**UNIVERSITATEA ALEXANDRU IOAN CUZA IAŞI**

**FACULTATEA DE INFORMATICĂ**



LUCRARE DE LICENŢĂ

**DOC Web**

propusă de

Plugariu Raluca Nicoleta

**Sesiunea**: Iulie, 2019

Coordonator ştiinţific

***Lect. Dr. Frasinaru Cristian***

**DECLARAŢIE PRIVIND ORIGINALITATEA ŞI RESPECTAREA DREPTURILOR DE AUTOR**

Prin prezenta declar că Lucrarea de licenţă cu titlul „DOC Web” este scrisă de mine şi nu a mai fost prezentată niciodată la o altă facultate sau instituţie de învăţământ superior din ţară sau străinătate. De asemenea, declar că toate sursele utilizate, inclusiv cele preluate de pe Internet, sunt indicate în lucrare, cu respectarea regulilor de evitare a plagiatului:

– toate fragmentele de text reproduse exact, chiar şi în traducere proprie din altă limbă, sunt scrise între ghilimele şi deţin referinţa precisă a sursei;

– reformularea în cuvinte proprii a textelor scrise de către alţi autori deţine referinţa precisă;

– codul sursă, imagini etc. preluate din proiecte open-source sau alte surse sunt utilizate cu respectarea drepturilor de autor şi deţin referinţe precise;

– rezumarea ideilor altor autori precizează referinţa precisă la textul original.

Iaşi,

Absolvent Plugariu Raluca Nicoleta \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**DECLARAŢIE DE CONSIMŢĂMÂNT**

Prin prezenta declar că sunt de acord ca Lucrarea de licenţă cu titlul „DOC Web”, codul sursă al programelor şi celelalte conţinuturi (grafice, multimedia, date de test etc.) care însoţesc această lucrare să fie utilizate în cadrul Facultăţii de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea Alexandru Ioan Cuza Iaşi să utilizeze, modifice, reproducă şi să distribuie în scopuri necomerciale programele-calculator, format executabil şi sursă, realizate de mine în cadrul prezentei lucrări de licenţă.

Iaşi,

Absolvent Plugariu Raluca Nicoleta

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Acord privind proprietatea dreptului de autor**

Facultatea de Informatică este de acord ca drepturile de autor asupra programele-calculator,

format executabil și sursă, să aparțină autorului prezentei lucrări, Plugariu Raluca Nicoleta

Închierea acestui acord este necesară din următoarele motive:

Decan Adrian Iftene Absolvent Plugariu Raluca Nicoleta

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Cuprins**

[1.Introducere](file:///C:\Users\raluca.plugariu\Desktop\Cosmin%20Cazacu%20Licenta.docx#_Toc486605826)

[2.Contribuții](file:///C:\Users\raluca.plugariu\Desktop\Cosmin%20Cazacu%20Licenta.docx#_Toc486605827)

[3. Aplicații similare](file:///C:\Users\raluca.plugariu\Desktop\Cosmin%20Cazacu%20Licenta.docx#_Toc486605828)

[4.Tehnologii utilizate](file:///C:\Users\raluca.plugariu\Desktop\Cosmin%20Cazacu%20Licenta.docx#_Toc486605829)

[4.1 Java](file:///C:\Users\raluca.plugariu\Desktop\Cosmin%20Cazacu%20Licenta.docx#_Toc486605830)

[4.2 Spring](file:///C:\Users\raluca.plugariu\Desktop\Cosmin%20Cazacu%20Licenta.docx#_Toc486605831)

[4.3 Java Server Faces (JSF)](file:///C:\Users\raluca.plugariu\Desktop\Cosmin%20Cazacu%20Licenta.docx#_Toc486605832)

[4.4 H2 Database](file:///C:\Users\raluca.plugariu\Desktop\Cosmin%20Cazacu%20Licenta.docx#_Toc486605833)

[4.5 Maven](file:///C:\Users\raluca.plugariu\Desktop\Cosmin%20Cazacu%20Licenta.docx#_Toc486605834)

[5. Analiză și proiectare](file:///C:\Users\raluca.plugariu\Desktop\Cosmin%20Cazacu%20Licenta.docx#_Toc486605837)

[5.1 Diagrama Use Case](file:///C:\Users\raluca.plugariu\Desktop\Cosmin%20Cazacu%20Licenta.docx#_Toc486605838)

[5.2 Diagrama de entități](file:///C:\Users\raluca.plugariu\Desktop\Cosmin%20Cazacu%20Licenta.docx#_Toc486605839)

[5.3 Arhitectura](file:///C:\Users\raluca.plugariu\Desktop\Cosmin%20Cazacu%20Licenta.docx#_Toc486605840)

[5.5 Structura bazei de date](file:///C:\Users\raluca.plugariu\Desktop\Cosmin%20Cazacu%20Licenta.docx#_Toc486605841)

[5.4 Structura Aplicației](file:///C:\Users\raluca.plugariu\Desktop\Cosmin%20Cazacu%20Licenta.docx#_Toc486605842)

[5.5 Detalii de implementare](file:///C:\Users\raluca.plugariu\Desktop\Cosmin%20Cazacu%20Licenta.docx#_Toc486605843)

[6. Modul de utilizare al aplicației](file:///C:\Users\raluca.plugariu\Desktop\Cosmin%20Cazacu%20Licenta.docx#_Toc486605844)

[6.1 Rolul Doctor](file:///C:\Users\raluca.plugariu\Desktop\Cosmin%20Cazacu%20Licenta.docx#_Toc486605845)

[6.2 Rolul Pacient](file:///C:\Users\raluca.plugariu\Desktop\Cosmin%20Cazacu%20Licenta.docx#_Toc486605846)

[7. Concluzii și direcții viitoare](file:///C:\Users\raluca.plugariu\Desktop\Cosmin%20Cazacu%20Licenta.docx#_Toc486605848)

[Blibliografie](file:///C:\Users\raluca.plugariu\Desktop\Cosmin%20Cazacu%20Licenta.docx#_Toc486605849)

1.Introducere

Lucrarea de față prezintă creearea unei aplicații menită să ușureze obtinerea interactiunii dintre un pacient si un doctor, precum si organizarea si monitorizarea pacientilor si a programarilor, din perspectiva unui doctor.

Ideea proiectului a venit odata cu necesitatea mentinerii unei bune conexiuni intre aceste doua entitati, doctor si pacient. Din experienta personala, am observat ca necesita mult timp investit in gasirea unui doctor de care noi am avea nevoie, care sa indeplineasca anumite criterii dupa care noi cautam, si nu in ultimul rand, disponibilitatea acestuia in vederea programarii unei vizite. Desigur ca aceste functionalitati descrise mai sus, pot fi gasite deja in diferite platforme, cum ar fi cele oferite de institutiile medicale, dar ideea acestui proiect este le aduca impreuna intr-o singura platforma.

Este o aplicatie care se axeaza mai mult pe nevoile unui pacient, deoarece acesta poate gasi un doctor in functie de recenziile pe care acesta le detine, provenind de la alti utilizatori, dupa locatia in care se afla, iar acesta este un lucru foarte important pentru persoanele care se afla intr-o zona sau oras in care nu detin informatii legate de reteaua de medici, si nu in ultimul rand, pacientii primesc informatii legate de programul doctorilor si intervalele libere pentru consultatii.

Din perspectiva unui doctor, cel mai mare avantaj pe care il aduce aceasta aplicatie este ca nu trebuie sa apartii unei anumite institutii pentru a avea un profil aici, ceea ce ajuta la o vizibilitate mult mai buna pe piata. Cum spuneam, platformele deja existente, sunt creeate de institutii medicale, in mare parte din mediul privat. Exemplul concret din Iasi este Arcadia, care pune la dispozitie profilul doctorilor din cadrul institutiei. Dar nu multe sunt acestea, drept urmare, exista in Iasi doctori foarte bine pregatiti dar care nu sunt vizibili publicului larg. Pe langa acest aspect, in cadrul aplicatiei doctorul isi poate gestiona lista pacientilor si a calendarului si poate raspunde la interactiunile cu pacientul : accepta sau refuza cereri de programari.

2.Contribuții

Platforma este un produs destinat doctorilor din orice domeniu, dar si publicului larg care are nevoia de a interactiona cu acestia.

Logica aplicației este concepută în totalitate de mine, ea oferă câte o interfața pentru fiecare tip de user – doctor , pacient. Acestea au fost creeate utilizât framework-uri precum Primefaces și Bootfaces, iar template-ul general este inspirat din produsul gratuit AdminLTE 2, o interfață bazată în mare parte pe Bootstrap, ce oferă utilizatorilor o experiență plăcută.

3. Aplicații similare

Acest capitol propune o analiză sumară a unor soluții similare existente pe piață, scoțând în evidență diferite abordări pentru acest tip de aplicație.

3.1 Clickdoc

3.2 Doctor On Demand

3.3 Online Doctor

**4.Tehnologii utilizate**

**4.1 Java**

“Java este o tehnologie inovatoare lansată de compania Sun Microsystems în 1995, care a avut un impact remarcabil asupra îıntregii comunități¸ a dezvoltatorilor de software, impunându-se prin calități deosebite cum ar fi simplitate, robustețe și nu în ultimul rând portabilitate. Denumită initțial OAK, tehnologia Java este formată dintr-un limbaj de programare de nivel înalt pe baza căruia sunt construite o serie de platforme destinate implementării de aplicații pentru toate segmentele industriei software.”

*API (Application Programming Interface)*

Un API este un set de reguli folosit pentru comunicarea dintre o aplicație cu o altă aplicație. API este de multe ori corelat cu o librărie software, primul descrie specificațiile, iar cel din urmă descrie implementarea acestora. Un lucru important de menționat este că atunci când trebuie să folosim un API, nu trebuie să cunoaștem nici un detaliu tehnic cu privire la modul în care a fost scris.

*J2EE (Enterprise Edition)*

J2EE este o platformă Java ce pune la dispoziție o colecție de API-uri, care vine în ajutorul dezvoltatorilor pentru a creea aplicații server-side, deținută în momentul actual de Oracle. Această platformă conține elemente ce pot fi împărțite în 2 categorii:

1. Componente :

* Servlets
* JSP (Java Server Pages)
* EJB (Enterpise Java Beans)

1. Servicii/API

* JMS (Java Message Service)
* JTA (Java Transaction API)
* JAAS (Java Authentification and Autheriisation Service)
* JNDI (Java Naming and Directory Interface)
* Java Mail API
* JAX-WS (Java API for XML Web Services)
* JAXB (Java Arhitecture for XML Binding)
* SOAP
* JPA (Java Persistence API)
* JSF (Java Server Faces)

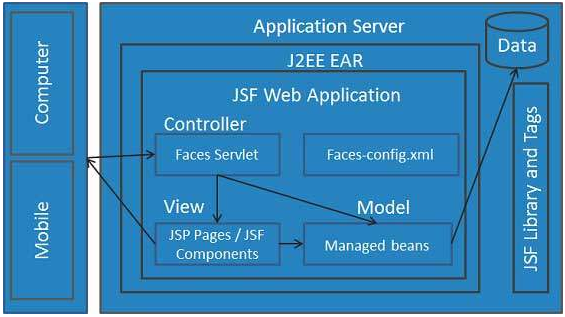
**4.2 Java Server Faces (JSF)**

JSF este un framework pentru aplicațiile Web ce intenționează să simplifice dezvoltarea interfețelor utilizatorului. Specificațiile JSF definesc un set standard de componente UI (User Interface) și oferă un API (Application Programming Interface) pentru dezvoltarea lor.

JSF reduce efortul pentru crearea si mententenanța aplicațiilor ce vor rula pe partea de server. Câteva dintre facilitățile pe care această tehnologie le oferă sunt:

* Furnizează componente User Interface reutilizabile.
* Face posibil tranferul de date rapid între componentele User Interface.
* Manageriază stările User Interface între multiple cereri către server.
* Permite implementarea componentelor proprii.

Tehnologia JSF este bazată pe arhitectura MVC (Model View Controller) cu scopul de a separa partea logică a aplicației de partea de prezentare.



**4.3 Spring Framework**

Spring este un framework pentru dezvoltarea aplicațiilor Java, mult mai ușor de folosit in comparatie cu modul de dezvoltare prevăzut în folosirea Java2EE. Acest framework utilizează diverse tehnici noi precum Aspect-Oriented-Programming (AOP), injecția de dependință (DI) sau Plain-Old-Java-Object (POJO).  Se axează în principal pe furnizarea de diverse modalități de a a ajuta la gestiunea bussines object-urilor. Dezvoltarea aplicațiilor Web este mult mai ușoară în comparație cu cadrul clasic Java și interfețele de programare a aplicațiilor (API), cum ar fi conectivitatea bazei de date Java (JDBC), paginile JavaServer (JSP) și Java Servlet.

Poate fi considerat ca o colecție de submodule sau layere, cum ar fi Spring AOP, Spring Object-Relational Mapping (Spring ORM), Spring Web Flow si Spring Web MVC, care pot fi folosite împreună pentru a furniza toate functionalitațile necesare în dezvoltarea unei aplicații web.

Cele mai importante elemente din cadrul Spring sunt:

* **Containerul IoC:**   
  Este containerul de bază care utilizează modelul DI sau IoC pentru a furniza implicit o referință de obiect într-o clasă în timpul rulării. Acest model acționează ca o alternativă la modelul de localizare a serviciului.
* **Data access:**   
  Permite dezvoltatorilor să utilizeze API-uri persistente, cum ar fi JDBC și Hibernate, pentru stocarea datelor de persistență în baza de date.
* **Spring MVC:**   
  Are la bază arhitectura MVC. Toate solicitările făcute de un utilizator parcurg mai întâi controlerul și apoi sunt expediate către diferite vizualizări, adică către diferite pagini JSP sau servlets.
* **Gestionarea tranzacțiilor:**   
  Ajută la gestionarea tranzacțiilor unei aplicații fără a afecta codul acesteia. Are la bază Java Transaction API (JTA) pentru tranzacțiile globale gestionate de un server de aplicații și tranzacțiile locale gestionate utilizând JDBC Hibernate, Java Data Objects (JDO) sau alte API-uri de acces la date.
* **Serviciul Web:**   
  Generează endpointurile si definițiile serviciului web bazate pe clase Java, dar este dificil să le gestionați într-o aplicație. Pentru a rezolva această problemă, serviciul Spring Web oferă abordări bazate pe straturi care sunt gestionate separat de parsarea Extensible Markup Language (XML) (tehnica de citire și manipulare a XML-ului).
* **Nivelul de abstractizare JDBC:**   
  Ajută utilizatorii să gestioneze erorile într-un mod ușor și eficient. Codul de programare JDBC poate fi redus când acest strat de abstractizare este implementat într-o aplicație Web. Acest strat gestionează excepții, cum ar fi DriverNotFound.

**4.4 H2 Database**

H2 este o bază de date open source bazată pe Java. Acesta poate fi încorporat în aplicații Java sau poate fi executat în modul client-server. În principal, baza de date H2 poate fi configurată să funcționeze ca bază de date in memory sau pe disc.

Această bază de date poate fi utilizată în modul embedded sau în modul server.

 Principalele caracteristici ale bazei de date H2 :

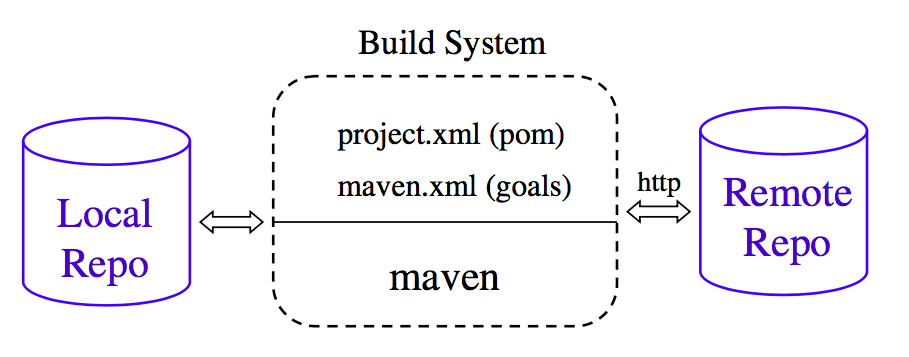
* Extrem de rapid, open source, JDBC API
* Disponibil în modurile embedded și server; în baze de date în memorie
* Consola bazată pe browser
* Amprentă mică - dimensiune de fișier de aproximativ 1,5 MB

**4.5 Maven**

Maven este o aplicație de tip open-source oferită de Apache menită să ușureze management-ul proiectului, automatizând infrastructura de compilare. Timpul de lucru scade, iar odată cu acesta costul de producție al unui produs software se micșorează. Acest lucru este realizat prin crearea unei structuri standard de directoare și un ciclu de viață prestabilit pentru compilare.

În esență, Maven este util și adeseori folosit în proiecte de dimensiuni mari cu multiple submodule, deoarece reușește să compileze proiectul ca un tot unitar, oferind în mod automat informații despre proiect și ajutând echipele de programatori să colaboreaze la același proiect.

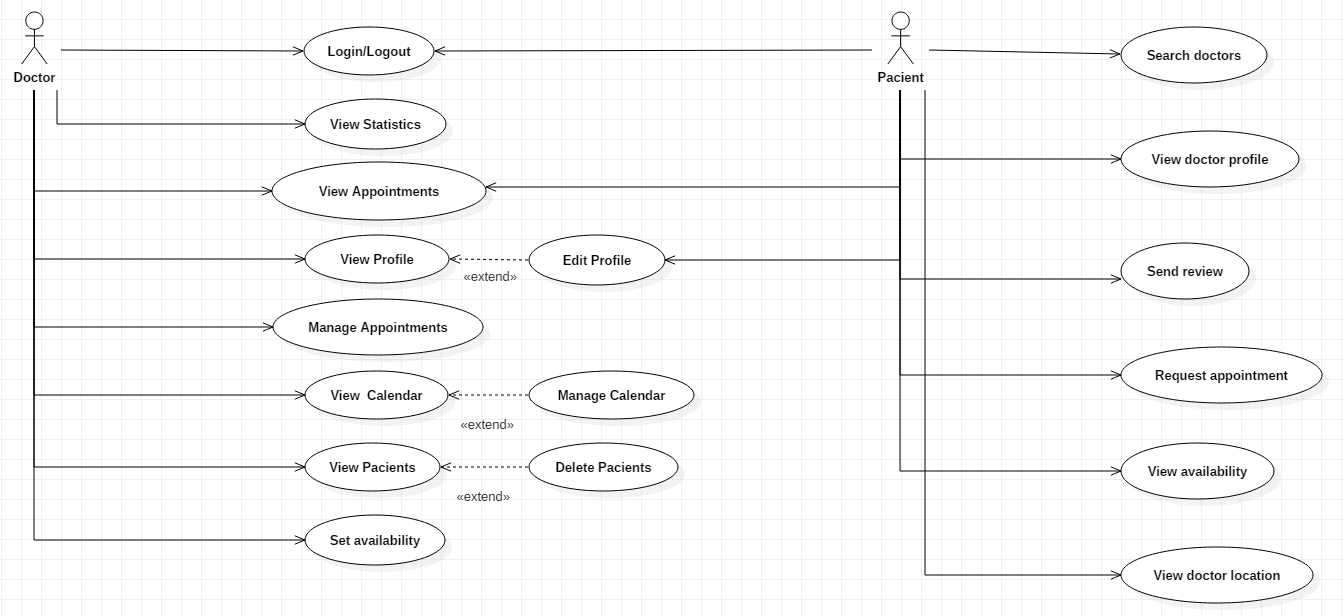
Structura proiectului este definită într-un fișier descriptiv unic, numit pom.xml, denumirea sa făcând referire la Project Object Model (POM). Acesta conține diferite informații, printre care detaliile ce țin de configurare, dependențe, informații despre resursele de testare, membrii echipei, etc. Dependențele sunt folosite pentru a importa automat librăriile necesare rulării.



**5. Analiza si proiectare**

Aplicația DOCWeb își propune să ofere utilizatorilor o platforma cat mai simpla si usor de accesat, in vederea interactiunii dintre doctori si pacienti, precum si sisteme de gestionare a unor diferite entitati.

**5.1 Diagrama Use Case**

****

Conform diagramei putem deduce că aplicația ar trebui să ofere:

* Autentificarea și autorizarea în funcție de rolul utilizatorului în aplicație

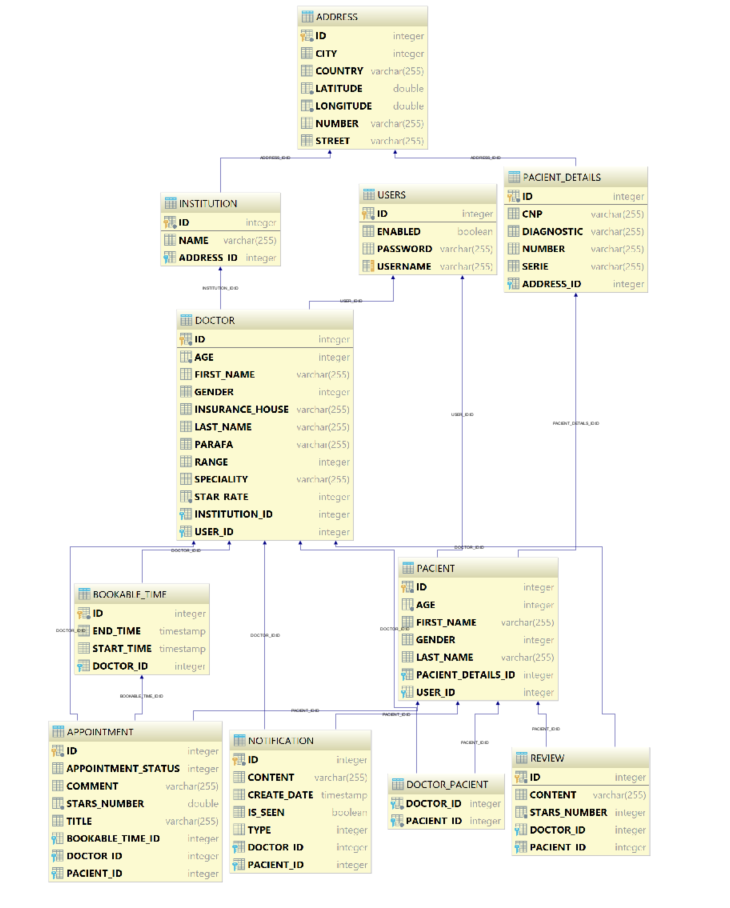
În cazul în care utilizatorul logat are rolul de doctor poate să

* Afiseze/Editeze programarile
* Stearga o programare
* Afiseze/Editeze profilul
* Afiseze calendarul
* Seteze orele si zilele la care este disponibil pentru programari
* Afiseze/Stearga pacientii
* Editeze calendarul
* Accepta/Refuza o programare
* Primeasca notificari in privinta cererilor de programare dar si pentru adaugarea unui nou review pe profilul sau

În cazul în care utilizatorul logat are rolul pacient poate să

* Afiseze programarile sale
* Caute un doctor in functie de locatie, scor si specializare
* Afiseze profilul doctorului
* Afiseze orele la care doctorul este disponibili pentru o programare
* Adauge un review pe profilul doctorului
* Trimite o cerere pentru o programare la o anumita ora
* Afiseze/Editeze profilul
* Primeasca notificari in privinta editarii unei programari deja stabilite

**5.2 Diagrama de entitati**

****

Conform diagramei putem deduce ca:

* Un doctor are mai multi pacienti
* Un pacient are mai multi doctori
* Un doctor are mai multe notificari
* Un doctor are mai multe programari
* Un doctor are mai multe review-uri
* Un doctor are mai multe intervale de timp destinate programarilor
* Un pacient are mai multe programari
* Un pacient are mai multe review-uri trimise doctorilor
* O programare are asignat un doctor si un pacient

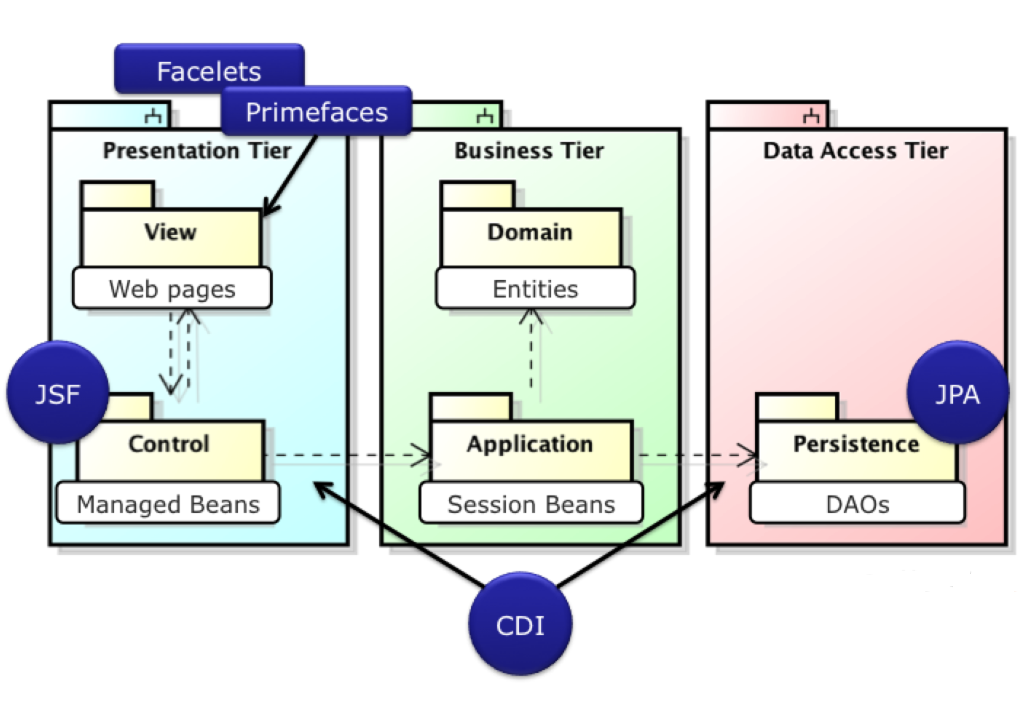
**5.3 Arhitectura**

DOCWeb este o aplicație bazată pe o arhitectură structurată pe 3 nivele :

1.Nivelul de prezentare sau client- conține elemente JSP, Servlet și framework-ul JSF, un framework bazat pe design patternul MVC – Model View Controller, modelul fiind constituit din Managed Beans, view-ul îl reprezintă pagini xhtml cu componente Primefaces, iar controller-ul este reprezentat de Faces Servlet, ce se ocupa de preluarea cererilor utilizatorului si rezolvarea acestora

2.Nivelul de mijloc denumit și nivelul logic – este alcătuit din servicii EJB ce mențin logica aplicației și entități ce constituie modelul aplicației.

3.Nivelul de date sau EIS (Enterprise Information Systems) – gestionează conexiunea dintre entități și baza de date. Persistența datelor este făcută de Hibernate - o implementare a interfețelor oferite de JPA.



**5.4 Structura bazei de date**

Baza de date