1 Mecanisme de dezvoltare a produselor software

1.1 Managementul proiectelor

Managementul modern al proiectelor software este o abordare relativ nouă în cadrul organizațiilor și oferă instrumente puternice prin care acestea îsi îmbunatațesc abilitățile de planificare, aplicare și control, precum și modul de utilizare al oamenilor și ale celorlalte resurse. Ultimele decenii au fost marcate de creșterea rapidă a utilizarii managementului proiectelor ca mijloc prin care organizațiile își ating obiectivele. Managementul proiectelor a apărut ca urmare a tendințelor din ultimele decenii legate de: creșterea exponențială a cunoștințelor și informației, cererea mare pentru servicii și bunuri personalizate, complexe si sofisticate, evoluția piețelor competitive, etc. [1]

Complexitatea crescândă a problemelor puse în fața proiectelor și creșterea rapidă a numărului organizațiilor orientate pe proiecte a contribuit la profesionalizarea managementului proiectelor. În 1969 s-a înființat Institutul pentru Managementul Proiectelor și a atins în anii ’90 aproximativ 10.000 de membri.

Un proiect este constituit dintr-un ansamblu de persoane şi alte resurse grupate temporar pentru realizarea unui anumit obiectiv, de regulă într-o perioadă de timp dată și cu utilizarea unui buget fixat. Astfel, un lucru foarte important în managementul unui proiect este managementul resurselor, și mai ales al resurselor umane.

În majoritatea marilor companii, managementul angajaților se face prin gruparea acestora pe echipe în funcție de cunoștințe, interese, studii, aptitudini și în funcție de proiectul pe care vor fi asignați. Echipele sunt în mod special propice pentru realizarea unor sarcini cu complexitate crescută şi care au mai multe sub-sarcini. Ele nu sunt izolate, ci interacţionează atât cu alte echipe din organizaţie cât şi cu organizaţia în sine.

1.2 Metodologia Agile

Agile este o metodologie de management al proiectelor ce încearcă să micşoreze riscurile de dezvoltare şi timpul de execuţie prin implementarea proiectelor în formă foarte flexibilă şi interactivă. Principala diferență între dezvoltarea agilă și iteractivă constă în faptul că metodele agile finalizează porțiuni mici ale rezultatelor în fiecare ciclu de livrare, în timp ce metodele iterative evoluează întregul set de rezultate în timp, completându-le aproape de sfârșitul proiectului. Ambele metode iterative și agile au fost dezvoltate ca o reacție la diferitele obstacole care s-au dezvoltat în forme secvențiale de organizare a proiectului. De exemplu, pe măsură ce proiectele tehnologice cresc în complexitate, utilizatorii finali tind să aibă dificultăți în definirea cerințelor pe termen lung fără a putea vedea prototipurile progresive. Proiectele care se dezvoltă în iterații pot obține în mod constant feedback pentru a rafina aceste cerințe, [2].

În februarie 2001, șaptesprezece dezvoltatori de software s-au întâlnit in stațiunea Snowbird din Utah pentru a discuta metodele de dezvoltare ușoară, printre care Jeff Sutherland, Ken Schwaber și Alistair Cockburn. Împreună, cei șaptesprezece au publicat Manifestul pentru Dezvoltarea Agilă a Software-ului, în care au împărtășit faptul că, prin experiența lor combinată de dezvoltare a software-ului și ajutării altora să facă acest lucru, au ajuns să valorizeze:

- Persoanele fizice și interacțiunile lor cu alte personae si cu instrumentele de lucru
- Software de lucru cu o documentație cuprinzătoare

- Colaborarea clienților pe baza negocierii contractului
- Răspunsul la schimbare în urma unui plan

1.3 Scrum

O metodă utiliă pentru gestionarea proiectelor este Scrum, metoda caracteristică programarii Agile. Prima abordare a acestei metode a fost descrisă de Takeuchi and Nonaka în „The New Product Development Game" (Harvard Business Review, Jan-Feb 1986). Ei au scris că de-a lungul timpului proiectele care folosesc echipe mici care au posibilitatea de a transfera una de la alta sarcini produc cele mai bune rezultate.

Scrum permite crearea de echipe cu organizare proprie încurajând mutarea membrilor astfel încât să fie în același spațiu fizic și comunicarea verbală între membrii echipei să fie de preferat altor metode.

Un sprint (sau o iterație) este unitatea de bază de dezvoltare în Scrum. Sprint-ul este un efort experimentat în timp si este limitat la o anumită durată. Durata este stabilită în avans pentru fiecare sprint și este, în mod normal, între o săptămână și o lună, două săptămâni fiind cele mai frecvente. Fiecare sprint începe cu un eveniment de planificare a sprintului, care urmărește definirea unei întârzieri a sprintului, identificarea lucrărilor pentru sprint și realizarea unui angajament estimat pentru atingerea obiectivului de sprint. Fiecare sprint se încheie cu o revizuire a sprintului și retrospectivă pentru sprint, care analizează progresele înregistrate părților interesate și identifică lecțiile și îmbunătățirile pentru următoarele sprinturi.

Metoda *Scrum* este implementată cu ajutorul unui ***ScrumMaster*** (stăpânul scrumului), a cărui sarcină principală este de a înlătura problemele ce împiedică echipa să atingă țelul sprintului. ScrumMaster-ul nu este conducătorul echipei, ci are rolul de intermediar între echipă și influențe care ar putea distrage echipa de la atingerea țelului. Scrum masterul ajută la asigurarea că echipa respectă procesele convenite în cadrul Scrum, facilitează adesea sesiunile cheie și încurajează echipa să se îmbunătățească.

Caracteristici ale metodei Scrum:

- Un set de sarcini nerezolvate care descriu ceea ce trebuie făcut și în ce ordine
- Îndeplinirea unui set fixat de sarcini nerezolvate în serii scurte numite *sprinturi*
- O întâlnire scurtă în fiecare zi (o ședință scrum) în care este stabilit progresul efectuat, munca ce urmează și eventualele impedimente
- O scurtă *sesiune de planificare a sprintului* în care vor fi definite sarcinile nerezolvate ce vor fi incluse în sprint
- O scurtă *retrospectivă a sprintului* în care toți membrii echipei reflectează asupra sprintului încheiat

Figura 1.1 prezintă modul de organizare a unei echipe de scrum și ciclul de livrare a unui produs. Tot aici se evidențiează și faptul ca durata unui sprint este de 2-4 saptamani, iar la sfârșitul fiecarui sprint se doreste livrarea unui produs cu toate funcționalitatile estimate.

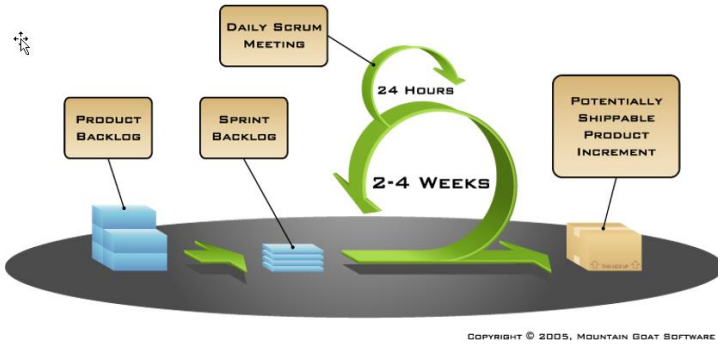

Fig.1.1 Ciclu Scrum [3]

Product Backlog cuprinde o listă ordonată a cerințelor pe care echipa de scrum le menține pentru un produs. Se compune din caracteristici, corecții de erori, cerințe non-funcționale și alte lucruri de care se ține cont pentru a livra cu succes un produs viabil. **Product Owner** scrie cerințele clientului, le acordă prioritate pe baza importanței și dependențelor și le adaugă la product backlog. Echipele Scrum ar trebui să aibă un singur proprietar de produs (Product Owner). Proprietarul produsului ar trebui să se concentreze pe partea de afaceri a dezvoltării produsului și să petreacă majoritatea timpului în legătură cu părțile interesate și nu ar trebui să dicteze modul în care echipa ajunge la o soluție tehnică.

Sprint Backlog este lista de lucru pe care echipa de dezvoltare trebuie să o abordeze în timpul următorului sprint. Lista este derivată de la echipa Scrum care selectează în mod progresiv articolele restante ale produsului în ordinea prioritară din partea de sus a backlog-ului produsului până când simt că au suficientă muncă pentru a umple sprintul. **Product increment** reprezintă suma tuturor elementelor restante ale produselor finalizate în timpul unui sprint, integrate cu activitatea tuturor sprinturilor anterioare. La sfârșitul unui sprint, incrementarea trebuie să fie completă.

1.4 Modelul V

Există mai multe forme de management al proiectelor, însă cele mai folosite sunt Modelul V și Managementul Agile, între care se gasesc multe diferente de concept. In Figura 1.2 se poate obesrva modelul V, care se bazează pe conectarea tuturor segmentelor din ciclul de dezvoltare a unui produs. Acest model este mai puțin flexibil decat managementul agile, se adreseaza progresului doar din cadrul echipei spre deosebire de cel agile unde progresul este monitorizat la nivel de organizație. Modelul V este un model de verificare și validare, aceste doua precese fiind făcute in paralel dar dupa implementarea tuturor specificațiilor proiectului. Această abordare este total diferita fața de modelul agile unde dezvoltarea și testarea se fac progresiv și în paralel. [4]

În dezvoltarea de software, modelul V reprezintă un proces de dezvoltare care poate fi considerat o extensie a modelului cascadă. În loc să se deplaseze în jos în mod liniar, etapele procesului sunt îndoite în sus după faza de codificare, pentru a forma forma tipică V. Modelul V demonstrează relațiile dintre fiecare fază a ciclului de viață al dezvoltării și faza de testare asociată acesteia. Axele orizontale și verticale reprezintă perioada de completare a proiectului sau a

proiectului (de la stânga la dreapta) și nivelul de abstractizare (cea mai mare abstractizare a granulelor de sus).

În principiu, Modelul V se folosește deobicei pentru proiecte mai mari fiind total nepotrivit pentru cele de scurtă durată, iar Managementul Agile in Scrum este unul mai flexibil, se adaptează mai ușor dacă cerințele varieaza și este mai eficient. Ambele metode se folosesc în marile organizații și companii de dezvoltare de produse tehnologice atât software cât și hardware.

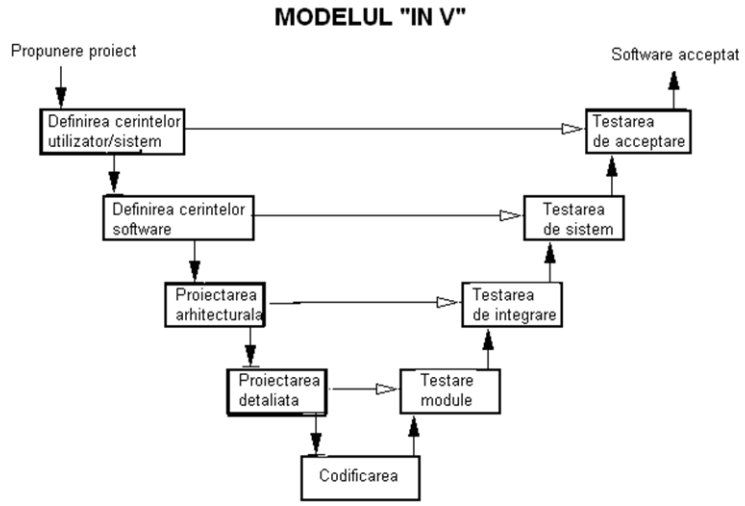

Fig.1.2 Modelul in V [5]

1.5 Aplicații software pentru managementul proiectelor

Pentru a facilita managementul proiectelor, s-au dezvoltat aplicații care sa gestioneze proiectele, echipele, angajații, cerințele produsului, sarcinile angajaților. Astfel de aplicații sunt utile în planificarea sarcinilor angajaților și organizarea capacității echipei în cel mai eficient mod. În funcție de cât de avansat este software-ul, acesta poate gestiona estimarea și planificarea, programarea, controlul costurilor și gestionarea bugetului, alocarea resurselor, software-ul de colaborare, comunicarea, luarea deciziilor, managementul calității și documentația sau sistemele de administrare.

Există numeroase soluții software pentru management de proiect bazate pe PC și browser și soluții software pentru managementul contractelor, cum ar fi : JIRA, ASANA, AGM, etc.

Jira este un produs de urmărire a problemelor, dezvoltat de Atlassian. Oferă funcții de urmărire a erorilor, de urmărire a problemelor și de gestionare a proiectelor. Acesta a fost dezvoltat din 2002. Din iunie 2017, Jira este instrumentul cel mai popular de gestionare a problemelor. Potrivit companiei Atlassian, Jira este utilizat pentru urmărirea problemelor și gestionarea proiectelor de către peste 25.000 de clienți din 122 de țări din întreaga lume . Jira este un produs software comercial care poate fi licențiat , prețurile depinzand de numărul maxim de utilizatori.

Asana este o aplicație web și mobilă concepută pentru a ajuta echipele să-și urmărească activitatea. Asana este un software menit să îmbunătățească colaborarea în echipă. Se concentrează asupra faptului că permite utilizatorilor să gestioneze proiecte și sarcini online fără a utiliza e-mailuri. Fiecare echipă poate crea un spațiu de lucru. Spațiile de lucru conțin proiecte, iar proiectele

conțin sarcini. În fiecare sarcină, utilizatorii pot adăuga note, comentarii, atașamente și etichete. Utilizatorii pot urmări proiectele și sarcinile și, atunci când se schimbă starea unui proiect sau o activitate, adepții primesc actualizări despre modificările primite.

HP **AGile Manager** (AGM) software acționează ca un centru de comunicare și sistem de suport decizional pentru organizarea, planificarea și furnizarea de proiecte Agile. HP AGM permite echipelor să își gestioneze backlogul produsului și sprijină metodologia populară Agile, cum ar fi SCRUM. HP AGM este disponibil fie pe bază de abonament de la o lună la alta, fie pe bază de abonament flexibil pe termen, respectând astfel termenele și bugetul practic al oricărui proiect.

În urma prezentării pe scurt a unor aplicații populare folosite pentru managementul proiectelor se pot obeserva criteriile principale pe care acestea se bazează. Aplicațiile Software pentru managementul proiectelor sunt bazate deobicei pe o singură metodologie de management, dar indiferent de metoda de management de proiecte aleasa, cam toate aceste aplicații trebuie să îndeplinească niște condiții de bază:

- Instrumente de planificare - diagrame Gantt, calendare, sarcini interdependente
- Gestionarea resurselor - gestionarea sarcinilor, interdependențe între sarcini
- Gestionarea timpului - facturare orară, revizuiri săptămânale
- Colaborare - instrumente online de chat / conferință, partajarea de fișiere, acces pe etape
- Instrumente pentru raportarea progresului

2 Fundamente Teoretice

2.1 Tehnologii folosite pentru dezvoltarea codului

Framework-urile nu sunt pur și simplu colecții de clase, ci au o bogată funcționalitate și interconexiuni puternice între clasele de obiecte care oferă o infrastructură dezvoltatorului. Ca o colecție coerentă de active reutilizabile, un Framework poate fi o investiție importantă pentru o organizație. Acestea sprijină refolosirea designului detaliat și al codului. Pe baza contextului utilizării sale, framework-urile sunt clasificate în următoarele categorii: categoria white-box, care este de obicei extinsă prin subclasarea claselor abstracte existente, în timp ce framework-urile de tip black-box sunt mai degrabă componente, deoarece acestea pot fi accesate numai prin intermediul interfețelor lor.

Una dintre întrebările cheie în timpul unei etape de luare a deciziilor este care este beneficiul în selectarea unui anumit cadru. Răspunsul este că framework-uri GUI nu au logică de afaceri, spre deosebire de framework-urile de aplicații care conțin mai mult de 70% din funcționalitatile de bază ca parte a cadrului. Pe parcursul timpului s-a observat că modelele ar putea reprezenta 36% din proiectarea cadrului.

Framework-urile foarte populare în lumea dezvoltării de software sunt următoarele:

- Microsoft Foundation Classes – MFC
- setul de instrumente Java pentru fereastra abstractă - AWT
- obiecte reutilizabile (ORO), care este un cadru open source
- Web Exchange Private Exchange este un cadru orizontal conceput pentru a construi aplicații B2B
- Cadrul Medical Business Object este un cadru vertical conceput pentru domeniul medical. [7]

Framework-urile oferă dezvoltatorilor un cadru general pentru construirea de aplicații ce respectă un anumit tipar (de exemplu, aplicații web). Infrastructura definește o serie de entități generice și asigură comunicarea de nivel scăzut între acestea (de exemplu, generarea de evenimente într-un context event-driven). Figura 2.1 ilustrează modul de crearea a unui aplicații folosind framework-uri și evidențiaza faptul că pe lângă aceastea, în implementarea unei aplicații este nevoie și de modele de design și alte componente. Elementul design patterns se referă la arhitectura folosită pentru realizarea aplicației, care poate fi spre exemplu MVC, Model-View-Controller, design care se va folosi și în aceasta lucrare.

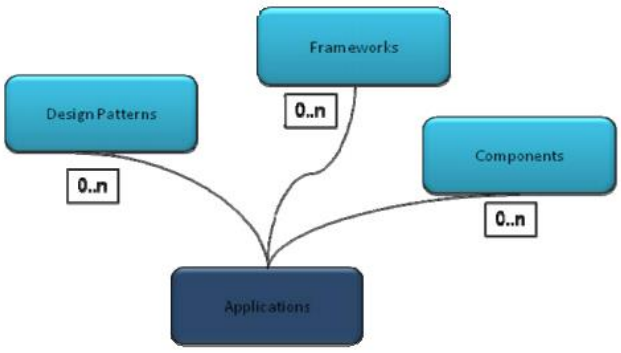

Fig.2.1 Schema realizării unei aplicații [7]

Un avantaj ce rezultă din utilizarea framework-urilor pentru realizarea aplicațiilor web este reutilizabilitatea. Urmărind desfășurarea firească a lucrurilor, obținem o mărire a fiabilității, deoarece utilizarea repetată a infrastructurii conduce la depistarea eventualelor erori și la îmbunătățirea corespunzatoare. Dezavantajul principal provine din limitarea flexibilității. Orice aplicație dezvoltată utilizând un anumit framework se va supune fluxului generic pe care acesta îl definește. Acest lucru poate conduce la cuplaje complicate chiar în situații simple.

2.1.1 Java

Java este unul dintre cele mai utilizate limbaje de programare la momentul actual. Este un limbaj de programare de nivel înalt lansat în 1995 de Sun Microsystems și conceput de James Gosling. In prezent este dezvoltat de Oracle. Limbajul este caracterizat de simplitate, robustețe, portabilitate („Write once, run anywhere”), performanță ridicată, dinamicitate și cel mai important, orientat pe obiecte. [8]

Java este un limbaj care poate fi compilat atât local, cât și la distanță prin intermediul unui server. O dată cu revoluția telefoanelor inteligente, Java a prins aripi și a devenit programul numărul unu în crearea de aplicații. Să enumerăm câteva domenii de aplicare: dezvoltarea aplicațiilor mobile, dezvoltarea site-urilor, conectarea la baze de date, dezvoltarea interfețelor grafice. Există două posibilităţi de a lucra în Java: în linie de comandă și prin crearea unei aplicații în unul dintre mediile de dezvoltare java: Eclipse, NetBeans, Intellij IDEA.

Fișierul cod sursă este un fișier text ce conţine implementarea programului într-un limbaj de programare. Acest fișier poate fii dezvoltat în editoare precum: Notepad++, Ultraedit, Emacs. Aceste fișiere vor fi executate din linia de comanada, sau dezvoltarea și executarea se poate face mult mai simplu în mediile de dezvoltare integrate (IDE) menționate anterior. Codul este mai întâi compilat, adica codul sursă este translatat de un program denumit compilator în cod mașină, după care poate fi executat. Mai mult codul sursă este compilat într-un cod de octeți intermiediar între codul sursa și codul mațina, lucru ce face ca java să fie un limbaj de programare portabil.

Java virtual machine(JVM) este mediul în care se execută programele Java. Astfel că programele java pot rula pe orice platforma pe care este instalată o mașina virtuala java, din asta rezultand potabilitatea acestui limbaj de programare. Java Runtime Environment (JRE) este un pachet software care conține tot ce este necesar pentru a putea rula un program java, adică include atât JVM cât și librării de clase java. Java Development Kit (JDK) este un set de JRE și conține instrumente pentru programatorii java. Acest kit este gratuit și se va folosi în proiectul dezvoltat în acesta lucrare. [9]

În prezent, se folosește Java 8, dar la începutul verii acestui an, 2017, va fi diponibilă următoarea versiune, mai exact Java 9. Această noua versiune va aduce cu sine unule îmbunătățiri cum ar fi: dezvoltare API pentru controlul și managementul proceselor sistemelor de operare, definirea unui nou API client Http care să implementeze HTTP/2 si WebSocket, adăugarea mai multor comenzi de diagnosticare, controller JavaFX UI, compilare inteligentă și multe altele. [10]

2.1.2 Maven

Maven este un sistem de execuție și management al proiectelor scrise în Java. Acesta face parte din Apache Software Foundation.

În orice proiect java este nevoie de anumite dependențe pentru a folosi diferite biblioteci. Într-un proiect simplu, se descarcă de pe internet aceste biblioteci, fișiere cu extensia ".jar", și se încarca în proiect. Cu Maven însă, acest ciclu anevoios nu mai trebuie făcut, deoarece acesta conține un fișier "pom.xml" în care se vor scrie toate datele proiectului inlusiv dependențele acestuia. La execuția unui proiect Maven, toate dependențele inscripționate în fișierul "pom.xml" vor fi automat descărcate și astfel lucrurile se simplifică.

Medii de programare care suportă Maven: Eclipse, NetBeans, Intellij IDEA, JDeveloper.

Proiectul Maven are setate toate datele în fișierul POM: numele, id-ul și group id-ul proiectului, versiunea, tipul de pachet și descrierea, dupa cum se poate vedea în codul din exemplul de mai jos. [11]

```
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.example.login</groupId>
  <artifactId>demo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>
  <name>demo</name>
  <description>Demo project for Spring Boot</description>
</project>
```

2.1.3 Eclipse

Eclipse este un mediu de dezvoltare Java gratuit. Poate fi folosit atât pentru programare în Java cât și în C/C++ sau PHP. Este unul dintre cele mai folosite medii de dezvoltare, fiind utilizat pe Windows, Linux sau Mac Os.

Versiunea Luna aparută în 2014 este prima versiune cu support Java 8 integrat. Ultima versiune este Eclipse Neon aparută în 2016, iar pentru urmatorii ani sunt pregatite noi versiuni cu functionalități multiple și diversificate. [12]

2.1.4 Intellij IDEA

Intellij IDEA este de asemenea un mediu de dezvoltare software, dar care spre deosebire de Eclipse are atat versiune gratuită "Community" cât și o versiune profesională "Ultimate" care conține diferite instrumente dedicate, care nu este open-source. Este dezvoltată de JetBrains și e compatibilă cu sistemele de operare : Windows, Linux si Mac Os. Câteva din limbajele suportate sunt: Java, Groovy, Python, XML, iar versiunea Ultimate aduce suplimentar suport pentru HTML, JavaScript, PHP, SQL, Ruby. [13]

Acest mediu ofera sugestii la completarea codului și imbunătățirea acestuia, asistență la refacerea codului, navigație și cautare prin tot proiectul.

Framework-uri suportate de Community Edition: Android, Gradle, JavaFX, JUnit, Maven, TestNG, iar Ultimate Edition mai adaugă și support pentru: JSF, JSP, Node.js, Ruby on Rail, Spring, Struts, Django, Web Services, Tomcat.

2.1.5 Apache Tomcat

Apache Tomcat mai numit și Tomcat Server este un container java. Acesta este dezvoltat și întreținut de către Apache Software Foundation și oferă un mediu de server web HTTP în care codul Java poate rula. [14]

Catalina este un container servlet al Tomcat și implementează specificațiile Sun Microsystems pentru servlet și JSP (Java Server Pages). Pentru pornirea unui server Tomcat este suficientă executarea din linia de comandă a fișierului "catalina.bat" prin comanda: run catalina.bat.

2.1.6 Spring Framework

Spring Framework poate fi utilizat de către orice aplicație Java, dar există si extensii pentru construirea de aplicații web peste platforma Java EE. Acest framework este open-source. Spring oferă un suport de infrastructură la nivel de aplicație, care permite programatorilor să se concentreze pe logica la nivel de aplicație, fără restricții legate de un anumit mediu de implementare. Unele dintre principalele caracteristici ale Spring sunt: injectarea de dependente, programarea orientata, aplicație web Spring MVC și cadru de servicii web REST, si suport pentru JDBC, JPA, JMS.[15]

2.1.7 Spring Boot

Spring Boot este o soluție convetion-over-configuration a framework-ului Spring pentru crearea de aplicații stand-alone (de sine stătătoare). Cu aceasta soluție se pot crea aplicații Spring care pur și simplu pot rula deoarece nu sunt necesare prea multe configurări, fiind folosite unele predefinite. [17]

Spring Boot conține un server Tomcat integrat, generează un fișier POM cu configurările Maven inițiale, nu necesită configurări in XML și furnizează proiecte gata de producție.

Un proiect Spring Boot se poate realiza cu Intellij IDEA,cu Eclipse sau cu Spring Tool Suit, ca un nou poiect de tipul Spring Initializer, sau se poate descarca de pe site-ul oficial Spring un proiect gata configurat ca proiect Spring Boot și mai apoi deschis în unul din programele mai sus menționate.

În cazul Intellij IDEA Community Version, nu exista opțiunea de spring initializer, dar se poate crea un proiect Maven pur și simplu și configura fisierul POM pentru a include bibliotecile necesare unui proiect Spring Boot.

Urmatoarele dependențe sunt necesare pentru simpla creare a unui proiect Spring Boot și vor fi adăugate după setările proiectului Maven în fișierul POM. Prin crearea secțiunii <parent> se inițializează proiectul ca fiind un copil al proiectului *org.springframework.boot,* și ca urmare proiectul copil ii va moștenii toate cofigurările.

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>1.5.2.RELEASE</version>
  <relativePath/>
</parent>
```

```
<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
  <java.version>1.8</java.version>
</properties>

<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-tomcat</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>

<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
```

2.1.8 Spring Tool Suite

Spring Tool Suite (STS) este un mediu de programare personalizat bazat pe Eclipse, care face dezvoltarea de aplicații Spring mult mai ușoara. Suitele oferă combinații de instrumente gata de utilizare de sprijin lingvistic, suport-cadru, și asistență în timpul rulării. Noile instrumente sunt combinate cu cele existente Java, Web și Java EE preluate de la Eclipse.

STS este gratuit însă nu este necesar pentru crearea de proiecte Spring deoarece acestea se pot realiza și în alte medii de dezvoltare.

2.2 Tehnologii folosite pentru accesarea datelor

2.2.1 Spring Data JPA

JPA (Java Persistance API) este o specificație Java pentru accesarea și gestionarea datelor între obiecte Java / clase și o bază de date relațională. Face posibila preluarea de date de la o baza de date și convertirea acestora la instanțe alea unor obiecte construite în clasele java. Clasele java, care sunt folosite ca modele, sunt configurate ca și clase @Entity ale caror atribute sunt direct mapate de către framework la coloanele bazei de date. [18]

Spring Date JPA face parte din Spring Data și face mai ușoara implementarea unui repository bazat pe JPA. Acest modul face mai posibilă dezvoltarea unei aplicații Spring, care utilizează tehnologii de acces la date. Pentru folosirea acestui model este nevoie de integrarea în proiect a dependențelor specifice:

```
<dependencies>
  <dependency>
    <groupId>org.springframework.data</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>
</dependencies>
```

2.2.2 Apache Derby

Apache Derby este un sistem de management al bazelor de date relaționale (Relational Database Management System - RDBMS), care poate fi integrat în programele Java. Este dezvoltat de Apache Software Foundation și este open-source. Nucleul tehnologiei, motorul bazei de date Derby, este o bază de date relațională încorporata suportand JDBC și SQL ca API-uri de programare. [19]

Acest sistem pornește o bază de data în momentul pornirii aplicației și este foarte utilă pentru dezvoltarea și testarea aplicațiilor care necesită baze de date. Nu se folosește pentru producerea aplicațiilor.

De asemenea și dependența acestui sistem trebuie adaugată în fișierul POM:

```
<dependency>
  <groupId>org.apache.derby</groupId>
  <artifactId>derby</artifactId>
  <scope>runtime</scope >
</dependency>
```

2.2.3 MySQL

MySQL, cel mai popular sistem de management de baze de date SQL open source, este dezvoltat, distribuit, și susținute de Oracle Corporation. MySQL Server poate rula confortabil pe un desktop sau laptop, alături de alte aplicații, servere de web, care necesită puțină atenție sau chiar deloc. Baza de date MySQL Software este un sistem client / server care constă dintr-un server multi-threaded SQL care suportă diferite backend-uri, mai multe programe diferite pentru partea de client și biblioteci, instrumente administrative, precum și o gamă largă de interfețe de programare a aplicațiilor (API).

De asemenea, MySQL Server ofera o bibliotecă multi-threaded încorporat, care poate fi inclusa în aplicația dumneavoastră pentru a obține mai rapid un produs independent si mai ușor de gestionat. [20]

2.3 Aspecte teoretice utilizate pentru realizarea interfeței cu utilizatorul

2.3.1 Vaadin Framework

Piesa de bază a framework-ului Vaadin este biblioteca Java care este proiectată pentru a face crearea și întreținerea interfetei web cu utilizatorul mult mai ușoare. Ideea-cheie în programarea bazată pe Vaadin este că permite programatorului sa uite de web și îl lasă să programeze interfața cu utilizator ca și pe oricare altă aplicație Java, dar mult mai simplu. Prin utilizarea acestei interfețe

utilizatorul nu este nevoit să învețe limbaje de programare specifice pentru web cum ar fi HTML sau JavaScript, ci poate să se concentreze pe logica aplicației.

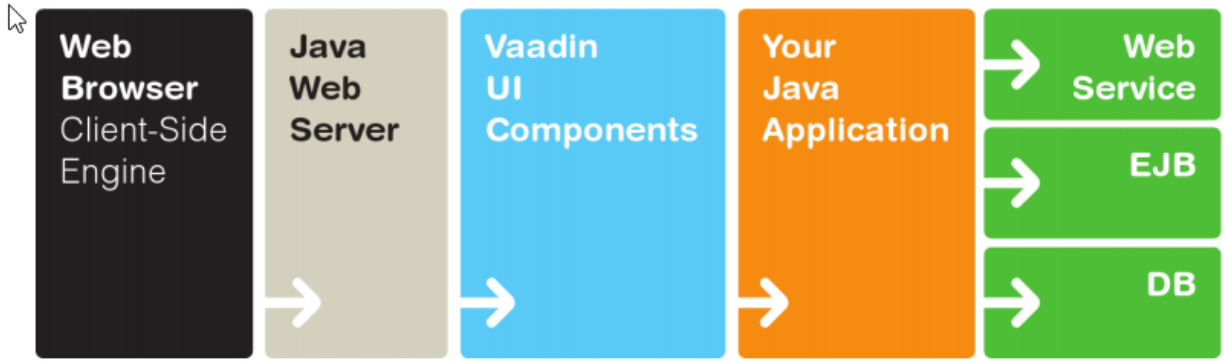

Fig.2.2 Arhitectura generală Vaadin [21]

Fig.2.2 ilustrează arhitectura de bază a unei aplicații web realizata cu Vaadin. Această arhitectură este compusă dintr-un framework pentru partea de server și un motor de rulare a părții clientului în browser ca și un program JavaScript, realizând o interfață interactivș cu utilizatorul. Cum aplicația ruleaza ca o sesiune Java Servlet într-o aplicație server, legatura dintre aplicația UI (user interface) si logica de server poate fi facută foarte usor. [21]

Începând cu Vaadin 6 se oferă un plugin pentru Eclipse și un editor de interfața vizibil tot în Eclipse. În acest an, 2017, a fost lansat Vaadin 8 care oferă mult mai multe funcționalități pentru ușurarea scrierii codului cum ar fi : converter pentru expresiile lambda introduse de Java 8, încarcarea mai ușoara a datelor de pe backend, suport pentru HTML5 History API, explicații ajutatoare în mesajele excepțiilor, și așa mai departe.

Bibliotecile Vaadin pot fi folosite în programarea atât în Eclipse cât și în Intellij IDEA. Dependențele Vaadin pot fi integrate într-un proiect Maven în fișierul *pom.xml*, iar mai apoi toate bibliotecile ce conțin componete UI și anotațiile necesare creării unei interfețe UI vor fi disponibile.

```
dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>com.vaadin</groupId>
      <artifactId>vaadin-bom</artifactId>
      <version>8.0.0</version>
      <type>pom</type>

      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

2.3.2 Vaadin UI class

De asemenea, acest framework poate fi folosit și pentru crearea de interfețe cu utilizatorul pentru aplicațiile Spring. Acest lucru se face prin integrarea în proiectul realizat cu Spring Boot a biblioteci necesare și anotarea corespunzatoare a claselor destinate interfeței Vaadin.

```
@SpringUI
@Theme("valo")
public class TodoUI extends UI {
   private VerticalLayout layout;
   @Autowired
   Repository repo;
   @Override
   protected void init(VaadinRequest vaadinRequest) {
      layout = new VerticalLayout();
      layout.setDefaultComponentAlignment(Alignment.MIDDLE_CENTER);
      setContent(layout);
      Label header = new Label("Aplicatie Spring Boot & Vaadin");
      header.addStyleName(ValoTheme.LABEL_H1);
      layout.addComponent(header);
     }
}
```

În codul din exemplu, @SpringUI este anotația necesară pentru ca Spring să considere clasa ca și clasa de tip UI. Această clasa moștenește clasa UI care aparține bibliotecilor Vaadin, și oferă posibilitatea utilizării elementelor grafice: butoane, tabele, elemnte de text și altele. Anotația @Autowired leagă partea de UI de o clasă Repository, un model al datelor din baza de date, și astfel poate accesa datele.

Pe deasupra, Vaadin mai oferă si o tema implicită “valo” prin care se pot stiliza elementele de UI și oferă o gamă largă de atribute care pot fii setate pentru aceste componente doar prin folosirea anotației @Theme(“valo”). Pe langă această temă, se pot crea teme și de către programator utilizand CSS (Cascadin Style Sheets), fișierele fiind integrate în proiectul Eclipse, dar majoritatea proiectelor folosesc o singură temă. Folosirea CSS pentru stilizarea elementelor grafice este rezultată din faptul că Vaadin rulează partea de client ca și un program JavaScript. Astfel că, dacă rulam în browser o aplicatie Vaadin și deschidem codul sursă acesta va fi un fișier HTML, ca și în Figura 2.3.

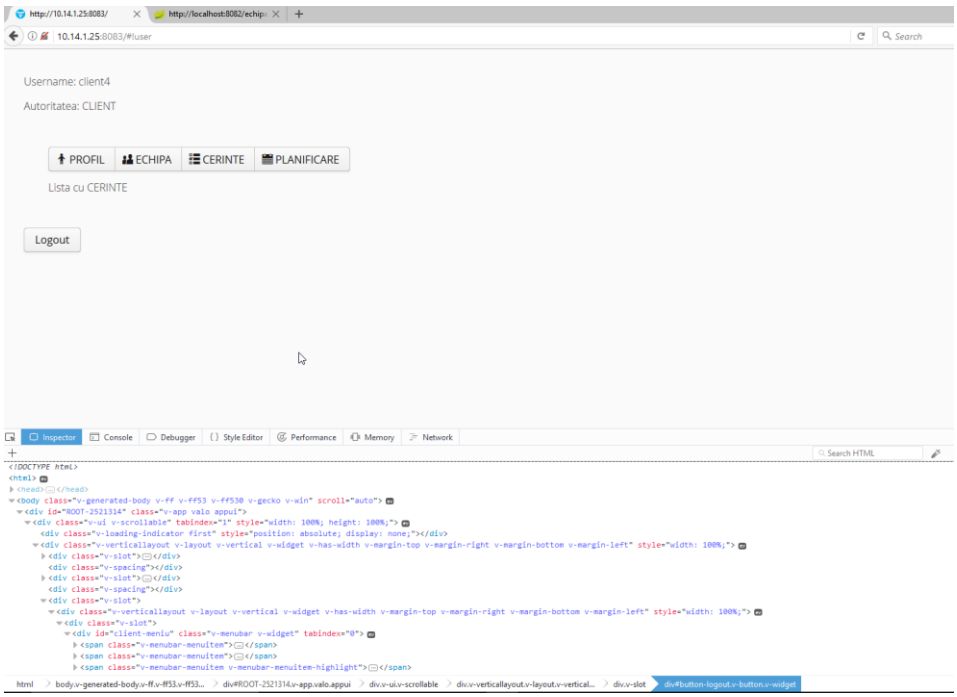

Fig.2.3 Fișierul HTML din spatele unei interfețe Vaadin

2.4 Aspecte teoretice folosite pentru implementarea serviciilor web

2.4.1 Model Class

Clasa model reprezintă o entitate care apare și în baza de date și are ca atribute coloanele din baza de date. Fiecare rând din baza de date poate fi transpus ca și un obiect Java cu atributele definite în clasa model. În codul exemplu se pot observa anotațiile corespunzatoare pentru conectarea atributelor la coloane @Column și a intregii clase model @Entity la un tabel @Table. În aceasta clasă mai sunt implementate și metodele pentru setareașsi preluarea valorii acestor atribute.

```
@Entity
@Table(name = "echipe")
public class EchipaModel extends Object {
   @Id
   private String numeEchipa;
   private String manager;
   public EchipaModel(String numeEchipa, String manager) {
      this.numeEchipa = numeEchipa;
      this.manager = manager;
   }
   public String getNumeEchipa() {
      return numeEchipa;
   }
   public void setNumeEchipa(String numeEchipa) {
      this.numeEchipa = numeEchipa;
   }
   public String getManager() {
      return manager;
   }
   public void setManager(String manager) {
      this.manager = manager;
   }
}
```

2.4.2 Interfața pentru maparea datelor din baza de data la clasa model

Interfața necesara pentru realizarea legaturii dintre baza de date și clasa model va extinde interfața “JpaRepository” care realizează implicit conversia detelor în obiecte java și invers.

```
public interface EchipaRepository extends JpaRepository< EchipaModel, String> {
   @Transactional
   void deleteById(String id);
}
```

Parametrii de tip EchipaModel și String din *JpaRepository< EchipaModel, String>* fac referire la faptul ca datele din baza de date se vor converti în obiect de tip EchipaModel identificat prin cheia principala de tip String, care poate fi id-ul.

Repository-ul creat moștenește de la clasa pe care o extinde metode de salvare, preluare și ștergere a datelor din baza de date. Pe langă aceste metode se pot definii și alte metode ca si cea

din exemplu *void deleteById(String id)*, care va șterge din baza de date elementul de pe o linie în funcție de valoarea coloanei "id" . Dupa regula: action+By+Attribute (ex: EchipaModel getByName(String name) sau void deleteByDescription(String descript ion)), se pot definii și alte metode, deoarece prin folosirea anotației @Transactional framework-ul va știi să interpreteze metoda și să o mapeze către implementarea sa.

2.4.3 Clasa Service

Clasa de tip service conține metode implementate care gestionează datele sub forma de obiecte preluate din repository-ul anterior creat. Aceste metode for fi mai apoi utilizate de controller-ele de Rest.

```
@Service
public class EchipaService  {
   @Autowired
   private EchipaRepository echipaRepository;

   public List<EchipaModel> getAllEchipe()
   {
      List<EchipaModel> echipaModels =new ArrayList<>();
      //echipeRepository.findAll().forEach(echipaModels::add);
      for(EchipaModel echipaModel : echipaRepository.findAll())
      {
         echipaModels.add(echipaModel);
      }
      return echipaModels;
   }
   public EchipaModel getEchipa(String numeEchipa)
   {
     return user.findOne(id);
      }
   public void addEchipa(EchipaModel echipaModel){
       echipaRepository.save(echipaModel);
   }
   public void updateEchipa(EchipaModel echipaModel, String id) {
      echipaRepository.save(echipaModel);
   }
   public void deleteEchipa(String nume) {
      echipaRepository.delete(nume);
   }
}
```

În codul exemplu se pot observa anotațiile specifice framework-ului Spring pentru clasa cu servicii @Service și pentru legarea repository-ului la aceasta clasa @Autowired.

De asemenea, în implementarea serviciilor se folosesc motodele moștenite de către repository-ul creat de la clasa părinte JpaRepository. În prima metodă se poate vedea utilizarea unei sintaxe mai diferite "topicRepository.findAll().forEach(topics::add)", care este o expresie lambda introdusă de Java 8 pentru minimizarea dimensiunii codului. Usual, această expresie ar putea fi inlocuita de mai multe linii din cod, cu care majoritatea programatorilor sunt obisnuiti:

```
List<EchipaModel> echipe=new ArrayList<>();

echipaRepository.findAll().forEach(echipe::add)
```

```
List< EchipaModel > echipe =new ArrayList<>();
for(EchipaModel echipa : echipaRepository.findAll())
{
   echipe.add(echipqa);
}
```

2.4.4 Controller Class

Clasa care conține rest controller-ele mapează serviciile create anterior la link-uri URL care pot fi accesate din browser și returnează datele cerute în format JSON(JavaScript Object Notation) sau modifică date din repository, în funcție de acțiunea care sa face: POST(salvează date), GET(preia date), PUT(actualizează date), DELETE(șterge date).

Toate aceste acțiuni se fac tot cu ajutorul anotațiilor oferite de Spring: @RestController pentru a identifica clasa ca și controller rest, @RequestMapping pentru maparea unui URL la metoda corespunzatoare cu specificarea tipului acșiunii dorite, @PathVariable pentru a identifica o variabila preluată din URL, @RequestBody pentru un parametru dat de utilizator printr-un JSON. Dupa cum se poate observa aceste framework simplifică foarte mult procesul de creare a parți de server și de gestionare a datelor doar prin anotașiile pe care le interpretează și mai apoi gestionează codul și setarile fără implicarea programatorului.

```
@RestController
@RequestMapping("/echipaservice")
public class EchipaController {
   @Autowired
   private EchipaService echipaService;
   @RequestMapping("/echipe")
   public List<EchipaModel> getallEchipe()
   {
      return echipaService.getAllEchipe();
   }
   @RequestMapping("/echipa/{id}")
   public EchipaModel getEchipa(@PathVariable String id)
   {
      return echipaService.getEchipa(id);
   }
   @RequestMapping(method = RequestMethod.POST, value = "/echipa")
   public void addEchipa(@RequestBody EchipaModel echipaModel){
      echipaService.addEchipa(echipaModel);
   }
   @RequestMapping(method = RequestMethod.PUT, value = "/echipa/{id}")
   public void updateEchipa( @RequestBody EchipaModel echipaModel,@PathVariable String numeEchipa ){
      echipaService.updateEchipa(echipaModel,numeEchipa);
   }
   @RequestMapping(method = RequestMethod.DELETE, value = "/echipa/{id}")
   public void deleteEchipa(@PathVariable String id ) {
      echipaService.deleteEchipa(id);
   } }
```

2.5 Tehnologii folosite pentru testarea aplicației

2.5.1 Aspecte teoretice folosite pentru testarea serviciilor web

2.5.1.1 Testare automată

Pentru testarea serviciilor web create cu String Boot se poate folosi modulul de testare oferit de același framework și utilizat prin anotațiile @RunWith și @SpringBootTest, dupa cum arată și exmplul de mai jos. Testele sunt construite prin consumarea serviciilor și verificarea datelor returnate. Pentru testare se poate folosi o baza de date integrata(își menține datele doar până la oprirea serverului), astfel ca la începutul testului baza de date se populează cu date, iar mai apoi se asteaptă ca datele returnate de serviciile web accesate să fie chiar cele introduse anterior.

```
@RunWith(SpringRunner.class)
@SpringBootTest
public class SpringapplicationApplicationTests {
  @Autowired
  private TestRestTemplate restTemplate;
  final String BASE_URL = "http://localhost:8080/";
  @Test
  public void contextLoads() {
    final String Manager_NAME = "Pop Aurel ";
    final String Echipa_Id = "echipa1";
//create Topic
     EchipaModel echipa = new EchipaModel ();
     echipa.setManager(Manager_NAME);
     echipa.setNumeEchipa(Echipa _Id);
//send post request to the server
    RestTemplate rest = new RestTemplate();
    ResponseEntity< EchipaModel > response =
        rest.postForEntity(BASE_URL, topic, EchipaModel.class);
    assertThat( response.getStatusCode() , equalTo(HttpStatus.CREATED));
//verify the created data
    EchipaModel echipaCreated = response.getBody();
    assertThat(echipaCreated.getNumeEchipa() , equalTo(Echipa_Id) );
  }
}
```

2.5.1.2 Testare manuală

Testarea manuală a serviciilor se poate face cu ajutorul unui program de testare de API, Postman, în care se poate introduce URL-ul serviciilui dorit și specifica acțiunea de: POST, PUT, GET, DELETE, ce urmează a fi executată. Acest program ofera posibilitatea ca pentru acțiuniile de POST și PUT, utilizatorul sa poate insera date sub forma de JSON spre exemplu, iar pentru GET să vadă raspunsul serverului. Acest program simplifică testarea serviciilor web și a comunicării serverului cu baza de date.

Postman funcționeaza ca aplicație gratuita pe Linux, Windows, Mac și Chrome, dar are și variantă profesională contra cost. [14]

Figura 2.4 reprezintă o imagine a interfeței Postman executând POST, iar figura2.5 afișează rezultatul pentru GET.

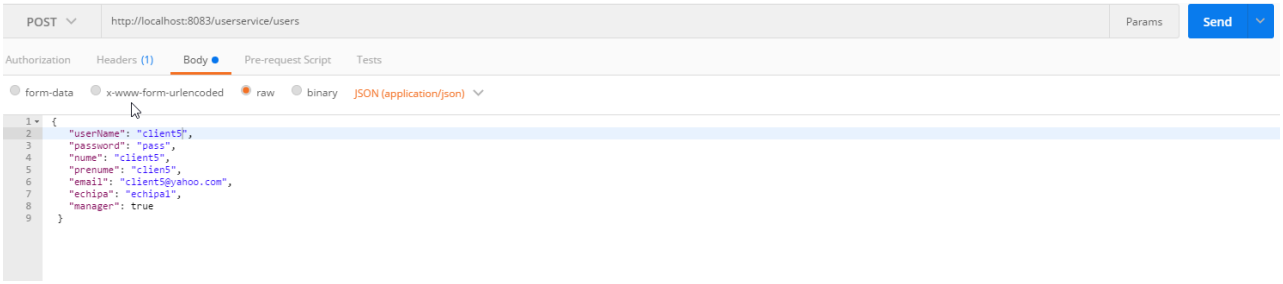

Fig.2.4 Postman UI- POST action

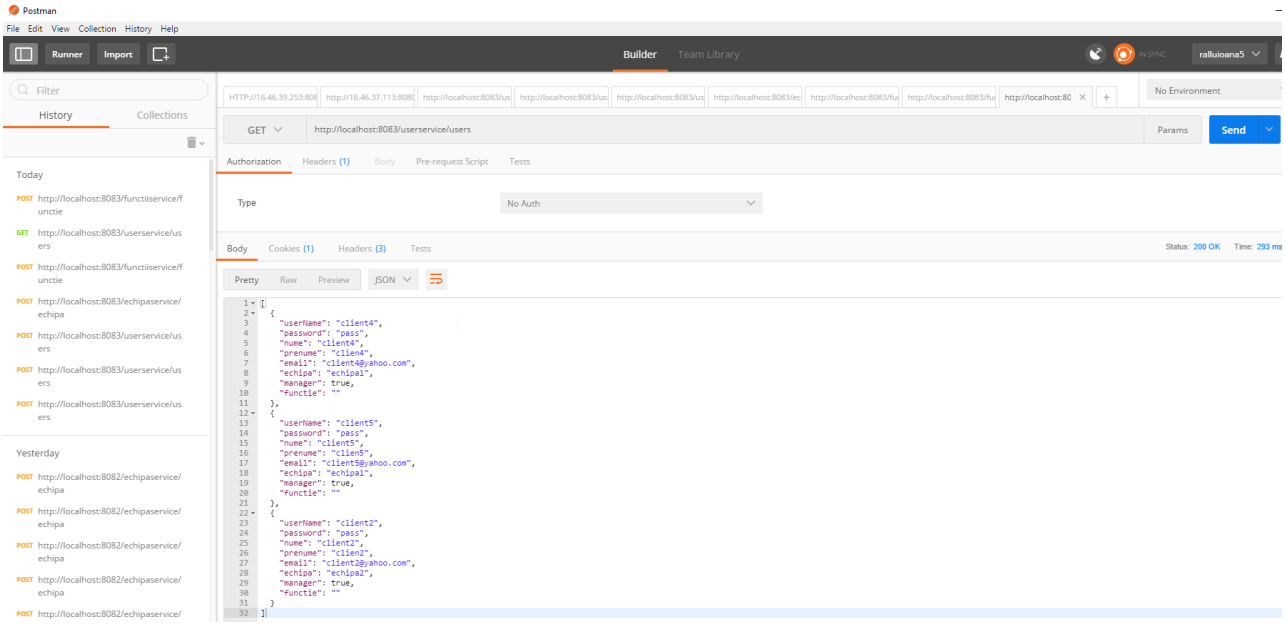

Fig.2.5 Postman UI- GET action

2.5.2 Aspecte teoretice folosite pentru testarea interfeței cu utilizatorul

1.1.1.1 LeantFT pentru testare automat

LeanFT este un framework produs de Hewlett Packard Enterprise (HPE) pentru dezvolatrea de teste automate pentru testarea interfeței cu utilizatorul a aplicațiilor web. Este o soluție puternică și ușoară de testare funcțională construită special pentru testare continuă și integrare continuă.

UFT Pro (LeanFT) este pe deplin integrat în IDE-urile de dezvoltare standard: Visual Studio, Eclipse și IntelliJ. Aceasta permite dezvoltarea de scripturi de testare bazate pe cadre standard de testare a unităților, cum ar fi NUnit, JUnit și TestNG. Testele pot fi scrise utilizand limbajele de programare JavaScript, C# si Java. LeanFt permite crearea de teste de selenium robuste cu ajutorul uneltelor UFT Pro (LeanFT), cum ar fi Centrul de identificare a obiectelor și alte utilitare. Testele create folosind acest instrument se pot executa pe platformele Mac, Linux sau Windows. [16]

LeanFT oferă suport pentru diferite browser-e și platforme pe care ruleaza aplicațiile ce urmează să fie testate, cum ar fi : Mozilla Firefoz, Google Chrom, Internet Explorer, Safari, Android.Acest produs nu este open source, dar se poate testa gratis timp de 60 de zile. O varianta asemanatoare pentru testare a interfeței cu utilizatorul, dar gratuită este Selenium.

Diferența dintre aceste doua produse este faptul ca LeanFT conține un centru de identificare al obiectelor, Figura 2.6, care sugerează atributele dupa care obiectul dorit ar putea fi selectat. Aceasta funcție permite selectarea atributelor dorite, reidentificarea obiectului în funcție de acestea și generarea codului de identificare scris în limbajul de programare dorit, Java în cazul acestei lucrări .

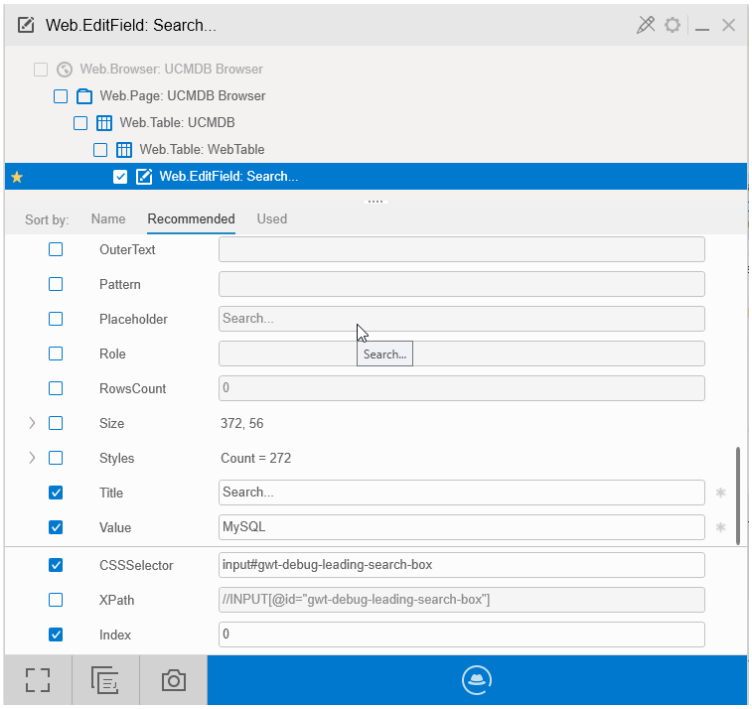

Fig.2.6 LeanFT –Centru de identificare al obiectelor

În urma selectarii atributelor : title, value, cssselector și index, centrul de identificare va genera un cod pentru obiectul identificat de tip EditField, cum apare iîn urmatorul exemplu:

```
browser.describe(EditField.class, new EditFieldDescription.Builder()
        .type("text").value("MySQL").cssSelector("input#gwt-debug-leading-search- box")
        .title("Search...").visible(true).index(0).build());
```

După instanțierea acestui obiect de tip EditField, LeanFT permite realizarea de operațiuni cu acest obiect cum ar fi: click(), setValue(“obiect de căutat”), getValue() și așa mai departe.

2.5.2.1 Selenium

Selenium este un set de instrumente software diferite, fiecare având o abordare diferită în sprijinul automatizării testării. Întreaga suită de instrumente generează un set bogat de funcții de testare adaptate în mod special nevoilor de testare a aplicațiilor web de toate tipurile. Aceste operațiuni sunt foarte flexibile, permițând numeroase opțiuni pentru localizarea elementelor UI și

compararea rezultatelor testelor așteptate cu comportamentul real al aplicațiilor. Una dintre caracteristicile cheie ale Selenium este suportul pentru efectuarea testelor pe mai multe platforme de browser. [23]

Selenium-WebDriver acceptă următoarele browsere, împreună cu sistemele de operare cu care sunt compatibile aceste browsere: Google Chrome, Internet Explorer , Windows 7, Windows 8 și Windows 8.1. Începând cu 15 aprilie 2014, IE 6 nu mai este acceptată. Produsul acceptă versiuni de browser pe 32 de biți și 64 de biți, dacă este cazul, cum ar fi: Firefox, Safari, Operă, HtmlUnit, Android, iOS.

3 Bibliografie

[1]https://biblioteca.regielive.ro/cursuri/management/managementul-proiectelor-5623.html [30.06.2017]

[2] Hass, Kathleen B. "The blending of traditional and agile project management." PM world today 9.5 (2007): 1-8.

http://chelsoftusa.com/uploads/3/4/1/3/34136265/agile_well_explained.pdf [30.06.2017]

[3] https://leankit.com/learn/agile/agile-development/ [30.06.2017]

[4]https://en.wikipedia.org/wiki/V-Model_(software_development) [30.06.2017]

[5] http://www.scritub.com/stiinta/informatica/Modele-ale-Procesului-de-dezvo1411914920.php [30.06.2017]

[6]http://www.cs.wustl.edu/~schmidt/PDF/patterns-intro4.pdf [30.06.2017]

[7]http://www.bptrends.com/publicationfiles/04-05-2011-ART-Anatomy%20of%20Software%20Frameworks-Paradkar-final.pdf [30.06.2017]

[8] Arnold, Ken, James Gosling, and David Holmes. The Java programming language. Addison Wesley Professional, 2005.

https://scholar.google.ro/scholar?q=java+programming+language&btnG=&hl=en&as_sdt=0%2C5&oq=java+p [30.06.2017]

[9] Venners, Bill. The Java Virtual Machine. McGraw-Hill, New York, 1998.

[10] http://www.oracle.com/technetwork/java/javase/8-whats-new-2157071.html [30.06.2017]

[11]http://webserver2.tecgraf.pucrio.br/~ismael/Cursos/Senac_MTSW/aulas/Modulo2_TecnicasDesnvolvimentoAgeis/2-Maven/articles/Maven2_Intro_jw-1205.pdf [30.06.2017]

[12] https://eclipse.org/ [30.06.2017]

[13] https://www.jetbrains.com/idea/documentation/ [30.06.2017]

[14] http://tomcat.apache.org/ [30.06.2017]

[15] Gupta, Praveen, and M. C. Govil. "Spring Web MVC Framework for rapid open source J2EE application development: a case study." Interface 2.6 (2010): 1684-1689.

[16] https://saas.hpe.com/en-us/software/leanft [30.06.2017]

[17] https://javabrains.io/courses/spring_bootquickstart [30.06.2017]

[18] http://projects.spring.io/spring-data-jpa/ [30.06.2017]

[19] https://en.wikipedia.org/wiki/Apache_Derby [30.06.2017]

[20] https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html [30.06.2017]

[21] Grönroos, Marko. Book of Vaadin. Lulu. com, 2011.

http://s3.amazonaws.com/academia.edu.documents/48834056/book-of-vaadin.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1498414156&Signature=MMgzHdtvz66ocVLZB31ADwK7Wdo%3D&response-content-disposition=inline%3B%20filename%3DBook_of_Vaadin_Vaadin_7_Edition_-6th_Rev.pdf [30.06.2017]

[22] https://saas.hpe.com/de-de/software/leanft [30.06.2017]

[23] http://www.seleniumhq.org/docs/01_introducing_selenium.jsp [30.06.2017]

[24]http://media.techtarget.com/tss/static/articles/content/StrutsFastTrack/StrutsFastTrack.pdf [30.06.2017]

[25]https://www.brainvire.com/six-benefits-of-using-mvc-model-for-effective-web-application-development/ [30.06.2017]

[26] Myers, Glenford J. The art of software testing. John Wiley & Sons, 2006.

http://ufrsciencestech.u-bourgogne.fr/master1/Qualite-Innovation-PI_archives/QUALITE%20INNOVATION%20M1%20STIC%20V2015/1_Qualit%C3%A9/TD%20AQL/The%20Art%20of%20Software%20Testing%20-%20Second%20Edition.pdf [30.06.2017]