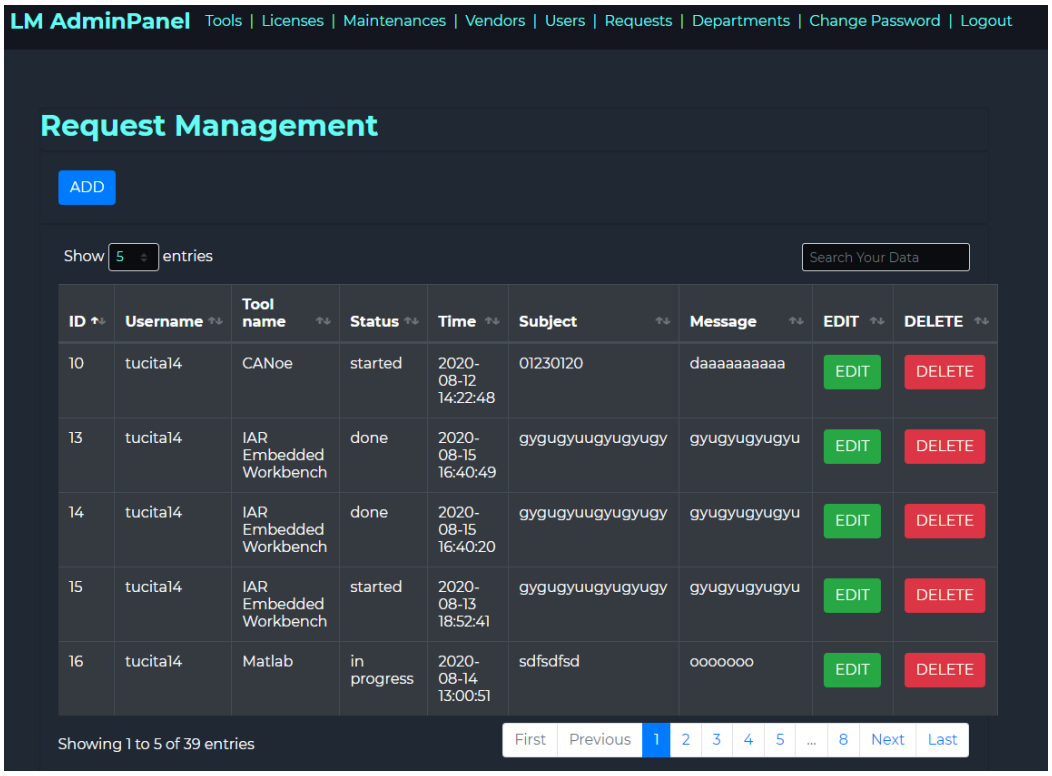


fig.43 Request Management

Această pagină e menită mai mult pentru a vizualiza edita cererile venite de la useri. Ca și la tabele anterioare avem username și tool name la afișare, în bază de date fiind stocate ca și id tool și id user. În această tabelă statusul se modifică către admin, după preluarea cererii, și după cum

am menționat la început coloana *Time* se actualizează după fiecare modificare, astfel putem urmări când a fost updatat ultima oară. La modificare, ca și la adăugare putem să modificăm doar username-ul, tool name-ul, statusul, subject-ul și eventual mesajul (vezi fig.44). Scopul acestei ferestre ar fi să modificăm doar statusul, dar permite modificarea oricărui câmp afișat.

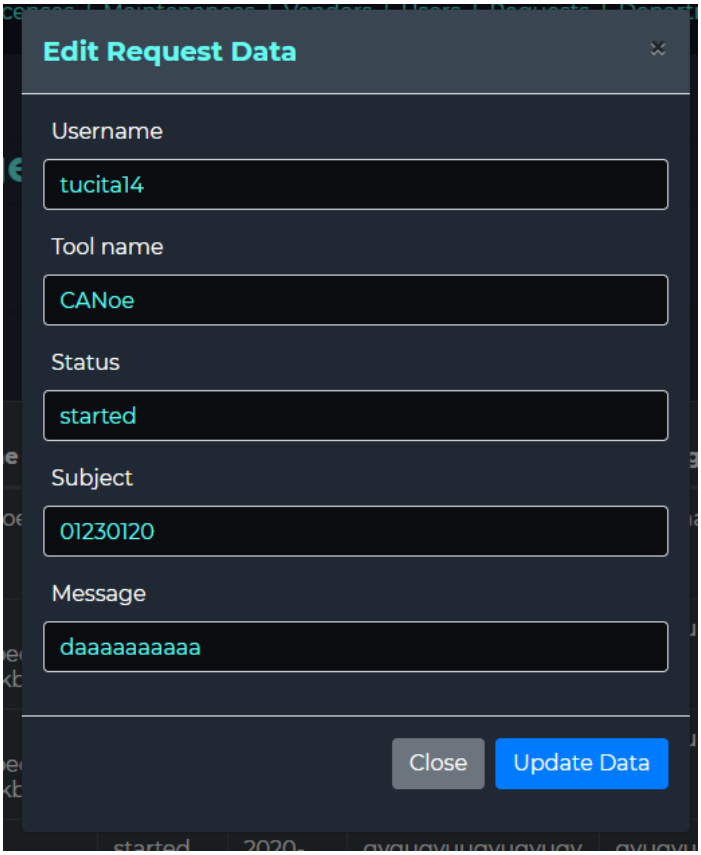


fig.44 Edit pop-up modal Request Management

4.5.8 Pagina de Departments

Pagina de Department Management a fost creat pentru a stoca date legate de departamente. Ca și design și funcționalități arată și funcționează idem ca paginile anterioare (vezi fig 45).

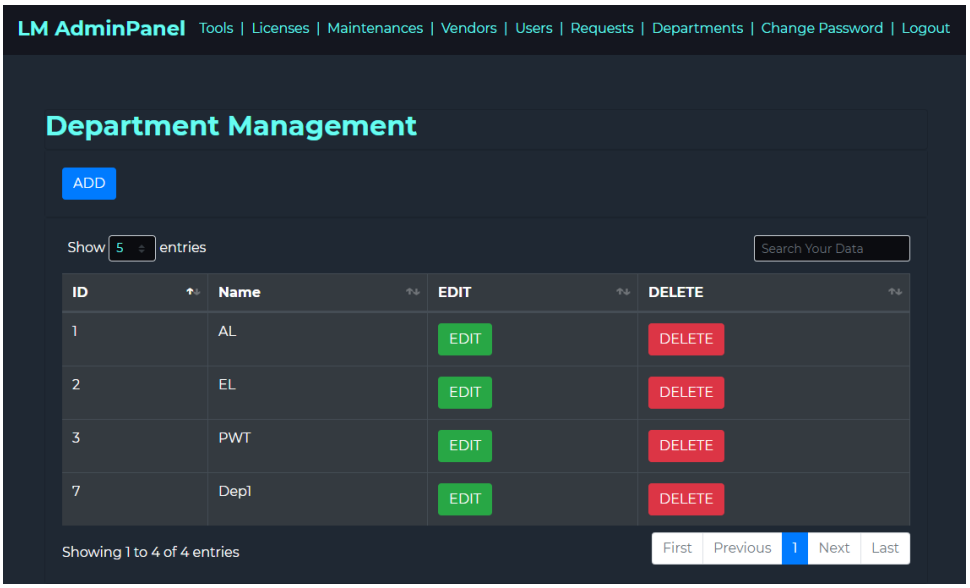


fig.45 Department Management

4.6 Hosting

Pagina https://tropiko.xyz găzduiește (hostează) aplicația mea pe internet (vezi fig.46). Am ales să public aplicația prin self-hosting.

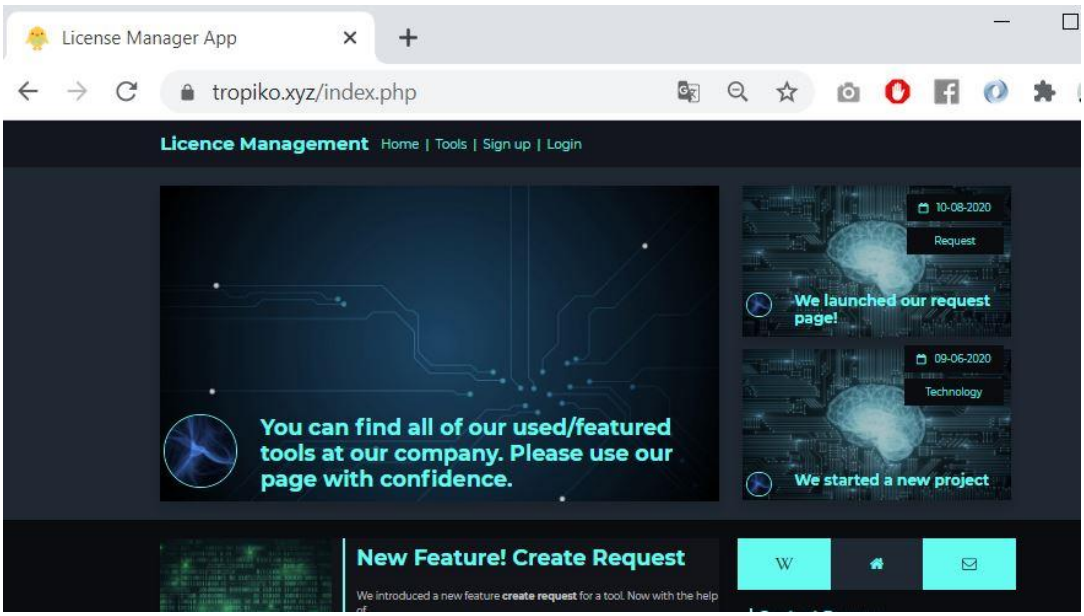


fig.46 Hosting

Am început prin a achiziționa numele de domeniu tropiko.xyz, de la un furnizor registru al domeniilor pe internet (Registrar). Am închiriat resurse hardware folosind o platforma VPS (Virtual Private Server), oferit de Vultr (11). Pe server-ul creat am instalat sistemul de operare Debian GNU/Linux versiunea 10 (Buster), kernel 4.19.0. După care am adus la zi pachetele și am început configurarea sistemului: am configurat componenta *sshd* să permită accesul SSH (Secure Shell) doar prin cheie SSL privată (Secure Sockets Layer), interzicând astfel accesul intrușilor. Apoi am început să instalez programele necesare pentru hostarea aplicației. Acestea se regăsesc în cadrul stack-ului software LEMP – Linux nginx MySQL PHP.

A urmat instalarea web server-ului nginx 1.14.2, urmată de configurarea acestuia. În acest pas am decis ca pagina să beneficieze de encripție HTTPS, un protocol securizat de a transmite date pe internet, îndeosebi important când vine vorba de parolele utilizatorilor aplicației. Pentru aceasta am folosit tool-ul *CertBot,* un set de scripturi care ajuta la instalarea HTTPS. Apoi am intrat pe pagina registrar-ului meu, și am configurat înregistrările DNS (Domain Name Server), astfel încât numele tropiko.xyz sa trimită către serverul meu. Am instalat PHP 7.3 și motorul de bază de date, MySQL 5.7.31, reconfigurând *nginx* să poată beneficia de acestea. Am ales să utilizez *sendmail* pentru mail server, pe care l-am instalat și configurat folosind numele domeniului (@tropiko.xyz), iar funcția de trimitere, *mail()* din PHP are trecut să afișeze numele celui care face un request în aplicație, la numele expeditorului în e-mailuri (vezi mai jos fig.47).

fig.47 Email primit în urma unui request al utilizatorului

În final, am încărcat codul sursă și resursele grafice ale aplicației pe server. Am restaurat baza de date pe server, folosind comanda din terminal: *mysql –u root –p lmapp < export_baza.sql*, totodată securizând-o cu o parolă root.

Am testat funcționalitatea paginii cu succes, am comparat cu mediul de dezvoltare și nu am descoperit nici o neconcordanță.

Partea a V - a: Concluzii

După estimările generale aproximativ 10-20% din costurile de software IT vin din lipsa controlului adecvat al gestionării licențelor în cadrul firmei. Căutând soluții, putem economisi o parte considerabilă din costurile software-ului achiziționat și utilizat anual, datorită faptului că firmele acordă mai multă atenție și sunt în ce în ce mai prudenți legat de costurile și bugetele de IT.

Folosirea aplicațiilor de gestionare de licențe au scos în evidență că ne ajută sa economisim bani, timp și uneori resurse umane. Indiferent dacă vorbim despre o întreprindere mică sau o firmă masivă cu întindere globală, gestionarea puternică a activelor este esențială pentru afacerea și bugetul companiei.

Scrierea acestei lucrări m-a ajutat să învăț și sa îmi structurez gândirea într-un mod mai pragmatic, ajutându-mă să rezolv o problemă reală, de la locul de muncă.

Realizarea acestei aplicații mi-a adus o mai bună înțelegere asupra modului de folosire a limbajului PHP, HTML și JavaScript pentru a dezvolta o aplicație completă de gestionarea licențelor de tip web.

La începutul dezvoltării aplicației, fiecare validare am scris-o pe pagina respectivă, unde am avut nevoia de validări (de exemplu la *login.php, sign-up.php*), iar când am ajuns la partea de admin am implementat o variantă mult mai elegantă. Toate adăugările pe paginile de admin le-am făcut separat într-o pagină, pe care am denumit-o *insertcode.php*. Această pagină este dedicată doar pentru adăugări, care sunt clar și ușor de urmărit. Totodată aici se verifică și ID-urile să fie corecte (de exemplu la tabela *request* unde afișăm username și tool name, iar în spatele aplicației de fapt verificăm ID-urile). Tot în această manieră am procedat și pentru acțiunile de editare și ștergere, care folosesc fiecare câte o pagină separată (*deletecode.php, updatecode.php)*, unde prin interogări SQL, fac update sau unde este cazul șterg elemente din baza de date.

Consider că această aplicație poate deveni mult mai complexă și își poate largi gama de funcționalități pe care le acoperă.

Posibilități de dezvoltare pentru aplicația License Management:

- Implementarea logării prin Windows, cu Active Directory.
- În momentele în care întâlnim licențe de tip multi user, să avem o listă în care să avem toți userii care folosesc licența respectivă.
- Să existe separat o tabelă pentru gestionarea activelor hardware din companie (laptop-urile utilizatorilor, legate prin cheia *laptopsn*, din tabela *user*).
- Dezvoltarea unei aplicații mobile.
- Înlocuirea stack-ului LEMP cu Windows Server, de asemenea și sendmail să fie înlocuit cu Microsoft Exchange, pentru o integrare mai bună și experiență user-friendly.
- Completarea implicită cu username-ul curent pentru pagina de request.
- Adăugarea implicită a userului "dummy" în momentul inserării unei licențe noi, dacă nu este completat câmpul *username.*
- Restricționarea câmpurilor editabile din pagina de Request Management, astfel încât să fie editabil doar statusul cererii.
- Whitelistarea serverului de e-mail încât să nu fie filtrate mesajele în cutiuța de Spam.

Partea a VI – a: Bibliografie

1. https://github.com/ChangemakerStudios/Papercut-SMTP/releases
2. https://www.apachefriends.org/index.html
3. https://ocw.cs.pub.ro/courses/eim/laboratoare/laborator07
4. http://twss.teaching.ro/
5. Manual oficial PHP
6. http://itee.elth.pub.ro/~vbucata/ia/laboratoare/modul2-programare-web/01-introducere.php
7. https://web.ceiti.md/lesson.php?id=16
8. https://sinups.net/javascript/despre-javascript/
9. https://bursasite.ro/ce-este-mysql/
10. https://datatables.net/
11. https://www.vultr.com/

- https://en.wikipedia.org/wiki/Software_license
- https://www.bmc.com/blogs/software-license-management/