Arhitekturni projekat

Raspored

Tim: *La Plávusha*

April 2016

Pregled izmena

Verzija 1.0 (24. april 2016)

Inicijalna verzija dokumenta.

Sadržaj

Glava 1

Cilj dokumenta

Cilj ovog dokumenta je opis arhitekture aplikacije **Raspored**.

Glava 2

Opseg dokumenta

Dokument se odnosi na aplikaciju **Raspored** koja će biti razvijana od strane tima *La Plávusha*. Namena aplikacije je mogućnost jednostavne personalizacije rasporeda časova za studente; i jednostavna podela studenata na grupe za razne tipove časova i aktivnosti za profesore.

Glava 3

Reference

[1] **Raspored** – *Predlog projekta*, V1.0.1; 2016, *La Plávusha*.

[2] **Raspored** – *Plan realizacije*, V1.0.1; 2016, *La Plávusha*.

[3] **Raspored** – *Planirani raspored aktivnosti na projektu*, V1.0; 2016, *La Plávusha*.

[4] **Raspored** – *Vizija sistema*, V1.0.1; 2016, *La Plávusha*.

[5] **Raspored** – *Specifikacija zahteva*, V1.1; 2016, *La Plávusha*.

Glava 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.

Glava 5

Ciljevi i ograničenja arhitekture

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

(1) Aplikacija **Raspored** će biti implementirana kao SPA (*single-page application*, jednostranična aplikacija) poštujući HTML5 i CSS3 specifikaciju, koristeći Angular 2.0 (frejmvork za JavaScript) i ASP.NET kao bekend podršku aplikaciji. Za bazu podataka će biti korišćen SQL Server. [4]

(2) Klijentski deo aplikacije **Raspored** će biti optimizovan za sledeće veb-čitače: prvenstveno poslednjih nekoliko verzija pregledačā *Google Chrome* i *Mozilla FireFox*, kao i njihove verzije za mobilne uređaje, mada će uzeti u obzir i *Opera*, *Internet Explorer 11* i poslednja verzija veb-pregledača *Edge*. Deo aplikacije za studente biće u potpunosti responzivan i prilagođen svim veličina ekrana, posebno uzimajući u obzir male ekrane sa tableta, mobilnih telefona i drugih pametnih uređaja, dok će deo aplikacije namenjen zaposlenima (asistentima i administraciji) biti optimizovan samo za velike desktop računare. [4]

(3) Svi zahtevi u pogledu performansi dati u *Specifikaciji zaheteva* [5] moraju biti uzeti u obzir pri izboru arhitekture i razvoju sistema.

Glava 6

Pogled na slučajeve korišćenja

U ovoj glavi je dat pogled na slučajeve korišćenja koji su detaljnije definisani u dokumentu *Specifikacija zahteva* [5].

Slučajevi korišćenja aplikacije **Raspored** su:

1. **Opšti**
   - **1.1.** Prijavljivanje na sistem
2. **Student**
   - **2.1.** Pregled rasporeda (globalnog, zvaničnog i ličnog)
   - **2.2.** Dodavanje časa u lični raspored
   - **2.3.** Dodavanje aktivnosti u lični raspored
   - **2.4.** Sakrivanje aktivnosti iz ličnog rasporeda
   - **2.5.** Pregled informacija vezanih za konkretan čas
   - **2.6.** Dodavanje taskova
   - **2.7.** Pretraživanje oglasne table
   - **2.8.** Dodavanje oglasa na oglasnu tablu
3. **Zaposleni**
   - **3.1.** Pregled rasporeda
   - **3.2.** Pregled grupa
   - **3.3.** Pregled studenata jedne grupe

3.4. *Manipulacija smerovima*

3.4.1. Pravljenje rasodele (podeli na x, podeli da ima x, manuelno)

3.4.2. Pregled svih studenata jednog smera

3.4.3. Pretraga studenata jednog smera

3.4.4. Prikaz globalnog rasporeda studenata jednog smera

3.5. *Manipulacija raspodelama*

3.5.1. Kopiranje raspodele

3.5.2. Prikaz svih grupa

3.5.3. Izmena raspodele

3.5.4. Izvoz raspodele

3.6. *Manipulacija grupama*

3.6.1. Otkazivanje časa

3.6.2. Prikazivanje kombinovanog rasporeda

3.6.3. Dodavanje i brisanje studenata

3.6.4. Dodavanje obaveštenja vezanih za konkretan čas

3.7. *Manipulacija studentima*

3.7.1. Dodavanje u grupu

4. Asistent

4.1. Dodavanje novog predmeta

4.2. Unos i ažuriranje podataka o predmetu

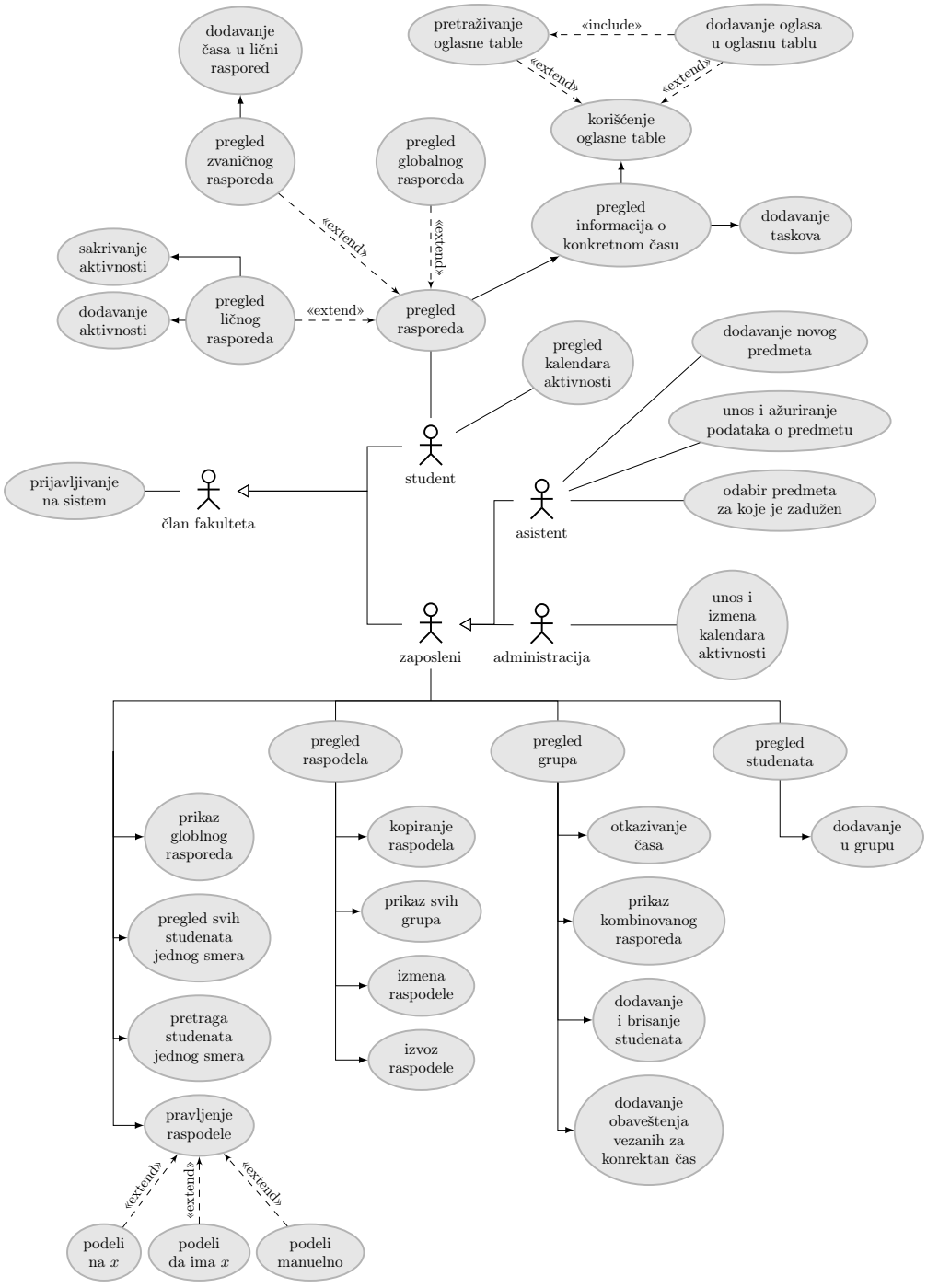
4.3. Odabir predmeta za koje je zadužen

5. Administracija

5.1. Unos i izmena kalendara aktivnosti

6.1 Dijagrami slučajeva korišćenja

UML dijagram koji prikazuje korisnike i slučajeve korišćenja aplikacije **Raspored**, kao i relacije između njih, prikazan je na slici 6.1.



Slika 6.1: UML dijagram slučajeva korišćenja.

6.2 Opis slučajeva korišćenja.

U nastavku je za svaki slučaj korišćenja dat pregled u vidu kratkog opisa i aktera koji iniciraju slučaj korišćenja.

6.2.1 Opšti

Prijavljivanje na sistem

Opis	Prijavljivanje korisnika na portal u cilju pristupa aplikaciji.
Akteri	Član fakulteta (student, asistent, administracija).

6.2.2 Student

Pregled rasporeda (globalnog, zvaničnog i ličnog)

Opis	Prikaz taba koji sadrži odgovarajući raspored.
Akteri	Student.

Dodavanje časa u lični raspored

Opis	Odabrani čas iz globalnog rasporeda se dodaje u lični raspored.
Akteri	Student.

Dodavanje aktivnosti u lični raspored

Opis	Uneta aktivnost se dodaje u lični raspored u skladu s izabranim terminom.
Akteri	Student.

Sakrivanje aktivnosti iz ličnog rasporeda

Opis	Odabrana aktivnost se sakriva iz ličnog rasporeda.
Akteri	Student.

Pregled informacija vezanih za konkretan čas

Opis	Prikaz pop-apa sa informacijama o određenom času.
Akteri	Student.

Dodavanje taskova

Opis	Dodavanje taskova (obaveštenja ili zadataka) vezanih za konkretan čas.
Akteri	Student.

Pretraživanje oglasne table

Opis	Pretraživanje informacija o studentima koji žele da promene svoj termin.
Akteri	Student.

Dodavanje oglasa na oglasnu tablu

Opis	Dodavanje novog uglasa na oglasnu tablu.
Akteri	Student.

6.2.3 Zaposleni

Pregled raspodela

Opis	Pregled svih raspodela studenata jedne godine na jednom smeru.
Akteri	Asistent, administracija.

Pregled grupa

Opis	Pregled svih grupa koje čine odgovarajuću raspodelu.
Akteri	Asistent, administracija.

Pregled studenata jedne grupe

Opis	Pregled svih grupa koje čine odgovarajuću raspodelu.
Akteri	Asistent, administracija.

Pravljenje raspodele

Opis	Kreiranje raspodele na željeni način (podeli na x, podeli da ima x, manuelno).
Akteri	Asistent, administracija.

Pregled svih studenata jednog smera

Opis	Pregled svih grupa koje čine odgovarajuću raspodelu.
Akteri	Asistent, administracija.

Pretraga studenata jednog smera

Opis	Pregled svih grupa koje čine odgovarajuću raspodelu.
Akteri	Asistent, administracija.

Prikaz globalnog rasporeda studenata jednog smera

Opis	Pregled rasporeda koji obuhvata sve studente jednog smera odnosno njihove grupe. (Trenutno imamo takve rasporede.)
Akteri	Asistent, administracija.

Kopiranje raspodele

Opis	Kopiranje trenutne raspodele u cilju stvaranja iste ili slične raspodele.
Akteri	Asistent, administracija.

Prikaz svih grupa

Opis	Prikazivanje svih grupa određene raspodele na pregledniji način od osnovnog.
Akteir	Asistent, administracija.

Izmena raspodele

Opis	Prikazivanje svih grupa određene raspodele na pregledniji način od osnovnog.
Akteri	Asistent, administracija.

Izvoz raspodele

Opis	Prikaz raspodele u `.pdf` formatu.
Akteri	Asistent, administracija.

Otkazivanje časa

Opis	Izbor časa i otkazivanje istog.
Akteri	Asistent, administracija.

Prikazivanje kombinovanog rasporeda

Opis	Prikazivanje kombinovanog (merged) rasporeda grupe odn. vreme kada su svi studenti slobodni.
Akteri	Asistent, administracija.

Dodavanje i brisanje studenata

Opis	Dodavanje i brisanje studenata iz konkretne grupe. qitem[Akteri] Asistent, administracija.

Dodavanje obevaštenja vezanih za konkretan čas

Opis	Dodavanje obevaštenja vezanih za konrektno izabran čas čas.
Akteri	Asistent, administracija.

Dodavanje u grupu

Opis	Dodavanje studenta u grupu korisnikove raspodele.
Akteri	Asistent, administracija.

6.2.4 Asistent

Dodavanje novog predmeta

Opis	Dodavanje novog predmeta u listu postojećih predmeta.
Akteri	Asistent.

Unos i ažuriranje podataka o predmetu

Opis	Unos i ažuriranje podataka o izabranom predmetu.
Akteri	Asistent.

Odabir predmeta za koje je zadužen

Opis	Odabir predmeta za koje je asistent zadužen..
Akteri	Asistent.

6.2.5 Administracija

Unos i izmena kalendara aktivnosti

Opis	Unos i izmena kalendara aktivnosti za tekući semestar.
Akteri	Administracija.

Glava 7

Pogled na logičku arhitekturu sistema

U ovoj glavi je dat pregled logičke arhitekture sistema. Ovaj pogled sadrži opis najznačajnijih klasa, njihove organizacije u pakete i podsisteme, kao i organizaciju podsistemā u slojeve. U cilju opisivanja dinamičkih aspekata arhitekture, ovaj odeljak može da uključi opise realizacije najznačajnijih slučajeva korišćenja.

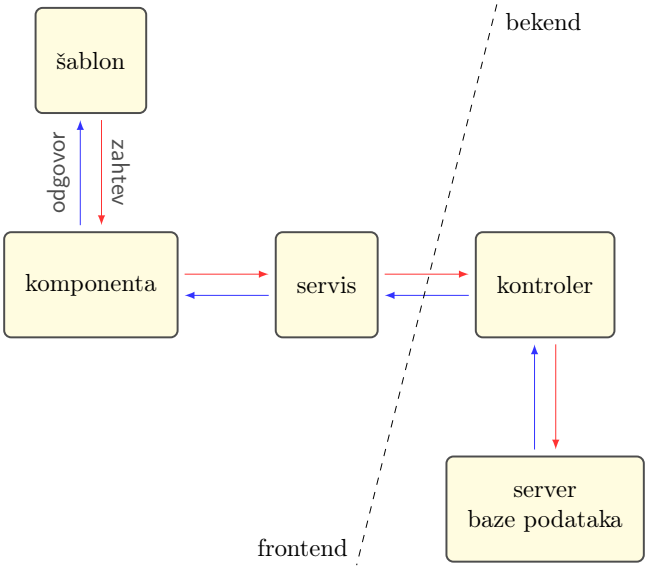
Logički pogled na aplikaciju **Raspored** obuhvata dve glavne logičke celine: serversku stranu i klijentsku stranu. Veza između podsistema se ostvaruje putem interneta. Ilustrovana je slikom 7.1.

Serverska strana predstavlja API servis za pristup bazi podataka, odnosno interakciju sa njom. Sastoji se od kontrolerā koji odgovaraju entitetima, te svaki kontroler poseduje najpre CRUD operacije za rad sa svakim entitetom, kao i akcije specifične za aplikaciju.

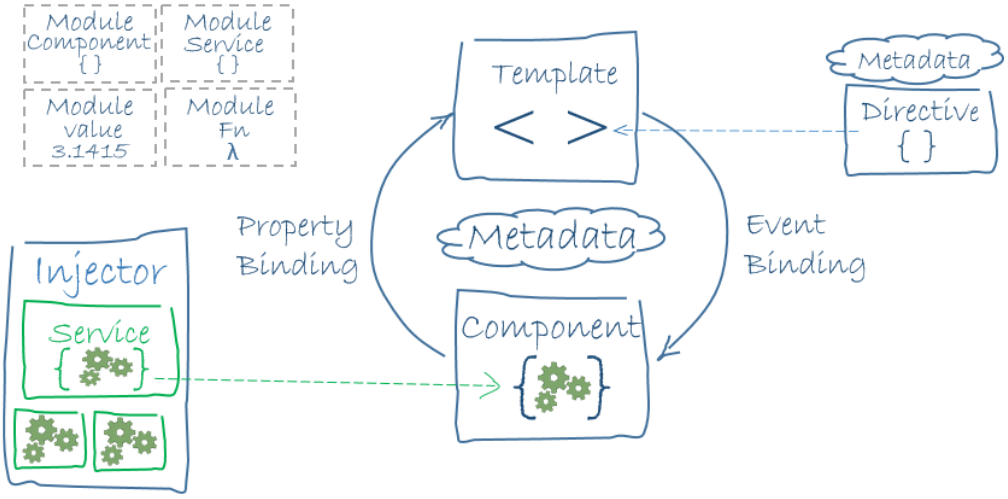
Klijentska strana je organizovana najpre preko Angular komponenti i servisā, pri čemu servisi obezbeđuju podatke komponentama tako što komuniciraju sa API-jem. Komponente predstavljaju određeni prikaz tih podataka, pri čemu omogućuju dvosmerno povezivanje (*two-way binding*) između podataka i pogleda na te podatke. Ovo je ilustrovano slikom 7.2.

7.1 Pregled arhitekture – organizacija komponenti u podsisteme

I serverska i klijentska strana se nalaze u istom projektu pa se ne može reći da su odvojeni slojevi sistema, iako oni logički to jesu.



Slika 7.1: Pogled na logičku arhitekturu sistema.



Slika 7.2: Arhitekturni dijagram koji identifikuje osam glavnih gradivnih blokova aplikacije bazirane na frejmvorku Angular 2: moduli, komponente, šabloni, meta-podaci, povezivanje podataka, servisi, direktive i ubrizgavanje zavisnosti.

7.1.1 Frontend

Tip Komponenta projekta.

Opis Frontend (tačnije Angularove komponente) se nalaze u folderu `app`. Svaku komponentu čine tri dela: logika (`.ts`), šablon za prikaz (`.html`) i stilovi (`.css`). Servisi sadrže samo logički deo za komunikaciju sa bekendom. Sve klase koje postoje na bekendu imaju svoj pandam i na frontendu, radi bolje organizacije i bezbednijeg kodiranja koristeći Angular.

Napomena Svi `.ts` fajlovi se kompajliraju u `.js` fajlove pre pokretanja, a `.css` fajlovi se dobijaju kompajliranjem `.scss` fajlova. Takođe, `.html` fajlovi predstavljaju samo šablone za komponente i ne predstavljaju gotove stranice, niti parcijalne stranice jer mogu da sadrže tagove koji nisu specificirani HTML5 specifikacijom. Nakon pokretanja projekta, frejmvork Angular će osnovu šablona realizovati validan HTML fajl, poštujući HTML5 specifikaciju.

7.1.2 Bekend

Tip Komponenta projekta.

Opis Sam projekat je ASP.NET Core 1.0, pa je organizovan kao MVC aplikacija. Modeli predstavljaju odgovarajuće objektno-orijentisano mapiranje baze podataka, a kontroleri su API servis za komunikaciju sa bazom. Pogledi se praktično ne koriste jer je ista funkcionalnost već obezbeđena frejmvorkom Angular na frontendu. Koristi se jedino `_Layout.cshtml` u kojem se, između ostalog, nalaze neophodne reference za pokretanje Angular aplikacije. On takođe predstavlja okvir jedine stranice koja postoji, `Index.cshtml`. Ova stranica ima selektor `<app>` koji služi za inicijalizovanje Angular aplikacije. Sva dalja aktivnost na sajtu se dešava na istoj stranici, a novi sadržaj se obezbeđuje AJAX pozivima, a ponašanje aplikacije kao REST veb-servisa je preneto na Angular u vidu ruta.

7.1.3 HTML5

Tip Tehnologija.

Opis Tehnologija HTML5 (*Hypertext Markup Language*) predstavlja standardizovan sistem za označavanje tekstualnih fajlova na *semantički* način, radi prikaza veb-stranica u veb-čitačima na internetu. *Izgled* stranice definiše se tehnologijom CSS3 (vidi pododeljak 7.1.4).

Napomena Pošto je **Raspored** dinamička veb-aplikacija, stranice neće imati statički definisan HTML, već će on biti generisan kao pogled od strane AngularJS-a (vidi pododeljak 7.1.8).

Veb `https://www.w3.org/TR/html5/`

7.1.4 CSS3

Tip Tehnologija.

Opis Tehnologija CSS3 (*Cascading Style Sheet*) predstavlja standarizovan sistem za opis *izgleda* veb-stranice. Softver za parsiranje CSS-a ugrađuje se u veb-čitače, što omogućava personalizaciju izgleda veb-stranice. Zbog načina na koji se selektiraju elementi, u tesnoj je vezi sa tehnologijom HTML5 (vidi odeljak 7.1.3).

Napomena Tokom razvitka aplikacije se neće direktno kreirati CSS fajlovi, već će oni biti rezultat kompajliranja SCSS fajlova (vidi pododeljak 7.1.5.

Veb `https://www.w3.org/Style/CSS/specs.en.html`

7.1.5 Sass

Tip Tehnologija.

Opis Sass (*Syntactically Awesome Stylesheets*) predstavlja proširenje CSS-a (vidi pododeljak 7.1.4) koje čini osnovni jezik moćnijim i elegantnijim. Uvodi korišćenje promenljivih, ugnježdenih pravila, miksinā, inlajn importā, itd. Kompatibilan je sa CSS sintaksom, što znači da svaki čist CSS fajl predstavlja i važeći Sass fajl. Korišćenjem jezika Sass je znatno lakše dobro organizovati strukturu CSS selektora.

Napomena Jezik Sass ima dve sintakse: staru indentovanu sintaksu i novu SCSS sintaksu. U ovom projketu se koristi SCSS.

Veb `http://sass-lang.com/`

7.1.6 JavaScript

Tip Tehnologija.

Opis Javascript je dinamički interpretatorski skript-jezik koji je razvio Netskejp. Kao takav, omogućuje lakše i brže kodiranje od jezika koji se kompajliraju kao što su C i C++, ali se zauzvrat sporije izvršavaju. Svi poznati moderni veb-pretraživači

imaju ugrađenu podršku za interpretiranje koda napisanog u JavaScriptu. JavaScript je jezik baziran na prototipovima što omogućuje programiranje u više paradigmi, uključujući objektno-orijentisanu, imeprativnu i funkcionalnu paradigmu.

Napomena U ovom projektu se neće direktno koristiti JavaScript jezik, već će `.js` fajlovi biti rezultat transpilacije `.ts` fajlova pisanih u TypeScriptu (vidi pododeljak 7.1.7).

Veb `https://developer.mozilla.org/en-US/docs/Web/JavaScript`

7.1.7 TypeScript

Tip Tehnologija.

Opis TypeScript je besplatan programski jezik otvorenog koda razvijen i održavan od strane Majkrosofta. On predstavlja strogi nadskup JavaScripta (vidi pododeljak 7.1.6) i jeziku dodaje opcionu statičku dodelu tipova podacima i objektno-orijentisano programiranje koje se bazira na klasama. Kôd TypeScripta se kompajlira u čist i jednostavan JavaScript koji se može izvršiti u bilo kom veb-pretraživaču koji podržava ECMAScript 3 (ili noviju verziju), što obezbeđuje visoku kompatibilnost sa starijim veb-čitačima.

Veb `https://www.typescriptlang.org/`

7.1.8 AngularJS 2.0

Tip Tehnologija.

Opis AngularJS 2.0 je strukturalni frejmvork za JavaScript (vidi pododeljak 7.1.6) za dinamičke veb-aplikacije. Pruža mogućnost da se proširena sintaksa HTML-a (vidi pododeljak 7.1.3) iskoristi kao šablonski jezik, što za posledicu ima jasniju podelu komponenata aplikacije po modulima. Vezivanje podataka (*data bidning*) koji koristi AngularJS eliminiše veliki deo koda koji bi inače morao da bude napisan.

Veb `https://angular.io/`

7.1.9 ASP.NET

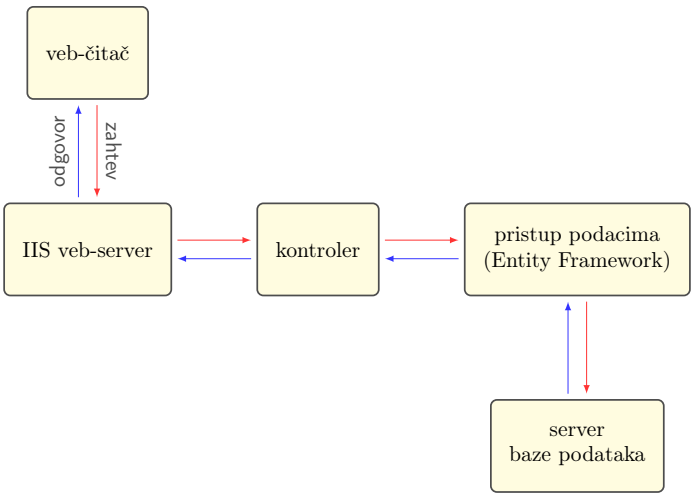
Tip Tehnologija.

Opis	ASP.NET je frejmvork otvorenog koda za razvijanje serverske strane veb-aplikacija razvijen od strane Majkrosofta koji služi za izradu dinamičkih veb-stranica, veb-aplikacija, kao veb-servisa. Omogućava lak razvoj MVC aplikacija, odnosno aplikacija koje se baziraju na projektnom obrascu Posmatrač (*Observer*). Radi zajedno a ostalim frejmvorkovima, kao na primer Entity Framework koji će biti korišćen u aplikaciji **Raspored**. On mapira bazu podataka na objektno-orijentisan model.
Veb	`http://www.asp.net/`

Glava 8

Pogled na procese

Veb-aplikacije zasnovane na ASP.NET-u imaju relativno jednostavan procesni koji je u potpunosti pod kontrolom veb-servera. Sa stanovišta projektanta ASP.NET aplikacije, nije potrebno voditi računa o načinu rada veb-servera i načinu izvršavanja koda, pa nema potrebe navoditi detalje. Pojednostavljena šema je data slikom 8.1.



Slika 8.1: Dijagram koji prikazuje procese u sistemu.

Glava 9

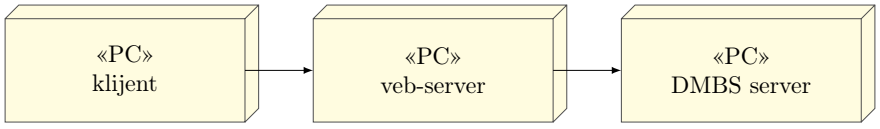
Pogled na raspoređivanje sistema

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

Na slici 9.1 dat je UML dijagram raspoređivanja aplikacije **Raspored**.

9.1 Klijent

Pristup aplikaciji **Raspored** se obavlja preko klijentskih računara na kojima se izvršava veb-pregledač. Za povezivanje klijenta sa veb-serverom se koristi infrastruktura interneta, tako da nema ograničenja u pogledu geografske lokacije klijenata.



Slika 9.1: UML dijagram koji pokazuje raspoređivanje sistema.

9.2 Veb-server

Računar na kome se izvršava veb-server opslužuje više klijenata koji pristupaju putem interneta. Na ovom računaru se izvršava proces koji realizuje sve neophodne funkcionalnosti veb-servera. U najopštijem slučaju, DBMS može biti konfigurisan tako da se izvršava na zasebnom računaru koji se nalazi u istoj lokalnoj mreži kao i veb-server.

9.3 DBMS server

DBMS server je računar na kome se izvršava SQL server. To je proces koji realizuje funkcionalnost sistema za upravljanje bazama podataka.

Glava 10

Pogled na implementaciju sistema

Pogled na implementaciju prikazuje različite aspekte bitne za implementaciju sistema. U slučaju aplikacije **Raspored**, ovaj odeljak sadrži model domena i šemu baze podataka.

10.1 Model domena

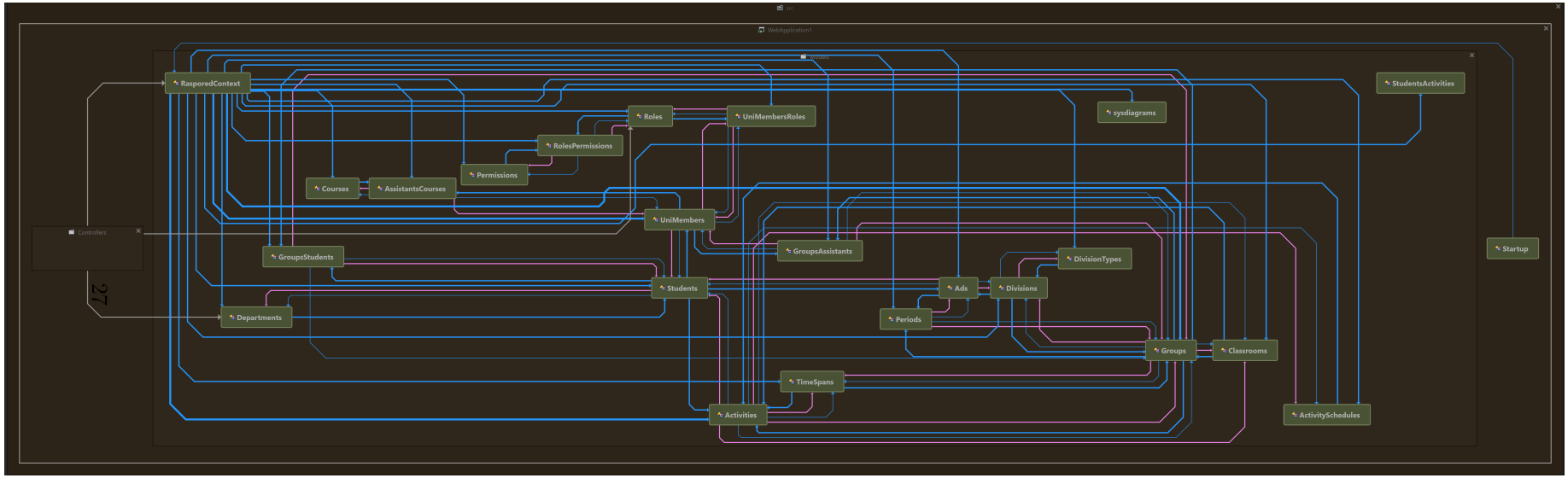
Model domena za koji se projektuje aplikacija **Raspored** je ilustrovan UML dijagramom klasa sa slike 10.1. Na slici su prikazane domenske klase, njihovi najznačajniji atributi, kao i veze koje se mogu identifikovati između njih.

10.2 Komponente korisničkog interfejsa

10.2.1 Panel za zaposlene

Dizajn korisničnog interfejsa je obuhvaćen sledećim Angular komponentama:

- **DepartmentItem** prikazuje jedan smer.
- **DepartmentsMenu** obuhvata sve smer koje sadrži naš fakultet, grupisani po godinama studija.
- **DivisionItem** prikazuje jednu podelu.
- **DivisionMenu** obuhvata sve podele jednog smera.
- **GroupItem** prikazuje jednu grupu studenata.



Slika 10.1: UML dijagram klasa.

- **GroupMenu** prikazuje sve grupe jedne raspodele.
- **StudentItem** prikazuje podatke o studentu.
- **StudentMenu** prikazuje sve studente jedne grupe.

Servisi koji obezbeđiuju podatke gorenavedenim komponentama su:

- **DepartmentsService**
- **DivisionsService**
- **GroupsService**

Admin panel

Ovaj odeljak je deo panela za zaposlene ali je logički odvojen od ostatka panela. Sastoji se od sledećih Angular komponenti:

- **AdminDepartments**
- **AdminCourses**
- **Classrooms**

Servisi koji obezbeđuju podatke gorenavedenim komponentama (pored **DepartmentsService** koji je već opisan) su:

- **CoursesService**
- **ClassroomService**

10.2.2 Panel za studente

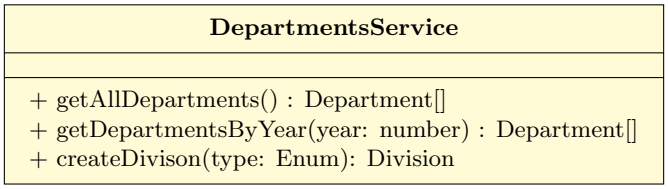
Dizajn korisničkog interfejsa se sastoji od sledećih Angular komponenti:

- **Schedule**.
- **Class** (10.20).
- **ClassDetail** (10.21).
- **Ad**.

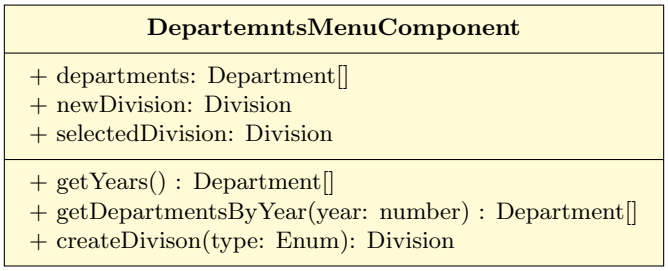
Kao i servise za časove: **ClassService**, **ScheduleService** i **AdService** koji poseduju osnovne CRUD operacije.

10.3 Šema baze podataka

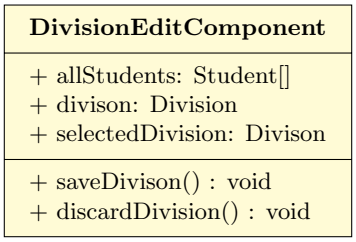
Detaljna šema baze podataka je prikazana na slici 10.22. Baza podataka i dijagram su kreirani korišćenjem SQL Servera.



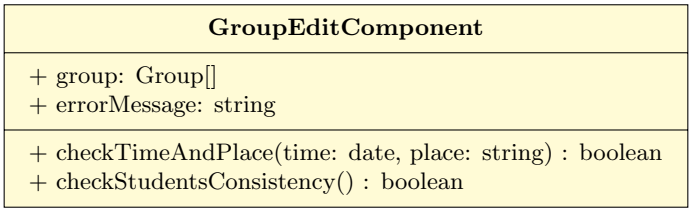
Slika 10.2: UML dijagram klase **DepartmentsService**.



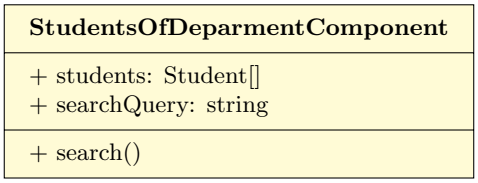
Slika 10.3: UML dijagram klase **DepartemntsMenuComponent**.



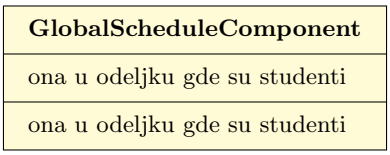
Slika 10.4: UML dijagram klase **DivisionEditComponent**.



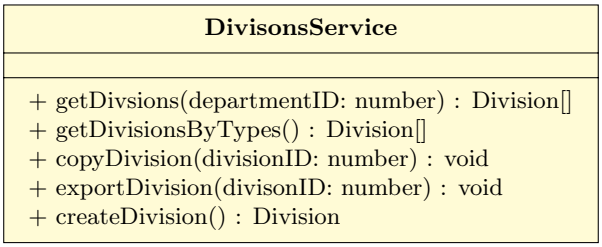
Slika 10.5: UML dijagram klase **GroupEditComponent**.



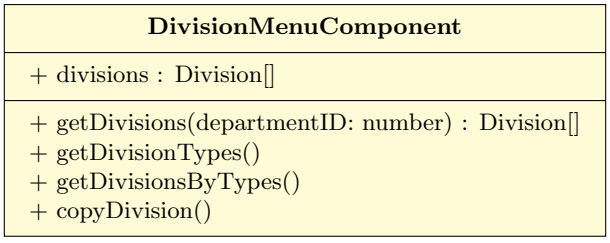
Slika 10.6: UML dijagram klase **StudentsOfDeparmentComponent**.



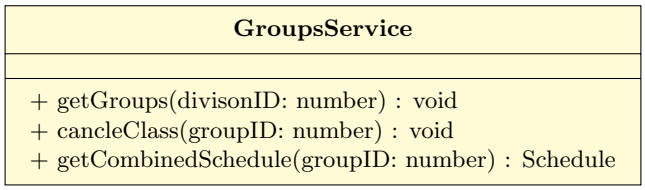
Slika 10.7: UML dijagram klase **GlobalScheduleComponent**.



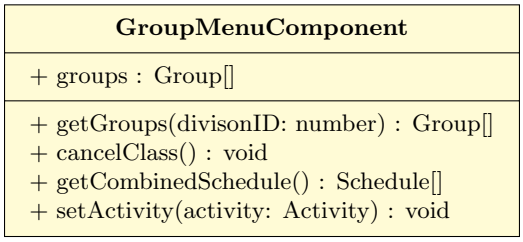
Slika 10.8: UML dijagram klase **DivisonsService**.



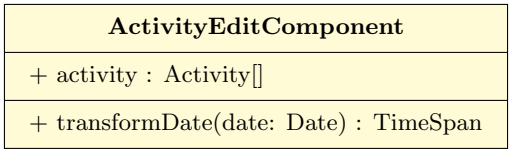
Slika 10.9: UML dijagram klase **DivisionMenuComponent**.



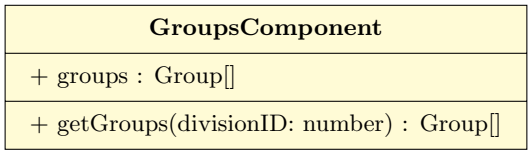
Slika 10.10: UML dijagram klase **GroupsService**.



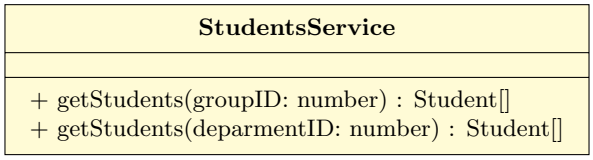
Slika 10.11: UML dijagram klase **GroupMenuComponent**.



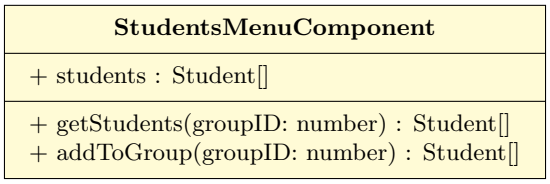
Slika 10.12: UML dijagram klase **ActivityEditComponent**.



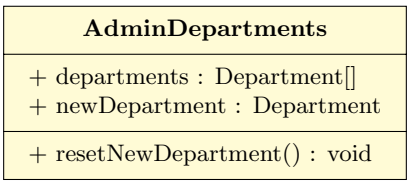
Slika 10.13: UML dijagram klase **GroupsComponent**.



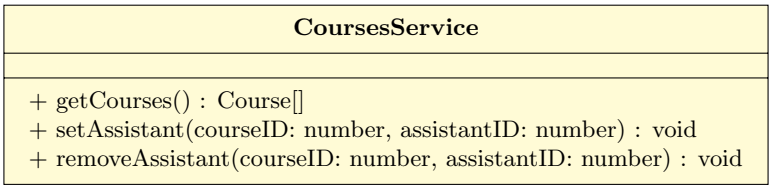
Slika 10.14: UML dijagram klase **StudentsService**.



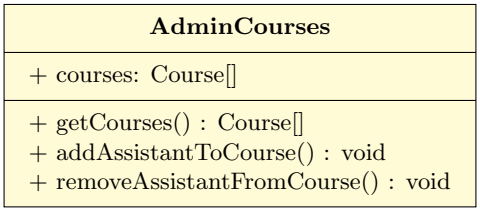
Slika 10.15: UML dijagram klase **StudentsMenuComponent**.



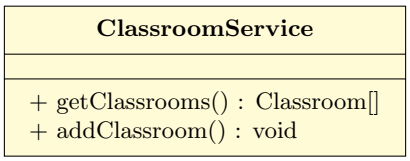
Slika 10.16: UML dijagram klase **AdminDepartments**.



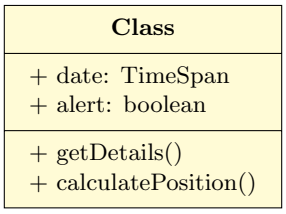
Slika 10.17: UML dijagram klase **CoursesService**.



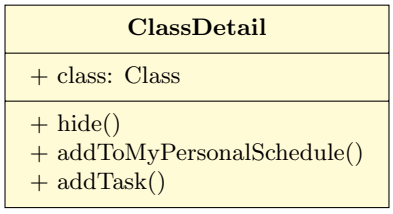
Slika 10.18: UML dijagram klase **AdminCourses**.



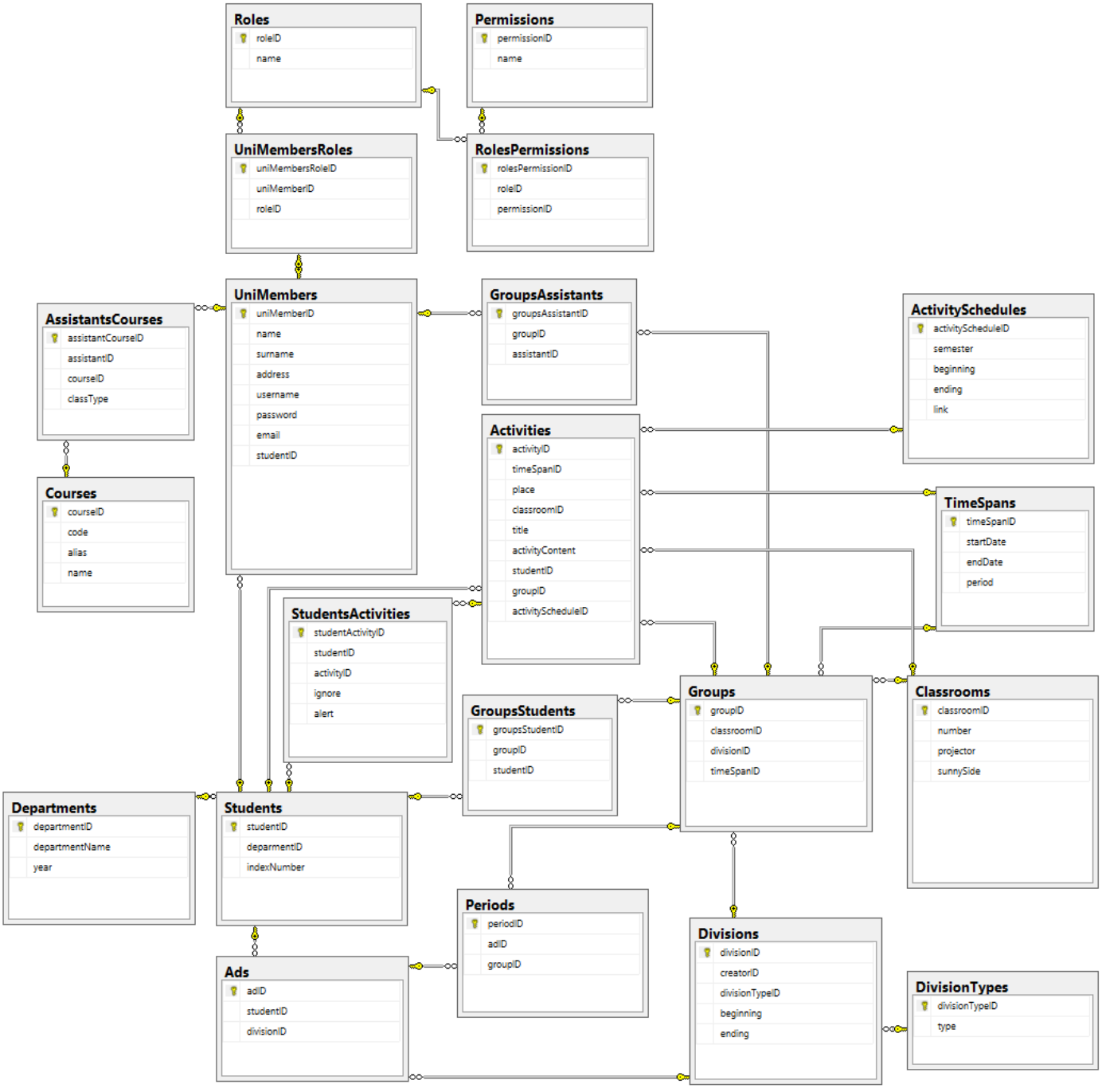
Slika 10.19: UML dijagram klase **ClassroomService**.



Slika 10.20: UML dijagram klase **Class**.



Slika 10.21: UML dijagram klase **ClassDetail**.



Slika 10.22: Relacioni dijagram baze podataka.

Glava 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 simultani pristup broja korisnika koji treba biti bar 70% od ukupnog broja studenata.

(2) Vreme potrebno za pristupanje bazi podataka u cilju izvršenje nekog upita ne sme da bude veće od 5 sekundi.

Zahtevane performanse su zadovoljene izborom tehnologija na kojima će sistem biti razvijen i definisane hardverske platforme [5].

Glava 12

Kvalitet

Izabrana arhitektura softvera podržava zahteve u pogledu dostupnosti i srednjeg vremena između otkaza specificirane u zahtevima u pogledu pouzdanosti [5]: **Raspored** će biti dostupan 24 časa dnevno, 7 dana u nedelji. Vreme kada portal nije dostupan ne sme da pređe 3%. Srednje vreme između dva sukcesivna otkaza ne sme da padne ispod 120 sati.