Co-funded by the European Union

ERASMUS+ PROJECT 2023-1-RS01-KA220-HED-000156660

EPIR | E-Procedure of Institutional Recognition of Foreign Higher Education Documents

WORK PACKAGE 3

WP3 - Development and improvement of IT systems for the recognition process of foreign students' HE documents

Testing report

Project	**E-Procedure of Institutional Recognition of Foreign Higher Education Documents**
Work Package 3	**Development and improvement of IT systems for the recognition process of foreign students' HE documents**
Activity 4	**Testing in which the programming phase is put through a series of structured tests at each partner (unit, system and user acceptance tests)**
Prepared by	**University of Novi Sad** **Center for Information Technologies** **Tatjana Zubić** **Tatjana Tošić** **Zoran Vojnović**

EPIR project partners:

UNIVERSITY OF NOVI SAD · UNIVERSITÀ POLITECNICA DELLE MARCHE · UAB ROMANIA 1 DECEMBRIE 1918 UNIVERSITY OF ALBA IULIA · 50 years of University of Split · REPUBLIC OF SERBIA Qualifications Agency

Contents

1. Introduction

The primary objective of Work Package 3 (WP3) within the EPIR project is the development of new or the improvement of existing IT systems that support the institutional recognition of foreign higher education documents at partner institutions. This objective is pursued through a complete software development lifecycle, including system analysis and design, programming, testing, and implementation of new or upgraded digital tools.

At the University of Novi Sad (UNS) a new IT solution, designed to meet the UNS specific needs and requirements is developing in the scope of this work package. This software system should enable candidates to submit recognition requests and accompanying documents online, simplify the tracking of the recognition process for all participants, provide more efficient reporting for institutional and regulatory purposes, and reduce the number of paper documents. The introduction of this system is expected to benefit all stakeholders involved in the recognition process.

The University of Novi Sad WP3 project team consists of a design team from the Centre for Information Technologies at UNS, responsible for system analysis and design and software development coordination, testing, and implementation, and a development team from the University of Novi Sad Faculty of Technical Sciences, tasked with coding and software development.

Following the completion of the programming phase (WP3 Activity 3.3 - from October 1, 2024 to December 15, 2025), the University of Novi Sad entered the **testing phase** (WP3 Activity 3.4 - from December 15, 2025 to March 1, 2026) with the objective of validating the functionality, reliability, and usability of the newly developed information system for the recognition of foreign higher education documents.

The testing phase focused on verifying that the implemented functionalities operate in accordance with the System Requirements Specification, institutional procedures, and applicable legal frameworks and administrative practices governing recognition procedures. Particular attention was given to validating complete business workflows, system integration, and user interaction before the system entered the implementation phase.

The testing activities were conducted using an iterative approach, allowing identified issues to be analysed and incorporated into subsequent system refinements.

2. Objectives

The primary objectives of the testing phase were to:

- functional verification of integrated modules;
- validation of end-to-end recognition workflows;
- evaluate system usability for different user groups (Candidate, Representative, Officer, Supervisor, Head of study programme, Vice-dean, Administrator);
- interaction between frontend, backend services and the database;
- verification of business rules and data validation;
- authentication and authorization mechanisms;
- document management;
- notification services;
- reporting functionalities.

3. Testing Approach and Methodology

Testing was carried out between 15 December 2025 and 1 March 2026 using a structured approach combining technical validation and user-oriented evaluation.

Testing consisted of several stages:

- Unit testing,
- System testing,
- User Acceptance Testing (UAT).

Unit testing was performed throughout the programming phase and continued during testing phase to verify functionality and correctness of individual software components and services. As part of unit testing, architectural validation tests were implemented to ensure compliance with the layered monolithic architecture adopted for the EPIR system. These tests verified architectural constraints, including permitted dependencies between application layers, enforcement of access rules, and protection of internal service methods through custom annotations.

System testing was aimed at validating the integration between software modules, backend services, databases, and user interfaces, and functional testing of integrated modules.

User Acceptance Testing (UAT) was carried out to verify that the implemented workflows, functionalities, and user interface met operational needs and institutional requirements.

To support the validation process, a dedicated testing environment (epir-test.uns.ac.rs) was established. This environment provided a controlled setting for all testing activities before implementation phase and deployment.

The testing process involved close cooperation between software developers from the Faculty of Technical Sciences, the design team from the Centre for Information Technologies, and representatives of the Legal Office with administrative staff responsible for recognition procedures. This multidisciplinary approach ensured that both technical performance and institutional requirements were evaluated throughout the testing process.

4. Testing Activities

Testing covered all major functional modules developed during the programming phase, including:

- user groups authentication and authorization;
- general system settings: registries, catalogues and reference data;
- online application submission;
- document upload and management;
- assignment of tasks according to user roles;
- workflow management throughout the recognition procedure;
- academic evaluation workflow module and internal processing;
- evaluation and final report generation;
- communication and notification services;
- workflow tracking and monitoring of application status and progress;
- other reporting functionalities.

Each module was evaluated individually and as part of complete end-to-end business scenarios representing real institutional recognition procedures.

Special attention was given to validating interactions between individual system components, correctness of workflow transitions, consistency of stored data, and compliance with administrative procedures.

5. Testing Results

Overall, testing confirmed that the developed system successfully supports the core business processes defined during the analysis and design phases.

The implemented functionalities performed reliably under regular operating conditions, while integration between frontend components, backend services, and the database proved stable throughout the testing process.

User acceptance testing confirmed that the system provides a clear and structured workflow supporting administrative staff during recognition procedures.

The testing process also demonstrated that the selected technical architecture provides a solid foundation for future system expansion and integration with additional institutional services.

6. Identified Issues and Corrective Actions

As expected during validation of a newly developed system, testing identified several areas requiring additional refinement.

The majority of observations related to:

- refinement of validation rules;
- optimization of individual workflow steps;
- correction of minor software defects;
- improvements to user interface functionality;
- clarification of certain administrative procedures within the application.

None of the identified issues affected the overall system architecture or core functionality.

All findings were documented and prioritised according to their impact. Critical issues were resolved during the testing phase, while minor improvements and usability refinements were scheduled for implementation as part of the deployment activities.

7. Readiness for Implementation

Based on the completed testing activities, the University of Novi Sad concluded that the system is technically stable and functionally mature for implementation phase.

Testing confirmed that the core system functionalities operate in accordance with the defined requirements and support the complete recognition workflow.

The testing process also identified several opportunities for further refinement, primarily related to usability improvements, validation rules, workflow optimisation, and minor functional adjustments. These refinements will be incorporated during the implementation phase to ensure a stable, efficient, and user-oriented production environment.

8. Conclusion

The testing phase successfully validated the functionality, reliability, and usability of the newly developed EPIR information system at the University of Novi Sad.

The activities carried out during this phase confirmed that the software solution satisfies the functional and technical requirements established during previous WP3 activities and provides effective support for the digitalisation of recognition procedures.

The iterative testing process verified system readiness and generated valuable feedback for further optimisation. These improvements will be incorporated during the implementation phase, ensuring a smooth transition from testing to operational use and contributing to the long-term sustainability and quality of the developed solution.

9. Appendix

Integral part of this report is the technical documentation prepared in Serbian language during the testing phase. It describes the architectural tests, integration and system testing procedures, and performance testing of the system.

APPENDIX: Testing documentation

Co-funded by the European Union

ERASMUS+ PROJECT 2023-1-RS01-KA220-HED-000156660

EPIR| E-Procedure of Institutional Recognition of Foreign Higher Education Documents

Dokument:	**Testiranje sistema**
Verzija:	1.0
Dokument pripremili:	Danijel Venus Igor Zečević Ljubomir Milašinović

APSTRAKT

EPIR project partners:

UNIVERSITY OF NOVI SAD

UNIVERSITÀ POLITECNICA DELLE MARCHE

ROMANIA 1 DECEMBRIE 1918 UNIVERSITY OF ALBA IULIA

50 years of University of Split

REPUBLIC OF SERBIA Qualifications Agency

Cilj digitalizacije procesa priznavanja stranih visokoškolskih isprava koju obavlja kancelarija za pravne poslove Univerziteta u Novom Sadu je omogućavanje kandidatima online podnošenje zahteva za priznavanje i pratećih dokumenata, kao i povećanje efikasnosti rada.

Na osnovu prethodno obavljene analize sistema, formirane specifikacije zahteva i kreiranja uzorka dizajna korisničkog interfejsa (mockup), kreiranog operativnog prototipa aplikacije kao i implementiranog konceptualnog, fizičkog modela strukture podataka kao i same baze podataka pristupilo se izradi arhitekture sistema kao i implementaciji API Backend servisa.

Arhitektura EPIR sistema zasniva se na višeslojnom modelu koji obuhvata korisnički interfejs za online podnošenje zahteva, aplikacioni sloj za obradu podataka i integracije, te centralizovanu bazu podataka. Kandidati putem web portala podnose zahteve i dokumenta, koja se automatski prosleđuju odgovarajućim službama. Aplikacioni sloj obavlja validacije, upravljanje tokovima i komunikaciju sa internim servisima Univerziteta. Sistem omogućava transparentno praćenje statusa, bržu obradu i pouzdano čuvanje svih podataka.

ABSTRACT

The goal of digitalizing the process of recognizing foreign higher education documents, carried out by the Legal Affairs Office of the University of Novi Sad, is to enable applicants to submit recognition requests and accompanying documents online, as well as to increase work efficiency.

Based on the previously conducted system analysis, the established requirements specification, the creation of a user interface mockup, the developed operational prototype of the application, and the implemented conceptual and physical data model including the database itself, work commenced on designing the system architecture and implementing the API backend service.

The EPIR system architecture is based on a multi-layer model that includes a user interface for online application submission, an application layer for data processing and integrations, and a centralized database. Applicants submit requests and documents through the web portal, which are automatically forwarded to the appropriate departments. The application layer performs validations, workflow management, and communication with the University's internal services. The system enables transparent status tracking, faster processing, and reliable storage of all data.

Sadržaj

Opseg sistema

Kancelarija za pravne poslove Univerziteta u Novom Sadu obavlja poslove priznavanja stranih visokoškolskih isprava za potrebe nastavka školovanja na Univerzitetu u Novom Sadu. Ova aktivnost predstavlja važan segment međunarodne saradnje i mobilnosti studenata, jer omogućava kandidatima iz drugih država da nastave svoje obrazovanje u okviru Univerziteta, u skladu sa važećim propisima i standardima.

Trenutno, proces priznavanja stranih visokoškolskih isprava nije digitalizovan, već se većina dokumenata i komunikacije odvija u papirnom obliku. Kandidati zahteve podnose lično ili putem pošte, dok zaposleni u Kancelariji za pravne poslove u upravljanju procesom koriste imejl komunikaciju i standardne kancelarijske softvere (tekstualne obrade, tabele, baze u ograničenom obliku). Ne postoji namensko softversko rešenje koje bi objedinjavalo sve aktivnosti i omogućilo integrisano praćenje toka priznavanja — od prijema zahteva do izdavanja odluke.

Digitalizacija ovog procesa ima za cilj da značajno unapredi način rada i iskustvo korisnika. Novi softverski sistem omogućiće kandidatima da elektronski podnesu zahtev za priznavanje i prate njegov status u realnom vremenu, dok će Kancelariji za pravne poslove i drugim organizacionim jedinicama Univerziteta obezbediti centralizovan, pregledan i efikasan način vođenja postupka.

Očekivani efekti uvođenja sistema su:

- Povećanje efikasnosti rada kroz automatizaciju administrativnih koraka i smanjenje manuelnog unosa podataka,
- Ubrzanje razmene informacija između svih učesnika u procesu,
- Smanjenje upotrebe papira i prelazak na elektronsku dokumentaciju,
- Poboljšano izveštavanje i jednostavnije generisanje statističkih podataka za potrebe Univerziteta i nadležnih organa, kao i povećanje transparentnosti i pristupačnosti procesa priznavanja za kandidate iz inostranstva.

Dodatno, očekuje se da digitalizacija ovog postupka doprinese jačanju međunarodne vidljivosti Univerziteta u Novom Sadu i povećanju broja stranih studenata zainteresovanih za nastavak studija na Univerzitetu, čime će se unaprediti i proces akademske mobilnosti i međunarodne saradnje.

Arhitektonska ograničenja

Zbog održanja konzistentnosti i poštovanja skupa ograničenja koja slojevita monolitna arhitektura propisuje, kao i idiomi u Java Sprint Boot sloju, napisani su bazični testovi kako bi se ovi elementi održali tokom razvojnog ciklusa sistema.

Testovi su pisani u okviru backend sloja (Java ekosistem). Definisani slojevi su:

- Data access sloj (repozitorijumi podataka, sloj perzistencije)
- Servisni sloj (poslovna logika sistema)
- Kontroler sloj (propisivanje skupa pristupnih tačaka od strane drugih delova sistema kao što je front)
- Aspect sloj (cross cutting logika - prožimajuća logika, kao što je prava pristupa resursima)
- Konfiguracioni sloj

Pravila pristupa slojevima:

- Kontroler sloju se ne može pristupiti. Obrnuto kontroler sloj pristupa servisnom sloju
- Aspect, konfiguracioni i kontroler sloj mogu pristupiti servisnom sloju
- Servisni i aspect sloj jedini imaju pristup Data access sloju

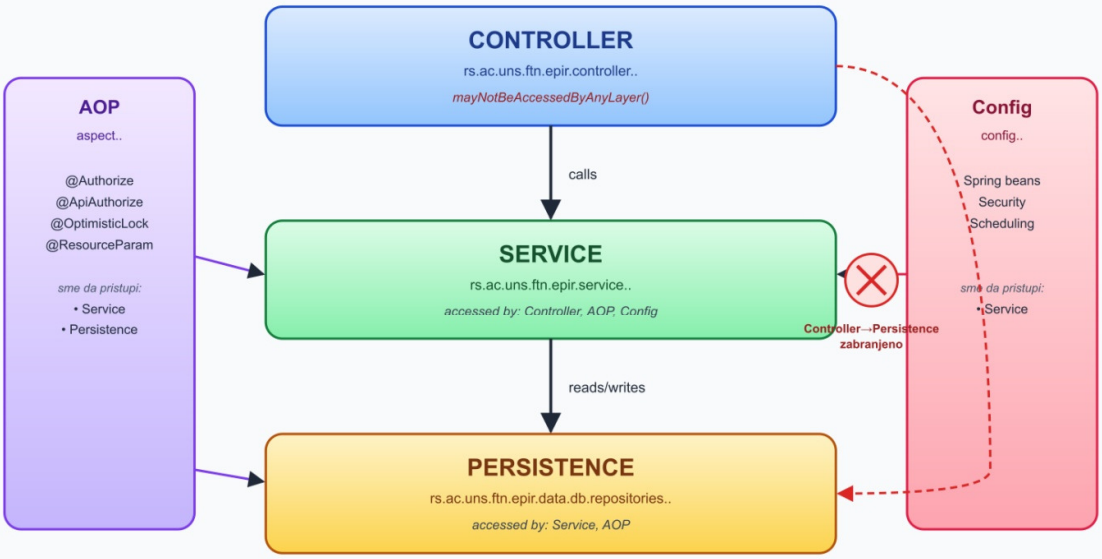
Komunikacija koja treba da ide isključivo između servisnih komponenti i nikada da ne izlazi izvan njihovog opsega se anotira anotacijom @InternalUseOnly. Postoji test koji takođe proverava da li je ovakvo arhitektonsko ograničenje ispoštovano. Ima potrebe da ne izlaze izvan servisnog sloja, jer nemaju na primer punu kontrolu pristupa ili su obuhvatane u veće poslovne transakcije. Zbog toga ima potrebe ograničiti da ne bi slučajno bili pozvani iz kontroler sloja na primer i time razbili kontrolu pristupa.

Arhitektonski test je pisan pomoću biblioteke ArchUnit (https://www.archunit.org/) i izvršava se automatski u CI/CD (Continuous Integration / Continuous Deployment) tokovima kao i u razvojnom okruženju programera prilikom standardnog građenja aplikacije korišćenjem maven alata.

Na sledećim slikama je dat opšti primer arhitektonskih ograničenja opisanih gore u tekstu.

Layered Architecture – ArchitectureTest.testLayers()

rs.ac.uns.ftn.epir



Legenda pravila

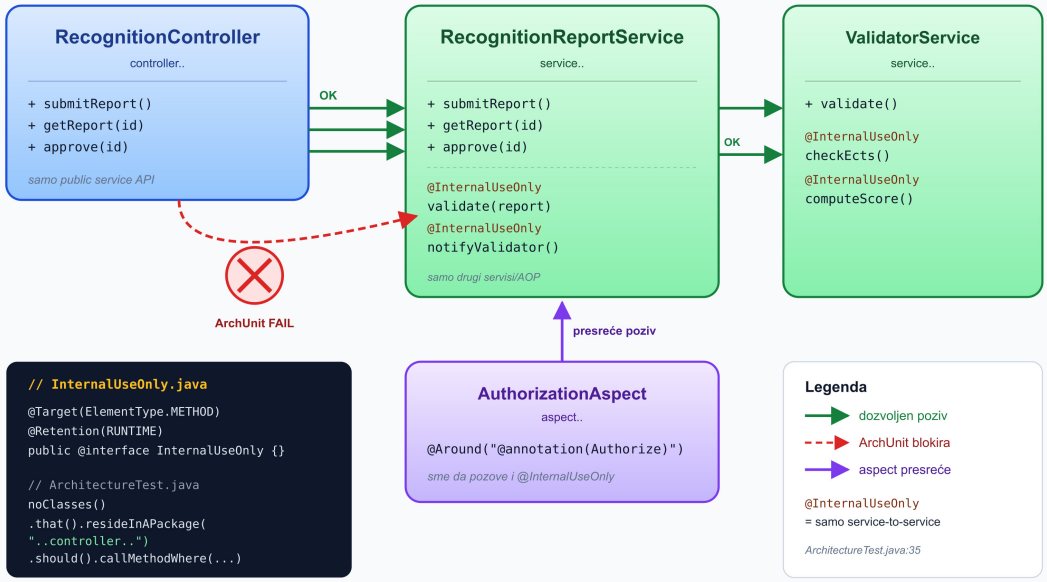
dozvoljen poziv — arhitektonski zabranjen poziv

Controller nije pristupljen ni od koga – **Service** samo iz Controller/AOP/Config – **Persistence** samo iz Service/AOP

Source: src/test/java/rs/ac/uns/ftn/epir/ArchitectureTest.java – testLayers()

Service-to-Service komunikacija – testInternalServiceMethods()

@InternalUseOnly metode ne smeju biti pozvane iz controllera



Source: src/test/java/rs/ac/uns/ftn/epir/ArchitectureTest.java, aspect/annotations/InternalUseOnly.java

Primer implementacije testa arhitektonskih ograničenja korišćenjem ArchUnit biblioteke:

```
class ArchitectureTest {

    private final JavaClasses classes = new ClassFileImporter()
            .withImportOption(new ImportOption.DoNotIncludeTests())
            .importPackages("rs.ac.uns.ftn.epir");

    @Test
    void testLayers() {
        Architectures.layeredArchitecture()
                .consideringAllDependencies()
                 .layer("Controller").definedBy("rs.ac.uns.ftn.epir.controller..")
                .layer("Service").definedBy("rs.ac.uns.ftn.epir.service..")
                .layer("Persistence").definedBy("rs.ac.uns.ftn.epir.data.db.repositories..")
                .layer("AOP").definedBy("rs.ac.uns.ftn.epir.aspect..")
                .layer("Config").definedBy("rs.ac.uns.ftn.epir.config..")
                .whereLayer("Controller").mayNotBeAccessedByAnyLayer()
                .whereLayer("Service").mayOnlyBeAccessedByLayers("Controller", "AOP", "Config")
                .whereLayer("Persistence").mayOnlyBeAccessedByLayers("Service", "AOP")
                .check(classes);
    }

    @Test
    void testInternalServiceMethods() {
        noClasses()
                .that()
                .resideInAPackage("rs.ac.uns.ftn.epir.controller..")
                .should()
                .callMethodWhere(DescribedPredicate.describe("annotated with @InternalUseOnly",
                        call -> call.getTarget().isAnnotatedWith(InternalUseOnly.class)))
                .check(classes);
    }
}
```

Integraciono testiranje/testiranje sistema

Pored testova arhitektonskih ograničenja, pisani su i testovi za ponašanje sistema u različitim primerima toka poslova (workflow), tj procesa podnošenja prijave za priznavanje stranih isprava.

Korišćen je standardni mehanizam *JUnit* platforme i *Spring Boot* testiranja u specijalnom kontekstu koji se podiže prilikom testiranja i koji simulira realnu server aplikaciju. Testiranje se oslanja na instancu baze podataka u kojoj već postoje popunjeni osnovni katalozi kao što su svrhe podnošenja, statusi prijava, ustanove, studijski programi, metapolja prijave i slično. Ostali zavisni servisi se pokreću kao docker kontejneri na dev mašini na primer na kojoj se testira proces. Topologija docker kontejnera je opisana u *docker-compose* fajlu kao standardni mehanizam podizanja sistema međusavisnih čvorova u docker ekosistemu.

Testovi su organizovani u grupe međuzavisnih testova (test suites) koji se zajedno izvršavaju kao celina i predstavljaju jedan primer toka posla podnošenja neke prijave.

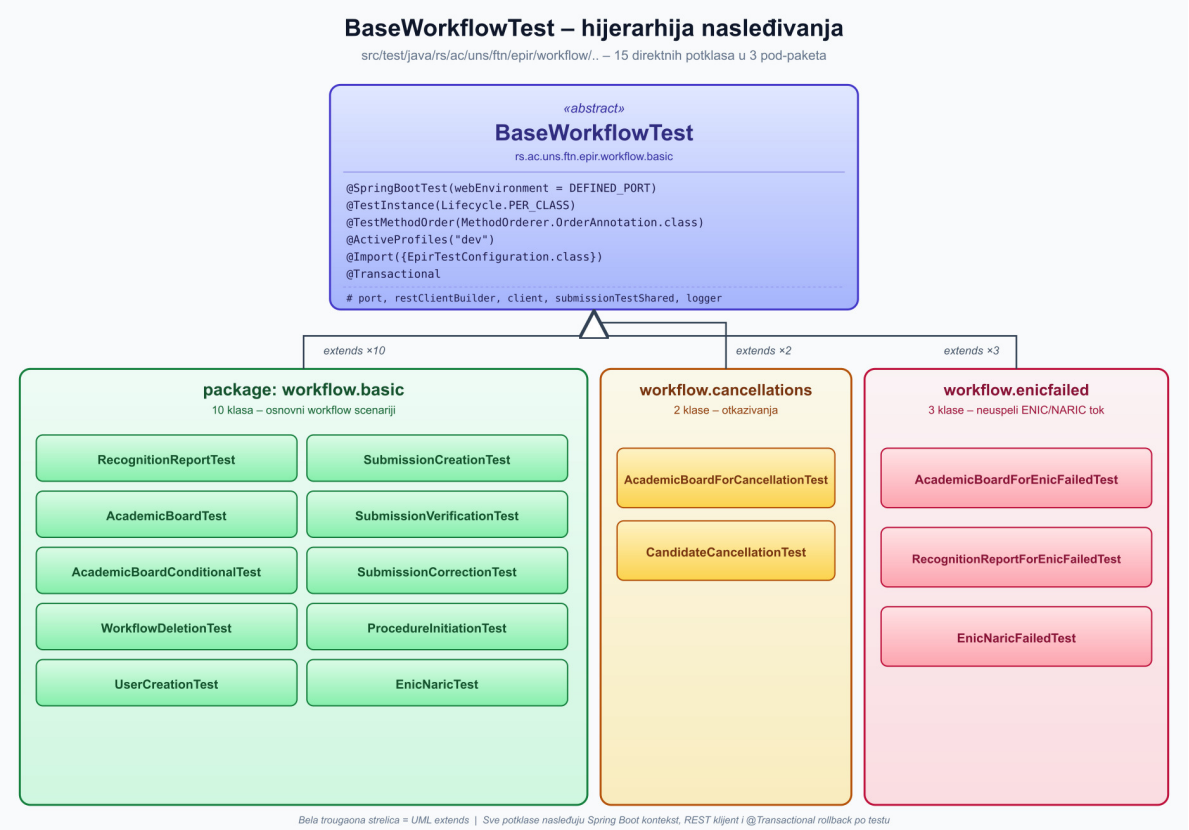
Svi testovi nasleđuju bazični test (*BaseWorkflowTest*) koji ima definisan mehanizam pristupa tačkama backend sistema. Na slici ispod je prikazana hijerarhija klasa testova koja nasleđuju *BaseWorkflowTest*.

Testovi rade po principu poziva endpoint-ova sistema i testiranja da li odgovor zadovoljava kriterijume. Simulira se kompletan proces. Ukratko:

- Kreiraju se testni korisnici u bazi (kandidat, supervizor, prodekan, rukovodilac, stručno veće, ...)
- Vrši se poziv endpoint-ova sistema gde se kreira prijava, popunjavaju podaci, šalju službeniku, fakultetu, stručnom veću, ...
- Na kraju svakog testa se brišu testni podaci i korisnici koji su privremeno bili kreirani

Na sledećoj slici je prikazan hijerarhija testnih klasa za uobičajene tokove posla prijave

Svako stanje u procesu prijave za priznavanje je predstavljena kao jedna klasa sa testnim metodama za testiranje pojedinačnih koraka. Na primer klasa SubmissionCreationTest testira kreiranje nove prijave. Poseduje metode na primer: testLogin, testNewSubmission, testUpdatePurpose, testCreateDfr, testPreviousEducation, testUploadDocuments za pokretanje testova prijave na sistem, kreiranja nove prijave, ažuriranja svrhe podnošenja, ažuriranje podataka o ispravi, prethodnom obrazovanju i otpremanje dokumenata respektivno.



Da bi test mogao da funkcioniše bilo je potrebe da se izvesni servisi zamene maketama (mockup servisi) u procesu testiranja, kako ne bi izvršavale realni posao. Na primer servis za slanje mejlova je potrebno zameniti servisom koji ne radi ništa, tj. vraća pozitivan odgovor slanja mejla iako ništa ne radi. Radni okvir za testiranje nudi fleksibilan mehanizam za zamene pojedinih servisa, bez menjanja samog koda aplikacije.

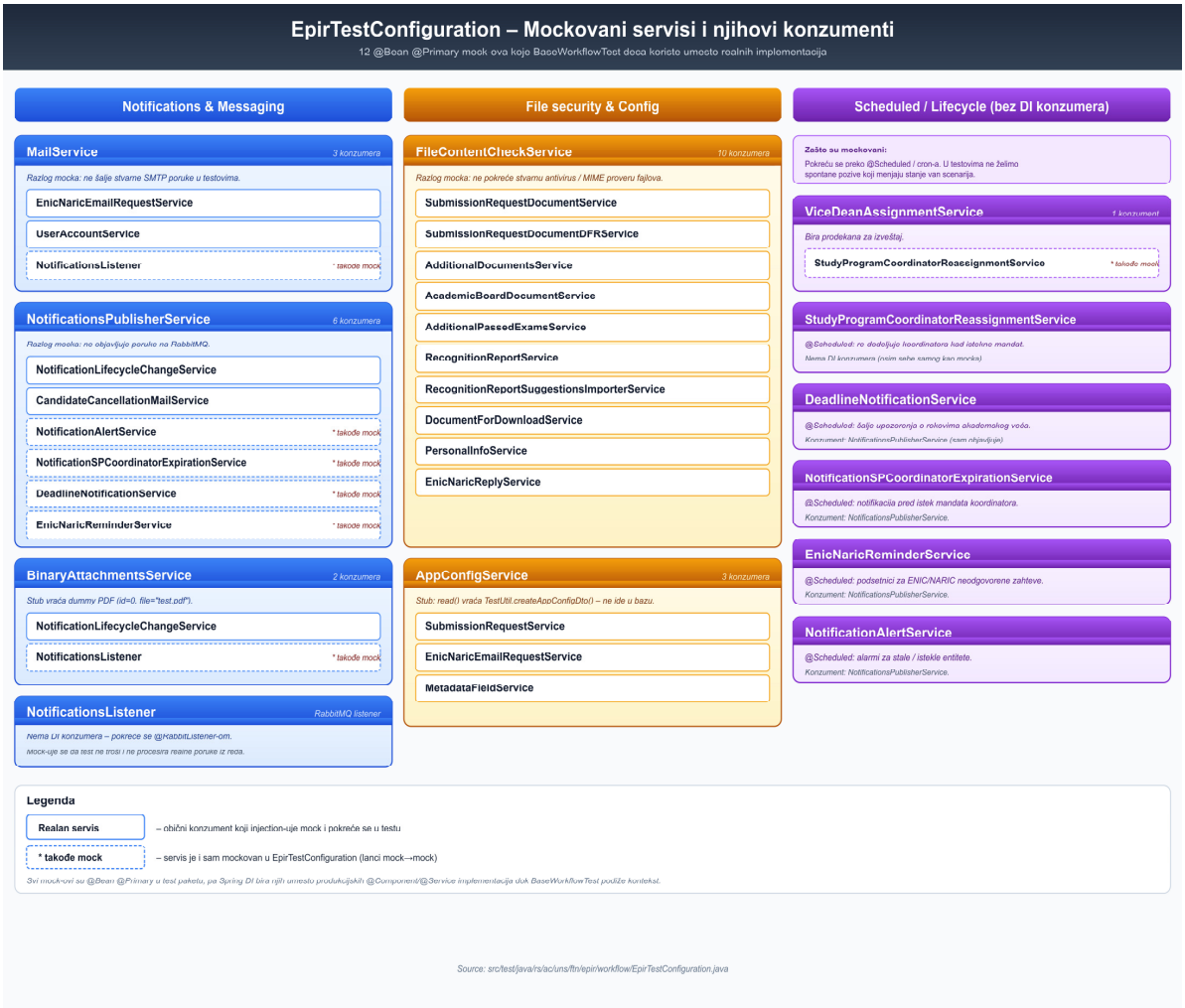
Servisi koji se zamenjuju u test fazi su:

- **MailService** – servis za fizičko slanje imejl poruka korisnicima
- **NotificationsPublisherService** – servis za slanje poruka u Kafka red koji se konzumira od strane servisa za slanje mejlova
- **NotificationsListener** – servis koji konzumira iz Kafka reda poruke za slanje imejl kanalom
- **FileContentCheckService** – servis za proveru da li fajl koji se otprema predstavlja potencijalno izvršnu verziju, zlonamernu za sistem. Koristi se prilikom provere datoteka koji se otpremaju u sistem kao deo dokumentacije zahteva za priznavanjem
- **DeadlineNotificationService** – servis koji šalje periodičnu kandidatima obaveštenje da su na polovini perioda za dostavljanje uverenja o položenim dodatnim ispitima
- **NotificationSPCoordinatorExpirationService** – servis koji šalje imejl obaveštenje prodekanu da su pojedini rukovodioci pri kraju sa svojim mandatima kao rukovodioci studijskih programa
- **EnicNaricReminderService** – servis za slanje notifikacije službenicima da se predugo čeka na odgovor od ENIC/NARIC centra

- **StudyProgramCoordinatorReassignmentService** – automatsko dodeljivanje novom rukovodiocu studijskog programa, kada starom istekne mandat.
- **ViceDeanAssignmentService** – automatska dodela izveštaja prodekanu za nastavu ako niko od rukovodioca ne preuzme izveštaj na popunjavanje
- **BinaryAttachmentService** – servis za kreiranje attachment-a u imejl poruci. Zamenjuje se sa praznim koji vraća rezultat da je attachovan fajl sa nazivom test.pdf na primer
- **NotificationAlertService** – servis za slanje obaveštenja da prijava stoji predugo u nekom status
- **AppConfigService** – servis koji vraća generalnu konfiguraciju sistema kao što je podrazumevani maksimalni broj dokumenata za upload ili email adresa ENIC/NARIC centra. Upravo zbog ovoga je servis zamenjen kako bi simulirao drugu imejl adresu za test slanja poruke ENIC/NARIC centru

Testiranje otpremanja dokumenata je urađeno tako da se generišu random bajtovi do određenog limita i kreira se na osnovu njega fajl koji je tipa pdf (iako nije ispravan) koji se zatim otprema u sistem.

Na sledećoj slici je prikazana zavisnost servisa od servisa koji su zamenjeni



Trenutni tokovi posla koji se testiraju su:

- Podrazumevani tok posla: kreira se prijava, popunjava, šalje službeniku. Službenik vrši verifikaciju i postavlja na nekoliko polja da nisu validna i vraća kandidatu. Kandidat ispravlja i vraća službeniku. Službenik pokreće postupak i prosleđuje fakultetu. Preuzima je prodekan za nastavu i popunjava izveštaj. Vraća službeniku na korekciju. Službenik zatim vraća fakultetu nazad. Fakultet generiše izveštaj, otprema i šalje službeniku. Službenik potvrđuje da je dostavljen potpisani primerak izveštaja. Generiše predlog slanja stručnom veću, bira veće i šalje. Veće donosi pozitivan odgovor. Službenik šalje notifikaciju kandidatu o preuzimanju rešenja, ...
- Tok posla sa izuzetkom da je ENIC/NARIC odgovor negativan, tj. studijski program trenutno nije akreditovan. Generiše se poseban predlog stručnom veću i šalje se....
- Tok posla sa izuzetkom da je uslovno priznavanje. Ubačeni su testovi za otpremanje uverenja o položenim dodatnim ispitima kandidata

Podrazumevani tok posla

Primer java koda za konstrukciju grupe međuzavisnih testova

```
@Suite
@SelectClasses({
	UserCreationTest.class,
	SubmissionCreationTest.class,
	SubmissionVerificationTest.class,
	SubmissionCorrectionTest.class,
	ProcedureInitiationTest.class,
	EnicNaricTest.class,
	RecognitionReportTest.class,
	AcademicBoardTest.class,
	WorkflowDeletionTest.class
})
public class WorkflowTestSuite {
}
```

Tok posla hronološki koji se testira grupisani po Java klasama i ispod spisak metoda unutar klase:

- UserCreationTest
  - testCreateCandidate – kreiranje kanidata kao korisnika u sistemu. Ovo spada više u pripremu podataka, a ne realni test kreiranja kandidata
  - testCreateSupervisor – kreiranje jednog supervizora. Supervizor će vršiti verifikaciju podataka i voditi ciklus podnošenja prijave umesto službenika u ovom procesu testiranja
- SubmissionCreationTest
  - testLogin – testiranje login kandidata
  - testNewSubmission – iniciranje nove prijave, popunjavanje personalnih podataka
  - testUpdatePurpose – izbor svrhe podnošenja i izbor studijskog programa/module i ustanove
  - testCreateDfr – kreiranje jedne isprave za priznavanje
  - testPreviousEducation – popunjavanje podataka o prethodnim obrazovanjima kandidata

    - testUploadDocuments – otpremanje dokumenata prijave
    - testUploadDfrDocuments – otpremanje dokumenata na nivou isprave za priznavanje
    - testSubmitSubmission – potvrda podnošenja prijave od strane kandidata
- SubmissionVerificationTest
    - testSupervisorLogin – login supervizora
    - testAssignSubmission – preuzimanje prijave za obradu od strane supervizora
    - testTranscription – unos transkribovanih podataka
    - testVerificationSetStatusOnSubmissionAll – postavljanje podrazumevano na sva polja da su uspešno pregledana, osim nekoliko polja koja su markirana da su nevalidna: država rođenja, mesto prebivališta, prethodno obrazovanje grad ustanove...
    - testVerificationSetStatusOnDfrLevelAll – postavljanje podrazumevano na sva polja u okviru prijave da su uspešno pregledana osim nekoliko polja koja su markirana da su nevalidna: naziv dodatne ustanove, trajanje studija u godinama...
    - testSendBackToCorrection – vraćanje kandidatu na dopunu
- SubmissionCorrectionTest
    - testCorrectionPersonalData – ispravljanje personalnih podataka
    - testCorrectionDfr – ispravljanje podataka u ispravi za piznavanje
    - testUploadDocuments – ispravljanje nevalidnih dokumenata. Brisanje starih i upload novih
    - testSubmitCorrection – potvrda ispravke podataka od strane kandidata
- ProcedureInitiationTest
    - testVerificationSetStatusOnSubmissionAllCorrection – postavljanje statusa na svim poljim da su validni
    - testVerificationSetStatusOnDfrLevelAllCorrection – postavljanje statusa na svim poljima da su validni na nivou isprave
    - testProcedureInitiated – potvrda verifikacije, tj. pokretanje postupka
    - testAdditionalDocumentationUpload – postavljanje dodatne dokumentacije od strane supervizora
    - testProcedureInitiatedReport – test generisanja izveštaja o pokrenutom postupku. Testira se tako što se očekuje da enpoint vrati niz bajtova. Zatim se taj niz bajtova parsira u pdf i proverava da li je pdf prazan. Razlog zašto se proverava da li je pdf prazan jeste zato što biblioteka jasper koja služi za generisanje izveštaja, ako dođe do neke greške vrati prazan pdf u nekim slučajevima
- EnicNaricTest
    - testEnicNaricCreate – kreiranje ENIC/NARIC zahteva kao kataloški podataka nezavisno od prijave. Testira se slučaj gde je ENIC/NARIC već dobio odgovor (pozitivan)
    - testEnicNaricToSubmission – vezivanje ENIC/NARIC zahteva za prijavu
    - testEnicNaricCreateAgain – ponovno kreiranje novog ENIC/NARIC zahteva (opet pozitivan)
    - testEnicNaricToSubmissionAgain – ponovno vezivanje novog ENIC/NARIC zahteva za prijavu
    - testListSubmissionByEnicNaricId – testiranje čitanja svih prijava koji su vezani za zadati ENIC/NARIC zahtev (ona koji je kreiran u ovom test scenariju kako bi se znao očekivani rezultat upita)
    - testEnicEmailRequestSendNoEmailConfigured – testiranje slanja imejl poruke ENIC/NARIC centru u slučaju kada u sistemu nije konfigurisana imejl adresa ENIC/NARIC centra
    - testEnicEmailRequestSend – testiranje slanja imejl poruke ENIC/NARIC centru u slučaju kada je definisana imejl adresa ENIC/NARIC centra u sistemu. Pošto je

imejl servis zamenjen maketom, ovde se samo proverava da li je uspešno izvršena operacija, tj. upisano u bazu događaj da je mejl poslat. Testira se oba tipa slanja (regularno i podsetnik)
    - testEnicEmailRequestRead – testiranje čitanja dnevnika slanja imejl poruke ENIC/NARIC centru za jedan ENIC/NARIC zahtev.
- RecognitionReportTest
    - testEnsureVicedeanCoordinator – ako već postoji aktivan rukovodilac studijskog programa na zadatoj ustanovi iz prijave uzima se on kao korisnik, s tim što se menja njegova lozinka na privremenu dok traje testiranje. Ako ne postoji rukovodilac studijskog programa (aktivan), kreira se novi sa primer trajanja mandata do godinu dana unapred. Na kraju testa se vraća stara lozinka koordinatoru, odnosno briše korisnik respektivno
    - testLoginCoordinator – test login koordinatora (rukovodioca studijskog programa)
    - testSendToFaculty – testiranje slanja fakultetu od strane supervizora
    - testAssignToCoordinatora – testiranje preuzimanja izveštaja od strane rukovodioca studijskog programa
    - testReadInitialReport – testiranje čitanja inicijalnog izveštaja za popunjavanje. Tada se popunjavanju inicijalno podaci iz originalne prijave
    - testUpdateRecognitionReport – testiranje popunjavanja izveštaja podacima. Dodaje se jedan dodatni predmet koji je potrebno položiti, tj. doneti uverenje o položenom dodatnom ispitu. Kreira se jedan predmet čije se priznavanje predlaže. Kreira se jedan primer predmeta čije se priznavanje ne predlaže
    - testPrepareSubmissionChangeDataForRecognitionReport – testiranje ako je u međuvremenu došlo do promene ličnih podataka kandidata na primer
    - testUpdateWithFreshReadonlyData – testiranje pokretanja osvežavanja readonly podataka iz prijave u izveštaj koji popunjava koordinator
    - testUploadReport – testiranje čitanja izveštaja o priznavanju (pdf) i upload istog nazad u sistem. Za pdf se proverava da li je generisanje uspešno i da li pdf nije prazan
    - testImportRecognitionReportSuggestions – testiranje import predmeta čije se priznavanje predlaže iz excel fajla. Sa opcijom override na true, tj. briše se postojeća lista predmeta i kreira nova na osnovu podataka iz excel fajla
    - testSendBackToCorrections – testiranje slanja nazad supervizoru na korekcije od strane koordinatora
    - testSendToFacultyFromCorrections – testiranja vraćanja nazad koordinatoru na osnovu korekcije od strane supervizora
    - testUploadReportFromCorrections – testiranje otpremanja izveštaja (pdf) u sistem ponovo, pošto se on briše svaki put kada se vraća fakultetu na dopunu
    - testSendBackToClerk – vraćanje nazad ponovo supervizoru, ali sa ciljem da se verifikuje popunjavanje izveštaja
    - testConfirmRecognitionReportByClerk – testiranje potvrde popunjavanja izveštaja od strane supervizora. Supervizor potvrđuje da je izveštaj u redu.
    - testConfirmRecognitionReportSigned – testiranje da je izveštaj potpisan i dostavljen (otpremljen u sistem takođe)
- AcademicBoardTest
    - testCreateAcademicBoard – priprema stručnog veća. Kreira se novi participant u sistemu, koji se briše na kraju testa, i lista osoba koja će biti u stručnom veću takođe.
    - testDownloadAllDocumentation – testiranje preuzimanja kompletne dokumentacije iz sistema
    - testCreateAcademicBoardForLifecycle – testiranje vezivanja stručnog veća za prijavu za koju će da odlučuje

- testUploadProposalDecision – testiranje otpremanja dokumenta predloga rešenja, tačnije zaključka pošto ovaj test scenario testira slučaj sa dva stručna veća
- testUpdateAcademicBoardForLifecycle – testiranje izmene podataka o relaciji stručnog veća sa prijavom. Postavlja se status odluke na primer na prihvaćen
- testSetSubmissionForwardedStatus – testiranje postavljanja statusa prijave da je odlučeno od strane stručnog veća
- testUploadAcademicBoardDecision – testiranje otpremanja odluke stručnog veća
- testUploadFinalDecision – testiranje otpremanja konačnog rešenje u sistem
- testSetSubmissionDecisionStatus – postavljanje statusa odluke zahteva
- testCreateAcademicBoardForLifecycleSecondAB – testiranje vezivanja novog stručnog veća kao drugo stručnog veće za zahtev koji se testira
- testUploadProposalDecisionSecondAB – testiranje otpremanja dokumenta predloga rešenja za drugo stručno veće
- testUpdateAcademicBoardForLifecycleSecondAB – izmena podataka relacije drugog stručnog veća za zahtev koji se testira. Postavlja se status da je prihvaćena odluka
- testSetSubmissionForwardedStatusSecondAB – postavljanje statusa prijave da je prihvaćen od strane drugog stručnog veća
- testUploadAcademicBoardDecisionSecondAB – testiranje otpremanja odluke drugog stručnog veća
- testUploadFinalDecisionSecondAB – testiranje otpremanja odluke drugog stručnog veća
- testSetSubmissionDecisionStatusSecondAB – testiranje postavljanja statusa prijave da je odlučeno od strane drugog stručnog veća
- testAcademicBaordsRead – testiranje čitanja stručnih veća za zahtev koji se testira
- testSendNotificationToCandidate – testiranje slanja obaveštenja kandidatu da preuzme rešenje
- testCandidateCollected – testiranje da je kandidat preuzeo rešenje

- WorkflowDeletionTest
  - testSubmissionDeletion – brisanje prijave sa svim kreiranim vezanim podacima
  - testEnicNaricDelete – brisanje kreiranih ENIC/NARIC test zahteva
  - testUserDeletion – brisanje kandidata, supervizora, potencijalno rukovodioca studijskog programa ako je kreiran ili vraćanje njegove lozinke na staru lozinku ako je promenjena

Primer metode testSendBackToCorrection:

```
@Test
@Order(6)
void testSendBackToCorrection() {
    var payload = new LifecycleChangeDto(submissionTestShared.getSubmissionId(), "Back to correction -
test workflow");
    var response = client.post()
            .uri(String.format("/submission-procedures/%s/back-to-correction",
submissionTestShared.getSubmissionId()))
            .header("Authorization", "Bearer " +
submissionTestShared.getSupervisorCredentials().tokens().accessToken())
            .body(payload)
            .retrieve()
            .toEntity(SubmissionLifecycleSimpleDto.class);

    assertThat(response.getStatusCode().is2xxSuccessful()).isTrue();
```

```
    assertNotNull(response.getBody());


assertThat(response.getBody().getSubmissionStatus().getCode().equals(SubmissionStatusEnum.UPDATE
.toString())).isTrue();
}
```

Tok posla kada je negativan odgovor od ENIC/NARIC centra

Primer koda za grupisanje međuzavisnih testova u ovom slučaju kada je odgovor od ENIC/NARIC centra negativan

```
@Suite
@SelectClasses({
        UserCreationTest.class,
        SubmissionCreationTest.class,
        SubmissionVerificationTest.class,
        SubmissionCorrectionTest.class,
        ProcedureInitiationTest.class,
        EnicNaricFailedTest.class,
        RecognitionReportForEnicFailedTest.class,
        AcademicBoardForEnicFailedTest.class,
        WorkflowDeletionTest.class
})
public class WorkflowEnicFailedTestSuite {
}
```

Neki testovi su uzeti iz podrazumevanog toka posla kao što je UserCreationTest, SubmissionCreationTest i SubmissionVerificationTest. Izmena je u delu kada se šalje ENIC/NARIC centru pa nadalje se umesto izveštaja sa fakulteta šalje stručnom veću.

EnicNaricFailedTest – slično kao i u podrazumevanom toku posla, vezuje se ENIC/NARIC zahtev sa zahtevom za priznavanje, testira se šalje imejl poruke ENIC/NARIC centru (regularna i podsetnika), izuzetak je što se status ENIC/NARIC zahteva postalja na nije akreditovan umesto akreditovan studijski program.

RecognitionReportForEnicFailedTest – testira se ponašanje šta ako se pokuša proslediti zahtev fakultetu na popunjavanje izveštaja, a ENIC/NARIC zahtev je odbijen (nije akreditovan).

Primer metode koja testira ovaj slučaj

```
@Test
@Order(3)
void testSendToFacultyMustFail() {
    var payload = new LifecycleChangeDto(submissionTestShared.getSubmissionId(), "Sent to faculty - test
suite");
    assertThatThrownBy(() -> client.post()
            .uri(String.format("/faculty-reports/%s/send-to-faculty", submissionTestShared.getSubmissionId()))
            .header("Authorization", "Bearer " +
submissionTestShared.getSupervisorCredentials().tokens().accessToken())
            .body(payload)
            .retrieve()
            .toEntity(ResponseErrorDto.class))
            .isInstanceOf(HttpClientErrorException.BadRequest.class)
            .satisfies(ex -> {
                var httpEx = (HttpClientErrorException.BadRequest) ex;
```

```
            assertThat(httpEx.getStatusCode()).isEqualTo(HttpStatus.BAD_REQUEST);

            ResponseErrorDto body = objectMapper.readValue(
                    httpEx.getResponseBodyAsString(),
                    ResponseErrorDto.class
            );
            assertThat(body.code()).isEqualTo(ExceptionCodes.SUBMISSION_STATUS_NOT_ALLOWED);
        });
}
```

Očekuje se greška od strane backend-a sa status codom SUBMISSION_STATUS_NOT_ALLOWED

Zahtev za odustajanjem od strane kandidata

```
@Suite
@SelectClasses({
        UserCreationTest.class,
        SubmissionCreationTest.class,
        SubmissionVerificationTest.class,
        SubmissionCorrectionTest.class,
        ProcedureInitiationTest.class,
        EnicNaricTest.class,
        RecognitionReportTest.class,
        CandidateCancellationTest.class,
        AcademicBoardForCancellationTest.class,
        WorkflowDeletionTest.class
})
public class WorkflowCandidateCancellationTestSuite {
}
```

Takođe se koriste postojeći testovi iz podrazumevanog toka posla kao što je priprema zahteva i popunjavanje osnovnih podataka o zahtevu od strane kandidata. Izuzetak je deo kada Kandidat podnosi zahtev za odustajanjem, a prijava je trenutno kod fakulteta na popunjavanje izveštaja.

Tok testiranja ovog slučaja:

- CandidateCancellationTest
    - testSendCancellationRequest – testiranje slanja zahteva za odustajanjem od strane kandidata. Kreiranje tokena za odustajanje koji se šalje preko mejla se radi ručno u testu preko servisne metode namenjen za to. Prednost Spring Boot testnog okruženja je što možemo direktno pristupiti servisnim komponentama i pravim i zamenjenim maketama kako bi odradili neki posao koji je inače sakriven enkapsulacijom, a nije moguće na drugi način da se konfiguriše testiranje da liči na black box testiranje.
    - testCancellationConfirmedByCandidate – testiranje potvrde od strane kandidata
    - testConfirmedByClerk – testiranje potvrde od strane supervizora
- AcademicBoardForCancellationTest
    - Testira se slučaj sa jednim stručnim većem. Šalju se odgovarajući statusi koji se odnose na zahtev za odustajanjem od proces prijave od strane kandidata

Testiranje slučaja dodatnih predmeta za polaganje

Primer test suite koda za konfiguraciju ovog toga posla vezano za dostavljanje uverenja o položenim ispitima od strane kandidata u toku godine

```
@Suite
@SelectClasses({
	UserCreationTest.class,
	SubmissionCreationTest.class,
	SubmissionVerificationTest.class,
	SubmissionCorrectionTest.class,
	ProcedureInitiationTest.class,
	EnicNaricTest.class,
	RecognitionReportTest.class,
	AcademicBoardConditionalTest.class,
	WorkflowDeletionTest.class
})
public class WorkflowConditionalExamsTestSuite {
}
```

Opet je proces u početku isti kao i u podrazumevanom slučaju toka posla prijave, tako da se koriste postojeće test klase za ovaj segment. Razlika je u testiranju u fazi stručnog veća

Klasa AcademicBoardConditionalTest opisuje skup testova za prosleđivanje odgovarajućih statusa stručnom veću vezano za uslovno priznavanje.

Dodatno se pozivaju test metode ove klase

- testCandidateUploadConditionalDocument – testiranje otpremanja dokumenata vezano za uverenje o položenim ispitima od strane supervizora. Supervizor vrši otpremanje na prijem dokumenata od strane kandidata u štampanom obliku na primer.
- testSetConditionalExamsRecievied – testiranje postavljanja statusa da su dokumenti dostavljeni vezano za uverenaj od položenim dodatnim ispitima

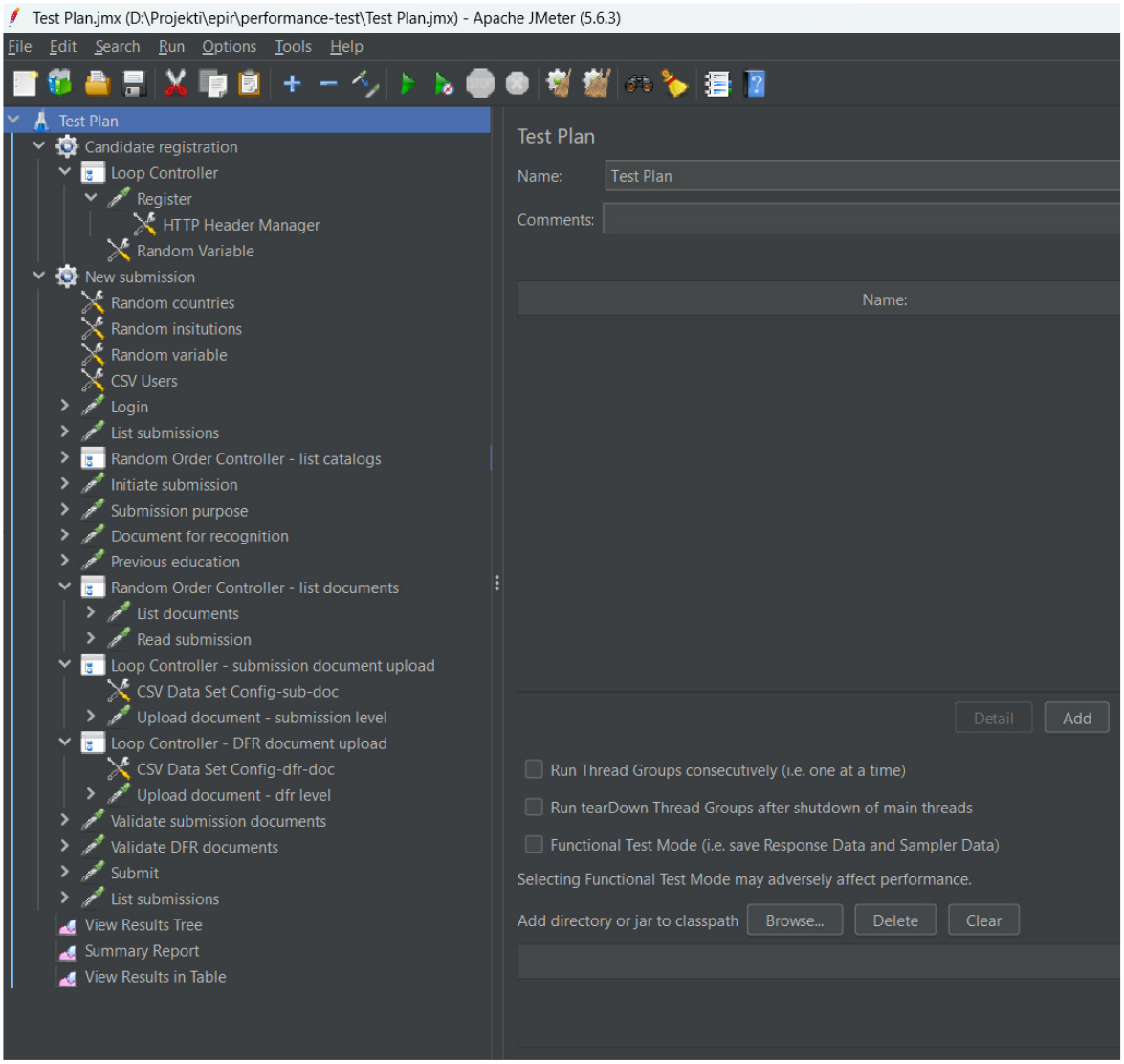
Testiranje performansi

Testiranje performansi je rađeno u alatu Apache JMeter (https://jmeter.apache.org/). Kreiran je primer toka posla od registracije i login korisnika, kreiranja prijave, verifikacije službenika. Cilj testiranja je merenje merformansi sistema pod opterećenjem kao i procenat grešaka koji nastaju zbog optimističkog zaključavanja određenih resursa kao što je inkrementer broja predmeta (redni broj kao deo delovodnog broja)

Greške nastaju kod određivanja broja predmeta zato što se vrši inkrement rednog broja koji je deo delovodnog broja zahteva. Da bi se osigurala jedinstvenost delovodnog broja potrebno je vršiti zaključavanje resursa koji broji redne brojeve. Da bi se osigurala ordinalnost takvih brojeva moguće je pesimističko zaključavanje i optimističko zaključavanje. Pesimističko zaključavanje bi dovelo do velikog broja zaključavanja resursa u režimu velike opterećenosti pristupu backend-u ka istim endpoint-ovima koji kalkulišu delovodni broj. Ovaj način zaključavanja bi se manifestovao u testu performansi pod velikim opterećenjem u vidu velikog broja grešaka koji su tipa DeadLockException gde se jedan zahtev za kreiranjem delovodnog broja određuje kao žrtva i poništava. Zbog velikog broja zahteva u trenutku, većina njih bih bila određena za žrtve, a samo mali broj bi prošao. Iako u realnom korišćenju ove aplikacije ovakve situacije neće dolaziti zbog manjeg broja obrada podnetih prijava, ipak je odlučeno da ne bude pesimističko zaključavanje već optimističko. Kod optimističkog zaključavanja ne dolazi do prethodno zaključavanja resursa nego se pokušavaju odraditi paralelno. Ako jedan od njih dođe u situaciju da treba da izmeni ordinal delovodnog broja na sledeću vrednost, a već je to neko drugi odradio vratiće grešku slično kao kod pesimističkog zaključavanja. Prednost optimističkog zaključavanja je veća propusna moć. Takođe dolazi do grešaka, ali one su se pokazale da su u fazi testiranja velikog opterećenja do maksimalno 25% od ukupnog broja zahteva za kreiranjem delovodnog broja u trenutku.

Parametri testiranja:

- Broj paralelnih korisnika: 100
- Ramp-up period, period zagrevanja, tj. postepeno uveđenje 100 paralelnih korisnika a ne sve odjednom: 10 sekundi
- Ukupan broj korisnika: 4800
- Unošenje podataka o državi stanovanja, državljanstvu, studijskom program i slično su randomizovani kako bi se pospešila diversifikacija podataka da ne bi došlo do izvršavanja uvek istog plana sql upita nad bazom podataka



Na slici gore je prikazana struktura testa performansi u Apache Jmeter. Kreće se od registracije, zatim se pokreće nit New Submission koja simulira novu prijavu za priznavanjem od strane kandidata. Kreirani su bazični csv fajlovi koji sadrže kataloške podatke i koji se radnom uzimaju prilikom testiranja performansi: Random countries, Random institutions, Random variable, CSV Users.

Takođe koristi se i Random Order Controller kako bi se simulirala dinamičnost u redosledu poziva određenih endpoint-ova da ne bi svaki korisnik išao isitim redosledom i time podstakao sql server manager-a da koristi keširane execution planove i podatke.

Primer sumarnih rezultata testiranja performansi

	Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughpu	Received k	Sent KB/se	Avg. Bytes
2	Login	1702	2342	107	11055	1531.84	0.06%	2.31697	7.89	0.77	3487.1
3	List submis	3310	5440	23	16299	3652.36	0.24%	4.50967	16.5	2.49	3746.9
4	List purpos	1699	1263	12	11021	1636.8	0.00%	2.35258	11.65	1.26	5073
5	List study p	1700	1425	44	12823	1709.71	0.00%	2.32333	62.81	1.42	27683.7
6	List institut	1700	1262	16	10357	1592.49	0.06%	2.32131	13.03	1.25	5746.2
7	List countr	1699	1552	68	12196	1961.42	0.00%	2.35295	109.29	1.24	47561
8	List metad	1699	1731	92	13470	2070.78	0.00%	2.35052	973.87	1.26	424265
9	Initiate sub	1699	482	29	4358	474.74	0.00%	2.41337	15.14	3.31	6422.4
10	Submissior	1699	428	37	11257	486.68	0.00%	2.41151	20.46	1.56	8687.1
11	Document	1699	408	41	4897	378.74	0.00%	2.41011	24.3	4.37	10323.5
12	Previous e	1699	404	45	3535	356.28	0.00%	2.41068	25.49	3.41	10828.2
13	Read subm	1699	415	51	3898	339.75	0.00%	2.4097	25.48	1.17	10828.2
14	List docum	1699	382	39	3688	307.29	0.00%	2.40913	15.8	1.19	6715.8
15	Upload do	10191	849	63	10265	491.61	0.02%	14.36022	9.62	8594.13	686
16	Upload do	6763	1246	63	12721	1104.03	0.49%	9.5791	6.51	5731.71	695.6
17	Validate su	1664	2021	91	14155	1673.78	1.02%	2.36598	1.2	1.25	519.5
18	Validate D	1647	2644	74	13352	2096.33	1.09%	2.34354	1.19	1.28	521.1
19	Submit	1629	5863	37	17403	3262.02	9.39%	2.3224	35.51	1.19	15657
20	TOTAL	45597	1599	12	17403	2196.84	0.51%	62.07195	1350.36	13839.03	22276.9
21											