Lambda

TestIT

Arhitekturni projekat

Verzija 1.0

Pregled izmena

Datum	Verzija	Opis	Autor
21.04.2019.	1.0	Inicijalna verzija	Lambda tim

Sadržaj

1. Cilj dokumenta 5
2. Opseg dokumenta 5
3. Reference 5
4. Predstavljanje arhitekture 5
5. Ciljevi i ograničenja arhitekture 5
6. Pogled na slučajeve korišćenja 5
6.1. Dijagrami slučajeva korišćenja 6
6.2. Kratak opis slučajeva korišćenja 8
6.2.1. Pregled informacija 8
6.2.2. Pregled podataka o predmetu 8
6.2.3. Pregled informacija o članovima 8
6.2.4. Pregled podataka o kvizovima 8
6.2.5. Pregled podataka o takmičenjima 8
6.2.6. Registracija 8
6.2.7. Login 8
6.2.8. Prijavljivanje na predmet 8
6.2.9. Ažuriranje sopstvenih podataka 8
6.2.10. Dodavanje novih kvizova 8
6.2.11. Brisanje postojecih kvizova 8
6.2.12. Izmena postojećih kvizova 8
6.2.13. Validacija kviza od strane moderatora 9
6.2.14. Kreiranje pitanja 9
6.2.15. Kreiranje novog člana 9
6.2.16. Brisanje postojećeg člana 9
6.2.17. Kreiranje turnira 9
6.2.18. Učestvovanje u turniru 9
7. Pogled na logičku arhitekturu sistema 9
7.1. Pregled arhitekture – organizacija paketa i podsistema u slojeve 10
7.1.1. Korisnički interfejs (sloj) 10
7.1.2. Aplikaciona logika (sloj) 10
7.1.3. Pristup podacima (sloj) 10
7.1.4. HTML (tehnologija) 10
7.1.5. Razor (tehnologija) 10
7.1.6. MySQL (DBMS) 10
8. Pogled na procese 11
8.1. Procesi 11
8.1.1. Web čitač 11
8.1.2. Web server 11
8.1.3. Controller 11
8.1.4. MySQL Server 11
9. Pogled na raspoređivanje sistema 11
9.1. Klijent 11
9.2. Web server 11
9.3. DBMS server 12
10. Pogled na implementaciju sistema 12
10.1. Model domena 12
10.2. Šema baze podataka 13
10.3. Komponente sistema 13
10.3.1. Komponente korisničkog interfejsa 13

10.3.2. Komponente aplikacione logike 15
10.3.3. Komponente za pristup podacima 16
11. Performanse 16
12. Kvalitet 16

Arhitekturni projekat

1. Cilj dokumenta

Cilj ovog dokumenta je detaljni opis arhitekture TestIT aplikacije.

2. Opseg dokumenta

Dokument se odnosi na TestIT aplikaciju koja će biti razvijen od strane Lambda tima. Namena TestIT aplikacije je interaktivno ucenje i takmicenje.

3. Reference

Spisak korišćene literature:

1. TestIT – Predlog projekta, V1.0, 2019, Lambda tim.
2. TestIT – TestIT_Raspored_Aktivnosti.mpp, V1.0, 2019, Lambda tim.
3. TestIT – Plan realizacije projekta, V1.0, 2019, Lambda tim.
4. TestIT – Vizija sistema, V1.1, 2019, Lambda tim.
5. TestIT - Specifikacija zahteva, V1.1, 2019, Lambda tim.

4. Predstavljanje arhitekture

Arhitektura sistema u dokumentu je prikazana kao serija pogleda na sistem: pogled na slučajeve korišćenja, pogled na logičku arhitekturu sistema, pogled na procese, pogled na razmeštaj komponenti sistema i pogled na implementaciju. Ovi pogledi su predstavljeni odgovarajućim UML dijagramima.

5. Ciljevi i ograničenja arhitekture

Ključni zahtevi i sistemska ograničenja koja imaju značajan uticaj na izbor arhitekture i projektovanje sistema su:

1. TestIT aplikacija će biti implementiran kao Web aplikacija zasnovana na ASP.NET Core MVC Framework i MySQL bazi podataka[4].
2. Klijentski deo TestIT aplikacije će biti optimizovan za sledeće Web čitače: Internet Explorer 11.0 i noviji, Opera, Firefox, Google Chrome i ostali “moderni” čitači[4].
3. Svi zahtevi u pogledu performansi dati u [5] moraju biti uzeti u obzir pri izboru arhitekture i razvoju sistema.

6. Pogled na slučajeve korišćenja

U ovom odeljku je dat pogled na slučajeve korišćenja definisane u specifikaciji zahteva.
Slučajevi korišćenja TestIT portala su:

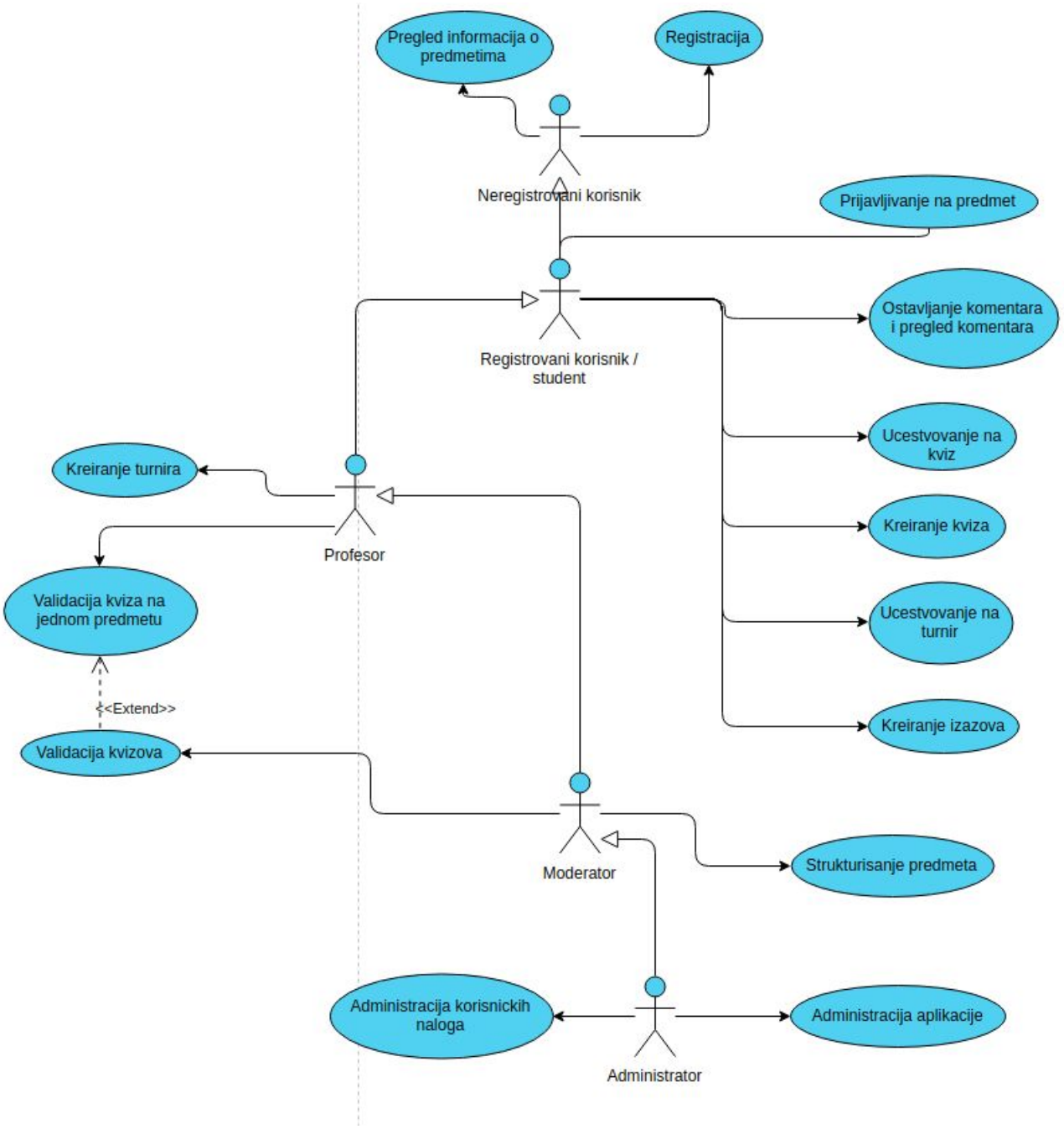
- *Pregled informacija*
  - Pregled osnovnih podataka o predmetu
  - Pregled podataka o članovima
    - Pregled registrovanih korisnika
    - Pregled licnih podataka
  - Pregled podataka o kvizu
  - Pregled komentara
  - Pregled podataka o takmičenjima
- Registracija
- *Login*
- Prijavljivanje na predmet
- Ažuriranje sopstvenih podataka
- *Kviz*
  - Dodavanje novih kvizova
  - Brisanje postojećih kvizova
  - Izmena postojecih kvizova
  - Učestvovanje na kviz
  - Validacija kviza od strane moderatora
- *Kreiranje pitanja*
  - Dodavanje pitanja na postojeće kvizove
  - Izmena postojećih pitanja
  - Brisanje pitanja
- *Turnir*
  - Kreiranje turnira
  - Učestvovanje na turnir

- *Predmet*
  - Dodavanje novog predmeta
  - Strukturisanje predmeta

Ove slučajevi korišćenja mogu da iniciraju: neregistrovani korisnik, registrovani korisnik, profesor, moderator ili administrator.

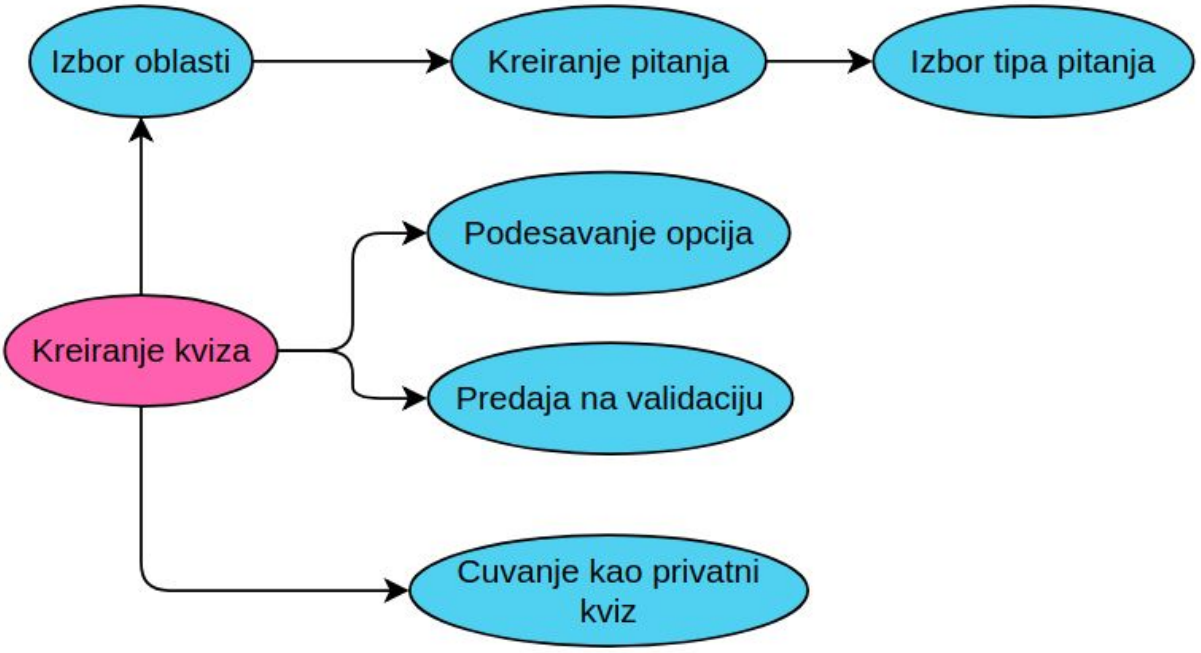
6.1 Dijagrami slučajeva korišćenja

Osnovni UML dijagram koji prikazuje korisnike i slučajeve korišćenja TestIT aplikacije prikazan je na sledećoj slici:

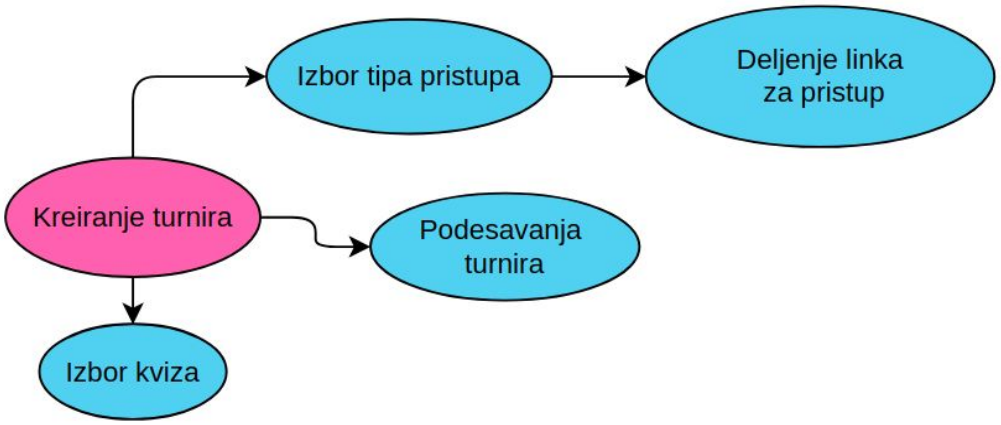


Slučajevi korišćenja *kreiranje kvizova* i *kreiranje turnira, strukturisanje predmeta i učestvovanje u kvizu* obuhvataju složenije radnje koje se mogu razložiti dalje razložiti na pojedinačne slučajeve korišćenja.

Detaljni UML dijagram za slučaj korišćenja *kreiranje kviza* je prikazan na sledećoj slici:

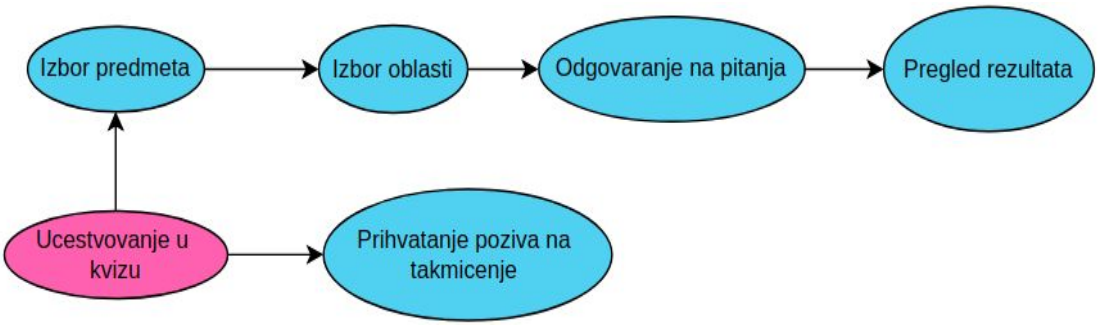


Detaljni UML dijagram za slučaj korišćenja *kreiranje turnira* je prikazan na sledećoj slici:



Detaljni UML dijagram za slučaj korišćenja *strukturisanje predmeta* je prikazan na sledećoj slici:

Detaljni UML dijagram za slučaj korišćenja *učestvovanje u kvizu* je prikazan na sledećoj slici



6.2 Kratak opis slučajeva korišćenja

6.2.1 Pregled informacija

Kratak opis:

Obuhvata pregled sopstvenih informacija, podataka o predmetu, registrovanim korisnicima, objavljenim kvizovima, pregled komentara na predmetima.

6.2.2 Pregled podataka o predmetu

Kratak opis:

Pregled svih informacija, komentara, i kvizova o predmetu.

6.2.3 Pregled informacija o članovima

Kratak opis:

Pregled registrovanih korisnika i sopstvenih informacija.

6.2.4 Pregled podataka o kvizovima

Kratak opis:

Kreiranje pitanja sa svim njegovim pitanjima i ograničenjima.

6.2.5 Pregled podataka o takmičenjima

Kratak opis:

Pregled informacija o tekućim takmičenjima.

6.2.6 Registracija

Kratak opis:

Registracija novog korisnika na web aplikaciji.

6.2.7 Login

Kratak opis:

Prijavljivanje korisnika na sistem radi korišćenja svih usluga TestIT sistema.

6.2.8 Prijavljivanje na predmet

Kratak opis:

Student moze da pregleda listu svih predmeta i da se prijavi na željeni.

6.2.9 Ažuriranje sopstvenih podataka

Kratak opis:

Korisnik će biti u stanju da promeni lične informacije koje se čuvaju o njemu u bazi podataka sistema.

6.2.10 Dodavanje novih kvizova

Kratak opis:

Kreiranje pitanja sa svim njegovim pitanjima i ograničenjima.

6.2.11 Brisanje postojecih kvizova

Kratak opis:

Uklanjanje postojećih kvizova iz baze podataka.

6.2.12 Izmena postojećih kvizova

Kratak opis:

Korisnik će biti u stanju da izmeni podatke o kvizu kao i svih pitanja i odgovora vezanih za taj kviz.

6.2.13 *Validacija kviza od strane moderatora*

Kratak opis:

Moderatori će biti u stanju da iz reda kvizova na čekanju odobri sve kvizove koji su adekvatni.

6.2.14 *Kreiranje pitanja*

Kratak opis:

Obuhvata dodavanje pitanja na postojeći kviz, brisanje i izmenu postojećih pitanja.

6.2.15 *Kreiranje novog člana*

Kratak opis:

Kreiranje korisničkog naloga za novog člana laboratorije.
Akteri koji iniciraju slučaj korišćenja: Šef laboratorije, Administrator.

6.2.16 *Brisanje postojećeg člana*

Kratak opis:

Brisanje korisničkog naloga i podataka za postojećeg člana laboratorije.
Akteri koji iniciraju slučaj korišćenja: Šef laboratorije, Administrator.

6.2.17 *Kreiranje turnira*

Kratak opis:

Kreiranje turnira zajedno sa svim limitacijama.

6.2.18 *Učestvovanje u turniru*

Kratak opis:

Korisnik moze učestvovati na turnir za koji je dobio poziv.

7. Pogled na logičku arhitekturu sistema

U ovom odeljku je dat pregled logičke arhitekture sistema. Ovaj pogled sadrži opis najznačajnijih klasa, njihove organizacije u pakete i podsisteme, i organizacija podsistema u slojeve. U cilju opisivanja dinamičkih aspekata arhitekture, ovaj odeljak može da uključi opise realizacije najznačajnijih slučajeva korišćenja. Da bi se ilustrovala veza između arhitekturno značajnih klasa, podsistema, paketa ili slojeva moguće je uključiti i odgovarajuće dijagrame klasa.

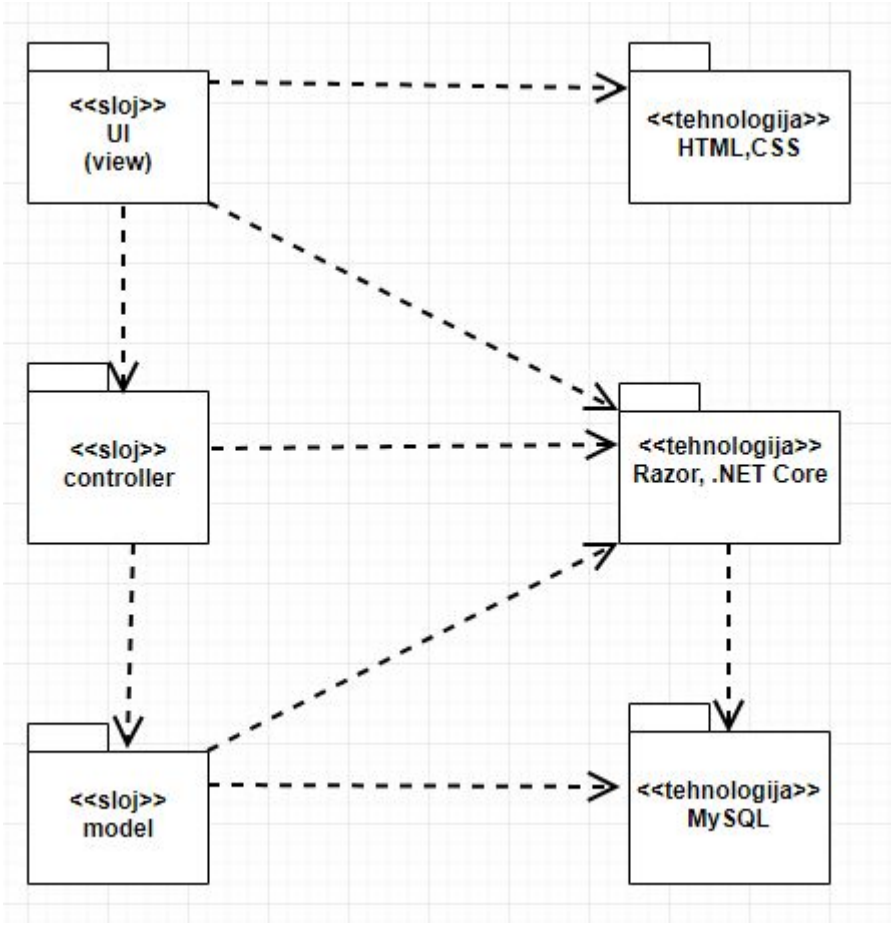
Logički pogled naTestIT aplikaciju obuhvata 3 glavna paketa: Korisnički interfejs (UI), Aplikaciona logika (Controller), Pristup podacima (model).

Paket *UI* sadrži Web stranice, Razor skripte i multimedijalni sadržaj koji realizuju grafički dizajn i forme preko kojih korisnici sistema komuniciraju sa sistemom.

Paket *controller* predstavlja srednji sloj sistema koji sadrži Razor skripte zadužene za realizaciju funkcionalnosti specifičnih za domen sistema koji se razvija.

Paket *Pristup podacima* sadrži Razor skripte i klase koje predstavljaju interfejs za pristup, dodavanje i ažuriranje podataka koji se čuvaju u bazi podataka.

7.1 Pregled arhitekture – organizacija paketa i podsistema u slojeve



7.1.1 Korisnički interfejs (sloj)

Ovaj sloj realizuje korisnički interfejs aplikacije. U njemu su sadržane sve HTML, multimedijalni sadržaji i Razor skripte koje generišu HTML stranice preko kojih korisnici komuniciraju sa sistemom.
Sloj korisničkog interfejsa zavisi od sloja aplikacione logike, kao i paketa HTML i Razor.

7.1.2 Aplikaciona logika (sloj)

Sloj aplikacione logike je srednji sloj u troslojnoj arhitekturi TestIT aplikacije. Sadrži Razor skripte i .Net core klase kontrolera koje realizuju funkcionalnost karakterističnu za domen primene aplikacije i uspostavljaju vezu između korisničkog interfejsa i sloja za pristup podacima.
Ovaj sloj zavisi od sloja za pristup podacima i Razor paketa.

7.1.3 Pristup podacima (sloj)

Sloj za pristup podacima se nalazi na dnu troslojne arhitekture i sadrži .Net core klase zadužene za pribavljanje, dodavanje i ažuriranje podataka koji se čuvaju u MySQL bazi podataka.
Ovaj sloj ne zavisi od drugih slojeva, ali je zavisan od paketa .Net core i MySQL baza podataka.

7.1.4 HTML (tehnologija)

Tehnologija HTML-a definiše gradivne elemente stranica koje se prikazuju u Web čitaču i koje omogućavaju prikaz formatiranih informacija i realizaciju formi za unos i ažuriranje podataka.

7.1.5 Razor (tehnologija)

Tehnologija Razor-a obezbeđuje mehanizam za pisanje i izvršavanje skripti na strani servera. Ove skripte mogu da generišu HTML kod koji realizuje korisnički interfejs i pristupaju bazi podataka u cilju pribavljanja, unosa i ažuriranja podataka.

7.1.6 MySQL (DBMS)

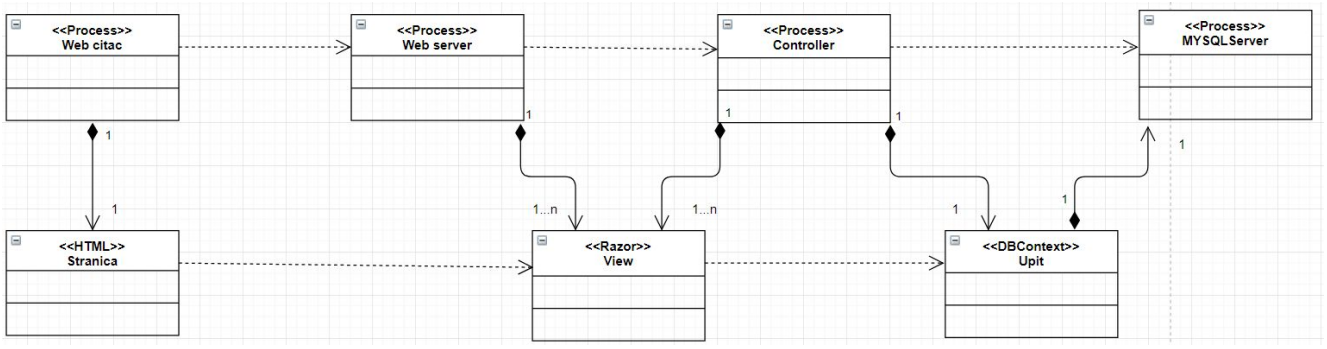
MySQL predstavlja sistem za upravljanje bazama podataka koji će se koristiti za realizaciju TestIT aplikacije.

8. Pogled na procese

U nastavku je dat opis procesa uključenih u izvršenje TestIT portala kao Web aplikacije.

8.1 Procesi

Na sledećem UML dijagramu klasa prikazani su procesi koji učestvuju u izvršenju TestIT portala.



8.1.1 Web čitač

Web čitač je proces koji izvršava funkcionalnost aplikacije za prikaz HTML stranica dobijenih od nekog Web servera. U najopštijem slučaju Web čitač u jednom trenutku može da prikazuje samo jednu HTML stranicu. Web čitač zavisi od Web servera koji generiše i vraća odgovarajuću HTML stranicu na zahtev.

8.1.2 Web server

Web server je proces koji izvršava funkcionalnost opsluživanja zahteva prispelih sa više Web čitača. Ukoliko je zahtevana stranica validnog kontrolera, Web server inicira izvršenje tog kontrolera koji obrađuje zahtev i generiše View koji se vraća čitaču. Web server može paralelno da izvršava više kontrolera.

8.1.3 Controller

Controller proces obavlja posao obrade prosledjenog zahteva i generiše odgovarajući View ili odgovor koji Web server šalje Web čitaču. Ovaj proces može da zahteva usluge MySQL servera-a. Komunikacija između kontrolera i MySQL servera se obavlja preko DatabaseContext-a.

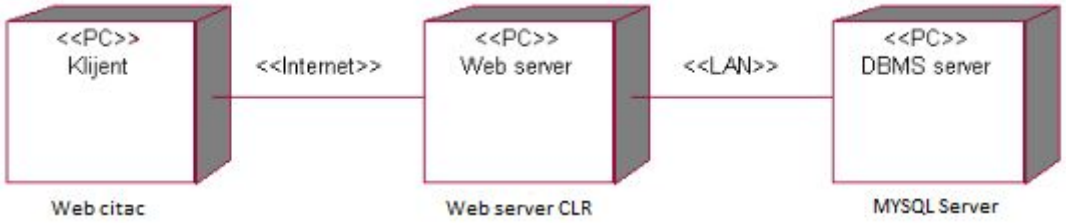
8.1.4 MySQL Server

MySQL Server je proces koji izvršava funkcionalnost MySQL sistema za upravljanje bazama podataka. Ovaj proces može konkurentno da prihvati određen broj upita, izvrši ih nad bazom podataka i vrati rezultate procesu koji je upite postavio.

9. Pogled na raspoređivanje sistema

Pogled na raspoređivanje sistema prikazuje različite fizičke čvorove za najopštiju konfiguraciju sistema. Fizičkim čvorovima koji predstavljaju procesore vrši se dodeljivanje identifikovanih procesa.

Na sledećoj slici dat je UML dijagram raspoređivanja TestIT aplikacije.



9.1 Klijent

Pristup TestIT aplikaciji se obavlja preko klijentskih računara na kojima se izvršava Web čitač. Za povezivanje između klijenta i Web servera koristi se Internet infrastruktura tako da nema ograničenja u pogledu lokacije klijenta.

9.2 Web server

Računar na kome se izvršava Web server opslužuje više klijenata koji pristupaju preko Interneta. Pored osnovnog procesa koji realizuje funkcionalnost Web servera, na ovom računaru mogu da se izvršavaju i procesi CLR-a koji vrše obradu zadatih Razor skripti. U najopštijoj konfiguraciji DBMS se izvršava na posebnoj mašini koja je sa Web serverom u lokalnoj mreži (LAN).

9.3 DBMS server

DBMS server je računar na kome se izvršava MySQL Server proces koji realizuje funkcionalnost sistema za upravljanje bazama podataka. Zbog sigurnosti podataka koji se na ovom računaru čuvaju pristup bazi je ograničen samo na računare iz lokalne mreže (LAN).

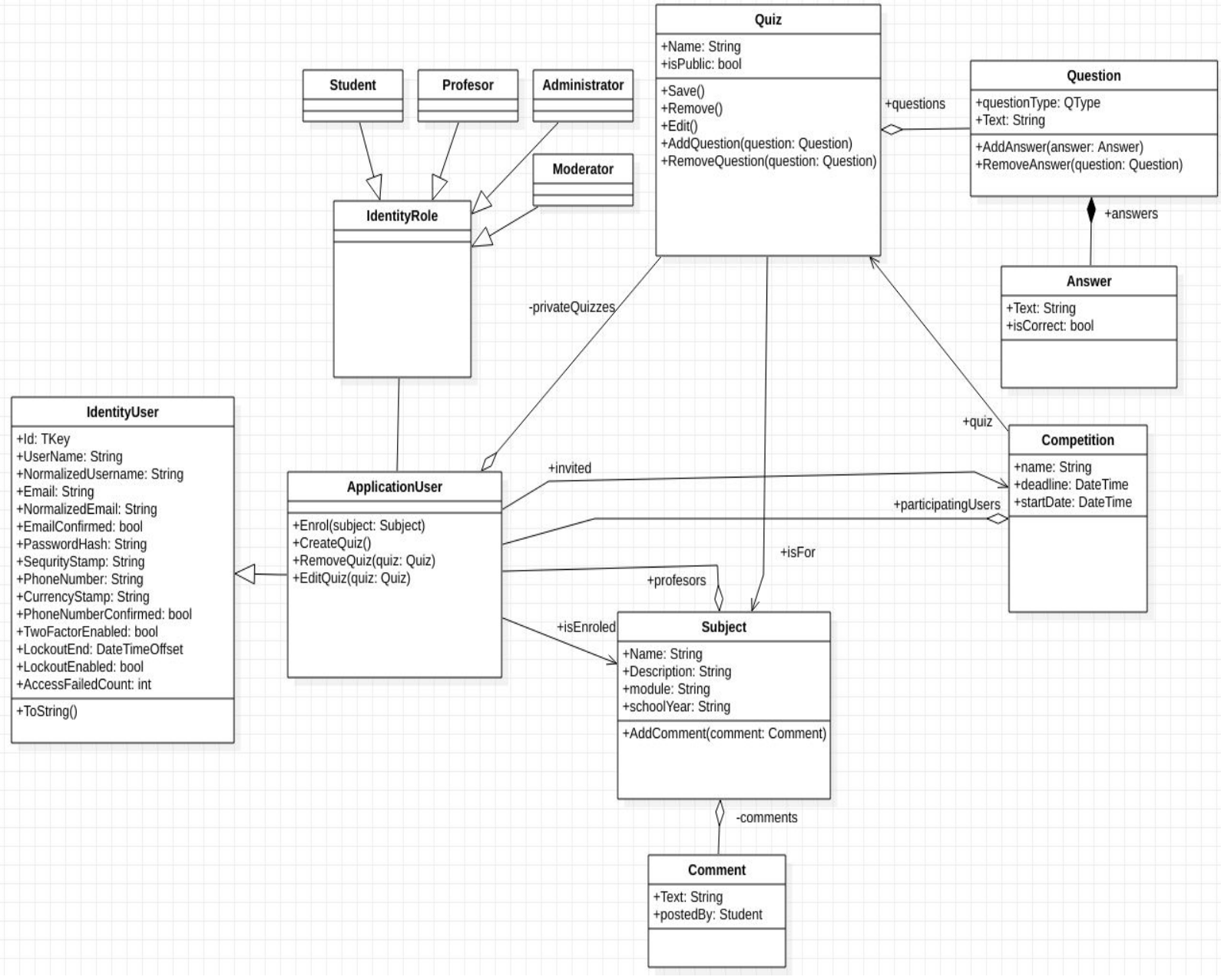
10. Pogled na implementaciju sistema

Pogled na implementaciju prikazuje različite aspekte bitne za implementaciju sistema. U slučaju TestIT aplikacije ovaj odeljak sadrži model domena, šemu baze podataka i prikaz komponenti sistema razvrstanih u ranije identifikovane pakete.

10.1 Model domena

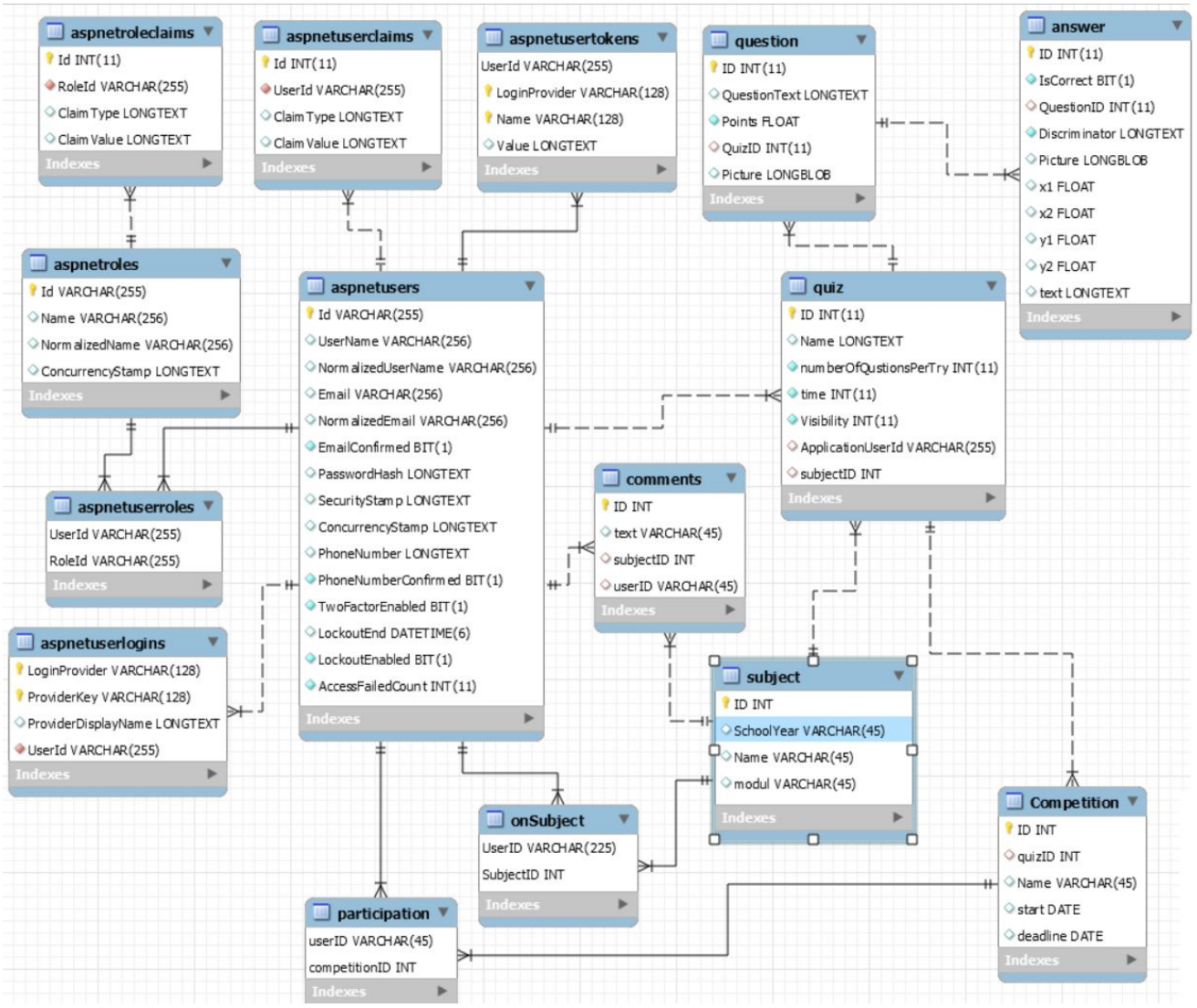
Model domena za koji se TestIT aplikacija projektuje je ilustrovan UML dijagramom klasa. U njemu su prikazane domenske klase, neki od njihovih atributa, kao i veze koje se mogu identifikovati između njih.

Model domena predstavlja osnovu za projektovanje baze podataka, ali i identifikaciju nekih od komponenti koje će biti implementirane.



10.2 Šema baze podataka

Detaljna šema baze podataka je prikazana na sledećem dijagramu. Za kreiranje baze podataka na osnovu modela entiteta koriste se Entity Framework migracije, pokretanje tih migracija na MySQL serveru se obavlja od strane funkcije koje se poziva prilikom startovanja aplikacije. Šema je kreirana pomoću MySQL workbench-a import-ovanjem SQL DDL naredbi.



10.3 Komponente sistema

10.3.1 Komponente korisničkog interfejsa

Dizajn korisničkog interfejsa je obuhvaćen sledećim komponentama:

- View (cshtml)
- JavaScript
- CSS

Komponenta **index.cshtml** implementira stranicu portala čiji sadržaj može da varira od parametra u view modelu koji se koristi. Ova komponenta zavisi od JavaScript i CSS komponenti.

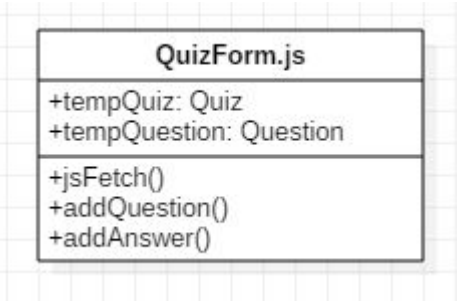
Komponenta **site.css** predstavlja opis stilova za pojedine HTML elemente koji se javljaju na različitim stranicama.

Pored CSS-a i HTML-a koristi se i Bootstrap uz pomoc kojeg se kreira dinamicki sadrzaj stranica koji odgovara velicini ekrana na kome se prikazuju stranice.

Komponenta **site.js** je zaduzena za dinamičko kreiranje elemenata u index.cshtml komponenti kao i globalnih funkcija korišćenih na svim stranicama..

Ako view treba da prikazuje kompleksniji skup podataka onda se za enkapsulaciju tog skupa koriste view Modeli. To su obicne C# clase (POCO), koje se prosledjuju u View. View čita podatke iz njemu prosleđenog modela i popunjava odgovarajuća polja za prikaz korisniku. U sledećem odeljku za svaki View koji ima parametre bice opisano koje parametre prima i u slučaju da koristi JavaScipt funkcije biće priloženi klasni dijagrami tih stranica.
U projektu se koristi veliki broj view-a:

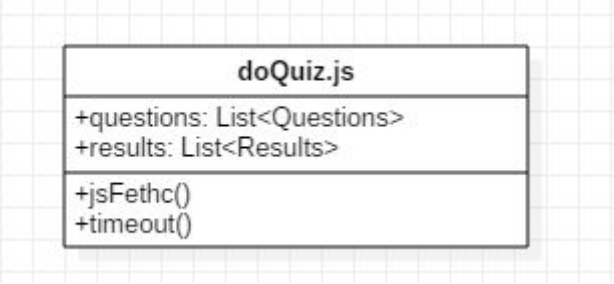
- Home/index.cshtml
  - Ne koristi view model
  - Prikazuje pocetnu stranicu aplikacije
  - Sadrzi linkove ka stranici za registraciju korisnika i kreiranje naloga,
- Home/privacy.cshtml
  - Ne koristi view model
  - Sluzi za prikazivanje klauzule o privatnosti informacija (GDPR compliant)
  - Sadrzi linkove ka stranicama za registraciju i kreiranje naloga
- Home/Navigation/Index.cshtml
  - Ne koristi view model
  - Sluzi za prikaz predmeta sortiranih po modulima i godinama
- Identity/Account/Login.cshtml
  - Koristi LoginModel
  - Sluzi za prikupljanje podataka o korisniku i i pripomaze prilikom logina
- Identity/Account/Logout.cshtml
  - Koristi LogoutModel
  - Sluzi za prikazivanje informacije o uspesnom odjavljivanju
- Identity/Account/Register.cshtml
  - Koristi RegisterModel
  - Sluzi za registraciju korisnika i prikupljanje podataka o korisnickom imenu i passwordu
- Identity/Account/Manage/Index.cshtml
  - Koristi IndexModel
  - Sluzi za pregled informacija o ulogovanom koristniku
  - Takodje omogucava promenu tih informacija
- Identity/Account/Manage/ChangePassword.cshtml
  - Koristi ChangePasswordModel
  - Sluzi za bezbendu zamenu sifre korisnika
- Quiz/index.cshtml
  - Koristi Quiz view model
  - Prikazuje listu svih kvizova ulogovanog korisnika
  - Sadrzi linkove ka stranicama za kreiranje, brisanje i izmenu qvizova
- Quiz/Create.cshtml
  - Ne koristi view model vec dinamicki pomocu JavaScripta kreira objekte koji se salju akciji za kreiranje u Quiz controlleru
  - Koristi QuizForm.js za prikupljanje podataka o kvizu i kreiranje odgovarajucih objekata koje se prosledjuju FetchCreate akciji iz Quiz Controllera
  - Omogucava dodavanje pitanja kao i njihovih odgovora



Ova klasa sluzi za prikupljane podataka o kvizu i pozivanje jsFetch() akcije iz Quizz kontrolera, podaci se prikupljaju pomocu addAnswer() i addQuestion metode(), addAnswer() ubacuje odgovore u tempQuestion a addQuestion() ubacuje tako nastalo pitanje u tempQuiz, ovaj proces se ponavlja dok kviz nije zavrsen, nakon cega ga jsFetch funkcija pretvara u form data objecat i prosledjuje jsFetch() funkciji iz Quiz Kontrolera

- Quiz/Delete.cshtml
  - Koristi Quiz view model
  - Na osnovu poziva akciju iz kontrolera za birsanje kviza kome prosledjuje

    - Izvrsavanje te akcije poziva se akcija za prikaz Quiz/index.cshtml
- Quiz/details
    - Koristi Quiz view model
    - Sluzi za prikaz informacija o prosledjenom kvizu
- Quiz/Edit
    - Koristi Quiz view model
    - Na osnovu prosledjenog modela popunjava edit komponente za izmenu kviza
    - Koristi QuizForm.js za prikupljanje podataka o kvizu i kreiranje odgovarajucih objekata koje se prosledjuju FetchEdit akciji iz Quiz Controllera
- Quiz/doQuiz
    - Koristi doQuiz view model
    - Prikazuje formu za pokusaj odrade kviza
    - Koristi doQuiz.js koji u sebi sadrzi timer koj odbrojava preostalo vreme na kvizu
    - Nakon isteka vremena ili nako submitovanja svih odgovora pokrece se akcija za prikaz Quiz/results.cshtml

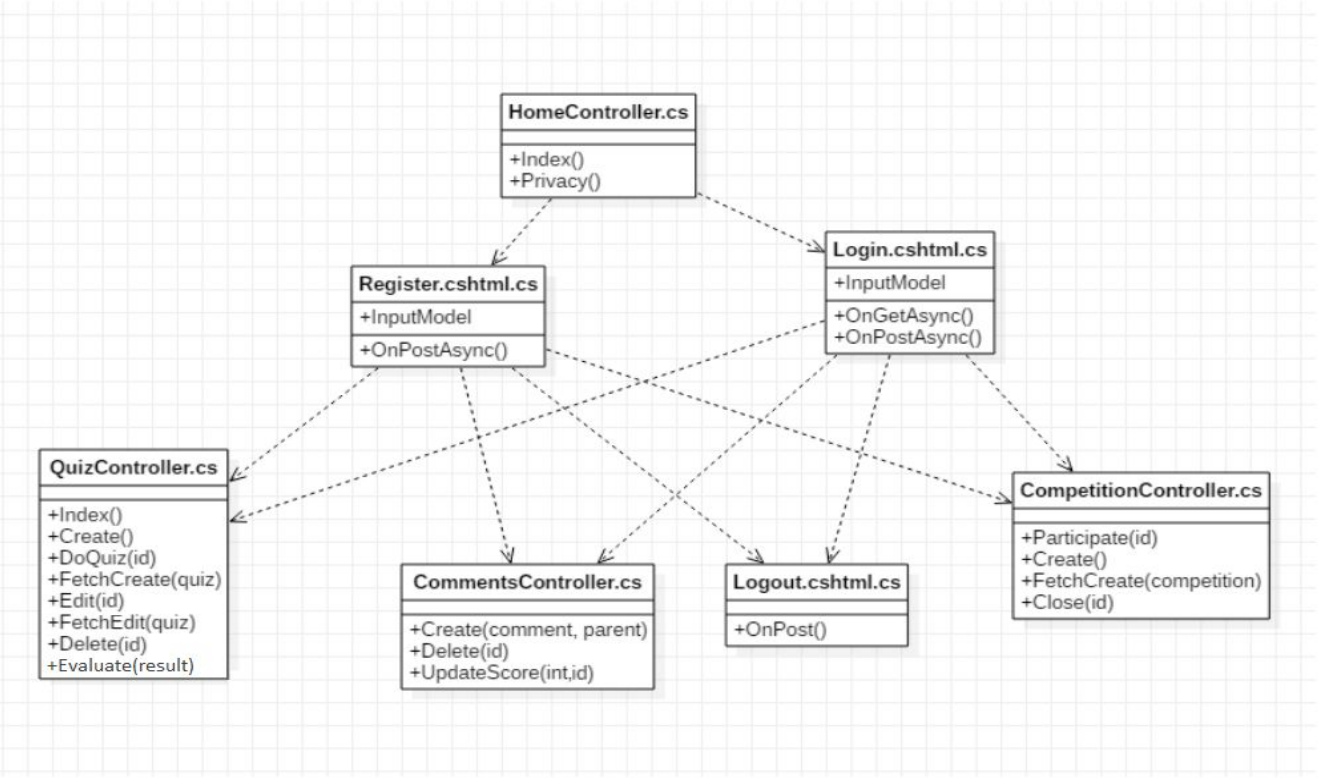


doQuiz.js u sebi sadrzi listu pitanja koje ce se prikazati korisniku. Korisnik ce imati odredjeno vreme za davanje odgovora na pitanja koje ce se cuvati u listi odgovora. Ta lista se pomocu jsFetch() funkcije salje evaluateResults() akciji iz Quiz kontrolera a korisniku se prikazuje Quiz/results.cshtml

- Quiz/results.cshtml
    - Koristi Results model
    - Prikazuje rezultate pokusaja kviza.
- Competition/index.cshtml
    - Ne koristi view model
    - Ucitava iz trenutno ulogovanog koristnika listu takmicenja na koje je pozvan
    - Izborom takmicenja poziva se akcija za pregled takmicenja
- Competition/details.cshtml
    - Koristi kompetition view model
    - Prikazuje podatke o takmicenju i omogucava korisnicima da ucestvuju odradom kviza pozivanjem Quiz/doQuiz(id)
- Competition/create.cshtml
    - Koristi competition view model
    - Omogucava korisniku da unosi osnovne podatke o turniru i da izabere odgovarajuci kviz
- Competition/delete.cshtml
    - Koristi competition view model
    - Sluzi za brisanje turnira iz baze podataka.

10.3.2 Komponente aplikacione logike

Komponente koje realizuju domen problema su kontroleri koji na serveru obrađuju sve prosleđene podatke i izvršavaju svu potrebnu logiku. Svaka stranica (View) zove neki kontroler koji u sebi ima akcije (funkcije) koje ili vraćaju nove poglede ili obrađuju prosleđene podatke. Na sledećem dijagramu su prikazane komponente ovog sloja i njihove međusobne zavisnosti:



HomeController u sebi ima akcije *Index* i *Privacy* koje su povezane sa istoimenim View stranicama navedenim gore i generišu ih.

Login i **Register** klase u sebi imaju kontrolere i **InputModel** model podataka, **InputModel** su podaci iz HTML forme te stranice koji se obrađuju, **Register** klasa radi samo sa POST zahtevom i čuva korisnika u bazu podataka dok **Login** klasa prihvata POST zahtev za (pokušaj) prijavljivanja i GET zahtev koji služi za čišćenje sesije.

Iako postoje više modela, TestIT aplikacija u okviru jednog kontrolera radi sa više modela, ne postoji specifični kontroler za svaki model.

QuizController rukuje svim akcijama kviza. *Index* akcija generiše View svih kvizova nekog korisnika. *Create* akcija otvara formu za kreiranje kviza. *FetchCreate* ima parametar quiz koji se generiše u JavaScript fajlu na klijentu i šalje se kontroleru za obradu: pravi se novi kviz sa tim podacima i čuva se u bazu podataka. *DoQuiz* ima parametar id koji govori kontroleru koji kviz da učita iz baze podataka i da generiše odgovarajući View za taj quiz kako bi korisnik mogao da radi taj kviz. *Evaluate* akcija čuva rezultat odrađenog kviza u bazu podataka. *Edit* akcija generiše View za menjanje postojećeg kviza čiji je id primljen kao parametar akcije. *FetchEdit* ima parametar quiz koji ako nije null, ažurira postojeći kviz u bazi podataka. *Delete* akcija briše kviz čiji je id primljen kao parametar.

CommentsController rukuje svim akcijama komentara. *Create* akcija čuva novi komentar u bazu podataka, parametri su comment koji je tekst komentara i parent koji označava objekat kome pripada komentar (kviz/predmet). Akcija *Delete* briše komentar čiji je id primljen kao parametar. Akcija *UpdateScore* ima parametar int koji ažurira poene komentara čiji je id takođe u parametru akcije.

Logout klasa ima *OnPost* akciju koja odjavi korisnika i očisti sesiju tog korisnika.

CompetitionController rukuje svim akcijama takmičenja. *Participate* akcija dozvoljava korisniku da pristupi takmičenju čiji je id parametar te akcije i vraća odgovarajući View. *Create* akcija generiše View za pravljenje novog takmičenja. *FetchCreate* ima parametar competition koji obrađuje i generiše novo takmičenje koje čuva u bazu podataka. Akcija *Close* zatvara turnir čiji je id parametar akcije i briše ga iz baze podataka.

10.3.3 Komponente za pristup podacima

Za pristup bazi podataka koristi se Entity Framework u okviru koga je implementirana klasa za pristup podacima pod imenom **IdentityDbContext**. U TestIT aplikaciji nasledjujemo **IdentityDbContext** u **ApplicationDbContext** koji u sebi sadrzi kolekcije podataka o kvizovima, korisnicima i predmetima. Za implementaciju tih kolekcija koriti se Entity Framework klase **DbSet**, koje predstavljaju jedan entitet iz relacione baze podataka.

Pristupanjem bilo kom DbSetu iz applicationDbContexta pozivaju se funkcije za pristup i citanje iz baze podataka koje te podatke pretvaraju u objekte za rad u applikacij. Upisivanjem podataka u DbSet kolekcije ima za posledicu upisivanje tih podataka u odgovarajuce tabele u relacionu bazu podataka.

11. Performanse

Izabrana arhitektura softvera podržava zahteve u pogledu broja korisnika koji mogu simultano pristupati sistemu i vremena odziva za pristup bazi podataka specificirane u zahtevima u pogledu performansi [5]:

1. Sistem će da podrži do 1000 simultanih pristupa korisnika portalu.
2. Vreme potrebno za pristupanje bazi podataka u cilju izvršenje nekog upita ne sme da bude veće od 5 sekundi.

Beta verzija će biti ograničena vremenom procesora na 60 minuta korišćenja procesora zbog Azure platforme.

12. Kvalitet

Izabrana arhitektura softvera podržava zahteve u pogledu dostupnosti i srednjeg vremena između otkaza specificirane u zahtevima u pogledu pouzdanosti [5]:

1. TestIT aplikacija će biti dostupna 24 časa dnevno, 7 dana u nedelji. Vreme kada aplikacija nije dostupna ne sme da pređe 5%.
2. Srednje vreme između dva sukcesivna otkaza ne sme da padne ispod 120 sati.