Razvoj platforme za učenje programiranja - serverska strana

Luka Dević, dr Bojana Dimić Surla

***Sadržaj* — Rad pruža kratak pregled istraživanja i opisa rezultata u vezi sa razvojem serverske infrastrukture za platformu za učenje programiranja. Fokus je na integraciji sa IntelliJ IDEA platformom, skalabilnosti servera, sigurnosti i agilnom razvoju. Opisuje se implementacija privatnog Git servera i REST API-ja, ključne funkcionalnosti platforme, te izazovi i rešenja tokom razvoja. Na kraju, istaknuti su planovi za budući rad, uključujući razvoj dodataka za IntelliJ IDEA i VS Code za profesore i studente.**

***Ključne reči* — agilni razvoj, Git server, IntelliJ IDEA, platforma za učenje programiranja, REST API, sigurnost, skalabilnost.**

I. UVOD

Da bi olakšale efikasno učenje i procenu znanja, obrazovne institucije zahtevaju robustne i prilagodljive platforme koje podržavaju programerske zadatke i ispite. Ove platforme moraju omogućiti besprekornu integraciju sa razvojnim okruženjima, podržavati saradničko učenje, i osigurati siguran i skalabilan rad [1]. Platforme za učenje programiranja značajno su napredovale, odražavajući napredak u obrazovnim tehnologijama i metodologijama.

Iako postoje brojne platforme za učenje programiranja, mnogim postojećim rešenjima nedostaju fleksibilnost i funkcionalnosti potrebne za podršku dinamičnim i evoluirajućim obrazovnim okruženjima. Česti izazovi uključuju ograničenu integraciju sa razvojnim okruženjima, nedovoljnu podršku za saradničke zadatke i nedostatak adekvatnih bezbednosnih mera. Da bi se prevazišli ovi izazovi, potrebna je platforma koja pruža robustnu serversku infrastrukturu, podržava različite obrazovne funkcije i usklađena je sa agilnim metodologijama razvoja [2].

Osnovni cilj ovog rada je da opiše jedan pristup razvoju i implementacije robustne serverske infrastrukture za platformu za učenje programiranja, sa posebnim fokusom na omogućavanju studentima da završe programerske ispite i zadatke u izabranom razvojnom okruženju.

L. D. Autor, Računarski fakultet, Srbija (email: luka.m.devic@gmail.com).
B. D. S. Autor, Računarski fakultet, Srbija (email: bdimicsurla@raf.rs).

II. POSTOJEĆA SOFTVERSKA REŠENJA ZA OBRAZOVANJE IZ PROGRAMIRANJA

Obrazovanje iz programiranja se brzo razvija, podstaknuto rastućom potrebom za veštinama programiranja. Pojavile su se brojne platforme koje olakšavaju ovu obrazovnu potrebu, svaka sa jedinstvenim karakteristikama i pristupima učenju.

A. GitHub Classroom

GitHub Classroom koristi GitHub-ovo okruženje za kodiranje kako bi olakšao obrazovanje iz programiranja. Instruktori kreiraju i upravljaju zadacima, dok studenti koriste Git repozitorijume za završavanje i predaju radova. Prednosti uključuju saradnju, integraciju sa GitHub-om i automatizaciju ocenjivanja. Međutim, platforma nema ugrađenu podršku za integrisana razvojna okruženja (IDE) i može biti izazovna za prilagođavanje automatizovanog ocenjivanja. Potrebna je stabilna internet konekcija, a postoji i krivulja učenja za studente koji nisu upoznati sa Git-om.

B. Codio

Codio nudi razvojno okruženje u oblaku, resurse za nastavni plan i automatsko ocenjivanje, omogućavajući studentima da kodiraju direktno u pregledačima. Prednosti uključuju jednostavan pristup, interaktivne tutorijale i efikasno automatsko ocenjivanje. Međutim, Codio može biti skup zbog pretplate, a njegova fleksibilnost je ograničena za specijalizovane zadatke. Takođe, zavisnost od internet konekcije može biti problematična.

C. Repl.it

Repl.it podržava više programskih jezika i omogućava saradničko kodiranje u realnom vremenu. Njegova jednostavnost upotrebe i širok spektar obrazovnih resursa čine ga idealnim za početnike. Međutim, platforma ima ograničene resurse na besplatnom nivou i može biti zabrinjavajuća u pogledu bezbednosti podataka.

D. IntelliJ IDEA Educational

IntelliJ IDEA Educational pruža integrisane lekcije, automatsko ocenjivanje i profesionalne alate, pripremajući studente za stvarna programerska okruženja. Iako je moćan alat, može biti preopterećujući za početnike i nema ugrađenu podršku za saradničko kodiranje.

E. Rezime postojećih rešenja

Svaka od ovih platformi nudi jedinstvene karakteristike i prednosti za obrazovanje iz programiranja. GitHub Classroom podržava saradničko učenje i automatizaciju, Codio pruža cloud-based IDE i resurse za nastavni plan, Repl.it omogućava saradničko kodiranje u realnom vremenu, a IntelliJ IDEA Educational nudi profesionalne alate sa integrisanim lekcijama. Međutim, svaka platforma ima ograničenja u vezi sa fleksibilnošću, bezbednošću i pristupom internetu. Nova platforma koja se predlaže u ovoj tezi nastoji da reši ova ograničenja integracijom sa različitim IDE-ovima, pružajući prilagodljive opcije procene i robustne bezbednosne mere, čime se obezbeđuje efikasno i sigurno okruženje za učenje.

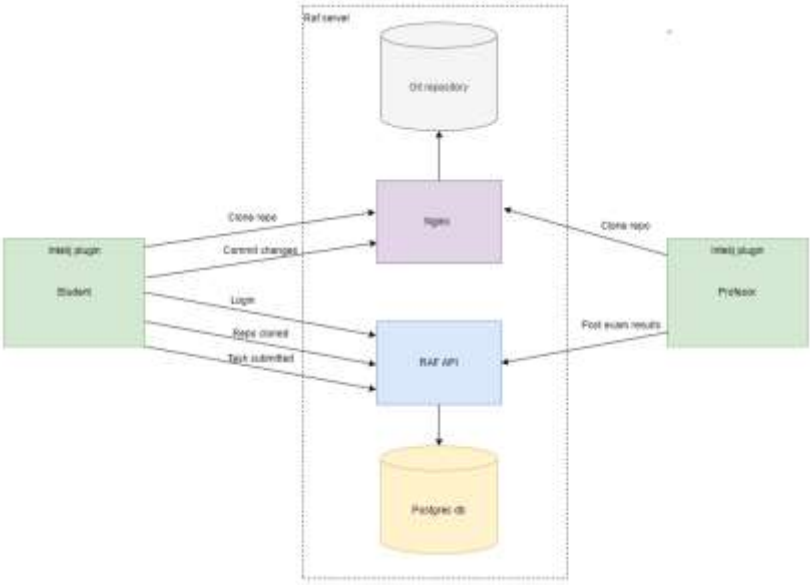
III. ARHITEKTURA I DIZAJN SISTEMA

Platforma za učenje programiranja koja je tema ovog rada usvaja arhitekturu klijent-server kako bi olakšala besprekornu interakciju između korisnika i backend sistema. Ovu arhitektura karakterišu sledeće komponente. Klijentska strana je implementirana kao dodatak za IntelliJ IDEA Integrated Development Environment (IDE), popularno okruženje za razvoj softvera. Ovaj dodatak omogućava studentima da interaguju sa platformom, šalju svoj kod i primaju povratne informacije direktno iz svog razvojnog okruženja. Klijent komunicira sa serverom putem RESTful API-ja, omogućavajući laku integraciju i komunikaciju. Trenutno se radi na razradi sličnog dodatka za Visual Studio Code (VS Code), koji je takođe popularan među studentima i profesionalcima, čime će se proširiti dostupnost i korisnička baza platforme [4].

Serverska strana je razvijena koristeći programski jezik Java i Spring framework, koji je poznat po svojoj robusnosti i skalabilnosti. Ova strana uključuje nekoliko ključnih komponenti. Na serverskoj strani su razvijeni RESTful API krajnje tačke koje omogućavaju klijentu interakciju sa platformom, poput slanja zadataka ili dobijanja povratnih informacija. Ove krajnje tačke su dizajnirane da budu intuitivne i jednostavne za korišćenje, čime se obezbeđuje efikasna komunikacija između klijenta i servera. Privatni Git server se koristi za upravljanje studentskim prijavama, olakšavanje kontrole verzija i održavanje istorije promena. Svaki student dobija svoj repozitorijum gde mogu da se prate promene i čuvaju predata rešenja. Git server omogućava praćenje svih verzija koda, olakšavajući procese evaluacije i povratne informacije. Relaciona baza podataka se koristi za čuvanje obrazovnih resursa, podataka studenata, zadataka i prijava. Baza podataka je dizajnirana tako da podržava skalabilnost i efikasno upravljanje velikim količinama podataka, obezbeđujući brz i pouzdan pristup informacijama kada

god je to potrebno.

Klijent komunicira sa serverom putem HTTP protokola. Kada student pošalje kod ili zatraži povratne informacije, klijent šalje odgovarajući zahtev serveru. Server obrađuje ove zahteve, interaguje sa bazom podataka po potrebi, i vraća odgovarajuće odgovore klijentu. Na primer, kada student pošalje zadatak, klijent šalje POST zahtev REST API-ju servera, koji zatim ažurira bazu podataka. Intelij plugin snima prijavu zadatka na Git server preko Nginx-a.

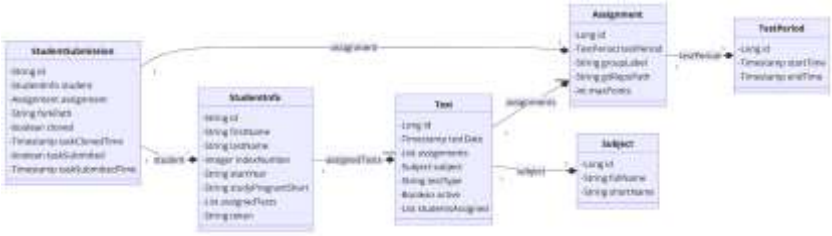


Sl. 1. Arhitektura platforme za učenje

Arhitektura klijent-server koju možete videti na slici 1, pruža jasnu separaciju odgovornosti, pri čemu je klijent usmeren na pružanje intuitivnog interfejsa za studente, a server se bavi backend obradom i upravljanjem podacima. Arhitektura je skalabilna, bezbedna i pogodna za evoluirajuće potrebe platforme za učenje programiranja. Korišćenje popularnih razvojnih okruženja kao što su IntelliJ IDEA i Visual Studio Code omogućava studentima da rade u poznatom okruženju, dok robusni serverski sistem osigurava pouzdano upravljanje podacima i efikasnu obradu zahteva.

A. *Baza podataka platforme*

Baza podataka za platformu za učenje programiranja bazirana je na relacionom modelu koji efikasno organizuje i upravlja različitim entitetima uključenim u obrazovanje iz programiranja. Dizajn uzima u obzir odnose između studenata, zadataka, predmeta, testova i drugih relevantnih komponenti. Dijagram ključnih entiteta se može videti na slici 2.



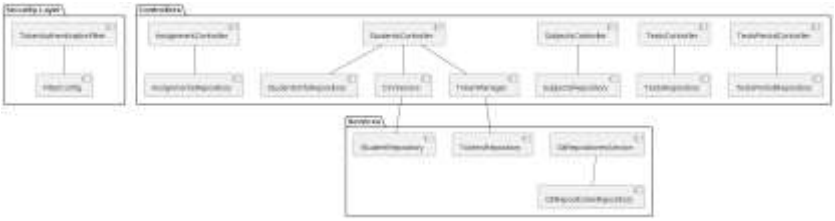
Sl. 2. Dijagram ključnih entiteta platforme za učenje

U predloženom rešenju koristi se niz klasa entiteta baze podataka za efikasno upravljanje komponentama obrazovnog i ispitnog sistema, integrisanog sa Git repozitorijumima. Klasa Assignment povezuje zadatke sa specifičnim test periodima (TestPeriod), detaljno opisujući kada je zadatak aktivan, dok ExamInfo beleži specifičnosti ispita, uključujući grupisanje zadataka i detalje učionice. StudentInfo pruža detalje o studentima, kao što su ime, broj indeksa, godina početka studija i program, zajedno sa jedinstvenim ID-om i tokenom za autentifikaciju. StudentSubmission prati podnošenja zadataka, beležeći informacije o studentu i zadatku. Klasa Subject obuhvata informacije o predmetima, nudeći pune i kratke nazive, dok TaskSubmissionInfo čuva logističke detalje podnošenja. Test modelira testove ili ispite, detaljno opisujući datum, zadatke, predmet, tip testa, aktivnost i listu studenata. TestPeriod definiše vremenske okvire za testove i zadatke, dok GitRepository upravlja Git repozitorijumima putem jedinstvenog ID-a i URL-a. Klasa Token upravlja autentifikacionim tokenima sa id i value poljima. Ovi entiteti zajedno stvaraju robustan okvir koji podržava integrisano obrazovno okruženje, olakšavajući detaljno upravljanje akademskim aktivnostima i usklađivanje obrazovnog sadržaja sa modernim tehnološkim alatima.

B. *Dizajn REST API-ja*

Dizajn REST API-ja koji je implementiran u ovom rešenju efikasno kombinuje sigurnost, upravljanje podacima i funkcionalnost unutar veb-bazirane obrazovne platforme. Arhitektura serverske strane aplikacije kojom

je implementiran REST API prikazana je na slici 3. Osnova sigurnosne infrastrukture postavljena je kroz klasu `FilterConfig`, koja konfiguriše `TokenAuthenticationFilter` za autentifikaciju API zahteva. Ovaj filter se primenjuje na sve rute koje počinju sa `/api/*`, osiguravajući da svaki zahtev bude autentifikovan preko zaglavlja 'Authorization' protiv unapred definisanog važećeg tokena pre bilo kakve dalje obrade. Ako autentifikacija ne uspe, zahtev se zaustavlja sa HTTP 401 Unauthorized statusom, štiteći osetljive krajnje tačke.



Sl. 3. Dijagram java klasa platforme za učenje

U upravljanju autentifikacionim tokenima, klasa `TokenManager` interaguje sa `TokensRepository` za generisanje, validaciju i brisanje tokena. Ova klasa osigurava da je svaki token jedinstveno povezan sa ID-om studenta, poboljšavajući personalizovanu kontrolu pristupa i sigurnost širom sistema.

Upravljanje podacima omogućavaju različite klase kontrolera koje rukuju specifičnim entitetima domena. Na primer, `AssignmentController` upravlja operacijama povezanim sa zadacima, nudeći krajnje tačke za pregled, kreiranje i brisanje zadataka. Slično, `StudentsController` se bavi funkcionalnostima specifičnim za studente, uključujući dohvatanje detalja o studentima, upravljanje registracijama i rukovanje tokenom za autorizaciju pristupa Git repozitorijumima. `SubjectsController`, `TestsController` i `TestsPeriodController` proširuju ove sposobnosti na predmete i testove, omogućavajući sveobuhvatno upravljanje obrazovnim sadržajem i rasporedima.

Pored toga, arhitektura integriše `GitRepositoriesService` za upravljanje Git repozitorijumima, što je suštinski važno za kontrolu verzija unutar upravljanja obrazovnim sadržajem. `CSVService` podržava operacije sa datotekama, posebno rukovanje masovnim uvozom podataka o studentima u CSV formatu, olakšavajući efikasnu obradu i unos podataka.

Upravljanje izuzecima je robustno kroz `FileUploadExceptionAdvice`, koji posebno cilja probleme povezane sa otpremanjem datoteka, kao što su

prekoračenja ograničenja veličine. Ovo osigurava da sistem graciozno rukuje greškama korisnika tokom interakcija sa datotekama.

Pomoćne klase kao što je `CSVHelper` pomažu u parsiranju CSV datoteka i održavanju integriteta podataka, dok prilagođeni deserializeri poput `LocalDateTimeDeserializer` osiguravaju tačno parsiranje podataka o datumu i vremenu, što je ključno za tačnu obradu spoljnih ulaznih podataka.

Ukratko, dizajn REST API-ja ne samo da naglašava sigurno i efikasno rukovanje podacima već takođe poboljšava interakciju korisnika kroz dobro definisane krajnje tačke za upravljanje obrazovnim resursima. Ovaj sveobuhvatan pristup osigurava da platforma ostane robustna, sigurna i visoko funkcionalna, efikasno zadovoljavajući potrebe obrazovnog okruženja.

IV. OPIS IMPLEMENTACIJE

Platforma se sastoji od dve ključne komponente, to su privatni Git repozitorijum i REST API. U ovom poglavlju opisan je postupak konfiguracije Git servera, integracija Git servera sa ostatkom platforme i implementacija REST API-ja.

A. Postavljanje i konfiguracija privatnog Git servera

Privatni Git server služi kao repozitorijum za studentske zadatke i predaje ispita, postavljen na virtuelnu mašinu fakulteta i konfigurisana za pristup putem HTTP-a koristeći web server. Proces uključuje instalaciju Git servera, konfiguraciju web servera, kreiranje direktorijuma za repozitorijume sa odgovarajućim dozvolama, i kreiranje korisničkih naloga za autentifikaciju. Nakon ažuriranja i nadogradnje Ubuntu Servera, instaliraju se potrebne zavisnosti, kreiraju se i inicijalizuju Git repozitorijumi, a korisnici se povezuju sa repozitorijumima pomoću git clone komande. Ovi koraci osiguravaju funkcionalnost HTTP Git servera, omogućavajući efikasnu razmenu i upravljanje zadacima i rezultatima ispita.

B. Integracija Git servera sa platformom

Privatni Git server se integriše u platformu putem skupa servisa koji upravljaju kreiranjem repozitorijuma, pristupom i praćenjem predaja:

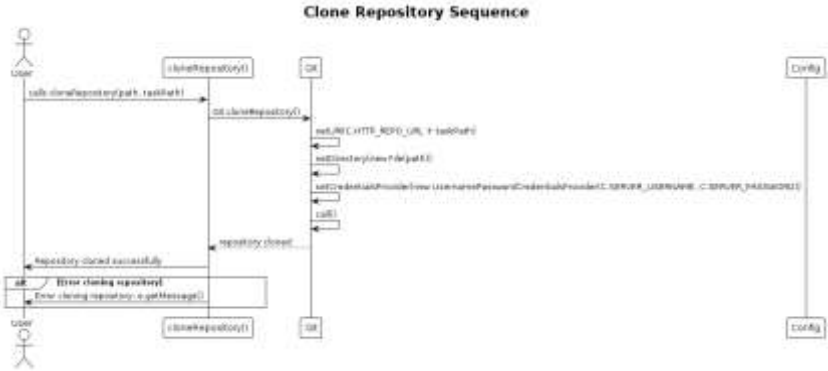
Klasa GitServerHttpService pruža metode za kloniranje i slanje na repozitorijum pomocu metoda `cloneRepository` i `pushToRepository`.

Metoda `cloneRepository` je dizajnirana za kloniranje Git repozitorijuma sa određenog URI-ja u lokalni direktorijum i javna je i statička, što znači da se može pozvati bez kreiranja instance klase u kojoj je definisana. Prima dva parametra: `path`, koji specificira putanju lokalnog direktorijuma gde treba klonirati repozitorijum, i `taskPath`, koji se koristi za konstrukciju pune URL

adrese repozitorijuma. Metoda koristi try-catch blok za rukovanje potencijalnim izuzecima koji mogu nastati tokom procesa kloniranja repozitorijuma, što je esencijalno za robustno rukovanje greškama u operacijama mreže poput pristupa udaljenom repozitorijumu.

Unutar try bloka, metoda koristi statički poziv metode `Git.cloneRepository()` iz JGit biblioteke, koja olakšava rad sa Git repozitorijumima u Javi. URI udaljenog repozitorijuma se postavlja dodavanjem `taskPath` na osnovni URL koji se nalazi u `Config.HTTP_REPO_URL`, omogućavajući dinamičko specificiranje različitih repozitorijuma. Lokalni direktorijum za kloniranje određuje se korišćenjem `path`, a autentifikacija se vrši preko `UsernamePasswordCredentialsProvider` koji koristi korisničko ime i lozinku iz klase `Config`. Nakon postavljanja svih parametara, metoda `call()` pokreće operaciju kloniranja koja može baciti `GitAPIException` ako dođe do problema.

Ako je repozitorijum uspešno kloniran, na konzoli se ispisuje poruka "Repository cloned successfully." Ako dođe do greške, catch blok hvata izuzetak i ispisuje poruku o grešci na standardni izlaz za greške, pružajući detalje o problemu. Ova metoda omogućava jednostavno kloniranje Git repozitorijuma, rukovanje autentifikacijom i izveštavanje o greškama, čineći je korisnom za ponovnu upotrebu u različitim delovima aplikacije gde je potrebno kloniranje repozitorijuma. Na slici 4 prikazan je UML dijagram sekvenci koji ilustruje operaciju kloniranja git repozitorijuma.



Sl. 4. Kloniranje repozitorijuma - sekvencijalni dijagram

Java metoda `pushToRepository` je specifično dizajnirana za upravljanje i slanje promena u Git repozitorijum, koristeći JGit biblioteku za efikasno rukovanje operacijama kao što su upravljanje granama, preuzimanje

ažuriranja, priprema datoteka, čuvanje promena i konačno slanje na udaljeni repozitorijum. Evo detaljnog pregleda funkcionalnosti i procesa metode:

Potpis metode i parametri: Definisana kao `public static void pushToRepository(String path, String branchName, String message)`, ova metoda je dostupna bez instanciranja klase kojoj pripada i prihvata tri parametra: `path` (putanja direktorijuma lokalnog Git repozitorijuma), `branchName` (ciljana grana za slanje promena) i `message` (poruka za čuvanje promena kada se vrše promene).

Na početku, metoda pokušava da otvori navedeni lokalni Git repozitorijum koji se nalazi na datoj putanji. Ova operacija može baciti `IOException` ako putanja ne ukazuje na validan repozitorijum ili ako postoje problemi sa pristupom, ističući potrebu za rukovanjem grešaka kako bi se uhvatili takvi izuzeci.

Metoda nastavlja sa proverom da li navedeni naziv grane postoji unutar repozitorijuma. Iterirajući kroz sve postojeće grane, proverava se postojanje ciljane grane. Ako grana ne postoji, nova grana se kreira i aktivira. Suprotno, ako grana već postoji, jednostavno se aktivira ta grana. Ova dvostruka funkcionalnost osigurava da operacije uvek teku na ispravnoj grani, bilo prilagođavanjem postojećem okruženju ili uspostavljanjem novog po potrebi.

Za postojeće grane, metoda izvodi operaciju preuzimanja da sinhronizuje lokalnu granu sa njenim udaljenim pandanom, osiguravajući da je lokalni repozitorijum ažuran. Ovaj korak uključuje podešavanje autentifikacije i izvođenje preuzimanja, sa povratnim informacijama o uspehu ili neuspehu operacije, što je ključno za održavanje integriteta repozitorijuma i sprečavanje konflikata.

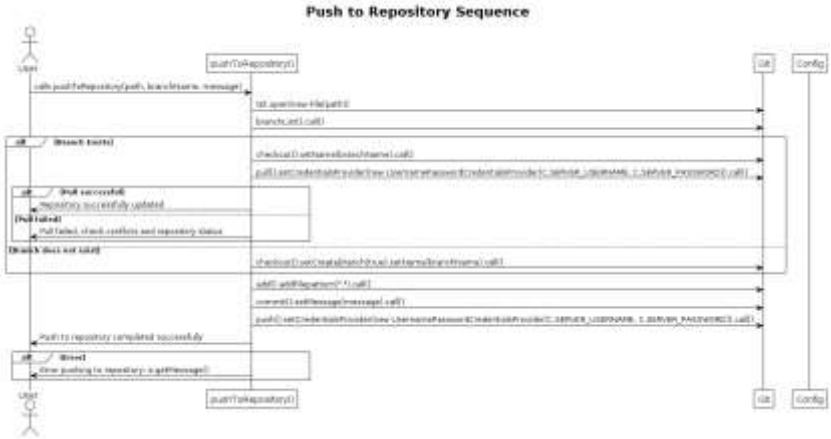
Sve promene u direktorijumu repozitorijuma se pripremaju za sledeće čuvanje koristeći širok obrazac datoteka, koji uključuje sve promene. Potom se te promene čuvaju lokalno sa datom porukom za čuvanje, kapsulirajući modifikacije u jednom čuvanju koje odražava najnovije stanje razvoja.

Nakon čuvanja promena lokalno, metoda postavlja i izvodi operaciju slanja na udaljeni repozitorijum. Autentifikacija se rukuje tokom ovog koraka kako bi se osigurala sigurna transmisija podataka. Poruka o uspehu potvrđuje završetak operacije, pružajući jasne povratne informacije o statusu izvršenja.

Kroz ceo proces, metoda je obavijena try-catch blokom dizajniranim za rukovanje i `GitAPIException` i `IOException`, osiguravajući robustno rukovanje greškama. Greške tokom bilo koje Git operacije se hvataju, i informativna poruka se štampa, detaljišući problem na osnovu poruke izuzetka.

Ukratko, metoda `pushToRepository` pruža sveobuhvatan i automatizovan pristup upravljanju Git operacijama unutar Java aplikacija (slika 5).

Dizajnirana je da precizno rukuje različitim scenarijima, od postavljanja novih grana do ažuriranja i slanja promena, sve dok pruža povratne informacije i osigurava rukovanje greškama kako bi se efikasno rešavali potencijalni problemi. Ova metoda je posebno vredna u automatizovanim okruženjima ili u scenarijima gde su česte izmene repozitorijuma neophodne.



Sl. 5. Slanje ka repozitorijumu - sekvencijalni dijagram

Obe metode koriste JGit biblioteku. Klasa `GitRepositoriesService` je strukturirana kako bi upravljala podacima Git repozitorijuma unutar aplikacije, prvenstveno služeći kao deo sloja poslovne logike. Ova klasa je zadužena za izvođenje CRUD (Create, Read, Update, Delete) operacija kroz interakciju sa `GitRepositoriesRepository`, koji deluje kao sloj pristupa podacima.

Konstruktor klase, `GitRepositoriesService(GitRepositoriesRepository repository)`, je dizajniran da primi objekat `GitRepositoriesRepository`. Ovaj repozitorijumski objekat je esencijalan za pristup bazi podataka koja čuva informacije o Git repozitorijumima. Jednom kada se primi, dodeljuje se privatnom polju unutar klase, osiguravajući da se sloj pristupa podacima održava odvojeno od logike servisa. Ova separacija sledi princip odvajanja briga, koji je osnovna smernica dizajna u softverskoj arhitekturi kako bi se poboljšala modularnost i olakšalo održavanje koda.

Klasa pruža tri glavne metode: `addRepository(String id, String url)`, `getRepository(String id)`, i `deleteRepository(String id)`. Metoda `addRepository` omogućava dodavanje novih Git repozitorijuma u bazu podataka. To postiže kreiranjem nove instance `GitRepository` sa datim `id` i `url` i zatim čuvanjem te instance korišćenjem metode `repository.save()`. Ovo

osigurava da je svaki Git repozitorijum jedinstveno identifikovan svojim `id`, a njegova lokacija je specificirana `url`-om.

Metoda `getRepository` dohvata URL Git repozitorijuma na osnovu njegovog jedinstvenog identifikatora. Koristi `repository.findById(id)` za pronalaženje repozitorijuma. Ako repozitorijum postoji, vraća URL koristeći `result.get().getUrl()`. Ako repozitorijum nije pronađen, što se proverava sa `result.isEmpty()`, metoda vraća `null`, što ukazuje na odsustvo repozitorijuma u bazi podataka.

Metoda `deleteRepository` je jednostavna; ona uklanja repozitorijum iz baze podataka koristeći njegov `id`. To se radi pozivanjem `repository.deleteById(id)`, što briše navedeni repozitorijum iz baze podataka. Ova metoda direktno menja bazu podataka uklanjanjem repozitorijuma i ne vraća nikakvu vrednost.

U celini, klasa `GitRepositoriesService` apstrakuje operacije vezane za upravljanje Git repozitorijumima, omogućavajući drugim delovima aplikacije da interaguju sa podacima Git repozitorijuma bez direktnog bavljenja operacijama baze podataka. Ova klasa je ključna komponenta servisnog sloja u aplikacijama, pružajući čistu separaciju briga i doprinoseći održivijem i skalabilnijem kodu [3] .

Git server omogućava kontrolu verzija za studentske predaje kroz korišćenje grana. Nova grana se kreira za svaku predaju studenta kako bi se održao zapis njihovog rada. Imena grana se tokenizuju kako bi se sprečio neovlašćen pristup.

C. Implementacija REST API-ja

Implementacija REST API-ja za platformu za učenje programiranja ključna je za olakšavanje komunikacije između klijenta i servera. API se razvija koristeći Spring okvir, koji pruža robustnu i skalabilnu platformu za izgradnju RESTful web servisa. Ključne komponente implementacije REST API-ja uključuju kontrolere, servise i bezbednosne mere.

D. Kontroler studenta

`StudentsController` unutar obrazovne platforme je pažljivo osmišljen da upravlja širokim spektrom funkcionalnosti vezanih za studente, značajno poboljšavajući korisničko iskustvo i efikasnost administracije. U srži njegovog dizajna je integracija sa raznim repozitorijumima i servisima, osiguravajući besprekorno manipulisanje podacima i robustnu implementaciju funkcija. Ovaj kontroler obavlja višestruke uloge, od upravljanja registracijama i brisanjem studenata do omogućavanja sigurnog pristupa obrazovnim resursima.

Osnovni aspekt `StudentsController-a` je njegova sposobnost da sveobuhvatno upravlja podacima o studentima. Interaguje sa `StudentsInfoRepository` za CRUD operacije, omogućavajući detaljno upravljanje profilima studenata. Dodatno, kontroler koristi `TokenManager` za generisanje i validaciju sigurnih tokena, što je ključno za autentifikaciju sesija studenata i zaštitu njihovih interakcija unutar platforme.

Štaviše, `StudentsController` je dizajniran da rukuje specifičnim akademskim funkcionalnostima kao što su predaja zadataka i pristup repozitorijumima. U partnerstvu sa `GitRepositoriesService`, upravlja vezama Git repozitorijuma za studente, omogućavajući im efikasan pristup i predaju zadataka. Ova integracija osigurava da studenti mogu preuzeti URL-ove repozitorijuma i upravljati svojim branch putanjama, što je esencijalno za održavanje individualnih radnih prostora i integriteta predaja.

Kontroler takođe koristi `CSVService` za uvoz podataka o studentima iz CSV datoteka, što olakšava operacije u velikim obimima i strukturira proces unosa podataka. Ovaj servis, zajedno sa `CSVHelper`, osigurava da uvezeni podaci odgovaraju zahtevanom formatu, poboljšavajući konzistentnost i pouzdanost podataka.

U pogledu operacija, `StudentsController` je opremljen mehanizmima za detaljno beleženje aktivnosti i izuzetaka, koristeći `LoggerFactory` za pružanje uvida u operacije sistema i brzo rešavanje problema. Ova funkcija je vitalna za održavanje transparentnosti i odgovornosti sistema.

Rukovanje izuzecima je još jedna ključna komponenta funkcionalnosti kontrolera, sa strukturama koje su postavljene za graciozno upravljanje greškama i pružanje smislenih povratnih informacija korisnicima. To uključuje rukovanje izuzecima vezanim za otpremanje datoteka putem `FileUploadExceptionAdvice`, osiguravajući da korisnici budu informisani o problemima kao što su prekoračene veličine datoteka bez kompromitovanja stabilnosti sistema.

Sigurnost je od primarne važnosti, adresirana kroz rigorozne provere tokena i sigurne prakse rukovanja podacima unutar kontrolera. Ove mere štite informacije o studentima i osiguravaju da je pristup obrazovnim resursima adekvatno regulisan.

Sve u svemu, implementacija `StudentsController-a` prikazuje sveobuhvatan pristup upravljanju interakcijama studenata unutar sofisticirane softverske arhitekture. Osigurava efikasnost, sigurnost i jednostavnost korišćenja, podržavajući raznolik spektar obrazovnih aktivnosti i administrativnih zadataka unutar platforme.

E. Kontroler zadataka

`AssignmentController` na obrazovnoj platformi je vešto izrađen kako bi upravljao zadacima putem RESTful API-ja, strukturiranog korišćenjem Spring frameworka i označenog anotacijom `@RestController`. Ova oznaka osigurava da se odgovori serijalizuju u JSON i besprekorno vraćaju klijentima. Kontroler je opremljen za obavljanje nekoliko ključnih funkcija kao što su listanje svih zadataka, preuzimanje detalja o specifičnom zadatku, kreiranje novih zadataka i brisanje postojećih.

Za listanje zadataka, metoda `getAssignment()` preuzima sve unose iz baze podataka koristeći `assignmentsRepository.findAll()` i dostupna je putem GET zahteva na putanji `/api/v1/assignments`. Kada preuzima specifičan zadatak sa `getAssignment(@PathVariable Long id)`, proverava postojanje entiteta koristeći `assignmentsRepository.findById(id)`. Ako je zadatak pronađen, vraća se, a ako ne, izdaje se status HTTP 400 sa odgovarajućom porukom o grešci.

Kreiranje zadataka se rukuje metodama `createAssignment(@RequestBody Assignment newAssignment)`, koji obrađuje POST zahteve. Ova metoda prihvata podatke o zadatku kao JSON objekat, koji se automatski pretvara u entitet `Assignment` i čuva u bazi podataka. Sačuvani entitet se zatim vraća u telu odgovora, potvrđujući kreiranje zadatka.

Brisanje zadataka upravlja `deleteAssignment(@PathVariable Long id)`, mapiran na DELETE zahtev. Ova funkcija takođe proverava prisustvo zadatka pre brisanja. Uspešno brisanje vraća potvrdnu poruku, dok neuspeh zbog ne-postojanja vraća poruku o grešci.

Što se tiče implementacije, kontroler koristi obrazac repozitorijuma preko `AssignmentsRepository`, koji apstrahuje interakcije s bazom podataka, pružajući čistu i održivu strukturu koda. Robusno rukovanje greškama je integralni deo dizajna kontrolera, osiguravajući da operacije kao što su preuzimanje ili brisanje zadataka budu praćene odgovarajućim HTTP status kodovima i porukama. To ne samo da održava integritet operacija već i poboljšava korisničko iskustvo pružajući jasne i izvodljive povratne informacije.

Dalje, upotreba `ResponseEntity` i `ResponseMessage` standardizuje API odgovore, osiguravajući konzistentan i dobro formatiran izlaz koji klijenti mogu lako tumačiti, bilo da se radi o uspešnim operacijama ili o rukovanju greškama.

Ukupno gledano, `AssignmentController` je dokaz efikasnog upravljanja resursima unutar sofisticirane web aplikacije, koristeći sposobnosti Spring-a i REST principa kako bi ponudio pouzdane i skalabilne krajnje tačke za

upravljanje obrazovnim zadacima, neophodne za administrativne i akademske funkcionalnosti platforme.

F. Docker Compose konfiguracija za platformu za učenje programiranja

Platforma koristi Docker Compose za orkestraciju REST API-ja i PostgreSQL baze podataka, omogućavajući laku i efikasnu primenu i skaliranje aplikacije. Docker Compose fajl definiše dva glavna servisa: aplikacioni servis (REST API) i servis baze podataka (PostgreSQL). Aplikacioni servis koristi unapred izgrađenu Docker sliku, dok servis baze podataka koristi zvaničnu Docker sliku. Definisanje volumena osigurava trajnost podataka baze podataka čak i nakon restarta kontejnera.

Implementacija REST API-ja i PostgreSQL baze podataka kao kontejnera korišćenjem Docker Compose-a predstavlja efikasno i fleksibilno rešenje za orkestraciju aplikacionih servisa. Ovaj pristup omogućava laku konfiguraciju, pokretanje i skaliranje servisa, osiguravajući konzistentnost okruženja i trajnost podataka. Docker Compose omogućava brzu primenu i upravljanje aplikacijom, čime se značajno pojednostavljuje razvojni i produkcioni proces platforme za učenje programiranja. Korišćenjem kontejnera, platforma postiže visoku efikasnost, brzinu i sigurnost, osiguravajući najbolja iskustva za svoje korisnike.

V. BEZBEDNOST U OBRAZOVNIM PLATFORMAMA

Obrazovne platforme za programiranje suočavaju se sa jedinstvenim sigurnosnim izazovima. Ključne mere zaštite uključuju implementaciju kontrole pristupa (RBAC) koja osigurava da samo ovlašćeni korisnici mogu pristupiti ili menjati podatke studenata, i dvofaktorsku autentifikaciju (2FA) za dodatnu sigurnost. Autentifikacija na osnovu tokena (JWT) omogućava sigurnu verifikaciju korisnika, dok enkriptovana komunikacija putem TLS protokola štiti podatke od prisluškivanja i manipulacije.

Mere poput detekcije plagijarizma, sigurnih okruženja za ispite i praćenja aktivnosti sprečavaju varanje. Enkripcija osetljivih podataka štiti informacije studenata, dok politike privatnosti u skladu sa zakonodavstvom (GDPR, FERPA) osiguravaju pravnu usklađenost. Redovno ažuriranje softvera i primena sigurnosnih zakrpa smanjuju rizik od napada. Edukacija korisnika o bezbednosnim praksama, uključujući prepoznavanje phishing napada i korišćenje jakih lozinki, ključna je za sigurnost platforme.

Plan za odgovor na incidente omogućava brzo reagovanje na sigurnosne pretnje. On uključuje procedure za identifikaciju, analizu, suzbijanje i

oporavak od incidenata, kao i komunikaciju sa relevantnim zainteresovanim stranama. Redovno testiranje i ažuriranje plana osigurava spremnost timova za efikasno reagovanje. Ove prakse osiguravaju da obrazovne platforme ostanu sigurne i otporne na napade, štiteći osetljive podatke korisnika i osiguravajući pouzdano okruženje za učenje.

A. Mere bezbednosti

Bezbednost je ključna briga za platformu za učenje programiranja, posebno za privatni Git server koji igra ključnu ulogu u obradi osetljivih podataka i predaja studenata.

B. Autentikacija zasnovana na tokenima

Platforma koristi autentikaciju zasnovanu na tokenima kako bi obezbedila API tačke, osiguravajući da samo ovlašćeni korisnici mogu pristupiti ili izmeniti resurse. Klasa TokenAuthenticationFilter obrađuje logiku autentikacije, proveravajući validne tokene pre nego što dozvoli pristup zaštićenim tačkama. Klasa FilterConfig konfiguriše filter kako bi osigurala relevantne API tačke, dok klasa TokenManager pruža metode za generisanje, brisanje i proveru tokena.

C. Tokenizacija imena resursa

Platforma koristi tokenizaciju za imena resursa, poput imena Git grana, kako bi sprečila neovlašćen pristup ili manipulaciju. Ovaj proces osigurava da osetljive informacije, poput ID-eva predavanja studenata, nisu direktno izložene, smanjujući rizik od zlonamernih uplitanja.

D. Kontrola pristupa

Platforma implementira kontrolu pristupa na osnovu uloga (RBAC) kako bi ograničila pristup osetljivim funkcijama i podacima na osnovu uloga korisnika. Ovo pomaže u sprečavanju neovlašćenih korisnika da pristupe ili izmene podatke do kojih ne bi trebalo da imaju pristup. Git server je konfigurisan da ograniči pristup na osnovu uloga korisnika, omogućavajući samo ovlašćenim korisnicima pristup određenim granama i osiguravajući da studenti mogu samo da pregledaju i menjaju svoje predaje.

E. Testiranje bezbednosti

Testiranje bezbednosti osigurava da platforma ne uvodi ranjivosti i da neovlašćeni pristup nije moguć. Fokus je na proceni rukovanja autentifikacionim tokenima i sigurnosti komunikacije između klijenta i servera. Testiranje uključuje simuliranje različitih napada kako bi se identifikovale i otklonile potencijalne ranjivosti.

Mere bezbednosti u platformi za učenje programiranja dizajnirane su kako bi zaštitile osetljive podatke, sprečile neovlašćeni pristup i očuvale integritet studentskih radova i rezultata ispita. Korišćenjem autentikacije zasnovane na tokenima, tokenizacije imena resursa i kontrole pristupa na osnovu uloga, platforma obezbeđuje sigurno i robustno okruženje za učenje, pružajući siguran i efikasan način upravljanja predajama studenata i unapređujući obrazovno iskustvo kako za studente tako i za nastavnike.

VI. Zaključak

Razvoj serverske infrastrukture za platformu za učenje programiranja obuhvatao je brojne izazove koji su pružili vredne uvide u tehničke i upravljačke aspekte projekta. Ključni izazovi uključivali su integraciju sa IntelliJ IDEA, skalabilnost servera, sigurnost, agilni razvoj, i testiranje. Integracija sa IntelliJ IDEA zahtevala je učenje specifičnog API-ja i obezbeđivanje sigurne komunikacije između dodatka i servera. Izgradnja skalabilne infrastrukture omogućila je obradu vršnih opterećenja, dok je implementacija robustnih sigurnosnih mera zaštitila osetljive podatke studenata.

U okviru razvoja platforme podignut je privatni Git server koji omogućava profesorima postavljanje studentskih ispita, dok studenti mogu preuzimati zadatke i vraćati rešenja u svoje grane. Git server je omogućio efikasno upravljanje kodom, praćenje promena i kolaboraciju na projektima, dok je centralizovano skladište zadataka obezbedilo sigurnu razmenu podataka.

Implementacija REST API-ja omogućila je efikasnu komunikaciju između klijenta i servera, olakšavajući ključne funkcionalnosti kao što su predaja zadataka, preuzimanje povratnih informacija i upravljanje repozitorijumima. Dizajn API-ja prioritetizovao je bezbednost, skalabilnost i jednostavnost korišćenja, podržavajući besprekornu integraciju sa klijentskim komponentama. RESTful arhitektura omogućila je lako proširivanje i održavanje sistema, osiguravajući da platforma može brzo da se prilagodi novim zahtevima i funkcionalnostima.

Agilna metodologija, posebno Scrum, olakšala je iterativni razvoj, koordinaciju tima i prilagođavanje promenljivim zahtevima. Sveobuhvatno testiranje osiguralo je funkcionalnost, performanse i sigurnost platforme, naglašavajući potrebu za pažljivim planiranjem i tehničkim znanjem. Ovi izazovi i rešenja rezultirali su robusnom i skalabilnom platformom koja zadovoljava potrebe studenata i edukatora, pružajući korisnicima angažovano i interaktivno iskustvo učenja.

Za budući rad planiramo razvoj dodatka za IntelliJ IDEA za profesore za

pregledanje i ocenjivanje studentskih zadataka, kao i dodatka za VS Code koji će studentima omogućiti preuzimanje ispita i predaju rešenih zadataka.

LITERATURA

[1] Donath, L., Mircea, G., & Rozman, T. (2020). E-Learning Platforms as Leverage for Education for Sustainable Development. European Journal of Sustainable Development, 9(2), 45-53.
[2] Decuypere, M., Grimaldi, E., & Landri, P. (2021). Introduction: Critical Studies of Digital Education Platforms. Critical Studies in Education, 62(1), 1-16.
[3] Martin, R. C. (2008). Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall.
[4] Richardson, L., & Amundsen, M. (2013). RESTful Web APIs. O'Reilly Media.

ABSTRACT

Educational institutions play a crucial role in equipping students with programming skills, making programming education a key segment of their curricula. To facilitate efficient learning and assessment, institutions require robust and adaptable platforms that support programming tasks and exams, ensure seamless integration with development environments, support collaborative learning, and provide secure and scalable operations. This paper addresses the challenges faced by existing programming learning platforms, such as limited integration, insufficient support for collaborative tasks, and lack of adequate security measures. It presents a solution involving the development and implementation of a robust server-side infrastructure for a programming learning platform. The platform integrates with IntelliJ IDEA, provides a private Git server, and utilizes a REST API to support various educational functions. The architecture, design, and key components of the system, including the database model and security measures, are discussed. The paper also highlights the benefits of an agile development methodology and presents plans for future enhancements, including developing extensions for IntelliJ IDEA and VS Code for teachers and students.

Development of a Programming Learning Platform - Server Side

Luka Dević, dr Bojana Dimić Surla