Universitatea Politehnica Timișoara
Facultatea de Automatică și Calculatoare
Departamentul Calculatoare și
Tehnologia Informației

Eficientizarea înscrierii studenților la orele de educație fizică și sport din cadrul UPT

Proiect de diplomă

Absolvent:
Antonyo TOPOLNICEANU

Coordonator ştiinţific:
Conf.dr.ing. Ciprian-Bogdan CHIRILĂ

Timișoara
Septembrie 2023

Cuprins

Capitolul 1

Introducere

1.1 Context

Aplicațiile software au devenit indispensabile într-o varietate de activități și s-au răspândit datorită diferitelor inovații tehnologice, precum dispozitivele mobile (laptop-uri, asistenți digitali personali, smartphone-uri) și infrastructura de rețea (WiFi, GSM), care facilitează comunicarea dintre utilizatori și sistemele informatice în aproape orice context.

Într-o epocă definită de progresul tehnologic rapid, instituțiile de învățământ adoptă soluții digitale pentru a simplifica și optimiza diferite procese. În acest context, prezenta teză introduce o aplicație concepută pentru a efectua, într-un mod simplu, acțiunea de înscriere și gestionare al studenților la un anumit curs într-o facultate.

În mod tradițional, procesul de înscriere la cursuri a fost afectat de proceduri administrative complexe și blocaje de comunicare, adesea îngreunând viața studenților. Acest capitol pune bazele prezentând o privire de ansamblu asupra provocărilor asociate cu sistemele convenționale de înscriere. Apoi trece la obiectivul de bază al acestei cercetări - de a prezenta o aplicație care redefinește experiența de selecție a cursurilor. Alimentată de tehnologie actuală, aplicația se străduiește să ofere o platformă accesibilă și centrată pe utilizator, care conectează fără probleme studenții cu cadrul academic.

1.2 Motivație

Motivația din spatele acestei lucrări provine din necesitatea de a simplifica procedura de înscriere la cursuri din instituțiile academice. Metodele tradiționale impun adesea pași complicați și sisteme învechite, fiind adesea ineficiente. Odată cu progresul rapid al tehnologiei în educație, apare o oportunitate de a moderniza acest proces.

Prin urmare, motivația se concentrează pe îmbunătățirea experienței generale pentru studenții care jonglează cu diferite angajamente. Abordând problema constrângerilor de timp, proiectul contribuie la un proces de înscriere la cursuri mai flexibil, asigurând că studenții își pot gestiona responsabilitățile academice alături de celelalte eforturi.

1.3 Descriere proiect

Lucrarea are la bază o aplicație web centrată pe utilizator. Această aplicație oferă profesorilor puterea de a crea și de a gestiona fără efort sloturile de înscriere la cursuri, alocând intervale de timp limitate pentru înregistrarea studenților. Cu accent pe adaptarea diverselor programe ale studenților, aplicația oferă flexibilitate, permițând profesorilor să desemneze intervale orare specifice săptămânal.

Printr-un sistem de conectare securizat, profesorii au acces la un dashboard unde își pot introduce detaliile cursului, pot defini intervalele de înscriere disponibile și pot stabili limite ale numărului de studenți pe slot. Acest sistem asigură că atât profesorii, cât și studenții au instrumentele necesare pentru a naviga fără probleme în procesul de înscriere.

Pentru studenți, aplicația oferă o interfață ușor de utilizat, care afișează sloturile de înscriere la cursuri disponibile în intervalele săptămânale desemnate. Studenții se pot conecta, pot vedea cursurile oferite și pot selecta intervalele de timp preferate.

În continuare, se va descrie succint structura lucrării.

1.4 Structura lucrării

Scopul acestui subcapitol este de a îmbunătăți înțelegerea și de a facilita navigarea, asigurând o experiență coerentă și cursivă. Lucrarea a fost structurată în șase capitole, fiind redate după cum urmează.

- Capitolul 1: Introducere

  Prezintă contextul si obiectivele acestei lucrări. Această secțiune oferă o viziune de ansamblu, creând scena pentru capitolele ulterioare ale lucrării.

- Capitolul 2: Stadiul actual al domeniului

  Acest capitol reprezintă o explorare a peisajului existent al cunoștințelor și progreselor în domeniu, aferent temei alese.

- Capitolul 3: Tehnologii utilizate

  În cadrul celui de-al treilea capitol, centru atenției este pus pe bazele tehonolgice care au fost folosite la realizarea acestei lucrări. Această secțiune analizează instrumentele, cadrele si metodologiile folosite pentru a duce proiectul la bun sfârșit, oferind o înțelegere a peisajului tehnologic.

- Capitolul 4: Fundamente teoretice

  În al patrulea capitol se analizează bazele teoretice, cadrele conceptuale și cercetările existente care oferă schelele necesare pentru dezvoltarea ulterioară a proiectului.

- Capitolul 5: Proiect și implementare

  Al cincilea capitol servește drept punct culminant al teoriei în practică. Această secțiune oferă o explorare a arhitecturii și a detaliilor de implementare a proiectului. Pe scurt, acest capitol oferă o perspectivă asupra execuției practice.

- Capitolul 6: Studiu de caz

  Al șaselea capitol prezintă o aplicabilitate reală a proiectului. Această secțiune oferă o demonstrație practică a utilității proiectului într-un context specific.

- Capitolul 7: Concluzie și direcție de dezvoltare

  Capitolul de încheiere marchează deznodământul. Această secțiune sintetizeaza rezultatele obținute pe parcursul lucrării, și scoate în evidență semnificația colectivă a fiecărui capitol.

Capitolul 2

Stadiul actual al domeniului

În acest capitol se vor prezenta principalele caracteristile ale aplicațiilor de management al programului studenților existente pe piață, fără a fi prezentat și modul lor de funcționare.

2.1 iStudiez Pro

iStudiez Pro este o aplicație pentru studenți care permite planificarea cursurilor și adăugarea temelor și notelor. Mai mult decât o simplă aplicație pentru liste de activități, iStudiez Pro prezintă programul și face ca vizualizarea următoarelor activități să fie facilă. Utilizatorii încep prin introducerea programului cursului, inclusiv zile și ore, și adaugă profesorul pentru fiecare clasă, furnizând informații de contact precum adresa de e-mail și numărul de telefon. Dacă este necesar, pot fi adăugate și orele de disponibilitate ale profesorilor. Cu posibilitatea de a atribui culori și pictograme distincte pentru fiecare clasă, calendarul devine ușor de parcurs și de înțeles rapid.[2]

Oferă sincronizare între majoritatea dispozitivelor de pe piață, ceea ce permite adăugarea unei sarcini pe telefon si accesarea acesteia pe laptop. iStudiez Pro oferă posibilitatea de a obține note pe sarcini și se pot seta scale pentru a vedea poziționarea userului comparativ cu ceilalți studenți.

Dezavantajul principal este că această aplicație nu facilitează comunicarea dintre studenți și profesori, deoarece studentul își compune practic singur programul, dictate de către cadrele academice, ceea ce poate duce la diferite erori, de la comunicare până la greșeli din

indolență din partea studentului, rezultând astfel o desincronizare completă a programului.

Pe de altă parte, avantajul este că iStudiezPro se prezintă printr-o interfață intuitivă de tip calendar-dashboard, ușor de folosit, după cum este prezentată in figura de mai jos.

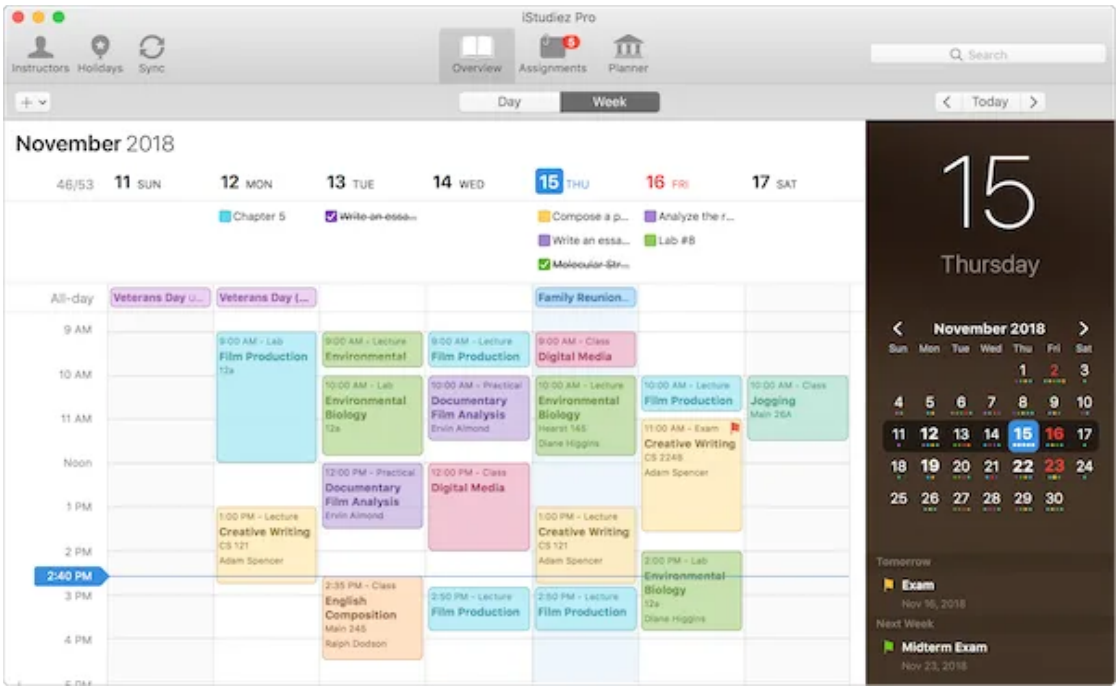

Fig. 2.1: Dashboard iStudiezPro

2.2 MyHomework

MyHomework este o aplicație de planificare simplă pentru educație. Mulți profesori au adoptat deja această aplicație pentru e-learning sau instruire online, dar poate fi folosită și pentru uz individual.

Aplicația este formatată pentru mai multe tipuri de programare a cursurilor, cum ar fi programările în bloc, pe perioade sau orar general. Utilizând MyHomework, studentul nu doar introduce sarcini, ci are și posibilitatea de a prioritiza și clasifica sarcinile, astfel încât să se poate concentra pe ceea ce este cel mai urgent într-un anumit moment. De asemenea, poate seta mementouri pentru termenele viitoare, evitând astfel ratarea termenelor limită din întâmplare sau amânarea lor.

Chiar dacă este nevoie de o conexiune la internet pentru a se sincroniza cu alte dispozitive, această aplicație menține funcționalitatea completă chiar și atunci când nu există conexiune la WiFi. Versiunea gratuită oferă o mulțime de funcții, dar versiunea plătită îi permite utilizatorului să se elibereze de reclame, să distribuie teme, să adauge atașamente de fișiere la teme și să-și schimbe tema, costul reprezentând si dezavantajul acestei aplicații.[5]

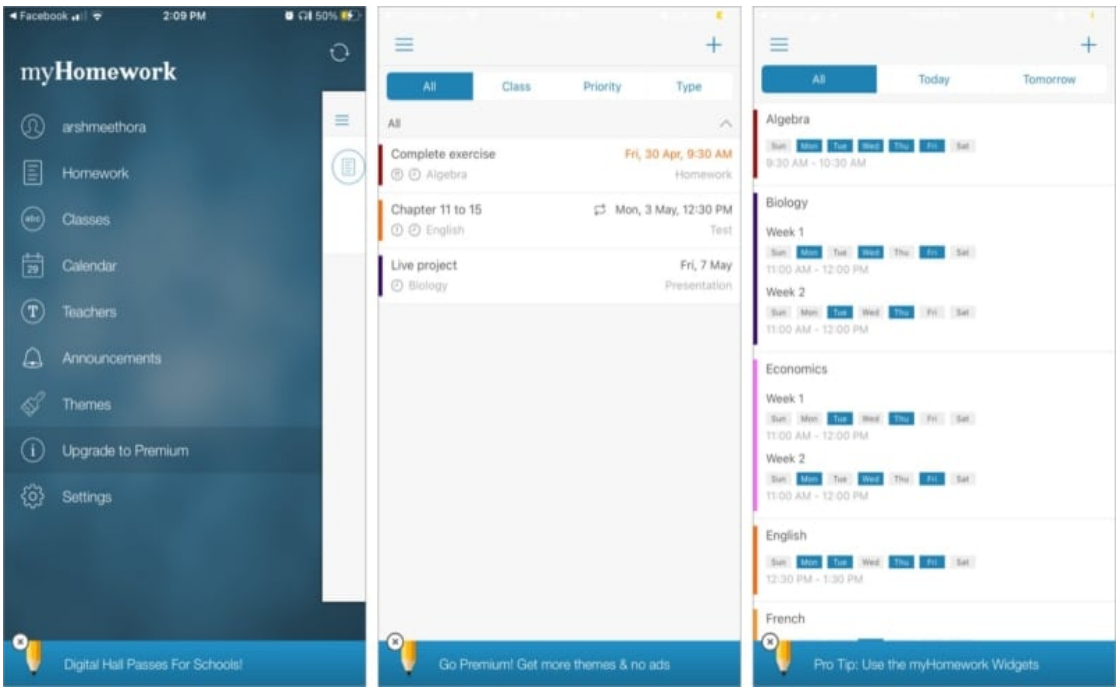

Fig. 2.2: Dashboard MyHomework

Capitolul 3

Tehnologii utilizate

Acest capitol prezintă tehnologiile folosite, fiind împărțite în două mari module, modulul reprezentând mediul în care acestea rulează: modulul clientului și modulul serverului, sau, folosiind termeni actuali, backend și frontend.

3.1 Modulul serverului

Partea de server este construită folosind Spring Boot, bazat pe limbajul de programare Java, având ca sistem de build și management de pachete Apache Maven. Backend-ul se ocupă de realizarea relațiilor dintre entități, folosind Spring Data JPA, prelucrearea datelor, precum și transmiterea acestora pe partea de client.

Limbajul de programare Java

Java a fost creat de către o echipă de cercetători condusă de către James Gosling, la Sun Microsystems, pentru a facilita comunicarea dintre dispozitivele electronice de consum. Edificarea limbajului de programare Java a început în anul 1991 și, în scurt timp, concentrarea echipei s-a schimbat pe o nouă nișă, World Wide Web. Java a fost lansat pentru prima dată în 1995, iar capacitățile acestuia de a oferi interactivitate și control multimedia a arătat că este deosebit de potrivit pentru Web.[3]

Până la sfârșitul anilor 1990, Java a adus multimedia pe internet și a început să se extindă dincolo de Web, alimentând dispozitive de consum, computere, chiar și computerul de bord al roverelor de

explorare pe Marte ale NASA. Datorită popularității, Sun Microsystems a creat diferite varietăți de Java pentru diferite scopuri, inclusiv Java SE pentru computerele de consum, Java ME pentru dispozitivele incorporate și Java EE pentru servere de internet și supercomputere.

Diferența dintre modul de funcționare Java și alte limbaje de programare a fost revoluționară. Codul în alte limbi este mai întâi tradus de un compilator în instrucțiuni pentru un anumit tip de computer. În schimb, compilatorul Java transformă codul în Bytecode, care este apoi interpretat de softwareul numit Java Runtime Environment (JRE) sau mașină virtuală Java. JRE acționează ca un computer virtual care interpretează Bytecode și îl traduce pentru computerul gazdă. Din acest motiv, codul Java poate fi scris în același mod pentru mai multe platforme ("Write Once, Run Everywhere"), ceea ce a contribuit la popularitatea sa pentru utilizarea pe Internet, unde multe tipuri diferite de computere pot prelua aceeași pagină Web.

În ciuda asemănării numelor, limbajul JavaScript care a fost conceput pentru a rula în browserele web nu face parte din Java. Inițial se numea Mocha și apoi LiveScript înainte ca Netscape să primească o licență de marketing de la Sun Microsystems.

Baza de date PostgreSQL

PostgreSQL este o bază de date relațională open source avansată, de clasă enterprise, care acceptă atât interogări SQL (relaționale), cât și JSON (non-relaționale). Este un sistem de management al bazelor de date extrem de stabil, susținut de peste 20 de ani de dezvoltare comunitară, care a contribuit la nivelurile sale ridicate de rezistență, integritate și corectitudine. PostgreSQL este utilizat ca bază de date pentru multe aplicații web, mobile, geospațiale și de analiză.[6]

Caracteristici cheie:

- Extensibilitate și Customizare: O trăsătură definitorie a PostgreSQL-ului este abilitatea sa de a fi extins cu ușurință prin adăugarea de tipuri de date personalizate, funcții stocate și operatori. Această extensibilitate îi permite să se adapteze cerințelor specifice ale proiectelor și aplicațiilor.

- Suport Complet pentru SQL: PostgreSQL respectă în mare măsură standardul SQL, facilitând dezvoltarea și întreținerea codului SQL într-un mod portabil și conform normelor.

- Tranzacții și ACID: Sistemul oferă suport robust pentru tranzacții, asigurând integritatea datelor și respectând proprietățile ACID, ceea ce înseamnă că operațiunile sunt fie realizate complet, fie nu sunt realizate deloc.
- Gestionarea Concurgenței: PostgreSQL gestionează cu succes concurgența în medii cu mai mulți utilizatori, evitând conflictele și garantând coerența datelor.
- Securitate Avansată: Baza de date dispune de caracteristici solide de securitate, inclusiv autentificare avansată, autorizare bazată pe roluri și criptare a datelor.
- Suport pentru JSON și Semi-Structured Data: Capacitatea de a stoca, interoga și indexa date în format JSON și JSONB permite lucrul cu date semi-structurate și flexibile, deschizând noi orizonturi pentru dezvoltarea aplicațiilor moderne.
- Triggere și Funcții Stocate: PostgreSQL permite definirea de triggere automate care sunt declanșate de evenimente în baza de date, precum și crearea de funcții stocate complexe, adăugând niveluri suplimentare de control și logica în cadrul bazei de date.
- Replicare și Scalabilitate: Pentru medii cu trafic intens, PostgreSQL oferă opțiuni de replicare pentru redundanță și scalabilitate orizontală, asigurând disponibilitatea datelor.

Managerul utilitar Apache Maven

Maven este un instrument popular de compilare open-source dezvoltat de Apache Group pentru a construi, publica și implementa mai multe proiecte simultan pentru un management mai bun al acestora.[1]

Maven este scris în Java și este folosit pentru a construi proiecte scrise în C#, Scala, Ruby etc. Bazat pe Project Object Model (POM), acest instrument a ușurat viața dezvoltatorilor Java, ocupându-se de:

- Builds
- Documentație
- Dependențe

- Rapoarte
- Managementul configurației software
- Distribuție
- Releases

POM este unitatea fundamentală de lucru în Maven. Este un fișier XML care conține informații despre proiect și detalii de configurare utilizate de Maven pentru a construi proiectul.

```
<?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 https
       ://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>3.1.2</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.management</groupId>
    <artifactId>app</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>app</name>
    <description>Managemenetul activitatilor sportive din
       cadrul facultatii UPT</description>
    <properties>
        <java.version>17</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</
               artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
```

```
28          ...
29      </dependencies>
30  </project>
```

Secvență de Cod 3.1: Exemplu structură pom.xml

Spring Framework

Spring Framework (Spring) este un cadru de aplicații open-source care oferă suport de infrastructură pentru dezvoltarea aplicațiilor Java. Unul dintre cele mai populare framework-uri Java Enterprise Edition (Java EE), Spring îi ajută pe dezvoltatori să creeze aplicații de înaltă performanță folosind obiecte Java simple (POJO - termen inventat de Martin Fowler).[7]

Un framework este un corp mare de cod predefinit la care dezvoltatorii pot adăuga cod pentru a rezolva o problemă dintr-un anumit domeniu. Există multe framework-uri Java populare, inclusiv Java Server Faces (JSF), Maven, Hibernate, Struts și Spring.

Lansat în iunie 2003 de Rod Johnson sub licența Apache 2.0, Spring Framework este găzduit de SourceForge.

Spring îmbunătățește eficiența codificării și reduce timpul general de dezvoltare a aplicațiilor, deoarece este lightweight și eficientă în utilizarea resurselor sistemului. Elimină munca de configurare, astfel încât dezvoltatorii să se poată concentra pe scrierea logicii. Spring se ocupă de infrastructură, astfel încât dezvoltatorii să se poată concentra asupra aplicației.

Biblioteca Spring Boot

Java Spring Boot (Spring Boot) este un instrument care face dezvoltarea aplicațiilor web și a microserviciilor cu Spring Framework mai rapidă și mai ușoară prin intermediul a trei capabilități principale:

- Autoconfigurare - aplicațiile sunt inițializate cu dependențe prestabilite care nu trebuie configurate manual. Deoarece Java Spring Boot vine cu capabilități de autoconfigurare încorporate, se configurează automat atât pachetele Spring Framework-ul de bază, cât și pachetele third-party, pe baza setărilor proiectului respectiv. Chiar dacă se pot suprascrie aceste valori implicite

odată ce inițializarea este completă, caracteristica de configurare automată a Java Spring Boot permite începerea rapidă a dezvoltării aplicațiilor bazate pe Spring și reduce posibilitatea erorilor umane.

- O abordare proprie a configurației - adăugarea și configurarea dependențelor de pornire, în funcție de nevoile proiectului. După propria sa judecată, Spring Boot alege ce pachete să instaleze și ce valori implicite să folosească, mai degrabă decât să solicite developerului să ia singur toate aceste decizii și să configureze totul manual.

- Abilitatea de a crea aplicații de sine stătătoare - Spring Boot îi ajută pe developeri să creeze aplicații care doar trebuie rulate. Mai exact, permite creerea aplicațiilor autonome care rulează pe cont propriu, fără a se baza pe un server web extern, prin încorporarea unui server web precum Tomcat sau Netty în aplicație în timpul procesului de inițializare.

Aceste caracteristici funcționează împreună pentru a oferi un instrument care permite configurarea unei aplicații bazată pe Spring cu o configurare minimă. [8]

Docker Framework

În ultimii ani, aplicațiile bazate pe microservicii au început să devină din ce în ce mai populare datorită beneficiilor aduse in comparație cu aplicațiile vechi bazate pe arhitectura de tip monolit. Aplicațiile bazare pe arhitectura de tip monolit primesc update-uri foarte rar, iar o schimbare în cod poate afecta întreaga aplicație. Adăugarea unei noi funcționalități solicită reconfigurarea și updatarea întregii aplicații. În general, acest proces este considerat foarte costisitor ceea ce poate duce la întârzierea livrării produsului software.

Arhitectura bazată pe microservicii a fost gândită și proiectată pentru a rezolva problemele anterioare create de sistemele de tip monolit. În acest tip de arhitectură aplicația este descompusă în mai multe componente care rulează independent si comunică prin protocolul sincron HTTP, peste care este expus de obicei un RESTful API (Representational State Transfer Application Programming Interface). Acest tip de arhitectură oferă un plus de flexibilitate în faza

de dezvoltare a aplicației deoarece respectă principiul SoC (Separation of Concern) care facilitează modularizarea aplicației in unități mai mici si independente.

Docker este o platformă open-source pentru containerizarea aplicațiilor software. Această platformă a fost concepută de compania Docker. Inc și fondată de Solomon Hykes. Punctele forte ale Docker sunt imaginile si containerele.[9]

Imaginile Docker sunt template-uri ce conțin un set de instrucțiuni pentru crearea container-ului. O imagine trebuie sa conțină toate pachetele și dependințele necesare pentru a rula aplicația. Acestea au un format stratificat și instrucțiunile se execută pas cu pas. Imaginile Docker oferă o portabilitate crescută, pot fi transferate ușor, stocate și updatate cu ușurință. Pot fi considerate ca fiind source code pentru containere.

Containerele Docker sunt instanțe care rulează o imagine. Putem crea, porni, șterge, restarta containerele folosind Docker API sau CLI. În același mod in care putem porni o mașină virtuală dintr-un șablon în care descriem specificațiile mașinii virtuale, la fel putem porni unul sau mai multe containere dintr-o singură imagine Docker. Un container care rulează este un proces izolat de gazda unde se află instalat Docker Engine.

3.2 Modulul clientului

Partea de pe client este edificată cu Angular v16, bazat pe limbajul de programare TypeScript, ce facilitează construirea aplicațiilor web de tipul single page application (SPA). Începutul SPA-ului în browser este un fișier HTML minimal, cu referințe către fișiere JavaScript, de obicei main.js, care este construit folosit un bundler, în acest caz Webpack. Randarea este una dinamică, JavaScript-ul fiind orchestratorul care va decide, pe partea de client, ce content să randeze la momentul respectiv, fară să reîncarce pagina. Astfel, utilizatorul va avea o experiență fluidă.

Limbajul de programare JavaScript

Dezvoltat de Netscape în colaborare cu Sun Microsystems, JavaScript este una dintre cele mai populare tehnologii de bază ale web. De la

începuturile sale, a fost o parte integrantă a aplicațiilor web, realizând interactivitatea și dinamica acestora.

Începând cu 2023, 98.7% dintre site-uri folosesc Javascript în partea clientului pentru prelucrarea comportamentului paginilor web, deseori încorporând librării third-party. Toate browserele web majore au un interpretor JavaScript dedicat pentru a executa codul pe dispozitivele utilizatorilor.

Este un limbaj de nivel înalt, adesea compilat just-in-time, la run-time, care se conformează standardului ECMAScript. Este un limbaj de programare orientat pe obiecte bazate pe prototip și funcții de primă clasă. Include tipuri dinamice. Este multi-paradigmă și acceptă stiluri de programare bazate pe evenimente, funcționale și imperative. Are interfețe de programare a aplicațiilor pentru lucrul cu text, date, expresii regulate, structuri de date și manipulare de Document Object Model.[4]

ECMAScript este un standard pentru limbajele de scriptare, incluzând JavaScript JScript și ActionScript. De asemenea este cunoscut ca un standard care să asigure interoperabilitatea paginilor web în diferite browsere.

Interpretoarele JavaScript au fost utilizate inițial doar în browserele web, dar acum sunt componente de bază ale unor servere și ale unor varietăți de aplicații, cel mai popular sistem de rulare folosit în aceste scopuri fiind Node.js.

Weak typing-ul limbajului de programare JavaScript poate creea foarte ușor erori. În alte limbaje, cum ar fi Java și C++, trebuie specificat în mod explicit tipul unui variabile. Cu toate acestea, în JavaScript, interpretorul deduce automat tipul de date al unei variabile pe baza valorii care îi este atribuită.

Constrângerea agresivă de tip este o caracteristică a JavaScript care obligă diferitele tipuri de date să fie compatibile între ele. De exemplu dacă se încearcă atribuirea unui număr unei variabile de tip string, JavaScript va converti automat numărul într-un string, indiferent de valoarea inițială. Prin urmare, acest lucru poate duce la rezultate neașteptate și poate fi dificil de depanat.

Din cauza acestor deficiențe, pentru dezvoltarea de aplicații pe scară largă, Microsoft a creat TypeScript.

Limbajul de programare TypeScript

Prin definiție, „TypeScript este JavaScript pentru dezvoltarea de aplicații scalabile”.

TypeScript este un limbaj puternic tipizat, orientat pe obiecte, compilat. A fost proiectat de Anders Hejlsberg la Microsoft. TypeScript este atât un limbaj, cât și un set de instrumente. TypeScript este un superset de JavaScript compilat în JavaScript. Cu alte cuvinte, TypeScript este JavaScript plus câteva caracteristici suplimentare.

A fost lansat publicului în octombrie 2012, cu versiunea 0.8, după doi ani de dezvoltare internă la Microsoft. La scurt timp după lansarea publică inițială, limbajul în sine a fost lăudat, dar s-a criticat lipsa suportului matur al multor IDE, în afară de Microsoft Visual Studio, care nu era disponibil pe Linux și OS X la acel moment. Din aprilie 2021, există suport în alte IDE-uri și editori de text, inclusiv Emacs, Vim, WebStorm, Atom și Visual Studio Code al Microsoft.

TypeScript este portabil peste browsere, dispozitive și sisteme de operare. Poate rula în orice mediu în care rulează JavaScript. Spre deosebire de omologii săi, TypeScript nu are nevoie de un VM dedicat sau de un mediu de rulare specific pentru a fi executat.

Este superior celorlalți omologi ai săi, cum ar fi limbajele de programare CoffeeScript și Dart, într-un mod în care TypeScript este JavaScript extins. În schimb, limbaje precum Dart, CoffeeScript sunt limbaje noi în sine și necesită un mediu de execuție specific limbii.

JavaScript este un limbaj interpretat. Prin urmare, trebuie rulat pentru a testa dacă este valid, ceea ce poate duce la ore întregi de debugging. Transpilerul TypeScript oferă caracteristica de verificare a erorilor. TypeScript va compila codul și va genera erori de compilare, dacă găsește erori de sintaxă. Acest lucru ajută la evidențierea erorilor înainte de rularea codului.

JavaScript nu este puternic tipizat. TypeScript vine cu un sistem opțional de tipizare statică și inferență de tip prin Serviciul de limbaj TypeScript, denumit TLS. Tipul unei variabile, declarată fără tip, poate fi dedus de TLS pe baza valorii sale.

Angular Framework

Angular este o platformă de dezvoltare software, construită pe TypeScript. Că platformă, Angular are un cadru bazat pe component

pentru construirea de aplicații web scalabile, o colecție de biblioteci bine integrate care acoperă o mare varietate de funcții, inclusiv rutare, gestionare a formularelor, comunicare client-server și multe altele. De asemenea, are o suită de instrumente pentru developeri, care facilitează dezvoltarea, construirea, testarea și actualizarea codului.

Ecosistemul Angular este format dintr-un grup divers de peste 1,7 milioane de developeri, autori de biblioteci și creatori de conținut.

Componente

Componentele sunt blocurile principale care compun o aplicație. O componentă include o clasă TypeScript cu un decorator @Component(), un șablon HTML și stiluri CSS. Decoratorul @Component() specifică următoarele informații:

- Un selector CSS care definește modul în care componenta este utilizată într-un șablon de HTML. Elementele HTML din șablon care se potrivesc cu acest selector devin instanțe ale componentei.
- Un șablon HTML care dictează la Angular cum să randeze componenta.
- Un set opțional de stiluri CSS care definesc aspectul elementelor HTML ale șablonului.

Următoarea este o componentă Angular minimală:

```
//hello-world.component.ts
    import { Component } from '@angular/core';

    @Component({
        selector: 'hello-world',
        template: `
            <h2>Hello World</h2>
            <p>This is my first component!</p>
        `
    })
    export class HelloWorldComponent {
        // Codul ce descrie comportamentul componentei.
    }
}
```

Secvență de Cod 3.2: Componentă Angular minimală

Pentru a utiliza această componentă, trebuie adăugat selectorul componentei într-un șablon:

```
//another.component.html
<hello-world></hello-world>
```

Când Angular randează această componentă, DOM-ul rezultat arată astfel:

```
<hello-world>
    <h2>Hello World</h2>
    <p>This is my first component!</p>
</hello-world>
```

Șabloane

Fiecare componentă are un șablon HTML care declară cum se randează acea componentă. Definirea acestui șablon se poate face fie în linie, fie într-un fiser separat.

Angular adaugă elemente de sintaxă care extind HTML, pentru a facilita inserarea de valori dinamice din componenta Angular. Actualizează automat DOM-ul atunci când starea componentei se schimbă. O aplicație a acestei caracteristici este inserarea de text dinamic, așa cum se arată în exemplul următor.

```
//hello-world-interpolation.component.html
<p>{{ message }}</p>
```

Valoarea mesajului provine din clasa de componente:

```
//hello-world-interpolation.component.ts
import { Component } from '@angular/core';

@Component ({
selector: 'hello-world-interpolation',
templateUrl: './hello-world-interpolation.component.html'
})
export class HelloWorldInterpolationComponent {
    message = 'Hello, World!';
}
```

Secvență de Cod 3.3: Componentă Angular interpolare

Când aplicația încarcă componenta și șablonul acesteia, utilizatorul vede următoarele:

```
//hello-world-interpolation.component.html
<p>Hello, World!</p>
```

Acoladele duble îi indică lui Angular să interpoleze conținutul din ele.

Angular acceptă, de asemenea, legături de proprietăți, pentru a facilita transmiterea valorilor dintre componente și pentru a seta diferite proprietăți.

```
1     <p
2         [id]="sayHelloId"
3         [style.color]="fontColor">
4         Se seteaza fontColor in clasa typescript al componentei
5     </p>
```

Utilizarea parantezelor drepte indică legătura dintre proprietatea sau atributul la o valoare din clasa componentelor.

Pentru a asculta acțiunile utilizatorului, cum ar fi apăsările de taste, mișcările mouse-ului, clickurile, se definesc ascultători de evenimente, specificând numele evenimentului între paranteze rotunde:

```
1     <button
2         (click)="sayMessage()">
3         functia sayMessage este definita in clasa typescript al
              componentei
4     </button>
```

Dependency Injection

Dependency Injection este partea din cadrul Angular care oferă componentelor acces la servicii și la alte resurse. Angular oferă posibilitatea de a injecta un serviciu într-o componentă pentru a oferi acelei componente acces la serviciu.

Decoratorul @Injectable() definește o clasă ca serviciu în Angular și îi permite lui Angular să o injecteze într-o componentă ca dependență. De asemenea, decoratorul @Injectable() indică faptul că o componentă, clasă sau NgModule are o dependență de un serviciu.

Injectorul este mecanismul principal. Angular creează un injector la nivel de aplicație în timpul procesului de bootstrap și injectoare suplimentare după cum este necesar.

Un injector creează dependențe și menține un container de instanțe de dependență pe care le reutilizează, dacă este posibil. Un furnizor este un obiect care îi spune unui injector cum să obțină sau să creeze o dependență.

Servicii

Serviciul este o categorie largă care cuprinde orice valoare, funcție sau caracteristică de care are nevoie o aplicație. Un serviciu este de obicei o clasă

cu un scop restrâns si bine definit. Angular stie sa distinga componentele de servicii pentru a creste modularitatea si reutilizarea.

Im mod ideal, sarcina unei componente este sa se ocupe doar de experienta utilizatorului. O componenta ar trebui sa prezinte proprietati si metode pentru legarea datelor pentru a media intre vizualizare si logica aplicatiei.

Serviciile sunt bune pentru sarcini precum preluarea datelor de pe server, validarea intrărilor utilizatorului sau conectarea direct la consolă. Prin definirea unor astfel de sarcini de procesare într-o clasă de servicii injectabilă, acele sarcini devin disponibile oricărei componente. De asemenea, pentru a face aplicația mai adaptabilă, se pot injecta furnizori diferiți ai aceluiași tip de serviciu, după caz, în circumstanțe diferite.

În Angular, Dependency Injection face aceste servicii disponibile pentru componente.

Module

Aplicațiile Angular sunt modulare și Angular are propriul său sistem de modularitate numit NgModules. NgModules sunt containere pentru un bloc coeziv de cod dedicat unui domeniu de aplicație, unui flux de lucru sau unui set de capabilități strâns legate. Ele pot conține componente, furnizori de servicii și alte fișiere de cod al căror domeniu este definit de NgModule care le conține. Ei pot importa funcționalități care sunt exportate din alte NgModules și pot exporta funcționalitatea selectată pentru a fi utilizate de alte NgModules.

Fiecare aplicație Angular are cel puțin o clasă NgModule, modulul rădăcină, care este denumit în mod convențional AppModule și rezidă într-un fișier numit app.module.ts.

3.3 Ghid de instalare

Capitolul 4

Fundamentare teoretică

Capitolul 5

Proiectare și implementare

Capitolul 6

Studiu de caz

Capitolul 7

Concluzie şi direcţie de dezvoltare

7.1 Concluzie

7.2 Direcții de dezvoltare

Listă figuri

Listă tabele

Listă secțiuni de cod

Bibliografie

[1] Apache maven. https://maven.apache.org. [Accessed: 15.08.2023].

[2] istudiezpro. https://istudentpro.com/. [Accessed: 15.08.2023].

[3] Java. https://www.britannica.com/technology/Java-computer-programming-language. [Accessed: 15.08.2023].

[4] Javascript. https://en.wikipedia.org/wiki/JavaScript. [Accessed: 15.08.2023].

[5] Myhomerwork. https://myhomeworkapp.com/. [Accessed: 15.08.2023].

[6] Postgresql. https://aws.amazon.com/rds/postgresql/what-is-postgresql. [Accessed: 15.08.2023].

[7] Spring. https://www.techtarget.com/searchapparchitecture/definition/Spring-Framework. [Accessed: 15.08.2023].

[8] Spring boot. https://www.ibm.com/topics/java-spring-boot. [Accessed: 15.08.2023].

[9] Sean P. Kane Karl Matthias. *Docker Up and Running*. O'REILLY, 2018.