**UNIVERSITATEA POLITEHNICA TIMIȘOARA**

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

SPECIALIZAREA INGINERIA SISTEMELOR

**LUCRARE DE LICENȚĂ**

**APLICAȚIE WEB PENTRU GESTIONAREA PLANURILOR DE ÎNVĂȚĂMÂNT**

**Candidat: Daniel-Ionuț MÎRZA**

**Coordonator științific: Conf. dr. ing. Ciprian-Bogdan CHIRILĂ**

Sesiunea: Februarie 2025

**CUPRINS**

1. INTRODUCERE
2. TEHNOLOGII WEB
   1. Limbajul de programare Java
   2. Framework Spring Boot
   3. Biblioteca Apache POI
   4. Sistem de gestionare de date MySQL
   5. Limbajul de programare Thymeleaf
   6. Bootstrap 5
3. CERINȚA ȘI SPECIFICAȚIILE PROIECTULUI
   1. Cerințe funcționale
   2. Cerințe nefuncționale
   3. Cerințe tehnice
4. PROIECTAREA
5. IMPLEMENTAREA
6. TESTARE ȘI INSTALARE
7. CONCLUZII
8. BIBLIOGRAFIE

# INTRODUCERE

Într-un mediu academic în continuă schimbare, gestionarea eficientă a informațiilor referitoare la planurile de învățământ reprezintă o provocare semnificativă. Lucrarea de licență intitulată *"Aplicație web pentru gestionarea planurilor de învățământ"* își propune să răspundă acestei provocări prin dezvoltarea unei soluții software care să automatizeze și să simplifice procesul de administrare a datelor academice.

Aplicația este construită folosind tehnologii moderne, cum ar fi Java, Apache POI, MySQL, Spring Boot și Thymeleaf/Angular, oferind o platformă scalabilă și intuitivă pentru utilizatori. Dezvoltarea aplicației se bazează pe procesarea fișierelor Excel și Word existente, care conțin planurile de învățământ și fișele de disciplină, asigurând astfel o integrare eficientă a datelor.

Proiectul include trei module principale:

1. **Modul administrator** – permite autentificarea, importul planurilor de învățământ, gestionarea fișelor de disciplină și alocarea cadrelor didactice la discipline.
2. **Modul cadru didactic** – oferă posibilitatea autentificării, vizualizării și încărcării de fișe de disciplină.
3. **Modul student** – facilitează accesul la fișele de disciplină filtrate după generație, ciclu de studii și an de studiu.

Lucrarea evidențiază procesul de dezvoltare al aplicației, pornind de la modelarea bazei de date relaționale și integrarea fișierelor de tip Excel și Word, până la implementarea funcționalităților de administrare și utilizare practică.

Prin această aplicație, se dorește crearea unei platforme care să îmbunătățească experiența utilizatorilor și să reducă eforturile administrative, contribuind astfel la o gestionare mai eficientă a resurselor academice.

In capitolul 2 prezentam tehnologiile web folosite in proiect. In capitolul 3 descriem specificatiile proiectului...

1. **TEHNOLOGII WEB**
   1. **Limbajul de programare Java**

Java este un limbaj de programare orientată pe obiecte, bazată pe clase care este proiectată să aibă cât mai puține *dependențe* posibile. Este renumită pentru portabilitatea sa, oferind programatorilor de a scrie cod care poate rula pe orice platformă ce suportă Java fără a necesita recompilare, conform sloganului „Write Once, Run Anywhere” (WORA) [XXX].

Introducere

Limbajul de programare Java a fost creat de Mike Sheridan, Patrik Naughton și James Gosling, cel din urmă cunoscut drept "părintele Java". Acesta a dezvoltat limbajul în timp ce lucra pentru compania Sun Microsystems, începând din 1991. Proiectul a fost inițial cunoscut sub numele de cod "Oak", iar scopul său era crearea unui limbaj portabil pentru dispozitive electronice. În cele din urmă, limbajul a fost redenumit "Java" și a fost lansat public în 1995.

Scurta istorie

James Gosling, un informatician de origine canadiana, este apreciat pentru contribuția sa semnificativă la tehnologia modernă a calculatoarelor și a fost influențat de alte limbaje precum C și C++, dar a dorit să creeze un limbaj mai simplu, mai sigur și orientat pe obiecte. Sub conducerea sa, Java a devenit unul dintre cele mai populare limbaje de programare din lume, folosit pe scară largă în dezvoltarea de aplicații pentru servere, desktop, dispozitive mobile și web.

După plecarea sa de la Sun Microsystems (care a fost achiziționată de Oracle în 2010), Gosling a continuat să lucreze în industrie și să contribuie la diverse proiecte tehnologice. Astăzi, el este considerat o figură emblematică în domeniul programării.

Parintele Java

Java este recunoscută pentru securitatea sa robustă, care este asigurată printr-o combinație de caracteristici integrate și mecanisme avansate. Printre cele mai importante măsuri se numără sandboxing-ul, care izolează codul și previne accesul neautorizat la resurse, precum și verificarea bytecode-ului de către JVM, care asigură integritatea și conformitatea codului. Un alt element central este Security Manager-ul, care controlează accesul aplicațiilor la fișiere, rețele și alte resurse critice.

Securitate

Java este un limbaj de programare cu mai multe caracteristici remarcabile care îl fac ideal pentru dezvoltarea de aplicații variate. În primul rând, Java este orientat pe obiecte, ceea ce înseamnă că aproape totul este definit ca un obiect, cu excepția tipurilor de date primitive. Această abordare face ca programele să fie mai ușor de întreținut și de extins, datorită modularității și reutilizării codului.

Intro OOP

Programarea orientată pe obiecte (OOP) este un model de dezvoltare software care organizează codul în jurul conceptelor de **clase** și **obiecte**. O **clasă** reprezintă o schiță sau un model abstract care definește atributele și comportamentele specifice unui tip de entitate. Pe baza acestei clase, se pot crea obiecte, care sunt instanțe concrete ce conțin valori specifice și pot efectua acțiuni definite de comportamentele clasei.

Clase si Obiecte

Un alt concept fundamental este **moștenirea**, care permite unei clase să preia atributele și metodele unei alte clase, sprijinind astfel reutilizarea codului și crearea unor relații ierarhice între clase. Acest mecanism facilitează extinderea funcționalităților fără a modifica clasele existente, permițând dezvoltarea software-ului într-un mod modular și flexibil.

Mostenire

Pentru a organiza mai bine codul, Java utilizează **pachetele**, care grupează clasele și interfețele în module logice. Acestea ajută la structurarea aplicațiilor mari și la evitarea conflictelor între numele claselor din diferite părți ale unui proiect sau din biblioteci externe.

Pachete

Un alt principiu esențial al OOP este **încapsularea**, care protejează datele interne ale unei clase prin ascunderea acestora și furnizarea unui acces controlat prin metode specifice. Acest lucru contribuie la menținerea securității și integrității datelor și permite modificarea logicii interne fără a afecta alte părți ale aplicației.

Incapsulare

Prin combinarea acestor concepte, programarea orientată pe obiecte permite crearea unor aplicații bine structurate, ușor de întreținut și scalabile. Fiecare concept contribuie la organizarea și gestionarea complexității, asigurând un proces de dezvoltare mai eficient și mai robust.

Concluzie OOP

O altă caracteristică esențială este independența platformei. Programele Java sunt compilate într-un format intermediar numit bytecode, care poate fi executat pe orice platformă ce suportă Java Virtual Machine (JVM). Această abilitate este cunoscută prin sloganul „Write Once, Run Anywhere”.

Indepententa platformei

Limbajul suportă multitasking prin funcționalități de multi-threading, permițând rularea simultană a mai multor fire de execuție, ceea ce duce la performanțe crescute în aplicații complexe.

Multi-threding

Biblioteca standard Java este extrem de bogată, oferind suport pentru o gamă largă de funcționalități, de la manipularea colecțiilor de date până la lucrul cu rețele. Toate aceste caracteristici fac din Java un limbaj versatil, utilizat pe scară largă în dezvoltarea de aplicații desktop, mobile, web și enterprise.

Biblioteca bogata

Java este împărțit în mai multe ediții, fiecare concepută pentru un anumit tip de aplicații și utilizatori. Cele trei ediții principale sunt Java Standard Edition (SE), Java Enterprise Edition (EE), și Java Micro Edition (ME). Fiecare dintre acestea are un set specific de funcționalități și biblioteci, adaptate cerințelor mediilor în care sunt utilizate.

Versiuni Java

**Java Standard Edition** (SE) este ediția de bază a platformei Java și oferă toate funcționalitățile necesare pentru dezvoltarea aplicațiilor independente sau desktop. Este cea mai utilizată ediție pentru învățarea limbajului Java și acoperă toate caracteristicile fundamentale ale acestuia.

Java SE

**Java Entreprise Edition** (EE), cunoscut acum ca Jakarta EE, este construit pe baza Java SE și extinde funcționalitățile acestuia pentru a permite dezvoltarea de aplicații complexe, distribuite și scalabile, utilizate în mediul enterprise.

Java EE

**Java Micro Edition** (ME) este o ediție ușoară și specializată a platformei Java, destinată dispozitivelor cu resurse limitate, cum ar fi telefoanele mobile, sistemele integrate (embedded systems), și dispozitivele IoT (Internet of Things).

Java ME

Java Persistence API (JPA) este un standard Java pentru gestionarea persistenței datelor, adică pentru stocarea și accesarea obiectelor Java într-o bază de date relațională. Este parte a platformei Java EE (Enterprise Edition), dar poate fi utilizată și în aplicații Java SE (Standard Edition). JPA oferă o modalitate de a interacționa cu bazele de date folosind concepte orientate pe obiecte, eliminând necesitatea de a scrie SQL manual.

JPA

Unul dintre principalele beneficii este utilizarea mapării obiect-relaționale (ORM), care asociază clasele Java cu tabelele din baza de date și atributele acestora cu coloanele corespunzătoare. Aceasta se realizează cu ajutorul adnotărilor precum @Entity pentru declararea entităților persistente, @Table pentru specificarea tabelului asociat sau @Id pentru identificatori unici.

Un alt aspect important al JPA este suportul său pentru un limbaj de interogare asemănător SQL, denumit JPQL (Java Persistence Query Language), care permite efectuarea interogărilor pe obiecte, păstrând însă un nivel ridicat de abstractizare. De asemenea, JPA gestionează automat ciclul de viață al entităților, de la crearea acestora și atașarea la contextul persistent, până la ștergere.

Caracteristici JPA

Un program simplu Java constă în general dintr-o clasă care conține o metodă principală (main), unde începe execuția programului.

O imagine care conține text, captură de ecran, software, Software multimedia

Descriere generată automat

public class HelloWorld {

Acesta definește o clasă numită HelloWorld. În Java, toate programele sunt organizate în clase, care reprezintă unități fundamentale ale codului. Cuvântul cheie public indică faptul că această clasă poate fi accesată de oriunde în program.

public static void main(String[] args) {

Aceasta este metoda principală care reprezintă punctul de intrare al programului. Fiecare aplicație Java pornește execuția din metoda main.

* public înseamnă că metoda poate fi apelată de JVM din afara clasei.
* static indică faptul că metoda aparține clasei și poate fi rulată fără a crea o instanță a clasei.
* void semnalează că metoda nu returnează nicio valoare.
* String[] args este un parametru care permite transmiterea de argumente către program din linia de comandă.

System.out.println("Salut, lume!");

Această linie utilizează metoda println din clasa System.out pentru a afișa textul „Salut, lume!” urmat de un rând nou în consolă.

* System.out este un obiect standard pentru ieșire, folosit pentru a trimite mesaje către consolă.
* println este metoda utilizată pentru afișarea mesajului și adăugarea unui caracter de rând nou la final.

Când acest cod este executat, rezultatul va fi:

Salut, lume!

Program simplu

Java este un limbaj de programare versatil, sigur și scalabil, care s-a impus ca un standard în dezvoltarea software. De-a lungul anilor, a evoluat pentru a satisface cerințele unor domenii diverse, de la aplicații enterprise și dezvoltare web, până la soluții pentru dispozitive mobile și IoT. Caracteristicile sale precum portabilitatea, gestionarea automată a memoriei și orientarea pe obiecte îl fac ușor de utilizat și de înțeles, fiind ideal atât pentru începători, cât și pentru profesioniști.

Scurta concluzie

* 1. **Biblioteca Spring Boot**

Spring Boot este un framework Java care facilitează crearea și execuția aplicațiilor Java. Simplifică procesul de configurare și setare, permițându-le developărilor să se focuseze mai mult pe scrierea de cod a aplicațiilor.

Intro

Spring Boot este dezvoltată de către Pivotal Team și este o combinație dintre Spring Framework și Embedded Servers. În octombrie 2012 un client pe nume Mike Youngstrom a făcut o cerere Jira cerând „bootstraping the spring framework” pentru a putea porni mai repede. Astfel, la începutul lui 2013 Spring Boot a fost creat.

Scurta istorie

Printre avantajele principale ale utilizării acestui framework se numără configurarea automată, care detectează librăriile incluse în proiect și configurează automat componentele necesare. Acest lucru elimină necesitatea unei configurări manuale complexe. De asemenea, Spring Boot permite crearea de aplicații autonome, care pot fi rulate direct ca fișiere JAR, fără a necesita un server extern, deoarece acesta este inclus în aplicație. O altă caracteristică importantă este utilizarea Starter POMs, care ajută la gestionarea simplificată a dependințelor prin includerea unui singur modul, evitând configurările complicate. În plus, Spring Boot Actuator oferă funcționalități avansate pentru monitorizarea și diagnosticarea aplicațiilor.

Avantaje

Din punct de vedere al componentelor de bază, Spring Boot include Starter Templates, care sunt seturi predefinite de dependințe pentru diferite tipuri de aplicații. Configurarea aplicației este simplificată prin utilizarea fișierelor application.properties sau application.yml, eliminând nevoia de a lucra cu fișiere XML lungi și complicate. O altă componentă utilă este Spring Boot Initializr, o interfață web ce permite generarea rapidă a proiectelor preconfigurate, oferind utilizatorilor opțiunea de a selecta dependințele, versiunea Java și alte detalii. Aceste caracteristici fac din Spring Boot un instrument ideal pentru dezvoltarea rapidă și eficientă a aplicațiilor Java.

Caracteristici

Crearea unei aplicații Spring Boot începe prin configurarea proiectului pe site-ul **Spring Initializr**, unde dezvoltatorul alege setările necesare, precum sistemul de build (Maven), limbajul (Java) și dependențele esențiale (Spring Web, Spring Data JPA, H2 Database). Proiectul descărcat este apoi importat într-un IDE, iar Maven se ocupă de gestionarea automată a dependențelor. Structura proiectului este deja configurată, incluzând foldere pentru cod sursă, fișiere de configurare și specificațiile Maven.

Pentru a adăuga funcționalitate, dezvoltatorul creează un controller simplu, o clasă Java adnotată cu *@RestController* și *@GetMapping*, care gestionează cererile HTTP și răspunde cu mesaje predefinite. Configurarea suplimentară a aplicației se face în fișierul *application.properties*, unde se pot personaliza detalii precum portul serverului și numele aplicației.

O imagine care conține text, captură de ecran, software, Software multimedia

Descriere generată automat

După ce aplicația este pornită prin rularea clasei principale, aceasta poate fi testată prin accesarea unei rute definite. În timp, funcționalitățile pot fi extinse prin adăugarea de persistare a datelor, servicii pentru logica aplicației și teste pentru validarea codului, transformând aplicația inițială într-un proiect scalabil și bine structurat.

Program simplu Spring Boot

Spring și Spring Boot sunt două instrumente din ecosistemul Spring, dar au scopuri diferite. Spring este un framework general pentru dezvoltarea aplicațiilor Java, oferind multă flexibilitate, dar necesită configurări manuale extinse. Pe de altă parte, Spring Boot simplifică procesul de dezvoltare prin configurări implicite și este ideal pentru aplicații standalone, gata de producție.

Când vine vorba de lansare și rulare, aplicațiile dezvoltate cu Spring au nevoie de un server extern, cum ar fi Tomcat, pentru a putea rula. În schimb, Spring Boot permite rularea aplicațiilor ca standalone, având servere încorporate, astfel încât aplicațiile pot fi lansate direct cu o simplă comandă java -jar.

În final, Spring necesită mai mult timp pentru configurare, dar oferă flexibilitate maximă, în timp ce Spring Boot este ideal pentru lansarea rapidă a aplicațiilor. Alegerea între cele două depinde de specificul proiectului: Spring este potrivit pentru aplicații complexe, în timp ce Spring Boot este soluția preferată pentru proiecte moderne și rapide.

Spring vs Spring Boot

Spring Boot este un framework modern și eficient, creat pentru a simplifica dezvoltarea aplicațiilor Java. Cu un sistem de configurare automat, suport pentru microservicii, integrare perfectă cu ecosistemul Spring și un set vast de funcționalități, Spring Boot accelerează procesul de dezvoltare și reduce complexitatea. Este o soluție ideală atât pentru aplicații mici, cât și pentru sisteme complexe și scalabile, fiind susținut de o comunitate activă și documentație extinsă. Alegerea Spring Boot pentru un proiect asigură flexibilitate, performanță și ușurință în utilizare, adaptându-se perfect cerințelor moderne de dezvoltare software. Concluzie

* 1. **Biblioteca Apache POI**

Apache POI este o bibliotecă Java utilizată pentru a citi și scrie fișiere Microsoft Office, inclusiv formate precum Excel (XLS, XLSX), Word (DOC, DOCX) și PowerPoint (PPT, PPTX). Este foarte utilă în aplicații care necesită manipularea documentelor Office în mod programatic, fie că este vorba de generarea rapoartelor în format Excel, modificarea documentelor Word sau crearea de prezentări PowerPoint.

Intro

Apache POI a fost dezvoltat ca parte a **Apache Software Foundation** pentru a oferi o soluție open-source destinată manipulării fișierelor Microsoft Office. Proiectul a început în anii 2000, cu obiectivul inițial de a crea un API pentru citirea și scrierea fișierelor Excel în formatul vechi (.xls). Aceasta a fost realizată prin API-ul **HSSF** (Horrible Spreadsheet Format), numele său având o notă ironică, caracteristică proiectelor din acea perioadă.

Istorie

Pe măsură ce Microsoft a introdus formatele bazate pe XML odată cu lansarea Office 2007, Apache POI și-a extins funcționalitățile pentru a include suport pentru noile formate (.xlsx, .docx, .pptx). Aceasta a condus la dezvoltarea API-urilor **XSSF** pentru Excel, **XWPF** pentru Word și **XSLF** pentru PowerPoint.

Istorie II

Apache POI oferă funcționalități extinse pentru manipularea fișierelor Microsoft Office. În ceea ce privește fișierele Excel, suportă atât formatele mai vechi (.xls) prin API-ul HSSF (Horrible Spreadsheet Format), cât și cele moderne (.xlsx) prin API-ul XSSF (XML Spreadsheet Format). Pentru documentele Word, bibliotecile HWPF și XWPF permit lucrul cu fișierele DOC și DOCX, respectiv. Similar, fișierele PowerPoint sunt gestionate prin HSLF pentru PPT și XSLF pentru PPTX. În plus, POI facilitează manipularea obiectelor precum foi de calcul, celule, paragrafe, tabele, imagini și stiluri, fiind astfel potrivit pentru o gamă variată de aplicații.

Caracteristici

Apache POI este un proiect open-source, ceea ce îl face gratuit și accesibil oricui. Este compatibil cu multiple formate Microsoft Office, oferind flexibilitate în manipularea fișierelor din diferite versiuni de Office. În plus, API-ul său este bine documentat și susținut de o comunitate activă, ceea ce facilitează învățarea și integrarea sa în proiecte software.

Avantaje

Exemplul următor demonstrează modul de utilizare a bibliotecii Apache POI pentru a citi și scrie fișiere Excel în format *.xlsx*. Procesul începe prin importul și deschiderea fișierului folosind un flux de intrare (*FileInputStream*). Fișierul este apoi încărcat într-un obiect *XSSFWorkbook*, care reprezintă întreaga structură a documentului Excel. Accesul la o anumită foaie se face fie prin index (de exemplu, *getSheetAt(0)* pentru prima foaie), fie prin nume.

Pentru citirea unei valori, este accesat rândul dorit folosind metoda *getRow()*, iar celula corespunzătoare este obținută cu *getCell()*. Valoarea este extrasă sub forma unui șir de caractere prin metoda *getStringCellValue()*. Dacă celula conține date de alt tip, cum ar fi numerice, se folosește metoda *getNumericCellValue()*.

O imagine care conține text, captură de ecran, software, Software multimedia

Descriere generată automat

Scrierea unei valori într-o celulă implică fie crearea unei celule noi cu *createCell()*, fie suprascrierea unei celule existente. Valoarea este apoi setată prin metoda *setCellValue()*.

Modificările aduse fișierului sunt salvate prin deschiderea unui flux de ieșire (*FileOutputStream*) și apelarea metodei *write()* a obiectului *XSSFWorkbook*. După ce modificările sunt scrise, fluxurile de intrare și ieșire sunt închise, împreună cu workbook-ul, pentru a elibera resursele utilizate.

Pentru utilizarea bibliotecii Apache POI într-un proiect, trebuie să incluzi dependința corespunzătoare în fișierul de configurare al proiectului, cum ar fi *pom.xml* pentru Maven. Gestionarea excepțiilor este esențială, deoarece operațiile de citire și scriere implică riscul de erori legate de I/O.

Exemplu de utilizare POI

Apache POI reprezintă o soluție robustă și flexibilă pentru manipularea fișierelor Microsoft Office în cadrul aplicațiilor Java. Prin funcționalitățile sale, permite citirea, scrierea și gestionarea completă a datelor din formatele Excel, Word și PowerPoint, atât în versiunile clasice, cât și în cele moderne bazate pe XML.

Concluzie

* 1. **Sistem de gestionare de date MySQL**

MySQL este un sistem de gestionare a bazelor de date relaționale (RDBMS) foarte popular și puternic, utilizat pentru a stoca și gestiona date. Este open-source și este folosit în mod frecvent în aplicații web, fiind compatibil cu diferite sisteme de operare, precum Linux, Windows și macOS.

Intro

O bază de date este o colecție organizată de informații sau date care sunt stocate și gestionate electronic, de obicei într-un sistem de gestionare a bazelor de date (DBMS – Database Management System). Baza de date permite stocarea, manipularea, accesarea și actualizarea eficientă a datelor.

Despre baza de date

MySQL a fost lansat oficial în 1995 de către Michael Widenius, David Axmark și Allan Larsson, fondatorii companiei MySQL AB din Suedia, fiind dezvoltat inițial ca un sistem de baze de date rapid și ușor de utilizat pentru aplicații web. În anul 2000, MySQL a devenit open-source sub licența GNU GPL, ceea ce a contribuit semnificativ la creșterea popularității sale în rândul dezvoltatorilor.

În 2008, compania MySQL AB a fost achiziționată de Sun Microsystems pentru suma de 1 miliard USD, marcând o etapă importantă în evoluția produsului prin adăugarea de resurse și investiții. Doi ani mai târziu, în 2010, Sun Microsystems a fost cumpărată de Oracle Corporation, care a preluat și dezvoltarea MySQL. Deși această schimbare a provocat controverse în comunitatea open-source, Oracle a continuat să îmbunătățească MySQL.

O etapă semnificativă în istoria MySQL a avut loc în 2013, odată cu lansarea versiunii 5.6, care a introdus îmbunătățiri majore în performanță, replicare și optimizarea interogărilor. În 2021, versiunea MySQL 8.0 a devenit standardul pentru cele mai noi funcționalități, incluzând suport pentru JSON, optimizări în performanță și administrare avansată.

Istoria MySQL

Numele MySQL provine din combinația cuvântului „SQL” (Structured Query Language), limbajul standard utilizat pentru gestionarea bazelor de date, și „My”, care face referire la fiica lui Michael Widenius (unul dintre fondatorii MySQL), al cărei prenume este „My”. Această combinație reflectă atât o notă personală, cât și un accent tehnic, ceea ce a contribuit la unicitatea brandului.

Denumire MySQL

Logo-ul MySQL este reprezentat de o delfin albastru, simbolizând agilitatea, viteza și ușurința de utilizare – trăsături care caracterizează acest sistem de gestionare a bazelor de date. Delfinul din logo este numit „Sakila”, nume ales de comunitatea MySQL printr-un concurs organizat de dezvoltatori.

Logo-ul MySQL

MySQL este un sistem de gestionare a bazelor de date relaționale (RDBMS), care organizează datele în tabele ce conțin rânduri și coloane. Utilizează limbajul SQL (Structured Query Language) pentru interogare și gestionare. Este recunoscut pentru performanța ridicată, oferind viteze mari de procesare a datelor și gestionând volume mari de informații. De asemenea, MySQL include controale de acces avansate și criptare pentru securitate, fiind capabil să gestioneze baze de date de dimensiuni mari. Este compatibil cu diverse limbaje de programare, cum ar fi PHP, Java și Python, ceea ce îl face ușor de integrat în diferite tipuri de aplicații.

Caracteristici

MySQL este un instrument open-source, ceea ce înseamnă că este gratuit și beneficiază de sprijinul unei comunități mari care contribuie la îmbunătățirea sa constantă. Este compatibil cu mai multe platforme, inclusiv Windows, Linux și macOS, ceea ce îl face extrem de versatil. Interfața prietenoasă și instrumentele ușor de utilizat îl fac accesibil pentru dezvoltatorii aflați la început de drum. Totodată, MySQL este recunoscut pentru simplitatea configurării și implementării, oferind performanță ridicată la un cost redus.

Avantaje

SQL și MySQL reprezintă concepte diferite, deși sunt strâns legate. SQL este un limbaj standardizat utilizat pentru gestionarea și interogarea bazelor de date relaționale. Este un instrument teoretic și practic care permite dezvoltatorilor să creeze, să actualizeze, să șteargă și să interogheze datele din tabelele bazelor de date. Printre comenzile SQL comune se numără CREATE TABLE pentru crearea tabelelor, SELECT pentru extragerea datelor, INSERT pentru adăugarea de date noi, UPDATE pentru actualizarea datelor existente și DELETE pentru ștergerea acestora. SQL este un limbaj universal, independent de platformă, utilizat de toate sistemele de gestionare a bazelor de date relaționale (RDBMS), inclusiv MySQL, Oracle, PostgreSQL și Microsoft SQL Server.

Pe de altă parte, MySQL este un sistem de gestionare a bazelor de date relaționale (RDBMS) care implementează limbajul SQL. MySQL este un software concret utilizat pentru stocarea, gestionarea și accesarea datelor. Spre deosebire de SQL, care este doar un standard teoretic, MySQL oferă o platformă practică și completă pentru administrarea bazelor de date, incluzând funcții avansate precum securitatea, replicarea și optimizarea performanței. MySQL este specific, fiind dezvoltat ca un software concret care extinde standardul SQL pentru a include caracteristici practice necesare gestionării bazelor de date.

MySQL vs SQL

MySQL este un sistem de gestionare a bazelor de date relaționale (RDBMS) care utilizează limbajul standard SQL pentru a stoca, administra și accesa datele. Dezvoltat inițial ca un instrument rapid, simplu și open-source, MySQL a devenit un standard de facto în industria IT datorită versatilității, performanței ridicate și compatibilității cu diverse platforme și limbaje de programare. Cu o istorie solidă și sprijin din partea comunității open-source, dar și al companiei Oracle, MySQL este folosit pe scară largă în aplicații web, sisteme de management al conținutului și platforme critice, fiind apreciat pentru combinația sa de accesibilitate și funcționalitate avansată.

Concluzie

* 1. **Biblioteca Thymeleaf**

Thymeleaf este un motor de template Java folosit pentru a genera cod HTML, XML, JavaScript, CSS și text. Este des utilizat în aplicațiile de tip Spring Boot pentru a crea interfețe de utilizator dinamice și intuitive. Thymeleaf permite generarea de pagini web pe partea de server și poate fi integrat cu ușurință în arhitectura MVC (Model-View-Controller), fiind destinat în special pentru proiectele de tip server-side rendering.

Intro

Thymeleaf are scopul principal de a oferi un mod elegant și foarte ușor de întreținut de a crea șabloane. Pentru a realiza acest lucru se bazează pe conceptul de Șabloane Naturale pentru a-și insera logica în fișierele șablon într-un mod în care să nu afecteze șablonul de la a fi folosit ca un prototip de design. Acest lucru îmbunătățește comunicarea designului și reduce decalajul dintre echipele de proiectare și dezvoltare.

Scop si concept

Thymeleaf a fost lansat pentru prima dată în 2011 de \*\*Daniel Fernández\*\*, având ca scop principal să ofere un motor de template Java care să fie ușor de utilizat și să producă șabloane HTML care să poată fi vizualizate direct în browsere, fără a necesita preprocesare. Inițial, a fost conceput pentru a înlocui motoarele de template existente, cum ar fi JSP (JavaServer Pages), oferind o abordare mai modernă și mai apropiată de standardele HTML.

Popularitatea Thymeleaf a crescut odată cu adoptarea pe scară largă a framework-ului Spring Boot, datorită integrării sale native și a ușurinței cu care dezvoltatorii puteau construi interfețe de utilizator dinamice. De-a lungul anilor, Thymeleaf a evoluat, adăugând suport pentru noi funcționalități, precum manipularea XML, reutilizarea fragmentelor și suportul pentru diverse expresii logice și condiționale.

Istorie Thymeleaf

Thymeleaf se remarcă printr-o sintaxă care se aseamănă mult cu HTML-ul, ceea ce îl face ușor de citit și întreținut. Se integrează fără probleme cu Spring Framework, oferind suport nativ pentru Spring MVC, iar această compatibilitate îl face o alegere populară în proiectele bazate pe Java. De asemenea, poate fi utilizat atât în mod standalone, cât și în aplicații web care rulează pe partea de server. O altă caracteristică importantă este capacitatea de a reutiliza fragmente de template, ceea ce simplifică organizarea și modularitatea codului. În plus, Thymeleaf permite utilizarea expresiilor logice, cum ar fi condiții și bucle, direct în șabloanele HTML.

Caracteristici

Avantajele principale ale Thymeleaf includ simplitatea și ușurința de utilizare, mai ales pentru cei care sunt deja familiarizați cu HTML. Datorită integrării native cu Spring Boot, dezvoltarea aplicațiilor web devine mai rapidă și mai eficientă. Thymeleaf facilitează generarea dinamică a paginilor pe partea de server și oferă posibilitatea de a reutiliza codul prin fragmentarea șabloanelor, ceea ce contribuie la o mai bună organizare a proiectului. În același timp, codul produs este ușor de întreținut și scalabil, fiind potrivit și pentru aplicații complexe.

Avantaje

Thymeleaf este o alegere excelentă pentru aplicațiile web care rulează pe partea de server și care sunt dezvoltate în Java, în special cele care folosesc framework-ul Spring Boot. Este ideal pentru proiectele în care colaborarea între dezvoltatori și designeri este esențială, deoarece permite crearea de șabloane care pot fi vizualizate direct în browser fără preprocesare. De asemenea, este potrivit pentru aplicațiile care necesită generarea dinamică de conținut și manipularea avansată a formularelor.

Cand sa folosesti Thymeleaf

Cu toate acestea, Thymeleaf are și câteva limitări. Nu este la fel de performant pentru aplicațiile single-page (SPA), unde framework-uri precum React sau Angular sunt mai potrivite. De asemenea, nu este recomandat pentru aplicațiile web care se bazează exclusiv pe logica de prezentare pe partea de client. În proiectele extrem de mari, unde există o complexitate ridicată în gestionarea logicii de prezentare, utilizarea sa poate deveni mai puțin eficientă.

Limitari

Thymeleaf este compatibil cu diverse tipuri de conținut, inclusiv HTML5, XML și JavaScript, ceea ce îl face potrivit pentru o gamă largă de proiecte. De asemenea, este extensibil, permițând adăugarea de funcționalități personalizate prin plugin-uri și dialecte proprii, ceea ce îl face extrem de flexibil și adaptabil la cerințele complexe ale proiectelor mari.

Compatibilitate si extensibilitate

Ecosistemul Thymeleaf este bine dezvoltat, incluzând extensii și module populare. De exemplu, **thymeleaf-layout-dialect** simplifică gestionarea layout-urilor și reutilizarea fragmentelor, în timp ce **thymeleaf-extras-java8time** oferă suport pentru manipularea datelor temporale din Java 8. Pentru securitate avansată, **thymeleaf-extras-springsecurity5** integrează funcționalități dedicate autentificării și autorizării. Aceste resurse adaugă un plus de putere și versatilitate Thymeleaf, făcându-l ideal pentru proiectele care necesită soluții personalizate.

Ecosistemul

Thymeleaf este un motor de template puternic și flexibil, ideal pentru aplicațiile Java server-side, în special cele construite cu Spring Boot. Datorită sintaxei sale naturale și integrării perfecte cu Spring, este o alegere excelentă pentru crearea de interfețe web dinamice și gestionarea formularelor. Deși nu este optim pentru aplicațiile single-page (SPA) și poate deveni mai puțin eficient în proiectele mari cu logică complexă de prezentare, Thymeleaf rămâne o soluție robustă pentru majoritatea aplicațiilor tradiționale Java. Sursele oficiale și tutorialele online oferă suport suplimentar pentru învățarea și utilizarea eficientă a acestui motor de template.

Concluzie

* 1. **Bootstrap 5**

Bootstrap 5 este cea mai recentă versiune a popularului framework front-end Bootstrap. Acesta este utilizat pentru a dezvolta interfețe web responsive și moderne cu mai puțin efort și cod personalizat. Versiunea 5 aduce îmbunătățiri semnificative, inclusiv renunțarea la dependența de jQuery, un design mai performant și o flexibilitate crescută.

Intro

Bootstrap 5 este cea mai recentă versiune a framework-ului Bootstrap, lansată oficial în mai 2021, continuând tradiția acestuia de a simplifica dezvoltarea web. Bootstrap a fost creat inițial de dezvoltatorii de la Twitter în 2011, ca un framework intern pentru a uniformiza interfețele.Bootstrap 1.0 a introdus sistemul de grilă și un set de stiluri predefinite pentru crearea de interfețe consistente. În 2012, Bootstrap 2 a adus design-ul responsive, făcând paginile web să se adapteze automat la diferite dimensiuni de ecran.

În 2013, Bootstrap 3 s-a concentrat pe abordarea mobile-first, oferind un sistem de grilă mai flexibil, ideal pentru dispozitivele mobile. Lansarea din 2018, Bootstrap 4, a făcut tranziția la Sass, un preprocesor CSS avansat, și a introdus variabile CSS și Flexbox pentru un control mai mare asupra layout-urilor. În cele din urmă, Bootstrap 5, lansat în 2021, a eliminat dependența de jQuery, a adăugat componente moderne precum offcanvas și a optimizat performanța utilizând variabile CSS pentru o personalizare mai simplă.

Istorie Bootstrap

Bootstrap 5 aduce o serie de îmbunătățiri notabile care îl fac mai eficient și mai flexibil pentru dezvoltarea interfețelor web. În primul rând, această versiune renunță la dependența de jQuery, ceea ce reduce dimensiunea proiectelor și crește performanța aplicațiilor. Sistemul de grilă a fost actualizat pentru a include cinci puncte de întrerupere, oferind o mai mare flexibilitate în crearea designurilor responsive. De asemenea, sunt introduse noi componente, precum offcanvas, iar componentele existente au fost simplificate și optimizate.

Un alt aspect important este utilizarea variabilelor CSS, care permite personalizarea temelor și a culorilor într-un mod mai simplu și mai eficient. Performanța generală a fost îmbunătățită, cu un consum mai redus de resurse ale browserului. În plus, documentația a fost extinsă și îmbunătățită, fiind mai ușor de utilizat pentru dezvoltatori, indiferent de nivelul lor de experiență.

Avantaje

Bootstrap 5 reprezintă un pas important în evoluția dezvoltării web, oferind un framework modern, performant și ușor de personalizat. Prin renunțarea la jQuery, introducerea variabilelor CSS și a unor componente noi precum offcanvas, această versiune răspunde cerințelor actuale ale dezvoltatorilor. Sistemul de grilă flexibil, tematicile personalizabile și suportul îmbunătățit pentru tehnologii moderne îl fac o alegere excelentă pentru proiecte responsive și elegante. Cu o documentație bine organizată și resurse disponibile, Bootstrap 5 este un instrument esențial pentru crearea de interfețe web rapide și atractive.

Concluzie

1. **CERINȚA ȘI SPECIFICAȚIILE PROIECTULUI**

Proiectarea aplicației web pentru gestionarea planurilor de învățământ reflectă un set bine definit de cerințe funcționale, nefuncționale și tehnice, fiecare având un rol esențial în realizarea unei soluții robuste, eficiente și scalabile.

* 1. **Cerințe funcționale**

Aplicația trebuie să ofere funcționalități adaptate rolurilor de utilizator, pentru a automatiza procesele legate de planurile de învățământ și fișele de disciplină. Administratorii trebuie să aibă posibilitatea de a se autentifica în aplicație, de a importa planuri de învățământ din fișiere Excel, de a genera fișe de disciplină pe baza șabloanelor Word existente și de a aloca cadre didactice la discipline. De asemenea, aceștia vor putea copia fișele disciplinelor în generații noi pentru actualizarea anuală a planurilor.

Pentru cadrele didactice, aplicația va permite autentificarea, vizualizarea fișelor de disciplină asociate și încărcarea unor fișe noi. În cazul studenților, aplicația va oferi acces la vizualizarea fișelor de disciplină filtrate în funcție de generație, ciclu de studiu (licență sau master) și an academic.

Cerințele funcționale se concentrează pe oferirea unui set clar de funcționalități destinate diferitelor categorii de utilizatori (administratori, cadre didactice, studenți), având ca scop principal automatizarea proceselor administrative. Acestea includ importul fișierelor Excel, gestionarea fișelor de disciplină și asigurarea unei interfețe intuitive pentru utilizatori.

* 1. **Cerințe nefuncționale:**

Aplicația trebuie să fie performantă, oferind timpi de răspuns rapizi, chiar și în cazul procesării unor fișiere mari. Arhitectura acesteia trebuie să permită scalabilitate, astfel încât să poată fi extinsă sau modificată pentru a răspunde cerințelor viitoare. Este esențial să se asigure securitatea datelor, prin protecția împotriva accesului neautorizat și utilizarea standardelor moderne pentru autentificare și stocare a parolelor.

Compatibilitatea aplicației cu formatele moderne de fișiere Excel și Word, precum și funcționarea sa corectă pe multiple platforme și browsere, sunt de asemenea cruciale. Aplicația trebuie să fie ușor de utilizat pentru toate categoriile de utilizatori și să gestioneze eficient erorile, oferind mesaje de feedback clare în caz de probleme.

Cerințele nefuncționale asigură calitatea și fiabilitatea aplicației prin standarde de performanță, securitate, scalabilitate și compatibilitate. Astfel, aplicația trebuie să funcționeze eficient chiar și în scenarii complexe, să protejeze datele utilizatorilor și să fie accesibilă pe multiple platforme și browsere moderne.

* 1. **Cerințe tehnice:**

Implementarea aplicației trebuie să se bazeze pe un set bine ales de tehnologii. Backend-ul va fi realizat folosind Java și Spring Boot, iar baza de date relațională va fi construită cu MySQL. Pentru manipularea fișierelor Excel și Word se va utiliza biblioteca Apache POI.

Frontend-ul poate fi dezvoltat fie cu Thymeleaf, pentru o integrare directă cu backend-ul, fie cu Angular, pentru o experiență mai modernă. Aplicația va respecta arhitectura Model-View-Controller (MVC) și va integra API-uri REST pentru a permite comunicarea între componentele frontend și backend. De asemenea, testarea va fi realizată printr-o combinație de teste unitare, de integrare și de performanță, pentru a garanta funcționalitatea, fiabilitatea și calitatea produsului final.

Cerințele tehnice oferă cadrul tehnologic și arhitectural pentru implementare. Alegerea unui stack tehnologic solid (Java, Spring Boot, MySQL, Apache POI, Thymeleaf/Angular) garantează o infrastructură flexibilă și ușor de întreținut, în timp ce arhitectura MVC și utilizarea standardelor moderne (REST API, Spring Security) asigură o dezvoltare modulară și bine structurată.

Prin îmbinarea acestor cerințe, aplicația urmărește nu doar satisfacerea nevoilor curente ale utilizatorilor, ci și crearea unei soluții sustenabile și extensibile, capabile să se adapteze viitoarelor cerințe și schimbări din mediul academic. Acest echilibru între funcționalitate, performanță și tehnologie reprezintă fundamentul pentru succesul proiectului.

**4. PROIECTAREA**