

*Zimski semestar 2024*

*PREDMET: SE201 – Projektni Zadatak*

Projektni Zadatak:Sistem za Kviz

***Student:*** *Nikola Ilic 5633*

***Profesor:*** *Dragan Domazet*

***Mentor:****Sara Nikolić*

**Odabrati agilni metod - Scrum**

Scrum je agilan okvir za timsku saradnju, često se koristi u razvoju softvera i drugim industrijama. Scrum preporučuje timovima da razbiju rad u ciljeve koji treba da budu završeni u vremenski ograničenim iteracijama, nazvanim sprintovi. Svaki sprint traje najviše mesec dana i obično traje dve nedelje.

**Tim (Scrum Team):**

- Članovi tima uključuju programere (Marko, Nikola, Janko), web dizajnere (Ana, Jovana, Danila), agente za rezervacije (Aleksandra, Iva) i marketing stručnjake (Danilo, Nataša).

**Product Owner:**

- Stefan Markovic, iskusni profesionalac u oblasti edukacije, odgovoran za vođenje Product Backlog-a. Fokusira se na poboljšanja koja će unaprediti korisničko iskustvo, kao što su intuitivan interfejs, kvizovi prilagođeni korisnicima i efikasna administracija pitanja.

**Scrum Master:**

- Nikolina Stefanovic, stručnjak za Scrum i iskusna u rešavanju prepreka, podržava tim u primeni Scrum principa. Razume specifičnosti edukativnog sektora i pomaže u optimizaciji procesa.

**Product Backlog:**

- Kreiranje i administracija pitanja

- Korisnički interfejs za studente i profesore

- Funkcionalnost prijave i odjave korisnika

- Praćenje rezultata kviza

- Mogućnost filtriranja rezultata po odgovorima

**Sprint:**

- Svaki Sprint traje 2 do 4 nedelje.

**Sprint Planning:**

- Tim se okuplja sa Product Owner-om da odabere prioritete za naredni Sprint. Fokusiraju se na poboljšanja u administraciji pitanja i korisničkom interfejsu.

**Daily Scrum (Daily Standup):**

- Tim se svakodnevno sastaje kako bi razmenio informacije. Programeri rade na implementaciji novih funkcionalnosti, dizajneri na poboljšanju korisničkog interfejsa, a agenti za rezervacije na praćenju rezultata kviza.

**Sprint Review:**

- Nakon završetka Sprint-a, tim prikazuje nove funkcionalnosti, a Product Owner izražava zadovoljstvo i predlaže eventualna unapređenja.

**Sprint Retrospective:**

- Tim se sastaje da analizira šta je dobro funkcionisalo i šta može biti poboljšano. Odlučuju implementirati nove funkcionalnosti i poboljšanja u narednom Sprint-u.

**Zašto baš Scrum?**

- Scrum se odlikuje transparentnošću, adaptivnošću i redovnim feedback-om, što ga čini prikladnim za projekte u kojima su zahtevi podložni promenama. Takođe, Scrum omogućava bolju kontrolu nad projektom i smanjuje rizik od neuspeha, što je posebno važno u dinamičnom okruženju edukacije.

**Murov obrazac za Viziju Proizvoda:**

**Ciljevi i Vrednosti:**

Koje su ključne vrednosti koje želimo pružiti korisnicima (studentima i profesorima)?

Koji su dugoročni i kratkoročni ciljevi aplikacije kviz menadžmenta?

**Korisnici:**

Ko su naši glavni korisnici?

Kakve funkcionalnosti očekuju od aplikacije kviz menadžmenta? (npr. lakša administracija kvizova, intuitivan korisnički interfejs)

**Inovacije i Tehnologija:**

Kako možemo koristiti inovacije kako bismo se izdvojili od konkurencije ?

Koje tehnologije možemo integrisati radi poboljšanja korisničkog iskustva (npr. AI za personalizaciju kvizova)?

**Ponuda proizvoda i Usluge:**

Koje vrste kvizova želimo omogućiti (obrazovni, zabavni, takmičarski)?

Kako možemo diferencirati našu aplikaciju od drugih sličnih?

**Branding i Marketing:**

Kakav imidž želimo stvoriti za našu aplikaciju?

Koje kanale marketinga ćemo koristiti kako bismo privukli studente i profesore?

**Mere Uspeha:**

Kako ćemo meriti uspeh aplikacije kviz menadžmenta?

Koje ključne metrike želimo pratiti (npr. broj korisnika, ocene zadovoljstva korisnika, efikasnost administracije kvizova)?

**Vizija Proizvoda:**

ZA studente i profesore KOJI žele da poboljšaju iskustvo učenja i organizaciju kvizova, kreirana je QuizMaster APLIKACIJA KOJA transformiše proces kvizova, čineći ga lakšim, interaktivnim i efikasnim. QuizMaster olakšava studentima pregled dostupnih kvizova, omogućava praćenje novih testova i interaktivno učestvovanje u kvizovima sa brzim rezultatima.

ZA razliku od tradicionalnih metoda organizacije kvizova, NAŠ PROIZVOD unapređuje iskustvo korisnika, povećava efikasnost učenja i omogućava bolji pregled kvizova.

QuizMaster APLIKACIJA je neophodan alat za studente i profesore, pružajući moderno, intuitivno i efikasno iskustvo u učenju i organizaciji kvizova.

**Spisak Funkcionalnih Zahteva:**

**Prijavljivanje i Registracija:**

Korisnici mogu odabrati da se prijave kao student ili profesor na početnoj formi.

Studenti se prijavljuju na svoj nalog popunjavanjem forme sa imenom, prezimenom, JMBG-om, adresom, i brojem telefona.

Profesori se prijavljuju na svoj nalog koristeći odgovarajuće dugme.

**Zapocinjanje Kviz-a (Student):**

Nakon prijave, studentu se prikazuje forma sa pravilima za kviz i opcijom "Zapocni Kviz".

Student započinje kviz koji se sastoji od 10 pitanja.

Pitanja su opciona, i student mora odabrati jedan od tri odgovora po pitanju.

Nakon završetka kviza, aplikacija se zatvara.

**Administracija Pitanja (Profesor):**

Profesori se prijavljuju na svoj nalog koristeći odgovarajuće dugme.

Dugme "Novo Pitanje" otvara formu za dodavanje novog pitanja sa opcijama 1, 2 i 3, i odgovorom koji mora biti jedan od tri ponuđena.

Dugme "Azuriraj Pitanje" omogućava profesoru da unese ID pitanja i izmeni tekst, opcije i odgovor.

Dugme "Sva Pitanja" prikazuje formu sa svim pitanjima u bazi podataka.

Dugme "Izbrisi Pitanje" briše pitanje na osnovu unetog ID-ja.

**Prikaz Rezultata (Profesor):**

Dugme "Student Rezultati" prikazuje formu sa rezultatima studenata koji su radili kviz.

Svaki odgovor je prikazan kao nov red u tabeli.

Omogućeno je filtriranje rezultata po odgovoru kako bi profesoru bilo lakše pregledati studente sa tačnim odgovorima.

**Odjava (Profesor):**

Dugme "Logout" omogućava profesoru da se izloguje sa svog naloga.

**Spisak Nefunkcionalnih Zahteva:**

**Performanse:**

Sistem treba obezbediti brz odziv prilikom interakcije sa formama za prijavu, kreiranje pitanja i pregled rezultata.

Minimalna vremena čekanja tokom započinjanja kviza i pretrage baze podataka.

**Dostupnost:**

Platforma treba obezbediti visok nivo dostupnosti kako bi korisnici mogli pristupiti aplikaciji u većini vremena.

Periodi nedostupnosti zbog održavanja treba biti minimizirani i najavljeni unapred.

**Sigurnost podataka:**

Sistem treba garantovati visok nivo sigurnosti i enkripcije podataka korisnika, uključujući informacije o studentima i profesorima.

Poseban fokus na ličnim podacima, JMBG-ovima i drugim osetljivim informacijama.

**Skalabilnost:**

Aplikacija treba biti skalabilna kako bi mogla rastuće brojeve studenata i profesora bez gubitka performansi.

Sposobnost rasta u broju pitanja, odgovora i rezultata kviza.

**Korisničko iskustvo:**

Platforma treba obezbediti intuitivan korisnički interfejs kako bi olakšala studentima i profesorima korišćenje svih funkcionalnosti.

Jednostavan pristup formama za prijavu, kreiranje i administraciju pitanja.

**Kompatibilnost:**

Aplikacija treba biti kompatibilna sa različitim pregledačima (Chrome, Firefox, Safari) i uređajima (računar, tablet, mobilni telefon).

Osigurati dosledno iskustvo korisnika bez obzira na uređaj ili pregledač koji koriste.

**Korisničke priče:**

**Studenti:**

**Jovana (20):**

Kao student, želim brzo i lako se prijaviti na sistem kako bih mogla pristupiti kvizu.

Kao student, želim imati jasnija pravila kviza pre nego što započnem sa odgovaranjem na pitanja.

**Stefan (22):**

Kao student, želim imati mogućnost popunjavanja podataka u formi jednostavnim koracima kako bih brzo pristupio kvizu.

Kao student, želim videti raznolika pitanja tokom kviza kako bih testirao svoje znanje.

**Tamara (25):**

Kao student, želim imati opciju vraćanja na prethodna pitanja tokom kviza kako bih proverila svoje odgovore.

Kao student, želim primati obaveštenje o rezultatima kviza nakon završetka.

**Milan (21):**

Kao student, želim imati opciju pauziranja kviza kako bih privremeno napustio aplikaciju i vratio se kasnije.

Kao student, želim pregledati statistiku svojih prethodnih kvizova kako bih pratio svoj napredak.

**Teodora (23):**

Kao student, želim dobiti preporuke za dodatne materijale ili resurse koji će mi pomoći u pripremi za kviz.

Kao student, želim imati mogućnost deljenja rezultata kviza na društvenim mrežama.

**Profesori:**

**Ana (Profesor, 35):**

Kao profesor, želim dodavati nova pitanja sa lakoćom kako bih ažurirao bazu pitanja.

Kao profesor, želim imati mogućnost precizne izmene postojećih pitanja kako bih održavao kvalitet kviza.

**Nikola (Asistent, 28):**

Kao profesor, želim lako pristupiti svim pitanjima u bazi kako bih ih pregledao i analizirao.

Kao profesor, želim brisati zastarela ili nepotrebna pitanja iz baze podataka.

**Vladimir (Profesor, 40):**

Kao profesor, želim imati pregled statistike ukupnih rezultata kviza kako bih procenio ukupnu uspešnost studenata.

Kao profesor, želim postavljati vremenska ograničenja za određene kvizove kako bih testirao brzinu rešavanja.

**Tara (Asistent, 27):**

Kao asistent, želim imati mogućnost dodavanja dodatnih informacija uz pitanja kako bih pružio dodatna objašnjenja studentima.

Kao asistent, želim pregledati rezultate kviza po pitanju kako bih identifikovao specifična pitanja koja su izazvala teškoće studentima.

**Milica (Profesor, 30):**

Kao profesor, želim imati funkciju praćenja rezultata studenata kako bih pravilno ocenjivao njihove kvizove.

Kao profesor, želim filtrirati rezultate po određenim kriterijumima kako bih lakše identifikovao uspešne studente.

**Persone:**

**Persona 1: Student - Marko Jovanović**

Marko Jovanović, 21 godina, je student koji redovno koristi QuizMaster za unapređenje svojih znanja. Studira informatiku na Univerzitetu u Beogradu. Marko želi efikasan način za prijavu na kvizove, brz pristup različitim pitanjima i mogućnost pracenja svojih rezultata. Njegove potrebe uključuju intuitivan interfejs, opciju za pauziranje kviza i pristup statistici njegovih prethodnih performansi. Marko koristi aplikaciju kako bi se efikasno pripremio za ispite i unapredio svoje znanje.

**Persona 2: Profesor - Ana Petrović**

Ana Petrović, 35 godina, je profesorka koja koristi QuizMaster za kreiranje i praćenje kvizova. Predaje na Fakultetu Informatike. Ana želi jednostavan način za dodavanje novih pitanja, praćenje rezultata studenata i lako upravljanje pitanjima i odgovorima. Njene potrebe uključuju brz pristup svim funkcijama, mogućnost ažuriranja pitanja i efikasno pregledavanje rezultata kvizova. Ana koristi aplikaciju kako bi unapredila nastavni proces i pružila interaktivno iskustvo svojim studentima.

**Persona 3: Administrator - Nikola Radulović**

Nikola Radulović, 40 godina, je sistem administrator na Univerzitetu i odgovoran je za održavanje QuizMaster aplikacije. Nikola želi stabilan sistem, brzo rešavanje eventualnih tehničkih problema i mogućnost upravljanja korisničkim nalozima. Njegove potrebe uključuju redovno ažuriranje sistema, bezbednost podataka studenata i efikasno održavanje celokupne aplikacije. Nikola koristi aplikaciju kako bi osigurao besprekorno funkcionisanje i visok nivo sigurnosti za sve korisnike.

**Scenario za Persone:**

**Scenario 1: Student - Marko Jovanović**

Marko Jovanović, student informatike, želi da koristi QuizMaster Aplikaciju kako bi se pripremio za ispite.

Marko otvara aplikaciju i prijavljuje se kao student.

Ulazi u kviz, gde pažljivo čita pitanja i odgovara na svako.

U toku kviza, ima opciju za pauziranje ako mu je potrebno više vremena za razmišljanje.

Kada završi kviz, prima trenutne rezultate, uključujući tačne i netačne odgovore.

Pregledava analizu svog performansa, uključujući oblasti koje treba dodatno proučiti.

Može ponovo raditi kviz kako bi pratio svoj napredak.

**Scenario 2: Profesor - Ana Petrović**

Ana Petrović, profesorka informatike, koristi QuizMaster Aplikaciju za pripremu i vođenje ispita.

Ana se prijavljuje kao profesor i pristupa svom nalogu.

Kreira novi kviz dodajući pitanja iz postojeće baze ili unosom novih pitanja.

Prilagođava postavke kviza, uključujući vreme trajanja, ocenjivanje i pravila.

Dodeljuje kviz određenim kursevima ili grupama studenata.

Prati napredak studenata tokom kviza i prima trenutne rezultate.

Pregledava analizu rezultata i identifikuje oblasti koje možda zahtevaju dodatno objašnjenje.

Postavlja opcionalna pitanja za dodatne diskusije ili dodatne zadatke za poboljšanje razumevanja gradiva.

**Scenario 3: Administrator - Nikola Radulović**

Nikola Radulović, sistem administrator na univerzitetu, koristi QuizMaster Aplikaciju za održavanje i nadzor sistema.

Nikola pristupa administratorskom panelu aplikacije.

Prati performanse sistema i identifikuje eventualne tehničke probleme.

Obezbeđuje redovna ažuriranja sistema kako bi održavao visok nivo performansi.

Upravlja korisničkim nalozima, dodajući nove profesore i studente prema potrebi.

Implementira sigurnosne mere kako bi osigurao privatnost podataka i sprečio neovlašćeni pristup.

Rešava tehničke izazove koji se pojave tokom korišćenja aplikacije.

**Mikroservisna Arhitektura:**

Mikroservisna arhitektura bi bila efikasan izbor za Kviz Menadžment Aplikaciju. Svaki deo sistema može biti zaseban servis s jasno definisanom funkcionalnošću.

**Opis Mikroservisne Arhitekture:**

**Autonomni Servisi:**

Svaki servis je posvećen specifičnoj funkcionalnosti, kao što su upravljanje korisnicima, vođenje kvizova, analiza rezultata, ili upravljanje profesorskim nalozima.

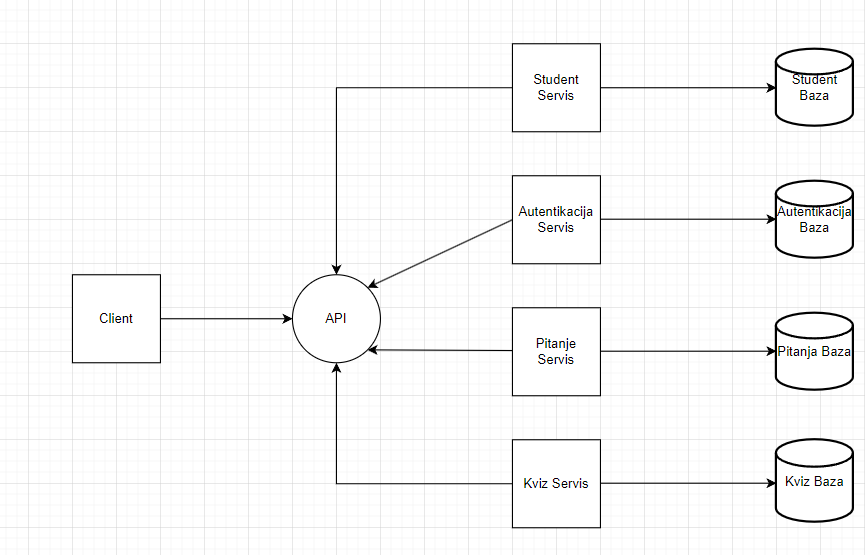
Autonomnost omogućava brži razvoj i održavanje svakog servisa.

**Komunikacija Preko Mreže:**

Servisi komuniciraju preko mreže, što omogućava nezavisno skaliranje i distribuiranu arhitekturu.

**Jednostavno Upravljanje:**

Svaki servis može biti nezavisno skaliran prema potrebama, omogućavajući bolje upravljanje resursima.

****

**Prednosti:**

Fleksibilnost i Skalabilnost: Svaki servis može biti skaliran nezavisno prema potrebama.

Lakše Upravljanje: Lakše praćenje i upravljanje manjim servisima.

Nezavisnost Razvoja: Svaki servis može koristiti različite tehnologije, omogućavajući brži razvoj.

**Mane:**

Kompleksnost Upravljanja: Povećava se kompleksnost u upravljanju infrastrukturom i komunikacijom između servisa.

Potrebna Dodatna Oprema: Zahteva sofisticirane alate za nadzor, upravljanje i praćenje rada svih servisa.

**Komponente Sistema koje mogu biti ponovo upotrebljene:**

⦁ **ClientHandler i Server** (Server-side komunikacija): Ove komponente su odgovorne za obradu zahteva od klijenta i upravljanje serverom putem soketa. Funkcionalnosti poput dodavanja, ažuriranja, i brisanja pitanja, prijave studenata i profesora, kao i prikazivanje podataka o studentima mogu se prilagoditi i ponovno koristiti u sličnim sistemima.

⦁ **PitanjaDAO, ProfesorDAO, RezultatDAO, StudentDAO** (Database Access Objects): Klase koje omogućuju interakciju sa bazom podataka, proveru i unos podataka. Ove komponente mogu biti ponovo upotrebljene u drugim sistemima koji koriste sličan pristup bazi podataka za rad sa pitanjima, profesorima, rezultatima i studentima.

**⦁ ProfesorDashboard, ProfesorLogin, StudentInfo, StudentLogin** (JavaFX interfejsi): Klase koje predstavljaju implementaciju korisničkih interfejsa za profesorski panel, prijavu profesora, informacije o studentima i prijavu studenata. Ove komponente mogu biti prilagođene i ponovno upotrebljene u drugim JavaFX aplikacijama koje imaju slične zahteve za korisničkim interfejsima.

⦁ **DatabaseConnector** (Povezivanje sa bazom podataka): Klasa koja omogućuje povezivanje sa MySQL bazom podataka. Ova komponenta može se ponovo upotrebiti u drugim sistemima koji koriste MySQL bazu podataka i zahtevaju sličan mehanizam povezivanja.

**Delovi sistema koji mogu biti primer računarstva u oblaku:**

**⦁ Baze podataka (IaaS, PaaS)**: Baze podataka sistema mogu se premestiti u računarstvo u oblaku, bilo kao Infrastructure as a Service (IaaS), koristeći virtualne mašine sa instaliranim sistemima za upravljanje bazama podataka, ili kao Platform as a Service (PaaS), koristeći upravljane baze podataka koje pružaju cloud provajderi.

⦁ **Server (IaaS)**: Server koji otvara serverski soket na određenom portu može se izdvojiti kao Infrastructure as a Service (IaaS). Ovo bi omogućilo skalabilnost i bolje upravljanje resursima putem cloud infrastructure.

⦁ **JavaFX Aplikacija (SaaS)**: Klasa Kviz koja predstavlja implementaciju korisničkog interfejsa kviza može se izdvojiti kao Software as a Service (SaaS). Pružanje ove aplikacije kao usluge putem web pregledača omogućava lak pristup korisnicima.

**Delovi sistema koji mogu biti zamenjeni već postojećim rešenjima:**

⦁ **Server (Serverless arhitektura)**: Umesto tradicionalnog serverskog modela, sistem može razmotriti prelazak na serverless arhitekturu gde se resursi izdvajaju dinamički prema potrebama, što može smanjiti operativne troškove.

⦁ **Database Connector (Cloud-native baza podataka)**: Umesto direktnog povezivanja sa MySQL bazom podataka, sistem može razmotriti upotrebu cloud-native baze podataka kao što su Amazon RDS, Google Cloud SQL ili Azure SQL Database.

**Delovi sistema za koje računarstvo u oblaku ne bi bilo najbolje rešenje:**

⦁ **JavaFX Aplikacija (Desktop aplikacije):** Ako aplikacija ima specifične zahteve koji se najbolje izražavaju u desktop okruženju, prelazak na računarstvo u oblaku (posebno SaaS) može dovesti do gubitka performansi i složenosti.

⦁ **ClientHandler (Jednostavna mrežna komunikacija)**: Ako mrežna komunikacija između klijenta i servera nije kompleksna i ne zahteva skalabilnost, prelazak na cloud servise može biti nepotreban.

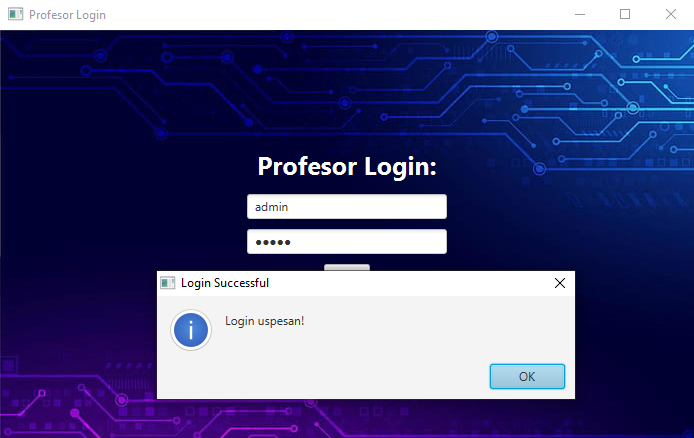
⦁ **Server (Mali projekti):** U manjim projektima gde nema potrebe za velikim brojem istovremenih konekcija, upotreba cloud servera može biti prekomerna, a lokalni server bi bio dovoljan.

**SQL Injection napad:**

SQL injection napad predstavlja tehniku napada na sistem putem manipulacije SQL upita koje izvršava aplikacija. Ova vrsta napada se javlja kada se korisnički unos ne san ili koriste parametrizovani upiti, već se direktno uključuje u SQL upit. Kroz ovu tehniku, napadač može ubaciti zlonamerni SQL kod koji će se izvršiti na strani baze podataka.

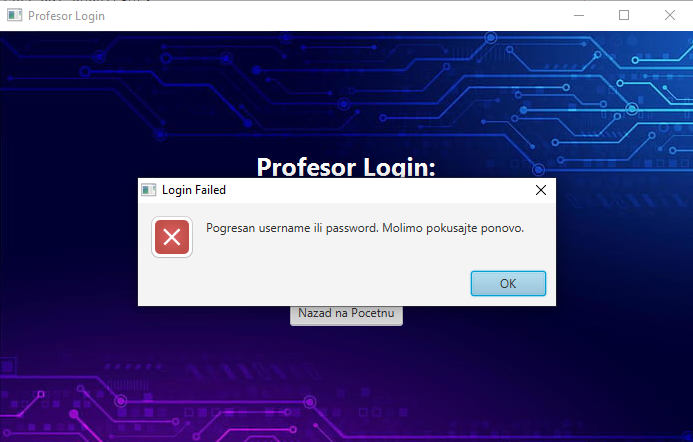
**Postupak:**

⦁ Ovako izgleda uspesno logovanje na profesor login:



*Slika 1.Normalno logovanje na Profesor login*

⦁ Neuspesno logovanje na profesor login:



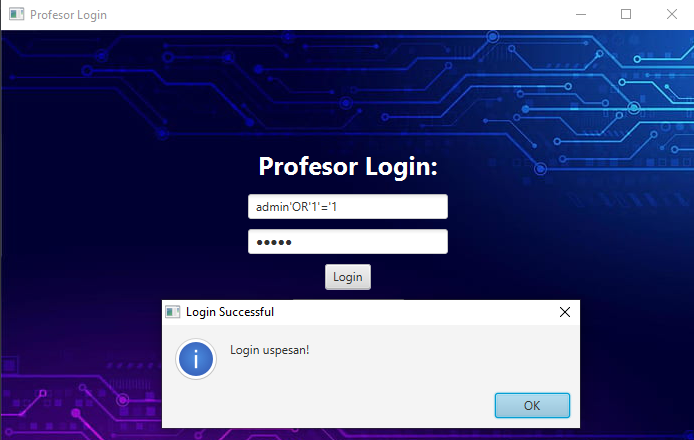
*Slika 2.Neuspesno Logovanje na Profesor login*

Primer upita(u Javi) koji prilikom logovanja se ispisuje u bazi podataka:

**String query = "SELECT \* FROM profesor WHERE Username ='" + username + "' AND Password = '" + password + "'";**

**OR PAYLOAD:**

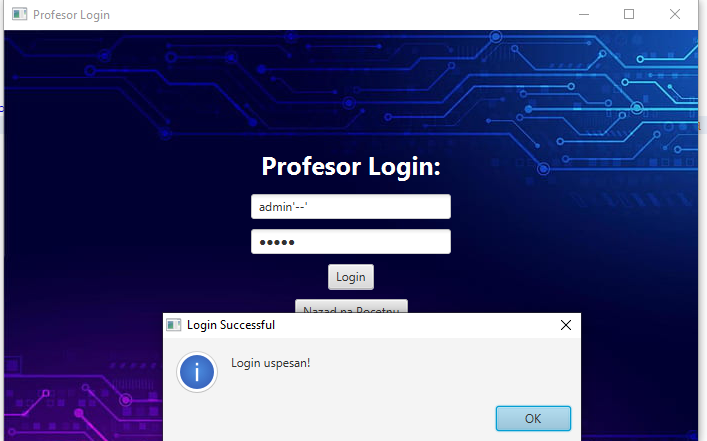
Ovom upitu dodajemo OR i '1' = '1'. Ako je korisničko ime 'admin', kako smo naveli, tada će, budući da AND ima prioritet u odnosu na OR, proveravati ispravnost lozinke. Iako bi rezultat trebao biti netačan, zbog dodavanja OR i '1' = '1' upit i dalje prolazi, a zatim će se proveravati da li se korisničko ime podudara i da li je 1 jednako 1, što će biti tačno, te će lozinka biti zanemarena.



*Slika 3.OR PAYLOAD*

**Comment PAYLOAD(--):**

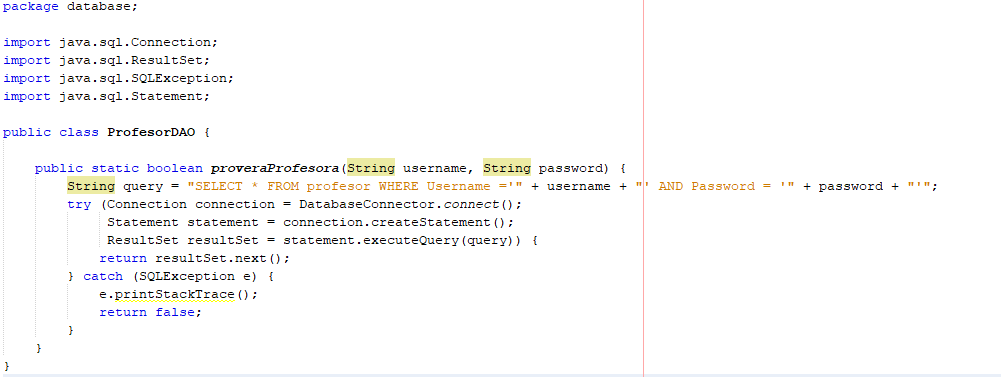
U ovom SQL upitu dodajemo dve crte koje komentarišu preostali deo koda. Ako je naše korisničko ime 'admin', nakon toga SQL upit se neće izvršavati, već će nas samo prijaviti, zahvaljujući prisustvu komentara.



*Slika 4.Comment PAYLOAD(--)*

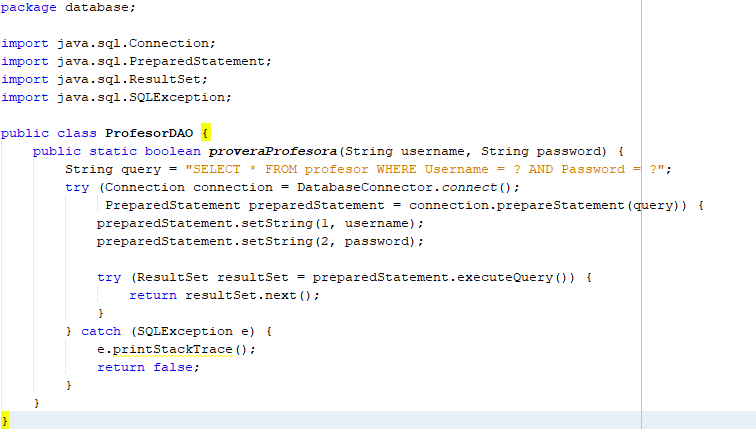
Kako bi smo resili ovaj problem:

Trenutno imamo ovaj kod u Javi:



*Slika 5.Neispravan kod*

Ovaj kod je los zato sto koristi Statement interfejs koji je podlozan SQL injection napadima. Kao sto se vidi na kodu u upitu se dodaje username i password. Kako bi smo resili ovaj problem koristicemo PreparedStatement interfejs koji izgleda ovako.:



*Slika 6.Ispravan kod*

**Atributi kvaliteta sistema**

**⦁ Pouzdanost:**

**Korišćenje Sockets i Servera:** Implementacija servera i korišćenje sockets za komunikaciju pružaju pouzdanost u vezi sa povezivanjem i komunikacijom između klijenata i servera.

**Transakcije i Baze podataka:** Aplikacija koristi baze podataka za čuvanje podataka o studentima, pitanjima, profesorima i odgovorima a implementacija transakcija omogućava doslednost i pouzdanost podataka.

**⦁ Fleksibilnost i Proširivost:**

**JavaFX Scene API:** Korišćenje JavaFX Scene API-a omogućava kreiranje modularnih korisničkih interfejsa. Nova funkcionalnost može se dodavati putem novih scena i prozora bez potrebe za velikim izmenama u postojećem kodu.

**DAO Klase:** Upotreba Data Access Object (DAO) klasa za interakciju sa bazom podataka pruža jasnu apstrakciju podataka i omogućava lakše dodavanje novih entiteta ili proširenje funkcionalnosti.

**⦁ Performanse:**

**ExecutorService:** Korišćenje ExecutorService-a za planirano osvežavanje podataka (kviz pitanja, rezultata) doprinosi efikasnosti i performansama sistema.

**Optimizovane SQL Upiti:** Ako se koriste SQL upiti, njihova optimizacija može poboljšati performanse prilikom pristupa bazi podataka.

**⦁ Održivost:**

**Čitljivost i Struktura Koda:** Održivost sistema olakšana je dobro organizovanim i čitljivim kodom. Razdvajanje odgovornosti u različite klase (npr. DAO klase) pomaže održavanju i izmenama.

**Test Paketi:** Postojanje test paketa, pruža mehanizme za proveru ispravnosti i održivost sistema kroz testiranje.

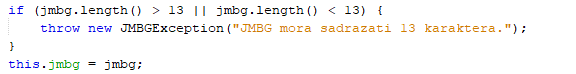
**⦁ Sigurnost:**

**Validacija Unosa:** Implementacija validacije unosa, posebno prilikom dodavanja novih pitanja ili informacija o studentima, može doprineti sigurnosti sistema sprečavajući potencijalne greške ili zloupotrebu.

**Koncepti pouzdanog programiranja**

* **Validacija input-a**

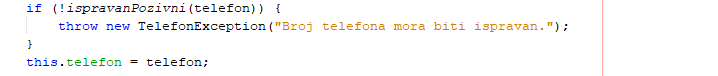
Validacija input-a je proces testiranja unosa primljenog od strane aplikacije radi usklađenosti s standardom definisanim unutar same aplikacije. Može biti jednostavna ili kompleksna poput korištcenja regularnih izraza ili poslovne logike za proveru unosa.U ovom primeru mozemo videti proveru jmbg-a za studenta.

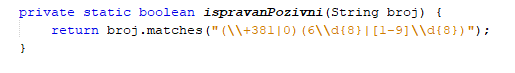


*Slika 7.Validacija Input-a*

* **Šabloni (Pattern)**

Šabloni su predefinisane strukture ili obrasci koje možemo koristiti za verifikaciju i manipulaciju tekstualnih podataka. U ovom primeru mozemo videti sablon za proveru adrese:

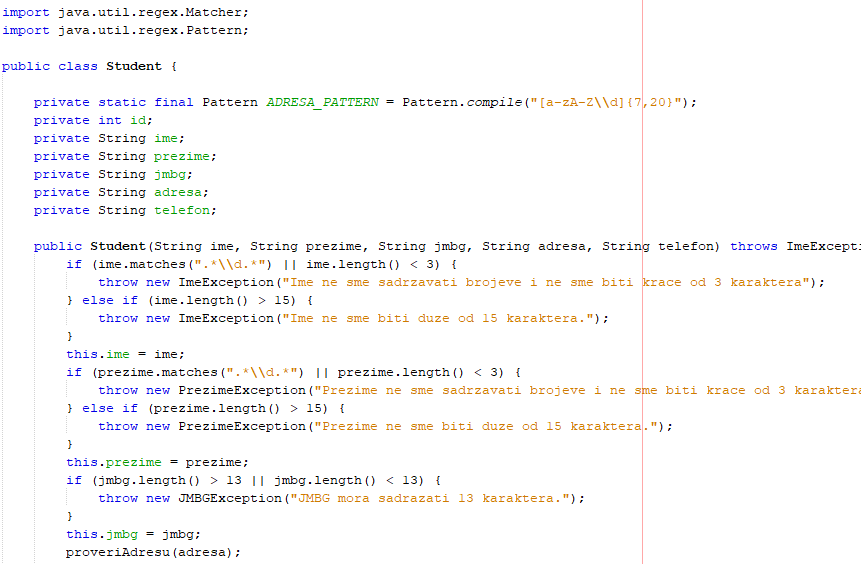


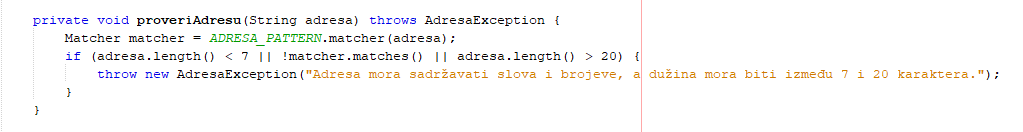


*Slika 9. Sabloni(Pattern)*

* **Regularni izrazi (RegEx)**

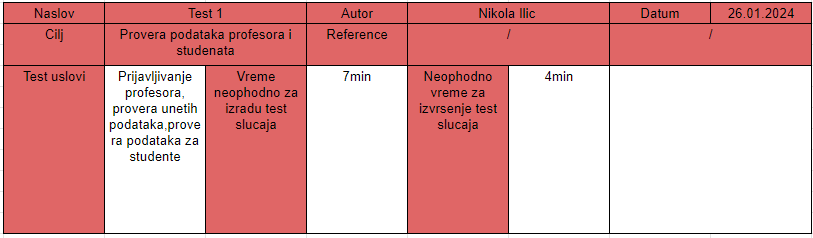
Regularni izrazi (RegEx) je moćan alat za pretragu, manipulaciju i analizu teksta, koji se često koristi u programiranju i obradi podataka.To su nizovi znakova koji opisuju šablone traženja u tekstualnim podacima.U ovom primeru mozemo videti koriscenje regularnog izraza za proveru adrese:

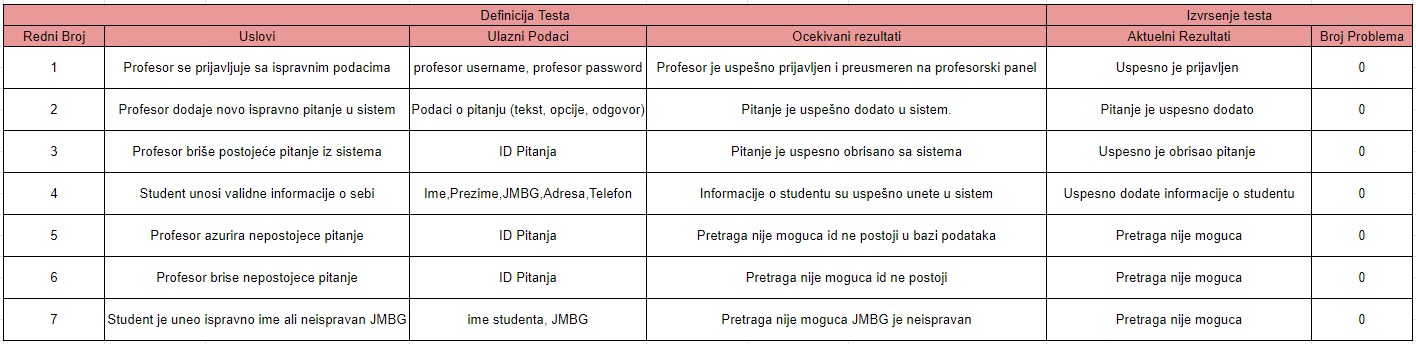




*Slika 11. Regularni izraz(RegEx)*

**Scenario Testiranja**

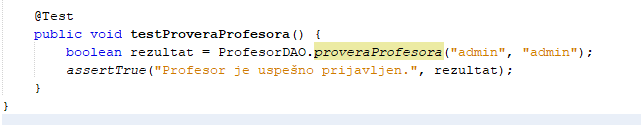




*Slika 12. Tabela sa scenarijima za testiranje*

**JUNIT Testovi**

Ovaj test proverava ispravnost metode proveraProfesora sa korisničkim imenom "admin" i lozinkom "admin". Očekuje se da će rezultat biti true, što ukazuje na uspešnu prijavu profesora sa datim korisničkim imenom i lozinkom.



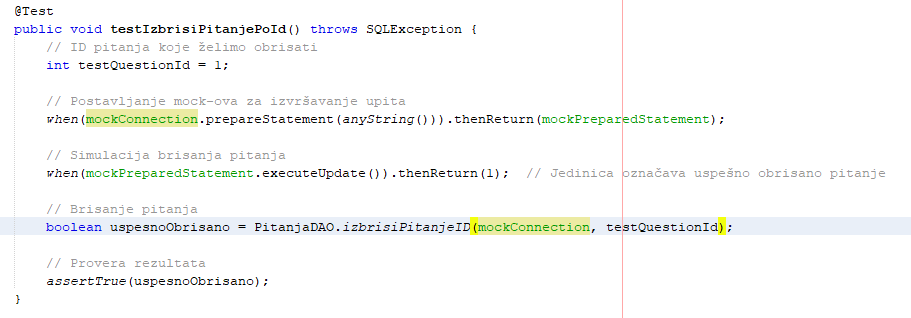
*Slika 13. Provera Profesora Test*

Ovaj test proverava ispravnost funkcionalnosti dodavanja pitanja u bazu podataka, koristeći mock objekte kako bi se izbeglo stvarno povezivanje sa bazom podataka tokom testiranja. Ovakvi testovi su korisni jer omogućavaju izolaciju testiranja od stvarne baze podataka, čime se obezbeđuje doslednost i ponovljivost testova.



*Slika 13. Dodavanje ispravnog pitanja Test*

Ova test metoda proverava da li će pozivanje izbrisiPitanjeID za dato pitanje ID vratiti true, što bi značilo da je pitanje uspešno obrisano. Takodje, i ova metoda koristi mockove da ne bi smo izbrisali neka pitanja iz nase baze podatka:



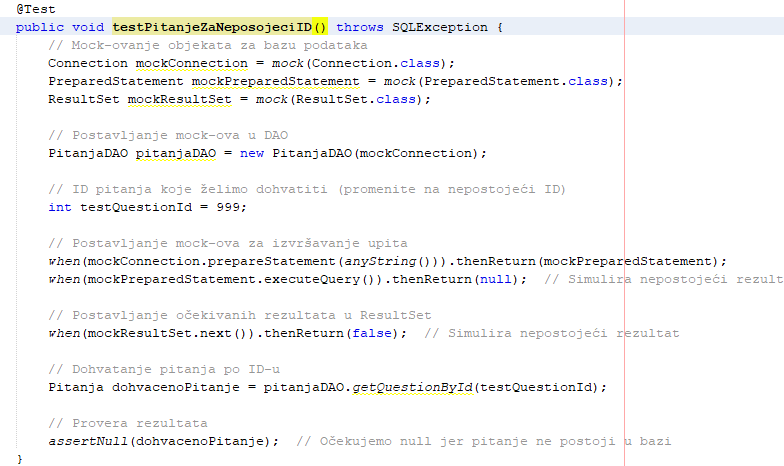
*Slika 14. Brisanje Pitanja Po ID Test*

Ovaj test testira metodu sacuvajStudenta klase StudentDAO. Koristi Mockito za mock-ovanje veza s bazom podataka. Testira se da li se uspešno unose informacije o studentu u sistem, simulirajući SQL upit i proveravajući rezultat, odnosno, da li je vraćena vrednost `true` što ukazuje na uspešno unošenje informacija.



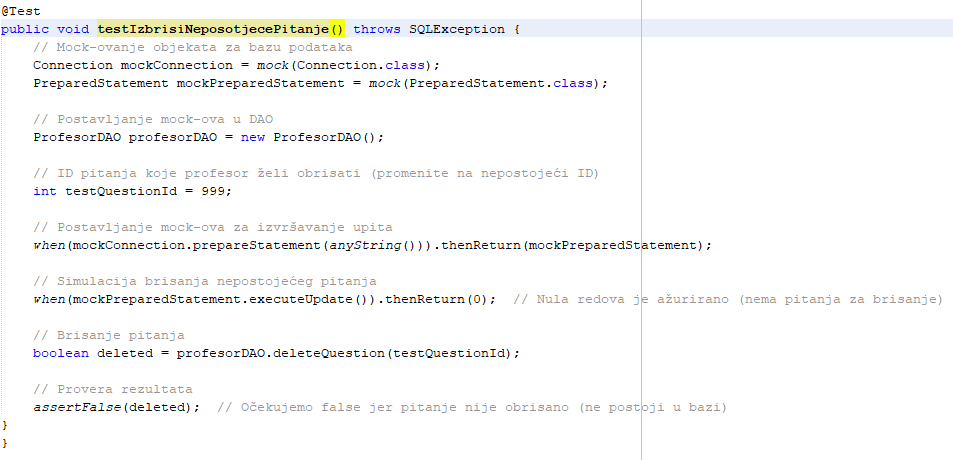
*Slika 15. Cuvanje studenta Test*

Ovaj kod testira metodu getQuestionById klase PitanjaDAO. Koristi Mockito za mock-ovanje veza s bazom podataka. Testira se scenario kada se pokušava dohvatiti pitanje sa zadatim ID-om koji ne postoji u bazi podataka. Proverava se da li rezultat dohvatanja pitanja za nepostojeći ID vraća `null`:



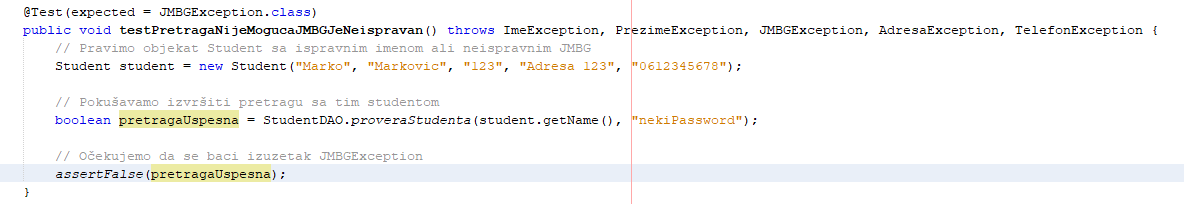
*Slika 16. Azuriranje nepostojeceg pitanja Test*

Ovaj kod jedinica testira metodu deleteQuestion klase ProfesorDAO. Takođe koristi Mockito framework kako ne bi izbrisao nesto iz baze podataka. Testira se scenario kada profesor pokušava obrisati pitanje sa zadatim ID-om koji ne postoji u bazi podataka. Proverava se da li metoda vraća `false`, što ukazuje na to da pitanje nije obrisano:



*Slika 17. Brisanje nepostojeceg pitanja Test*

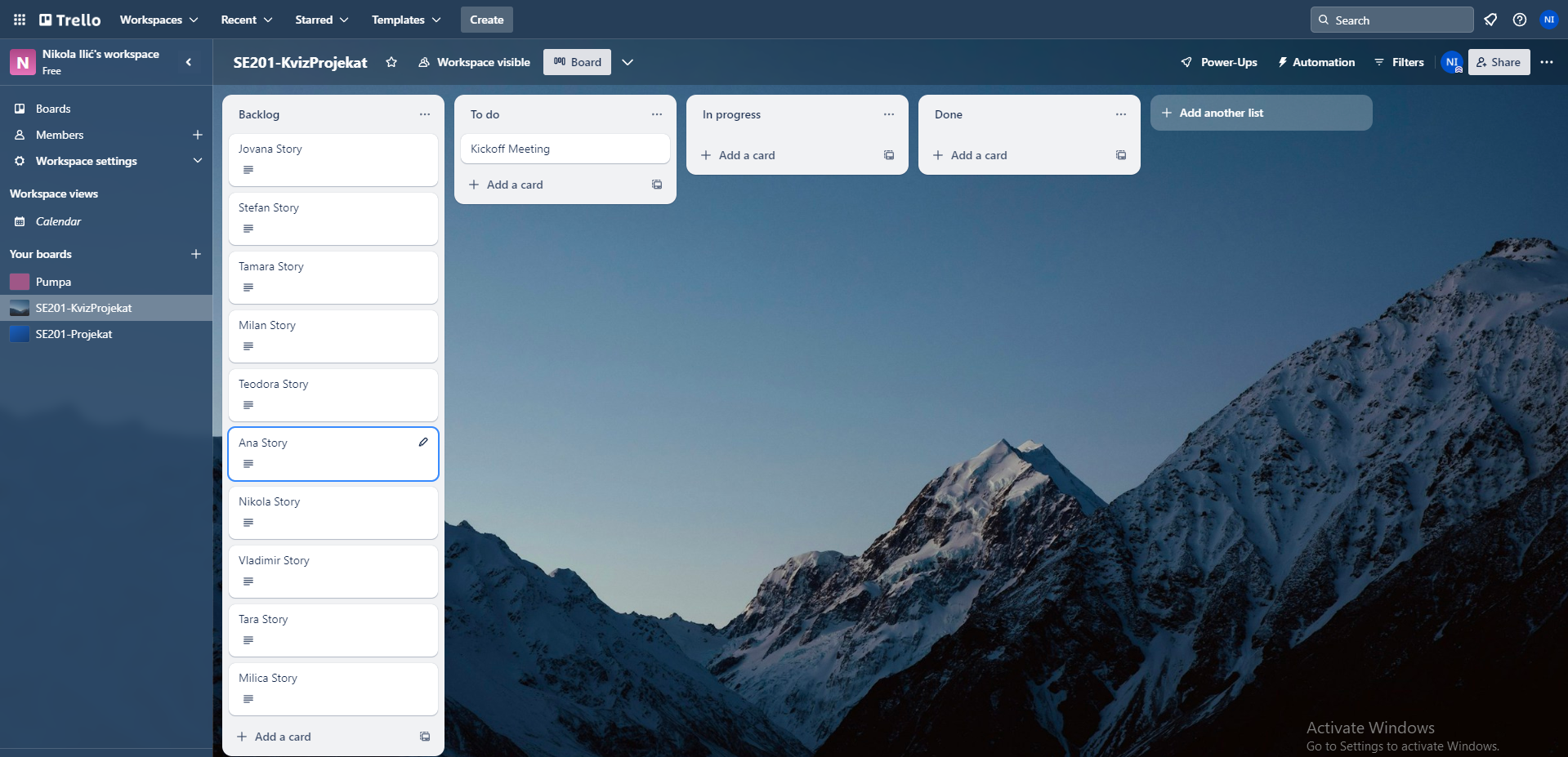
Ovaj test služi za proveru da li se baca izuzetak tipa JMBGException kada se pokuša izvršiti pretraga studenta sa ispravnim imenom, ali neispravnim JMBG-om. Očekuje se da poziv StudentDAO.proveraStudenta(student.getName(), "nekiPassword") izazove izuzetak tipa JMBGException, i test prolazi ako se izuzetak baci:

*Slika 18. Pretraga sa neispravnim JMBG-om Test*

**Trello**

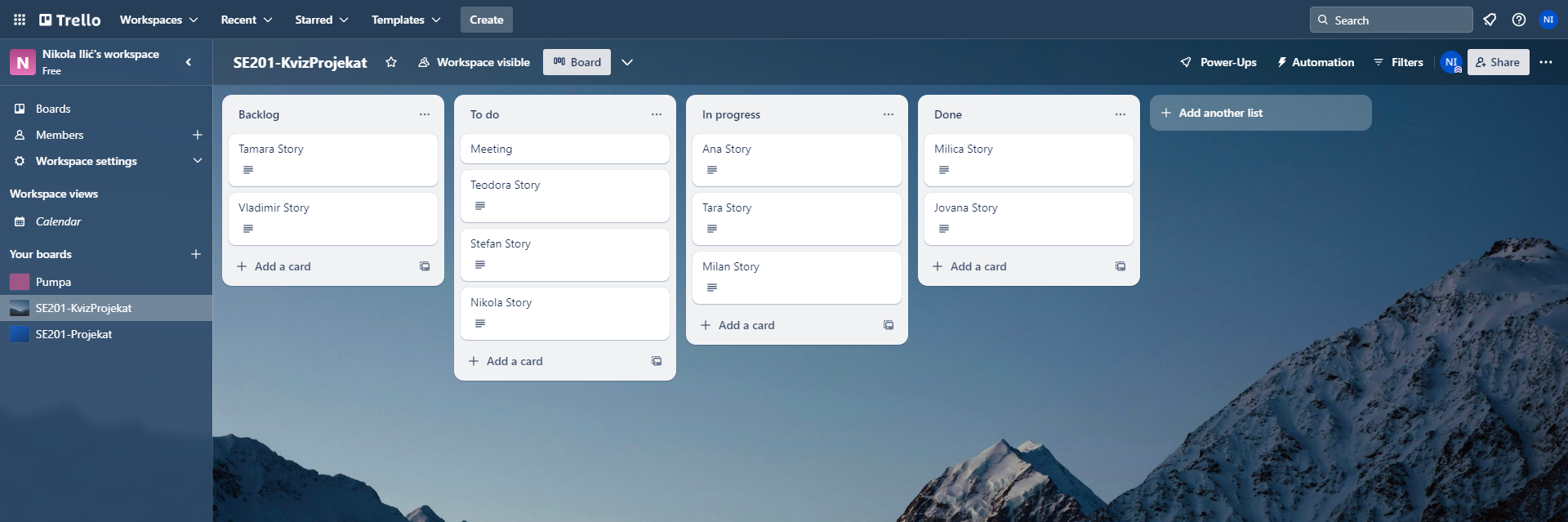
Trello je online alat za upravljanje projektima i organizaciju zadataka. Omogućuje korisnicima da kreiraju virtualne ploče na kojima mogu postavljati kartice koje predstavljaju zadatke, ideje ili projekte. Kartice se mogu premeštati između različitih lista koje predstavljaju različite faze ili statusne korake u projektu.

* Na ovoj slici mozemo videti izgled nase aplikacije u alatu Trello.Sastoji se od "backlog"-a koji obuhvata pojedince i njihove korisničke priče, koje služe kao osnova za razvoj novih funkcionalnosti u našoj aplikaciji.



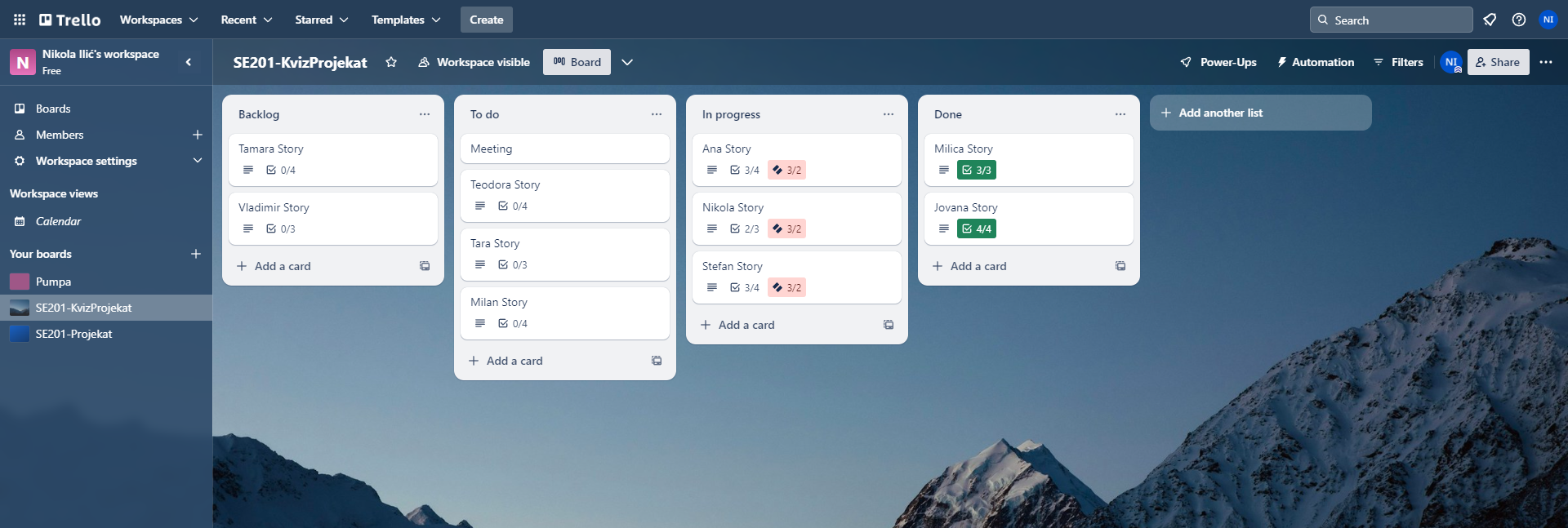
*Slika 19. Ubacivanje Korisnickih prica i pravljenje kartica*

* Na ovoj slici primjećujemo da smo organizirali sve korisničke priče u nekoliko lista: Backlog, To do, in progress i done. Ova organizacija nam pruža bolji pregled o tome koje korisničke priče trebaju još biti obavljene, koje su trenutno u procesu izvršavanja i koje su već završene.



*Slika 19. Rasporedjivanje korisnickih prica*

* Ovde možemo primetiti da smo, osim dodavanja u liste, uneli nove stavke kao kontrolnu listu kako bismo imali bolji pregled funkcionalnosti koje su već obavljene i koje tek treba obaviti.Takodje za karticu In progress smo stavili Kanban WIP power-up koji nam omogucava da se najvise dve kartice izvrsavaju tokom procesa.



*Slika 19.Checklist i Kanban*

**Zakljucak**

Tokom realizacije ovog projekta, uspešno smo primenili agilnu metodologiju razvoja softvera, posebno se opredelivši za Scrum kao agilan metod koji se pokazao izuzetno efikasan u upravljanju projektima. Ova metodologija nam je omogućila fleksibilnost i adaptabilnost tima tokom čitavog procesa razvoja.

Murov obrazac se pokazao kao izvanredan alat za oblikovanje vizije proizvoda, postavljanje jasnih ciljeva i usmerenja projekta.

Naša vizija proizvoda fokusirala se na stvaranje efikasnog sistema za kviz, a kroz precizno definisane funkcionalne i nefunkcionalne zahteve, korisničke priče i persone, stvorili smo jasnu sliku o potrebama i zahtevima korisnika. Scenariji su dodatno obogatili naše razumevanje korisničkih potreba.

U arhitekturi sistema smo se odlučili za mikroservisnu arhitekturu, pružajući nam skalabilnost i fleksibilnost. Identifikovali smo delove sistema koji su mogli biti ponovno upotrebljeni i implementirali ih kao mikroservise sa odvojenim bazama podataka.

Analiza računarstva u oblaku obuhvatila je prepoznavanje delova sistema koji su odgovarali IaaS, SaaS i PaaS pristupima, uzimajući u obzir i one koji nisu imali značajne prednosti od ovakvog pristupa.

Bezbednost sistema bila nam je od suštinskog značaja, te smo proaktivno identifikovali i rešili potencijalne bezbednosne ranjivosti, koristeći demonstraciju Brute Force i SQL injection napada.

Implementirali smo koncepte pouzdanog programiranja kroz validaciju input-a, upotrebu šablona (Pattern) i regularnih izraza (RegEx), čime smo osigurali stabilnost sistema. Testiranje je sprovedeno kroz scenarije testiranja, omogućavajući nam proaktivno otkrivanje i ispravljanje grešaka.

JUnit testovi pružili su dodatnu sigurnost u kvalitet softvera. Efikasno praćenje tokom celog procesa razvoja ostvareno je korišćenjem alata Trello, dok je verzioniranje na Git-u omogućilo precizno praćenje razvoja koda.

**Literatura**

[**http://lams.metropolitan.ac.rs/**](http://lams.metropolitan.ac.rs/) **- SE201 Uvod u softversko inžinjerstvo**

[**https://www.youtube.com/watch?v=6drUzoeHZkg**](https://www.youtube.com/watch?v=6drUzoeHZkg)

[**https://www.youtube.com/watch?v=MMa4AVdBCZY**](https://www.youtube.com/watch?v=MMa4AVdBCZY)