|  |  |  |
| --- | --- | --- |
|  | UNIVERZITET U NOVOM SADU  **FAKULTET TEHNIČKIH NAUKA U NOVOM SADU** |  |

Marko Stanić

**Naslov rada**

Diplomski rad

- Osnovne akademske studije -

Novi Sad, 2020.

|  |  |  |
| --- | --- | --- |
|  | UNIVERZITET U NOVOM SADU  **FAKULTET TEHNIČKIH NAUKA**  21000 NOVI SAD, Trg Dositeja Obradovića 6 | Datum: |
|  |
| **ZADATAK ZA IZRADU DIPLOMSKOG (BACHELOR) RADA** | List: |
| 1/1 |

*(Podatke unosi predmetni nastavnik - mentor)*

| Vrsta studija: | **Osnovne akademske studije** |
| --- | --- |
| Studijski program: | **Softversko inženjerstvo i informacione tehnologije** |
| Rukovodilac studijskog programa: | **prof. dr Miroslav Zarić** |

|  |  |  |  |
| --- | --- | --- | --- |
| Student: | **Marko Stanić** | Broj indeksa: | **SW 32/2016** |
| Oblast: | **Sistemi bazirani na znanju** | | |
| Mentor: | **prof. dr Siniša Nikolić** | | |
| NA OSNOVU PODNETE PRIJAVE, PRILOŽENE DOKUMENTACIJE I ODREDBI STATUTA FAKULTETA IZDAJE SE ZADATAK ZA DIPLOMSKI RAD, SA SLEDEĆIM ELEMENTIMA:   * problem – tema rada; * način rešavanja problema i način praktične provere rezultata rada, ako je takva provera neophodna; * literatura | | | |

**NASLOV DIPLOMSKOG (BACHELOR) RADA:**

|  |
| --- |
|  |

**TEKST ZADATKA:**

|  |
| --- |
|  |

|  |  |
| --- | --- |
| Rukovodilac studijskog programa: | Mentor rada: |
|  |  |

|  |
| --- |
| Primerak za:  - Studenta;  - Mentora |

Sadržaj

[1. Uvod 4](#_Toc50546222)

[2. Stanje u oblasti 5](#_Toc50546223)

[2.1 MyFitnessPal 5](#_Toc50546224)

[2.2 Cronometer 6](#_Toc50546225)

[2.3 Diskusija 7](#_Toc50546226)

[3. Tehnologije 9](#_Toc50546227)

[3.1 React i React Native 9](#_Toc50546228)

[3.2 Spring 10](#_Toc50546229)

[3.3 REST 12](#_Toc50546230)

[3.4 Drools 12](#_Toc50546231)

[4. Specifikacija sistema 14](#_Toc50546232)

[4.1 Dijagram klasa 14](#_Toc50546233)

[4.2 Dijagram slučajeva korišćenja 16](#_Toc50546234)

[5. Implementacija 19](#_Toc50546235)

[5.1 Serverska aplikacija 19](#_Toc50546236)

[5.2 Drools aplikacija 22](#_Toc50546237)

[5.3 Administratorska aplikacija 30](#_Toc50546238)

[5.4 Mobilna aplikacija 30](#_Toc50546239)

[6. Prikaz implementiranog rešenja 31](#_Toc50546240)

[6.1 Administratorska aplikacija 31](#_Toc50546241)

[6.2 Mobilna aplikacija 32](#_Toc50546242)

[7. Zaključak 36](#_Toc50546243)

[Literatura 37](#_Toc50546244)

# Uvod

Ishrana je potreba svakog čoveka, a to je proces gde ljudsko biće putem konzumacije hrane unosi sve potrebne elemente, materije u organizam da bi njegovo telo kompletno funkcionisalo, kao i da bi zadovoljio glad i svoja čula. Ishrana savremenog čoveka je veoma loša, najnezdravija u istoriji čovečanstva, iako tehnologija, nauka i medicina napreduju iz dana u dan. Potrebna su novija rešenja koja će ljudima obezbediti bolju edukaciju i promenu loših navika vezanih za ishranu.

Sistemi bazirani na znanju mogu biti ključno rešenje za automatizaciju nekih procesa. Ovi sistemi simuliraju ljudsko rezonovanje u određenom domenu. Jedan tip sistema baziranih na znaju su ekspertski sistemi. Dizajnirani su za rešavanje problema pomoću znanja eksperta, najčešće reprezentovanog preko *if-then* pravila umesto kroz konvencionalno rešenje u programskom kodu [[1](#_Literatura_1)]. Koriste se kada je domen problema uzak, jasan i dobro definisan, kada znanje možemo predstaviti činjenicama i pravilima nad njima i ako je izlaz iz sistema moguće obrazložiti [[2](#_Literatura_1)]. Za razliku od veštačke inteligencije i mašinskog učenja akcije su određene pravilima koje je napisao čovek (ekspert) [[3, 4](#_Literatura_1)]. Primeri sistema u kojima su se za predstavu ekspertsko znanja iskoristila formalno definisana pravila može se videti na [X,X]U ovom radu će biti opisan razvoj sistema baziranog na pravilima koji omogućava korisnicima računanje idealnog dnevnog unosa kalorija na osnovu fizičkih karakteristika kao i praćenje dnevnika ishrane sa preporukama. Sistem je namenjen korisnicima između 15 i 80 godina.

Motivacija za razvijanje ovakvog sistema je povećanje broja bolesti uzrokovanih nezdravom ishranom. Prema podacima Svetske zdravstvene organizacije gojaznost je danas skoro tri puta veća nego 1975. godine. U 2016. godini 1,9 milijardi odraslih (osamnaestogodišnjaka i starijih) su imali prekomernu težinu. Od njih je 650 miliona gojazno [[5](#_Literatura)].

Cilj je da se pokuša pomoći korisnicima koji žele da se hrane zdravije i kvalitetnije da promene svoje navike. Treba podići svest o važnosti pravilne ishrane i da je ljudi prilagode svojim potrebama kroz dnevnik ishrane.

U nastavku rada će biti opisano stanje u oblasti, korišćene tehnologije, detalji implementacije i uputstvo za korišćenje.

# Stanje u oblasti

Praćenje ishrane i računanje kalorija donosi određene pogodnosti, bez obzira da li je cilj povećanje ili smanjivanje telesne težine. Pomenuti aktivnosti doprinose donošenju odluke da se pređe na zdraviju ishranu i samim tim se poveća energija organizma i dostigne željena težina. Prethodno će uticati na poboljšanje fizičkog izgleda, što dovodi do povećanja zadovoljstva i odlučnosti da se nastavi tim putem. Istraživanje pokazuje da ljudi koji prate unos kalorija gube više na težini i imaju veću šansu da ostanu dosledni [[6, 7](#_Literatura)].

Svakako da je alat koji će tako nešto olakšati pametni telefon, što zbog svoje sveprisutnosti tako i zbog mogućnosti različitih rešenja. Iz tog razloga će biti razmatrane aplikacije za *Android* i *iOS* uređaje. Ovakve aplikacije prvenstveno treba da utiču da se više razmišlja o ishrani i velikom uticaju na zdravlje. Omogućavajući planiranje unapred doprineće se boljem usklađivanju dnevnih potreba.

Pored svih dobrih strana, smatra se da ovakav tip aplikacija ponekad može doneti negativno iskustvo svojim korisnicima. U naučnom radu, koji se bavi ulogom aplikacija za zdravu ishranu i fitness na formiranje neadkvatne ishrane, pokazano je da otprilike jedna trećina ispitanika prijavila da su prestali sa korišćenjem aplikacije navodeći da im je bila previše zahtevna i nedostajalo im je motivacije za dugoročno korišćenje [[8](#_Literatura)]. Ispitanici su objasnili kako je pritisak da se ostvari željeni cilj doveo do negativnog iskustva i ponašanja, kao što je preskakanje obroka. Zbog toga bi trebalo da aplikacija bude što jednostavnija, da ne oduzima previše vremena i da ne omogućava postavljanje nerealnih ciljeva, čije neispunjavanje će dovesti do pomenutih problema.

U narednom delu će biti prikazane najpopularnije aplikacije i opisane funkionalnosti koje pružaju kao i njihove prednosti i mane. Kriterijum za odabir sledećih aplikacija je bio popularnost i broj preuzimanja na *AppStore* i *GooglePlay* prodavnicama.

## 2.1 MyFitnessPal

MyFitnessPal je trenutno jedna od najpopularnijih aplikacija za računanje kalorija, sponzorisana od strane *Under Armour*-a. Prati težinu i računa preporučeni dnevni unos kalorija. Sadrži i dnevnih ishrane i vežbanja. Registrovanje se može izvršiti koristeći Facebook nalog.

Na početnoj stranici se može videti preporučeni dnevni unos kalorija, broj unetih kalorija kroz hranu, broj potrošenih kalorija kroz trening i koliko je još preostalo da se unese kalorija do kraja dana (slika 2.1.1). Dnevnik ishrane sadrži unete obroke podeljene u četiri grupe: doručak, ručak, večera i užina (slika 2.1.2). Svaki obrok ima naziv, broj kalorija i veličinu porcije. Obroke je moguće dodavati i skeniranjem bar koda na pakovanju. Recepti se mogu pregledati i podeljeni su u nekoliko kategorija: za jačanje imuniteta, obroci pre treninga, obroci posle treninga, visoko proteinski obroci, vegeterijanski, bez glutena, ispod 500 kalorija... (slika 2.1.3). Moguće je izabrati željeni plan prema preferencijama, dužini trajanja i nivou težine (slika 2.1.4). Pojedine mogućnosti su dostupne samo u plaćenoj Premium verziji koja košta 49,99 dolara godišnje.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| Slika 2.1.1 Početna stranica | Slika 2.1.2 Dnevnik ishrane | Slika 2.1.3 Recepti | Slika 2.1.4 Planovi |

## 2.2 Cronometer

Cronometer nudi sličan skup funkcionalnosti kao i prethodna aplikacija sa nešto drugačijim korisničkim interfejsom. Registracija nije moguća putem spoljnih servisa kao što su Gmail ili Facebook. Potrebni podaci za registraciju su pol, datum rodjenja, visina i težina. Korisnik može birati da li želi da održi, poveća ili smanji telesnu težinu.

Na početnoj stranici je dnevnik ishrane (slika 2.2.1) koji sadrži listu dodatih obroka sa osnovnim informacijama. Korisnik može sam kreirati recepte i dodavati namirnice. Obroci se dodaju skeniranjem bar koda ili pretragom po nazivu. Omogućen je i prikaz izveštaja i grafikona (slika 2.2.2) na kojima su prikazane informacije o broju unetih kalorija, sa raspodelom makro nutrijenata, po danima. Profil korisnika (slika 2.2.3) sadrži dodatne podatke kao što su indeks telesne mase, bazalni metabolizam i procenjena vrednost telesne masnoće. Plaćene verzije GOLD i PROFESSIONAL sa cenama od 6,99 i 24,95 dolara mesečno pružaju dodatne funkcionalnosti kao što su predlozi obroka i isključivanje reklama.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Slika 2.2.1 Dnevnik ishrane | Slika 2.2.2 Grafikoni i izveštaji | Slika 2.2.3 Profil korisnika |

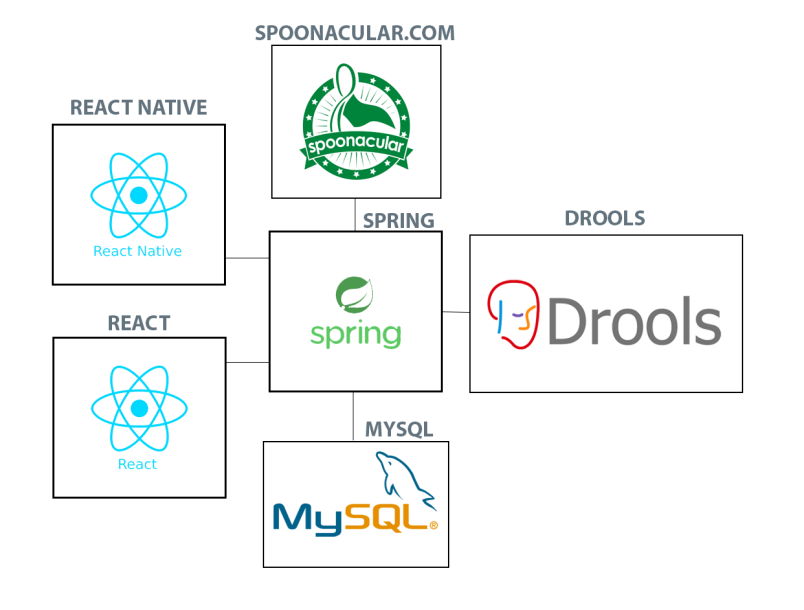
## 2.3 Diskusija

Analizom sličnih rešenja i pregledom sadržaja opisanih aplikacija može se zaključiti šta je od funkcionalnosti potrebno za adekvatno korišćenje. U nastavku će se sagledati pozitivne karakteristike i neki nedostaci jedne u odnosu na drugu aplikaciju.

Broj koji predstavlja ukupan unos dnevnih kalorija i koliko je još neophodno uneti kalorija do kraja dana se nalazi na početnom ekranu obe aplikacije. Grafički prikaz kod Cronometer-a u formi kružnog dijagrama je čitljiviji jer se vizuelno može primetiti koji je procenat unetih kalorija. MyFitnessPal na početnom ekranu sadrži samo brojeve a da bi se došlo do dijagrama nephodno je proći više koraka kroz navigaciju. Prikaz unetih obroka sadrži podatke kao što su naziv obroka količina kalorija i broj porcija. MyFitnessPal odvaja unete obroke u grupe: doručak, ručak, večera i užina. Takođe pruža dodatne informacije na primer ukoliko obrok sadrži visok nivo šećera, ispod naziva će se prikazati maksimalan dnevni unos šećera. Cronometer ima mogućnost preporuke obroka ali samo u plaćenoj Gold verziji. Takođe je mana što ne postoji probni period za plaćenu verziju, dok MyFitnessPal ima probni period od 30 dana. Obe aplikacije imaju prikaz dijagrama na kom se može pratiti telesna težina kroz vreme. Prednost MyFitnessPal aplikacije u odnosu na Cronometer su i obaveštenja ukoliko korisnik zaboravi da unese obrok.

# Tehnologije

Sistem se sastoji od klijentske strane, serverske strane, baze podataka, sistema baziranog na pravilima i eksternog API-ja spoonacular.com [[9](#_Literatura)] (slika 5.1). Za razvoj klijentskih aplijacija je korišćen *React* i *React Native* [[10](#_Literatura)], za serversku aplikaciju *Spring* [[11](#_Literatura)] koristeći *REST* [[12](#_Literatura)], za bazu podataka *MySQL* [[13](#_Literatura)] i za sistem baziran na znanju *Drools* [[14](#_Literatura)]



Slika 5.1 Arhitekura aplikacije

## 3.1 React i React Native

React je JavaScript biblioteka otvorenog koda za izgradnju korisničkih intefejsa, razvijen od strane Facebook-a [[15](#_Literatura)]. React aplikacija je sagrađena od komponenti koje omogućavaju podelu korisničkog interfejsa na delove koji su nezavisni i mogu se ponovo koristiti. Svaka komponenta kao povratnu vrednost ima React element koji opisuje šta treba da se pojavi na ekranu, a opciono može primati i parametre (props). Komponente se mogu pisati kao *JavaScript* klase ili funkcije. *React* i *React Native* koriste *JSX*, sintaksu koja omogućava pisanje elemenata unutar *JavaScript* koda.

*React Native* je radni okvir otvorenog koda za kreiranje mobilnih aplikacija.. Koristi se za razvoj aplikacija za *Android* [[16](#_Literatura)], *iOS* [[17](#_Literatura)], *Web* i *UWP* [[18](#_Literatura)] omogućujući programerima da koriste *React* zajedno sa ugrađenim mogućnostima određene platforme. Pisanjem *JavaScript* koda pristupa se *API*-u platforme i opisuje izgled i ponašanje korisničkog interfejsa koristeći *React* komponente.

Za *Android* i *iOS* *View* je osnovni gradivni blok korisničkog interfejsa. Predstavlja mali pravougaoni element na ekranu koji se može koristiti za prikazivanje teksta teksta, slika ili za interakciju sa korisnikom. Pored zajedničkih komponenti možemo koristiti i komponente specifične za platformu. *React Native* omogućava dva načina za organizovanje koda za različite platforme: korišćenjem *Platform* modula i korišćenjem ekstenzija za datoteke specifične za platformu.

## 3.2 Spring

*REST API* je implementiran pomoću *Spring* radnog okvira. *Spring* uključuje oko 20 modula koji pružaju niz servisa (slika 3.2.1).



Slika 3.2.1 Spring moduli [[19](#_Literatura)]

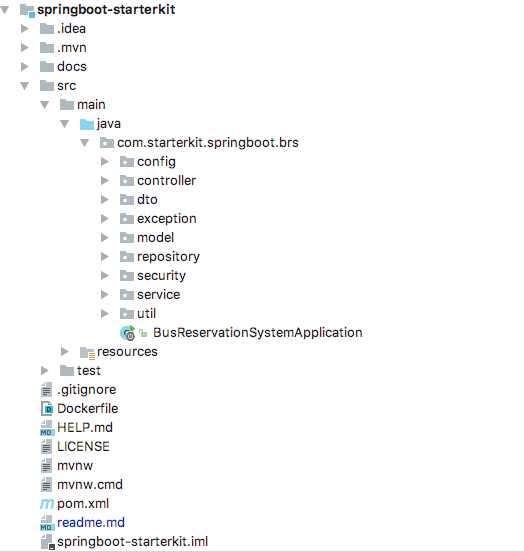
*Core* i *Beans* moduli pružaju osnovne delove radnog okvira, uključujući *IoC* (*Inversion of Control*) i *Dependency Injection* [[20](#_Literatura)]. Dependency Injection je koncept u kojem objekti dobijaju druge potrebne objekte. Predstavlja jednu formu šireg pojma koji se naziva Inversion of Control.

*Data Acces* / *Integration* modul u osnovi pruža podršku za interakciju sa bazom podataka. *Hybernate* [[21](#_Literatura)] je alat za objektno-relaciono mapiranje za *Java* programski jezik. Primarna karakteristika je mapiranje *Java* klasa na tabele u bazi podataka i mapiranje *Java* tipova podataka na *SQL* tipove podataka. Takođe pruža mogućnost upita i pronalaženja objekata i tako značajno smanjuje potrebno vreme za razvoj.

*Web* modul omogućavaja podršku za kreiranje web aplikacija. Pruža osnovne *web*-orijentisane funkcionalnosti kao što su prenos datoteka i inicijalizovanje *IoC* kontejnera. Sadrži i *model-view-controller* (*MVC*) implementaciju za *web* aplikacije.

Test modul podržava testiranje *Spring* komponenti pomoću alata *Junit* [[22](#_Literatura)] ili *TestNG* [[23](#_Literatura)]. Omogućava konzistentno učitavanje *ApplicationContext*-a i keširanje, kao i kreiranje *mock* objekata koji se mogu koristiti za izolovano testiranje.

Struktura organizovanja paketa u Spring projektu može biti proizvoljna. Međutim najbolja praksa je da se projekat podeli u module, koji će sadržati više slojeva (slika 3.2.2). *Config* paket sadrži klase koje služe za konfiguraciju projekta. *Controller* sadrži klase anotirane sa @RestController koje predstavljaju krajnje tačke za prihvatanje *HTTP* zahteva. U DTO (*Data Transfer Object*) paketu se nalaze klase čiji objekti se koriste za prenos podataka. *Exception* paket sadrži klase za implementaciju rukovanja izuzecima.



Slika 3.2.2 Struktura *Spring* projekta

Za kreiranje *Spring* projekta korišćen je *Spring Initializer* koji na jednostavan način omogućava da se konfiguriše projekat. Postoji izbor alata za izgradnju sistema, verzije i dodavanje zavisnosti (*Dependencies*)

## 3.3 REST

*Representational state transfer* (REST) stil softverske arhitekture koji definiše skup ograničenja za kreiranje Web servisa. Da bi servis bio označen kao RESTful treba da budu zadovoljeni sledeći principi:

1. *Client-server*: Razdvajanjem korisničkog interfejsa i poslovne logike sistema povećava se prenosivost korisničkog interfejsa širom različitih platformi i povećava se mogućnost skaliranja.
2. *Stateless*: Svaki zahtev od klijenta do servera mora sadržati sve informacije neophodne za razumevanje zahteva. Stanje se čuva isključivo na klijentskoj strani a ne na serveru.
3. *Cacheable*: Keširanje je mogućnost čuvanja kopija često pristupanih podataka na nekoliko mesta duž putanje zahtev-odgovor.
4. *Uniform interface*: REST je definisan sa 4 ograničenja interfejsa. Za identifikaciju resursa se koristi URI standard. Za manipulaciju resursima se koriste HTTP metode i URI. Samoopisujuće poruke su one koje sadrže sve informacije koje je potrebno da bi ih primalac razumeo, tj. ne bi trebalo da postoje dodatne informacije u posebnoj dokumentaciji ili poruci. REST API pruža hiperveze (*hyperlinks*) sa svakim odgovorom koji se povezuju sa drugim srodnim resursima.
5. *Layered system*: Slojevita arhitektura omogućava da komponente ne mogu da „vide“ izvan neposrednih slojeva sa kojima komuniciraju.
6. *Code on demand*: Jedino neobavezno ograničenje, koje znači da server može proširiti funkcionalnost klijenta u vremenu izvođenja (*runtime*), slanjem koda koji bi trebalo da se izvršava

## 3.4 Drools

*Drools* je sistem koji koristi pristup zasnovan na pravilima za implementaciju ekspertskih sistema. Napisan je u *Java*-i i otvorenog je koda, podržan od strane *Jboss*-a i *Red Hat*-a. Za razumevanje principa rada *Drools*-a neophodno je objasniti osnovne pojmove sistema baziranih na pravilima.

Činjenice predstavljaju delove podataka koji se obrađuju i skladište se u radnoj memoriji. Mogu se dodavati menjati ili uklanjati iz radne memorije. *Rule engine* izvršava određene procedure koristeći bazu znanja i činjenice u radnoj memoriji kao ulaz. Bazu zanja čine mašinski čitljiva pravila u kojima se nalazi domensko znanje za problem koji se rešava i na osnovu kojih se vrši rezonovanje. Proces u kom se proverava zadovoljenost činjenica za pravila se naziva *pattern matching*, a za implementaciju je korišćen *Rete* Algoritam.

Pravila se pišu uz pomoć *DRL* jezika u okviru .drl fajlova čija struktura se može videti na slici 3.4.1.



Slika 3.4.1 Stukura *Drools* pravila [[24](#_Literatura)]

Leva strana pravila predstavlja uslove koji treba da se zadovolje da bi pravilo bilo izvršeno. Desna strana pravila sadrži akcije koje će se izvršiti kada se aktivira pravilo. Ove akcije mogu izmeniti stanje radne memorije tj. dodavati, menjati ili brisati činjenice.

Za aktiviranje pravila neophodno je da svi uslovi u levoj strani pravila budu zadovoljeni. Međutim kada više pravila zadovoljava sve uslove potrebno je odrediti prioritet izvršavanja pravila. Razrešavanje konflikta je postupak kojim se određuje koje pravilo će se prvo aktivirati. Metode za razrešavanje konflikata su definisanje prioriteta, izbor prema poziciji pravila ili specifičnosti pravila, definisanje metapravila ili izbor pravila na osnovu činjenica koje su najskorije dodate.

*Drools* podržava ulančavanje unapred (*forward chaining*) i ulančavanje unazad (*backward chaining*). Ulančavanje unapred je vođeno podacima, zna se samo koje je trenutno stanje radne memorije, ali ne i kojim putem će krenuti zaključivanje i koji će zaključak biti. Proces se odvija tako što se u svakom koraku izvršava jedno pravilo i zaustavlja kada nema više pravila koja zadovoljavaju činjenice u radnoj memoriji. Ulančavanje unazad je vođeno upitom, kreće se od zaključka ka činjenicama. Rule engine pokušava da potvrdi hipotezu iz radne memorije prolazeći kroz niz pravila. Ukoliko takva pravila ne postoje hipoteza je oborena.

# Specifikacija sistema

Aplikacija je namenjena korisnicima za računanje kalorija i praćenje dnevnika ishrane. Jedna od glavnih funkcionalnosti je i preporuka obroka na dnevnom nivou koji zadovoljavaju potreban broj kalorija. Ova funkcionalnost je dodata kako bi se poboljšalo korisničko iskustvo u korišćenju aplikacije.

Da administrator ne bi ručno unosio obroke u sistem upotrebljen je *spoonacular.com*. Ovaj *API* omogućava pretragu preko pet hiljada obroka sa potpunim nutritivnim informacijama. Pokriva veliki broj slučajeva korišćenja kao što su analiza namirnica, generisanje planova ishrane, uputsvo o vinima, vizuelizacija sastojaka za obrok i nutritivnih vrednosti...

Prilikom pretrage obroka kao i generisanja dnevnog plana ishrane moguće je kao parametar navesti tip ishrane. Neki od podržanih tiopva ishrane su vegeterijanska u kojoj se ne koristi meso ili veganska u kojoj se pored mesa ne koriste ni jaja, mlečni proizvodi i med. Celokupna lista podržanih tipova ishrane može se naći u zvaničnoj dokumentaciji *spoonacular.com* *API*-a. Prilagođavanje tipa ishrane je omogućeno samo naprednim korisnicima zbog pretpostavke o većem domenskom znanju.

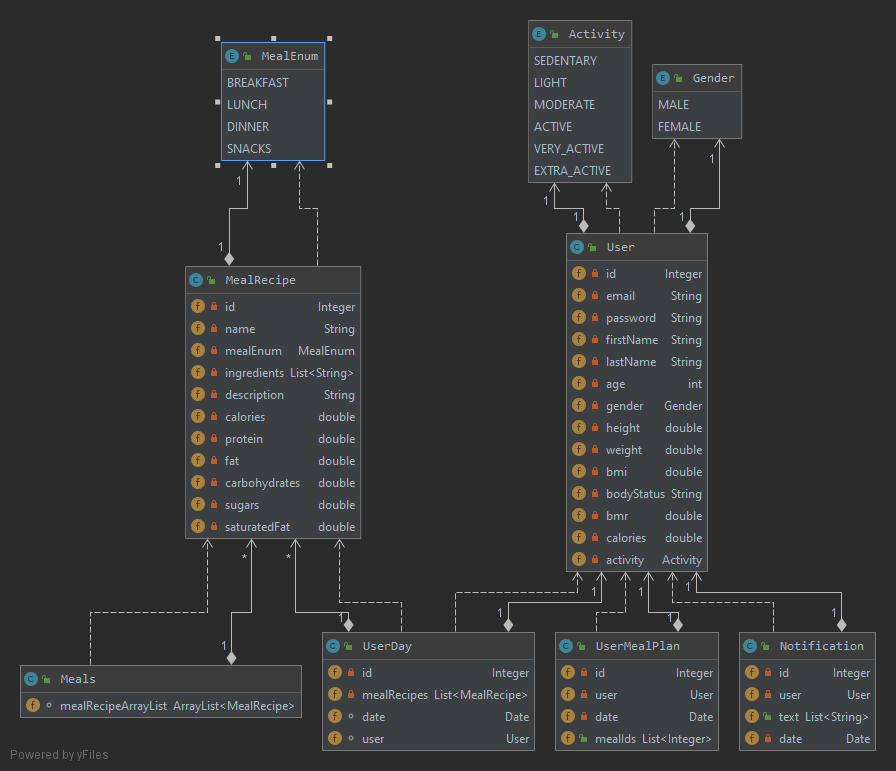
U nastavku će biti objašnjeni slučajevi korišćenja, dijagram klasa i dijagram aktivnosti.

## 4.1 Dijagram klasa

Model aplikacije čini 6 klasa i 3 enumeracije:

* User
* UserDay
* UserMealPlan
* MealRecipe
* Meals
* Notification
* Activity
* Gender
* MealEnum

Veze između klasa se mogu videti na slici 4.1.1. Svaka klasa je mapirana na odgovarajuću tabelu u bazi podataka. Klasa UserDay sadži sve obroke koje je korisnik uneo za određeni dan. Obrok je predstavljen klasom MealRecipe koja sadrži osnovne nutritivne informacije. Klasa Notification sadrži listu poruka koja se prikazuju korisniku za određeni dan.

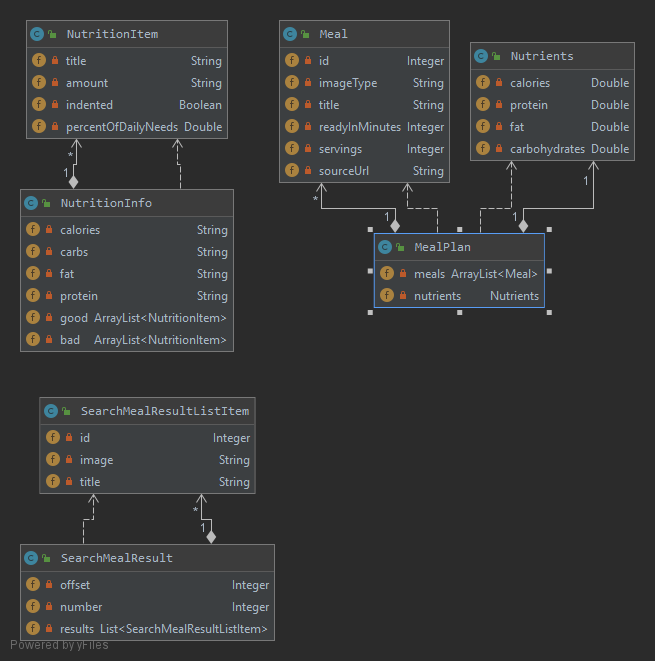


Slika 4.1.1 Dijagram klasa

Za komunikaciju sa spoonacular.com servisom je korišćeno 7 klasa:

* Meal
* MealPlan
* Nutrients
* NutritionInfo
* NutritionItem
* SearchMealResult
* SearchMealResultListItem

Veze između klasa mogu se videti na slici 4.1.2. Klasa Nutrients sadrži informacije o broju kalorija, masti, proteina i ugljenih hidrata, dok je klasa NutritionInfo proširena i listom dobrih (vitamini, vlakna, gvožđe, kalcijum, magnezijum...) i loših stavki (zasićene masti, šećeri, holesterol...). Klase SearchMealResult i SearchMealResultListItem se koriste za pretragu, koja podržava unos različitih parametara, za pronalazak nekih od stotine hiljada različitih recepata za obroke.



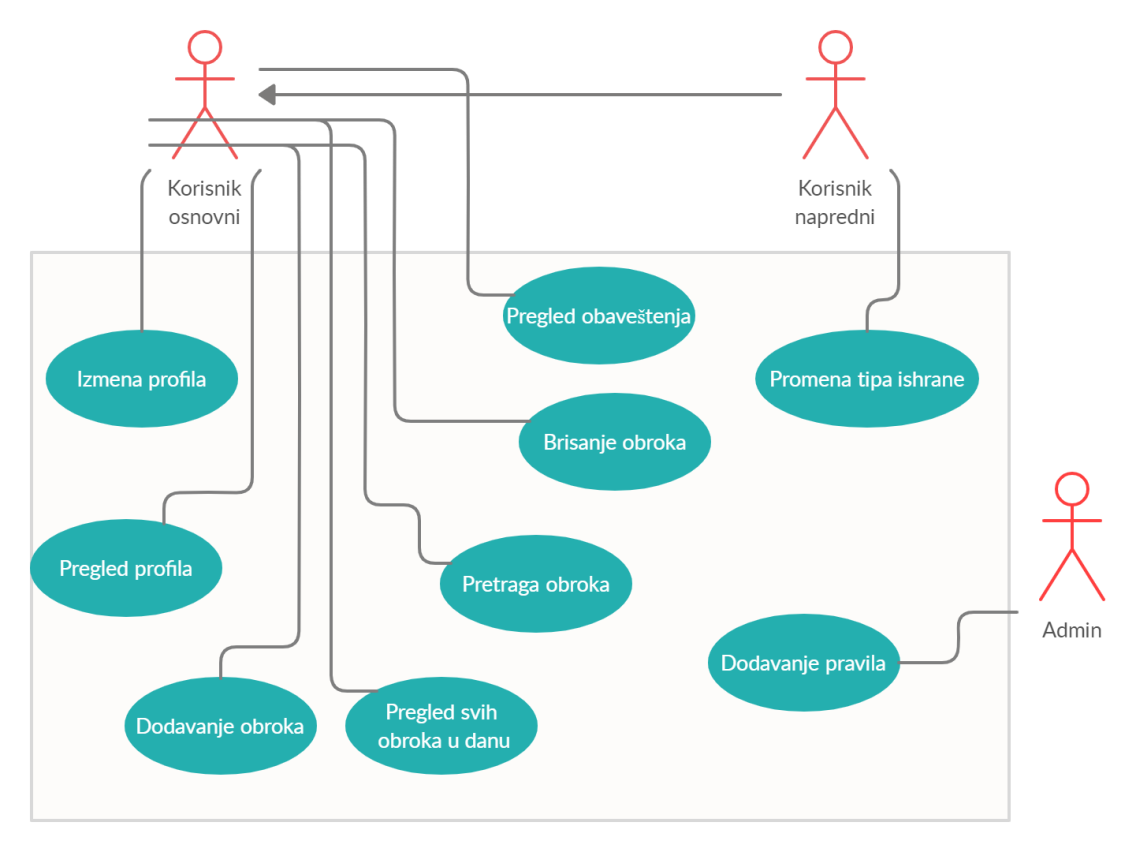
Slika 4.1.2 Klase za komunikaciju sa spoonacular.com

## 4.2 Dijagram slučajeva korišćenja

Aplikacija je namenjena za 3 tipa korisnika:

* Običan korisnik
* Napredni korisnik
* Administrator

Na slici 4.2.1 je prikazan UML dijagram slučajeva korišćenja (*use-case*). Ovaj dijagram služi krajnjim korisnicima da razumeju šta sistem radi, ali tako da se ne prikazuju detalji implementacije, koji će biti objašnjeni u narednom poglavlju. Elementi ovog dijagrama su akteri, slučajevi korišćenja i relacije.

Slika 4.2.1 UML dijagram

Uloga administratora je dodavanje pravila u sistem. Adminstrator predstavlja eksperta koji uz pomoć *Drools*-a piše pravila koja će biti korišćena. U ovom sistemu ekspert ima ključnu ulogu jer upravo od tih definisanih pravila zavise sve informacije koje će dobijati ostali korisnici.

Napredni korisnik se razlikuje od običnog po informacijama koje je obezbedio prilikom registracije. Procenat masti nije toliko poznata informacija korisnicima kao visina i težina, prema tome ovaj podatak se može iskoristiti za razdvajanje dve grupe korisnika. Iz tog razloga je naprednijim korisnicima omogućeno da izmene tip ishrane. Pored ove napredni korisnik može koristiti sve funkcionalnosti kao i običan korinik.

Sledi opis funkcionalnosti koje su omogućene običnom korisniku.

1. Pregled liste svih unetih obroka evidentiranih u toku dana, podrazumevano današnjeg. Korisnik ima pristup samo svojoj listi dodatih obroka, ne i listi drugih korisnika.
2. Pretraga obroka, tačnije recepata sa njihovim nazivima, koje mogu pregledati kao stavke u listi rezultata. Za svaku stavku u listi će biti prikazna slika i naziv uz dodatne informacije o obroku. Takođe je i za većinu obezbeđen link ka *web* stranici na kojoj se korisnici mogu dodatno informisati, kao i pročitati upustsvo za spremanje ukoliko je neophodno.
3. Generisanje dnevnog plana ishrane uz pomoć informacija o korisniku. Na osnovu predviđenog broja kalorija biće predloženo tri obroka koji zadovoljavaju korisnikove potrebe. Korisnik ima pravo da prihvati i sačuva predloženi plan, da zatraži ponovno generisanje ili da odustane.
4. Dodavanje obroka iz liste pretraženih obroka ili iz svog dnevnog plana, kao što je prethodno objašnjeno. Nakon dodavanja taj obrok će se pojaviti u listi svih dnevnih obroka tog korisnika.
5. Korisnik u svakom trenutku može poništiti izbor određenog obroka tako što će ga obrisati iz svoje liste. Ovim se korisniku obezbeđuje veća sigurnost prilikom dodavanja obroka znajući da može povući svoju odluku.
6. Pregled obaveštenja koja kreiraju prilikom dodavanja ili brisanja nekog obroka. Generisanje obaveštenja direktno zavisi od sistema baziranog na pravilima odnosno eksperta koji je napisao pravila. Obaveštenja predstavljaju poruke koje su rezultat primene pravila za obroke koje je korisnik dodao.
7. Za korišćenje sistema Korisnik treba da unese i redovno ažurira svoje osnove podatke kao što su visnina, težina, itd., jer ti podaci utiču na računanje idealnog unosa kalorija.

# Implementacija

U prethodnim poglavljima su opisane tehnologije korišćene za ovaj projekat kao i arhitektura sistema (poglavlje [3. Tehnologije](#_Tehnologije)). U ovom poglavlju će biti predstavljeni detalji implementiranog rešenja za svaku pojedinačnu aplikaciju. Sledi prikaz i objašnjenje implementacije serverske aplikacije, sistema baziranog na pravilima, administratorske i mobilne aplikacije. Kao što je već rečeno za korišćenje aplikacije je potrebno imati aktivan nalog tako da će prvo biti opisan postupak registracije i prijave korisnika, a zatim ostale funkcionalnosti. Kod aplikacije je dostupan na <https://github.com/Marko131/diplomski>.

## 5.1 Serverska aplikacija

Serverska aplikacija je jedan od glavnih delova sistema i služi za komunikaciju sa bazom podataka i sistemom baziranim na pravilima. Preko krajnjih tačaka obezbeđuje komunikaciju sa bilo kojom klijentskom aplikacijom. Prima *HTTP* zahteve koje obrađuje i zatim vraća odgovore.

Izvorni kod je, zbog bolje organizacije, podeljen u više paketa: *controller*, *dto*, *exceptions*, *model*, *repository*, *security* i *service*. U drugom poglavlju, tačnije u delu gde se govori o *Spring*-u je objašnjeno šta generalno sadrže ovi paketi tako da se neće ponavljati i ovde, već će biti prikazani samo konkretni delovi implementacije.

Jedine metode kojima može pristupiti neautentifikovani korisnik su registracija i prijava. Za registraciju je potrebno poslati *POST* zahtev i proslediti objekat u kom se nalaze potrebne inofrmacije za registraciju: email, šifra, potvrda šifre, ime, prezime, broj godina, pol, visina, težina i aktivnost. Ono što je bitno za registraciju korisnika je implementirano u sistemu baziranom na pravilima (objašnjeno u poglavlju 5.2). Potvrda identiteta je implementirana uz pomoć *json web token*-a (*JWT*). Nakon uspešne prijave na sistem, što znači da je korisnik uneo tačan *email* i šifru, korisnik dobija token. Svaki zahtev, za funkcionalnost koja zahteva prijavljenog korisnika, mora u zaglavlju da sadrži token. Ukoliko token nije prisutan u zaglavlju dobiće se odgovor 401 *unauthorized*, a ukoliko token nije validan odgovor će biti 403 *forbidden*. Važne informacije koje se nalaze u tokenu su *email* korisnika, uloga i vreme kada ističe token.

Kontroleri, klase anotirane sa *RestController*, pružaju mogućnost rukovanja zahtevima. Kreiraju se po entitetima, tako da npr. *UserController* sadrži sve krajnje tačke vezane za funkcionalnosti o korisniku. Tu spadaju registracija, prijava, pregled i izmena profila. Primer kontrolera sa metodom za prijavu na sistem prikazan je na listingu 5.1.1.

@RestController   
public class UserController {  
 @Autowired  
 AuthenticationManager authenticationManager;  
  
  
 @Autowired  
 private UserDetailsServiceImpl userDetailsService;  
  
  
 @Autowired  
 TokenUtils tokenUtils;  
  
 @PostMapping("/login")  
 public ResponseEntity<String> login(@Valid @RequestBody LoginUserDTO loginUserDTO) {  
 try {  
 UsernamePasswordAuthenticationToken token = new UsernamePasswordAuthenticationToken(  
 loginUserDTO.getEmail(), loginUserDTO.getPassword());  
 authenticationManager.authenticate(token);  
 UserDetails details = userDetailsService.loadUserByUsername(loginUserDTO.getEmail());  
 return new ResponseEntity<String>(tokenUtils.generateToken(details), HttpStatus.*OK*);  
 } catch (Exception ex) {  
 return new ResponseEntity<String>("Invalid login", HttpStatus.*BAD\_REQUEST*);  
 }  
 }  
}

Listing 5.1.1 Primer kontrolera sa metodom za prijavu

Kontroleri uz pomoć mehanizma *Dependeny Injection* imaju pristp servisnom sloju. Servisi su klase anotirane sa *Service*, što će omogućiti da budu registrovani kao *bean*. U tim klasama se pišu metode koje sadrže svu poslovnu logiku vezanu za neki entitet. Za razliku od klasične aplikacije u kojoj bi domensko znanje bilo implementirano u kodu, a nalazilo bi se u servisnom sloju, to domensko znanje je u implementiranoj aplikaciji odvojeno u posebnom sistemu baziranom na pravilima.

Na listingu 5.1.2 se može videti primer servisa i metode za kreiranje novog korisnika. Prvo se proverava da li korisnik sa unetim email-om već postoji u sistemu i kodira njegova šifra. Nakon toga se poziva rezoner, korisnik se dodaje u radnu memoriju i okidaju se pravila. Kada se izvrše pravila korisnik se čuva u bazi podataka. U narednom delu će biti opisano povezivanje sa *drools* aplikacijom i kako se obezbeđuju objekti klasa *KieContainer* i *KieSession*.

@Service  
public class UserDetailsServiceImpl implements UserDetailsService {  
  
 @Autowired  
 private UserRepository userRepository;  
  
 @Autowired  
 private PasswordEncoder passwordEncoder;  
  
 @Autowired  
 private KieContainer kieContainer;  
  
   
 public User createUser(User user){  
 if (userRepository.findByEmail(user.getEmail()) != null) throw new UserAlreadyExistsException();  
 user.setPassword(passwordEncoder.encode(user.getPassword()));  
  
 KieSession kSession = kieContainer.newKieSession();  
  
 kSession.insert(user);  
 kSession.fireAllRules();  
  
 return userRepository.save(user);  
 }  
  
}

Listing 5.1.2 Primer servisa sa metodom za kreiranje korisnika

Repozitorijumi su interfejsi anotirani sa *Repository* i omogućavaju komunikaciju sa bazom podataka. Nasleđuju *JpaRepository* interfejs pri čemu je neophodno definisati za koji tip entiteta se koristi i koji je tip primarnog ključa. *JpaRepository* obezbeđuje osnovne *CRUD* operacije za upravljanje entitetima kao i metode za paginaciju i sortiranje. Ukoliko su potrebne neke specifičnije metode za rukovanje entitetima dovoljno je napisati deklaraciju funkcije u odgovarajućem formatu bez pisanja *SQL* upita, koji će biti automatski generisan. Na listingu 5.1.3 se može videti primer repozitorijuma za upravljanje korisnicima sa dodatom metodom za pronaženje korisnika po korisničkom imenu.

@Repository  
public interface UserRepository extends JpaRepository<User, Integer> {  
 User findByEmail(String email);  
}

Listing 5.1.3 Primer repozitorijuma sa jednom dodatom metodom

Povezivanje sa *drools* aplikacijom je omogućeno tako što se instancira *bean* tipa *KieContainer* (listing 5.1.4) i u pom.xml fajl doda *dependency*. Takođe je definisan i *KieSession* *bean* koji će biti aktivan od pocetka pokretanja aplikacije. Postojanje sesije koja nije na nivou zahteva već aplikacije će biti objašnjeno u delu o *drools* aplikaciji.

@Bean

public KieContainer kieContainer() {

KieServices ks = KieServices.Factory.get();

KieContainer kContainer = ks

                .newKieContainer(ks.newReleaseId("sbnz.integracija", "drools-spring-kjar", "0.0.1-SNAPSHOT"));

KieScanner kScanner = ks.newKieScanner(kContainer);

kScanner.start(10\_000);

return kContainer;

}

Listing 5.1.4 KieContainer bean

## 5.2 Drools aplikacija

Sistem baziran na pravilima je implementiran pomoću *drools*-a, a za kreiranje projekta je korišćen *Maven*, alat za automatsku izgradnju *Java* projekata. Ovim se značajno smanjuje vreme potrebno za konfigurisanje projekta i može se koristiti za druge projekte kao *dependency*.

Svi .drl fajlovi se nalaze u *resource* folderu. Fajlovi su podeljeni tako da se u jednom nalaze pravila koja se aktiviraju nakon dodavanja novog obroka, a u ostalim pravila koja se aktiviraju prilikom registracije korisnika.

Kao što je objašnjeno u prethodnom delu o serverskoj aplikaciji, kada je u pitanju registracija, korisnik pored informacija potrebnih za autentifikaciju popunjava i podatke potrebne za drools aplikaciju. Prvi korak je računanje indeksa telesne mase (*body mass index* - *BMI*), za šta je potrebna samo visina i težina korisnika. U zavisnosti od indeksa telesne mase, prema podacima Svetske zdravstvene organizacije [[25](#_Literatura_1)], postoji šest kategorija prikazanih u tabeli 5.2.1.

| **BMI** | **Nutritional status** |
| --- | --- |
| Below 18.5 | Underweight |
| 18.5–24.9 | Normal weight |
| 25.0–29.9 | Pre-obesity |
| 30.0–34.9 | Obesity class I |
| 35.0–39.9 | Obesity class II |
| Above 40 | Obesity class III |

Tabela 5.2.1 Kategorije u zavisnosti od indeksa telesne mase

Sledeći korak je računanje bazalnog metabolizma, na koje utiče više faktora nego za *BMI*. Postoji više formula za računanje bazalnog metabolizma, a izbor u ovom sistemu će zavisiti od podataka koje korisnik unese prilikom registracije. *Mifflin*-*St* *Jeor* *Equation* je osnovna formula i za nju su potrebni podaci koje korisnik uglavnom već zna. Formula se razlikuje prema polu, a neophodni podaci su: visina, težina i broj godina.

*Mifflin*-*St* *Jeor* *Equation* za muškarce

*Mifflin*-*St* *Jeor* *Equation* za žene

Gde je *W* težina u kilogramima, *H* visina u centimetrima i *A* je broj godina.

Postoje i formule koje koriste različite parametre kao što su procenat masti. Ovaj podatak je moguće dobiti na različite načine, ali je za potpunu preciznost potrebna nešto profesionalnija oprema [[26](#_Literatura_1)]. Iz tog razloga korisnik koji popuni neobavezno polje za procenat masti može se smatrati naprednijim od običnog. U tom slučaju se koristi *Katch-McArdle* formula za koju su potrebni težina i procenat masti.

*Katch-McArdle* formula

Gde je *F* procenat masti i *W* visina u centimetrima.

Na listingu 5.2.2 su prikazana pravila za izračunavanje bazalnog metabolizma.

rule "Mifflin-St Jeor Female" salience 20

    when

        u: User(gender == Gender.FEMALE)

    then

        u.setBmr(10 \* u.getWeight() + 6.25 \* u.getHeight() - 5 \* u.getAge() - 161);

        update(u)

end

rule "Mifflin-St Jeor Male" salience 20

    when

        u: User(gender == Gender.MALE)

    then

        u.setBmr(10 \* u.getWeight() + 6.25 \* u.getHeight() - 5 \* u.getAge() + 5);

        update(u)

end

rule "Katch-McArdle" salience 21

    when

        u: User($bf: bodyFat)

        eval($bf != null)

    then

        u.setBmr(370 + 21.6 \* (1 - u.getBodyFat() / 100) \* u.getWeight());

end

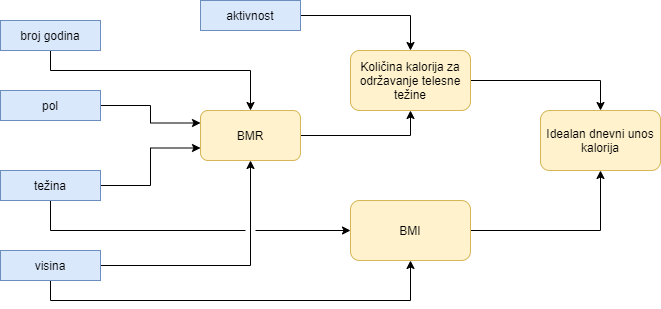
Listing 5.2.2 Pravila za računanje bazalnog metabolizma

Za broj kalorija potreban za održavanje telesne težine potrebno je razmatrati i fizičku aktivnost. Količina fizičke aktivnosti se svodi na izbor jedne stavke iz liste koja sadrži sledeće:

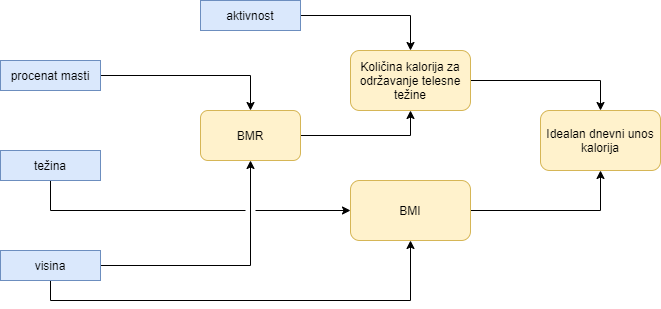
* *Sedentary*: bez fizičke aktivnosti
* *Light*: vežbanje od jednom do tri puta nedeljno
* *Moderate*: vežbanje od tri do četiri puta nedeljno
* *Active*: svakodnevno ili intenzivno vežbanje tri do četiri puta nedeljno
* *Very* *active*: intenzivno vežbanje šest do sedam puta nedeljno
* *Extra* *active*: veoma intenzivno svakodnevno vežbanje ili fizički posao

Broj dobijen računanjem bazalnog metabolizma se množi odgovarajućim koeficijentom vezanim za fizičku aktivnost.

U zavisnosti od indeksa telesne mase prethodno dobijeni broj se množi sa koeficijentima za smanjivanje ili dodavanje telesne težine tako da cilj bude dolazak do normalnog indeksa telesne mase. Da bi se lakše razumeo postupak povezivanja formula tj. ulančavanja unapred informacija na slici 5.2.3 se može videti dijagram koji prikazuje tok podataka od ulaza do krajnjeg rezultata sa pravilima koja se primenjuju za običnog korisnika, a na slici 5.2.4 za naprednog korisnika.



Slika 5.2.3 Postupak ulančavanja unapred za običnog korisnika



Slika 5.2.4 Postupak ulančavanja unapred za naprednog korisnika

Prikaz pravila u kojima se računa količina kalorija potrebna za održavanje telesne težine se može videti na listingu 5.2.5, a za idealan dnevni unos kalorija na listingu 5.2.6.

rule "Calorie SEDENTARY" salience 10

    when

        u: User(activity == Activity.SEDENTARY)

    then

        u.setCalories(u.getBmr() \* 1.2);

        update(u);

end

rule "Calorie LIGHT" salience 10

    when

        u: User(activity == Activity.LIGHT)

    then

        u.setCalories(u.getBmr() \* 1.35);

        update(u);

end

rule "Calorie MODERATE" salience 10

    when

        u: User(activity == Activity.MODERATE)

    then

        u.setCalories(u.getBmr() \* 1.5);

        update(u);

end

rule "Calorie ACTIVE" salience 10

    when

        u: User(activity == Activity.ACTIVE)

    then

        u.setCalories(u.getBmr() \* 1.65);

        update(u);

end

rule "Calorie VERY\_ACTIVE" salience 10

    when

        u: User(activity == Activity.VERY\_ACTIVE)

    then

        u.setCalories(u.getBmr() \* 1.8);

        update(u);

end

rule "Calorie EXTRA\_ACTIVE" salience 10

    when

        u: User(activity == Activity.EXTRA\_ACTIVE)

    then

        u.setCalories(u.getBmr() \* 1.95);

        update(u);

end

Listing 5.2.5 Pravila za računanje količie kalorija za održavanje telesne težine

rule "Calories 1" salience 5

    when

        $u: User(getCalories() > 0, getBmi() < 10)

    then

        $u.setCalories($u.getCalories() \* 1.25);

end

rule "Calories 2" salience 5

    when

        $u: User(getCalories() > 0, getBmi() >= 15 && getBmi() < 16)

    then

        $u.setCalories($u.getCalories() \* 1.20);

end

rule "Calories 3" salience 5

    when

        $u: User(getCalories() > 0, getBmi() >= 16 && getBmi() < 18.5)

    then

        $u.setCalories($u.getCalories() \* 1.1);

end

rule "Calories 4" salience 5

    when

        $u: User(getCalories() > 0, getBmi() >= 25 && getBmi() < 30)

    then

        $u.setCalories($u.getCalories() \* 0.9);

end

rule "Calories 5" salience 5

    when

        $u: User(getCalories() > 0, getBmi() >= 30 && getBmi() < 35)

    then

        $u.setCalories($u.getCalories() \* 0.85);

end

rule "Calories 6" salience 5

    when

        $u: User(getCalories() > 0, getBmi() >= 35 && getBmi() < 40)

    then

        $u.setCalories($u.getCalories() \* 0.8);

end

rule "Calories 7" salience 5

    when

        $u: User(getCalories() > 0, getBmi() >= 40)

    then

        $u.setCalories($u.getCalories() \* 0.75);

end

Listing 5.2.6 Pravila za računanje idealnog unosa kalorija

Pravila koja se aktiviraju prilikom dodavanja obroka vezana su za kalorijski deficit, kalorijski suficit, unos šećera i zasićenih masti, kao i razmak između obroka. Preporuka je da vremenski period protekao između dva obroka ne bude manji od tri sata, ni veći od šest sati. Ovo je implementirano uz pomoć upravljanja događajima u *drools*-u, tako što se prilikom dodavanja novog obroka kreira novi događaj koji u radnoj memoriji ostaje tri sata. Ukoliko se za to vreme doda novi obrok od strane istog korisnika, on će dobiti odgovarajuće obaveštenje (listing 5.2.4).

declare MealEvent

    @role(event)

    @expires(3h)

    userId: Integer

    mealEnum: MealEnum

end

query checkDifferentMeals(List allMeals)

     $first: MealEvent() from allMeals.get(0)

     MealEvent(mealEnum != $first.mealEnum) from allMeals

end

rule "Wait time"

when

    $user: User()

    $list: List(size > 0) from collect(MealEvent(userId == $user.id))

    checkDifferentMeals($list;)

    $notification: Notification($notifications: text not contains "The wait time between meals should be at least 3 hours", user == $user)

    then

    delete($user);

    $notification.addToList("The wait time between meals should be at least 3 hours");

end

Listing 5.2.4 Pravilo za vreme između dva obroka

Slično kao i prethodno pravilo, ukoliko se u radnoj memoriji ne nađe događaj, koji označava da je dodat novi obrok, više od 6 sati, korisnik će takođe dobiti obaveštenje (listing 5.2.5).

rule "Wait time 2"

when

    $user: User()

    not (MealEvent(userId == $user.id) over window:time(6h))

    $notification: Notification($notifications: text not contains "The wait time between meals should not be more than 6 hours", user == $user)

    then

    delete($user);

    $notification.addToList("The wait time between meals should not be more than 6 hours");

end

Listing 5.2.5 Pravilo za vreme između dva obroka

Ostala pravila koja su vezana za kalorijski deficit odnosno suficit i unos šećera i zasićenih masti su veoma slična. Leva strana pravila se svodi na funkcije iz *droolsa* *accumulate* i *sum* koja računa zbir elemenata liste i graniču vrednost preporučenu od strane eksperta. Primer jednog takvog pravila prikazan je na listingu 5.2.6

rule "Amount of sugar Men"

agenda-group "day-rules"

    when

        $day: UserDay($meals: mealRecipes, $user: user, $user.gender == Gender.MALE )

        Number (doubleValue > 37.5) from accumulate(

            MealRecipe(

                $sugar: sugars

            ) from $meals,

            sum($sugar)

        )

        $notification: Notification()

    then

        $notification.addToList("You should not take more than 37.5g of sugar");

end

Listing 5.2.6 Primer pravila sa *accumulate* i *sum*

## 5.3 Administratorska aplikacija

Administrator za pisanje *drools* pravila koristi web aplikaciju napisanu u *React*-u. Aplikacija se sastoji od dve stranice, jedne za prijavu i druge za pisanje pravila. Forma za unos pravila sadrži polje na naziv fajla i polje za definisanje pravila. Prilikom potvrde forme šalje se zahtev serverskoj aplikaciji, nakon čega će *Maven* ponovo instalirati *Drools* aplikaciju.

## 5.4 Mobilna aplikacija

Klijentska mobilna aplikacija je razvijena u *React* *Native* radnom okviru. *AppRegistry* predstavlja ulaznu tačku za pokretanje svake *React* *Native* aplikacije. Ovde će biti registrovana korenska komponenta, tako da izvorni sistem može učitati sve što je neophodno za pokretanje aplikacije kada bude spremna.

U korenskoj komponenti *App*.*js* definisan je router koji korisnicima omogućava navigaciju na neku od tri definisane stranice (listing 5.3.1). Svaka stranica predstavlja jednu kompoenentu koja će biti prikazana na ekranu mobilnog uređaja.

const App = () => {

  return (

    <Router>

      <Stack key="root">

        <Scene key="home" component={HomePage}

hideNavBar={true} />

        <Scene key="login" component={LoginPage}

hideNavBar={true} />

        <Scene key="register" component={RegisterPage}

hideNavBar={true} />

      </Stack>

    </Router>

  );

};

Listing 5.3.1 *App* komponenta sa *router*-om

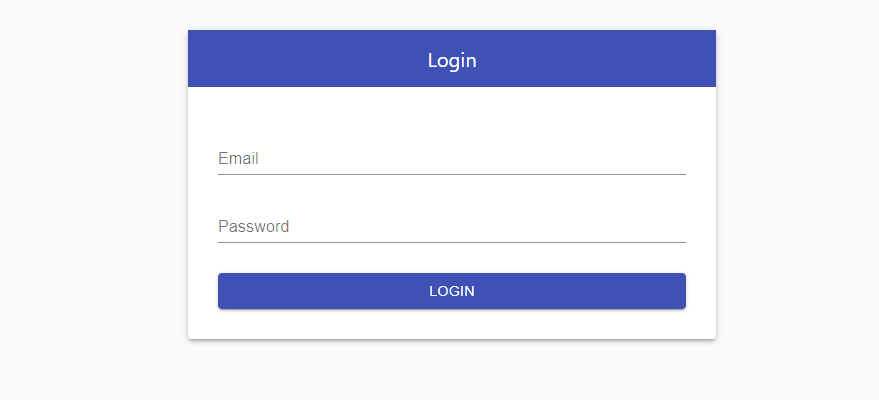
Sve komponente su definisane kao funkcije bez obzira da li imaju stanje ili samo prikazuju sadržaj na osnovu prosledjenih parametara. Ovo je omogućeno uz pomoć mehanizma koji se zove *Hooks* i dodat je u verziji 16.8 *React*-a. Komunikacija sa serverskom aplikacijom se obavlja preko *axios*-a, *HTTP* klijenta zasovanog na *Promise*-ima. Funkcije koje služe sa slanje zahteva ka serverskoj aplikaciji su pisane u komponentama dok je kod većih aplikacija bolja praksa da se te funckije izdvoje iz komponenti zbog bolje preglednosti.

# Prikaz implementiranog rešenja

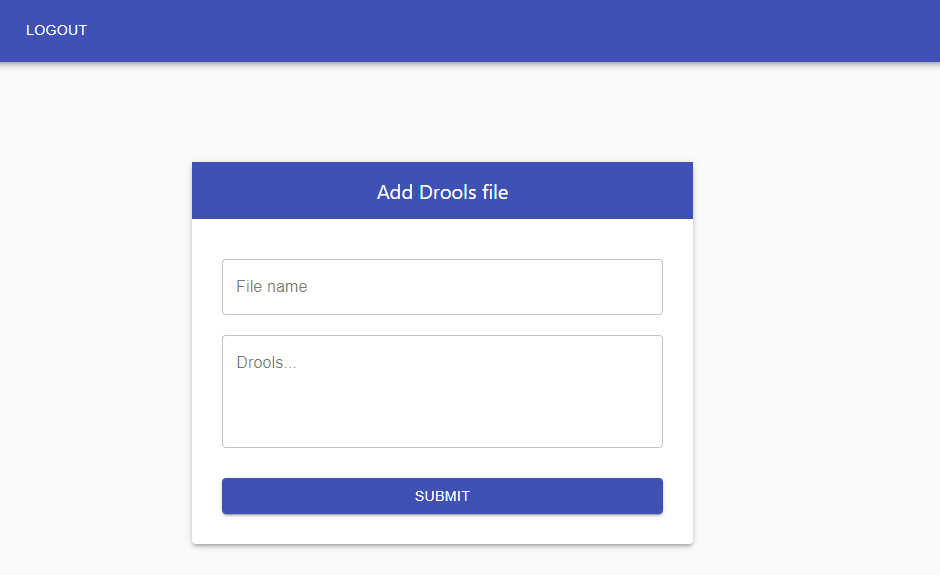
U ovom poglavlju će biti prikazane administratorska *web* aplikacija i mobilna korisnička aplikacija. U nastavku biće opisan postupak korišćenja iz ugla onog ko koristi aplikaciju, prvo administratora a zatim i ostalih korisnika.

## 6.1 Administratorska aplikacija

*Web* aplikacija koju koristi administrator za dodavanje pravila je prilično jednostavna jer ima samo dve stranice. Stranica za prijavu (slika 6.1.1) sadrži formu za unos *email*-a i šifre, a pritiskom na dugme *login*, ukoliko je kombinacija *email*-a i šifte tačna, administratoru se prikazuje glavna stranica za dodavanje pravila (slika 6.1.2).



Slika 6.1.1 Stranica za prijavu



Slika 6.1.2 Stranica za dodavanje pravila

Administrator popunjava formu u kojoj navodi ime fajla i samo pravilo i klikom na dugme *submit* pravilo će biti dodato. Ukoliko administrator želi da se odjavi potrebno je da klikne na dugme *logout* u gornjem levom uglu i nakon toga će se prebaciti ponovo na stranicu za prijavu.

## 6.2 Mobilna aplikacija

Kao što je već pomenuto da bi koristio mobilnu aplikaciju korisnik mora imati aktivan nalog. Prvi korak je registracija na stranici koja se može videti na slici 6.2.1. Korisnik popunjava podatke u odgovarajuća polja forme a pritiskom na dugme register, ukoliko su sva polja ispravno popunjena, prebacuje se na stranicu za prijavu (slika 6.2.2).

|  |  |
| --- | --- |
|  |  |
| Slika 6.2.1 Stranica za registraciju | Slika 6.2.2 Stanica za prijavu |

Nakon što unese tačan email i šifru i klikne na dugme login, korisnik prelazi na glavnu stranicu odakle može pristupati svim funkcionalnostima aplikacije. Početna stanica nakon prijave se može videti na slici 6.2.3. Pritiskom na odgovarajuće dugme za navigaciju korisnik bira da li želi da mu se prikaže profil (slika 6.2.4), lista obroka ili obaveštenja (6.2.5).

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Slika 6.2.3 Početna stranica | Slika 6.2.4 Profil korisnika | Slika 6.2.5 Obaveštenja |

Generisanje dnevnog plana ishrane je prikazano na slici 6.2.6. Korisniku su ponuđena tri obroka. Detaljne informacije o svakom obroku korisnik može videti u svom internet pretraživaču pritiskom na naslov obroka sa zelenom ikonicom. Korisnik može da prihvati i sačuva plan ishrane, da zatraži nova tri obroka ili da odustane.

Dodavanje obroka se može vršiti pretragom koja je prikazana na slici 6.2.7 ili iz prethodno sačuvanog plana ishrane (slika 6.2.8). Korisnik iz liste bira da li je u pitanju doručak, ručak ili večera i pritiskom na odgovarajuće dugme dodaje izabrani obrok.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Slika 6.2.6 Generisanje dnevnog plana | Slika 6.2.7 Pretraga obroka | Slika 6.2.8 Dodavanje obroka iz dnevnog plana |

Nakon dodatog obroka korisniku se osvežavaju podaci o unetim kalorijama, proteinima, mastima i ugljenim hidratima (slika 6.2.9). Pritiskom na izabranu stavku iz liste obroka korisniku se prikazuje dijalog za uklanjanje obroka (slika 6.2.10).

|  |  |
| --- | --- |
|  |  |
| Slika 6.2.9 Lista obroka | Slika 6.2.10 Uklanjanje obroka iz liste |

# Zaključak

U ovom radu je predstavljen razvoj sistema za računanje kalorija i preporuku obroka korišćenjem sistema baziranog na pravilima. Sistem čine serverska aplikacija implementirana u Spring-u, sistem baziran na pravilima u Drools-u i klijentske aplikacije za koje je korišćen React i ReactNative. Implementaciju je znatno olakšalo korišćenje spoonacular.com API-a, koji je omogućio određene funkcionalnosti.

Opisana su slična rešenja, korišćene tehnologije i specifikacija sistema. Na kraju rada je prikazano korišćenje administratorske web aplikacije i korisničke mobilne aplikacije. Mogući dalji pravci unapređenja sistema bilo bi dovavanje novih pravila za računanje BMI koja će koristiti neke druge/napradne podatke o korisniku, proširiti podatke o korisnicima sa alergenima i implementirati pravila za upozorenja ukoliko korisnika odabere obroke koje sadrže alergene, i proširiti administratorski deo aplikacije koji bi administratorima omogućio lakšu izgradnju pravila.

# Literatura

1. Liu, Han & Gegov, Alexander & Stahl, Frederic. (2014). Categorization and Construction of Rule Based Systems. Communications in Computer and Information Science. 459. 10.1007/978-3-319-11071-4\_18.
2. R. J. K. Jacob and J. N. Froscher, "A software engineering methodology for rule-based systems," in IEEE Transactions on Knowledge and Data Engineering, vol. 2, no. 2, pp. 173-189, June 1990, doi: 10.1109/69.54718.
3. <https://becominghuman.ai/the-key-differences-between-rule-based-ai-and-machine-learning-8792e545e6>, preuzeto septembra 2020.
4. <https://www.applozic.com/blog/rule-based-v-s-ai-based-approaches-to-building-ai/>, preuzeto septembra 2020.
5. <https://www.who.int/news-room/fact-sheets/detail/obesity-and-overweight>, preuzeto septembra 2020.
6. Kong A, Beresford SAA, Alfano CM, et al. Self-monitoring and eating-related behaviors are associated with 12-month weight loss in postmenopausal overweight-to-obese women. *J Acad Nutr Diet*. 2012;112(9):1428-1435. doi:10.1016/j.jand.2012.05.014
7. Burke LE, Wang J, Sevick MA. Self-monitoring in weight loss: a systematic review of the literature. *J Am Diet Assoc*. 2011;111(1):92-102. doi:10.1016/j.jada.2010.10.008
8. Honary M, Bell BT, Clinch S, Wild SE, McNaney R. Understanding the Role of Healthy Eating and Fitness Mobile Apps in the Formation of Maladaptive Eating and Exercise Behaviors in Young People. *JMIR Mhealth Uhealth*. 2019;7(6):e14239. Published 2019 Jun 18. doi:10.2196/14239
9. <https://spoonacular.com/food-api>, preuzeto septembra 2020.
10. <https://reactnative.dev/>, preuzeto septembra 2020.
11. <https://spring.io/>, preuzeto septembra 2020.
12. <https://restfulapi.net/>, preuzeto septembra 2020.
13. <https://www.mysql.com/>, preuzeto septembra 2020.
14. <https://www.drools.org/>, preuzeto septembra 2020.
15. <https://about.fb.com/>, preuzeto septembra 2020.
16. <https://www.android.com/>, preuzeto septembra 2020.
17. <https://www.apple.com/ios/ios-13/>, preuzeto septembra 2020.
18. <https://docs.microsoft.com/en-us/windows/uwp/get-started/universal-application-platform-guide>, preuzeto septembra 2020.
19. <https://docs.spring.io/spring/docs/4.0.x/spring-framework-reference/html/images/spring-overview.png>, preuzeto septembra 2020.
20. <https://docs.spring.io/spring/docs/4.0.x/spring-framework-reference/html/overview.html>, preuzeto septembra 2020.
21. <https://hibernate.org/>, preuzeto septembra 2020.
22. <https://junit.org/junit5/>, preuzeto septembra 2020.
23. <https://testng.org/doc/>, preuzeto septembra 2020.
24. <https://docs.jboss.org/drools/release/5.2.0.Final/drools-expert-docs/html/ch05.html>, preuzeto septembra 2020.
25. <https://www.euro.who.int/en/health-topics/disease-prevention/nutrition/a-healthy-lifestyle/body-mass-index-bmi>, preuzeto septembra 2020.
26. <https://www.healthline.com/nutrition/ways-to-measure-body-fat>, preuzeto septembra 2020.

# Biografija

Marko Stanić je rođen 31. maja 1997. godine u Šapcu. U istom gradu je završio Osnovnu školu „Nikolaj Velimirović“. Nakon toga upisuje Šabačku gimnaziju, informatički smer i završava je 2016. godine. Kasnije te godine upisuje Fakultet tehničkih nauka, smer Softversko inženjerstvo i informacione tehnologije. Sve ispite polaže i studije završava u roku, 2020. godine.

# Ključna dokumentacijska informacija

|  |  |
| --- | --- |
| Redni broj, **RBR**: |  |
| Identifikacioni broj, **IBR**: |  |
| Tip dokumentacije, **TD**: | Monografska publikacija |
| Tip zapisa, **TZ**: | Tekstualni štampani dokument |
| Vrsta rada, **VR**: | Diplomski rad |
| Autor, **AU**: | Marko Stanić |
| Mentor, **MN**: | prof. dr Siniša Nikolić, FTN Novi Sad |
| Naslov rada, **NR**: |  |
| Jezik publikacije, **JP**: | Srpski |
| Jezik izvoda, **JI**: | Srpski / engleski |
| Zemlja publikovanja, **ZP**: | Srbija |
| Uže geografsko područje, **UGP**: | Vojvodina |
| Godina, **GO**: | 2020 |
| Izdavač, **IZ**: | Autorski reprint |
| Mesto i adresa, **MA**: | Novi Sad, Fakultet tehničkih nauka, Trg Dositeja Obradovića 6 |
| Fizički opis rada, **FO**: | br. poglavlja 7/ stranica 38/ citata 26/ tabela 1/ slika 28 |
| Naučna oblast, **NO**: | Softversko inženjerstvo i informacione tehnologije |
| Naučna disciplina, **ND**: | Sistemi bazirani na znanju |
| Predmetna odrednica /  ključne reči, **PO**: |  |
| **UDK** |  |
| Čuva se, **ČU**: | Biblioteka Fakulteta tehničkih nauka, Trg Dositeja Obradovića 6, Novi Sad |
| Važna napomena, **VN**: |  |
| Izvod, **IZ**: |  |
| Datum prihvatanja teme, **DP**: |  |
| Datum odbrane, **DO**: |  |
| Članovi komisije, **KO**: |  |
| predsednik |  |
| član |  |
| mentor |  |
| Potpis mentora | |

# Key words documentation

|  |  |
| --- | --- |
| Accession number, **ANO**: |  |
| Identification number, **INO**: |  |
| Document type, **DT**: | Monographic publication |
| Type of record, **TR**: | Textual material |
| Contents code, **CC**: | BSc thesis |
| Author, **AU**: | Marko Stanić |
| Mentor, **MN**: | Siniša Nikolić, PhD |
| Title, **TI**: |  |
| Language of text, **LT**: | Serbian |
| Language of abstract, **LA**: | Serbian / english |
| Country of publication, **CP**: | Serbia |
| Locality of publication, **LP**: | Vojvodina |
| Publication year, **PY**: | 2020 |
| Publisher, **PB**: | Author’s reprint |
| Publication place, **PP**: | Novi Sad, Faculty of Technical Sciences, Trg Dositeja Obradovića 6 |
| Physical description, **PD**: | no. of chapters 7/ pages 38/ quotes 26/ tables 1/ pictures 28/ graphs 0/ appendix 0 |
| Scientific field, **SF**: | Software engineering and information technologies |
| Scientific discipline, **SD**: | Knowledge based systems |
| Subject / Keywords, **S/KW**: |  |
| **UDC** |  |
| Holding data, **HD**: | Library of the Faculty of Technical Sciences, Trg Dositeja Obradovića 6, Novi Sad |
| Note, **N**: |  |
| Abstract, **AB**: |  |
| Accepted by sci. Board on, **ASB**: |  |
| Defended on, **DE**: |  |
| Defense board, **DB**: |  |
| president |  |
| member |  |
| mentor |  |
| Mentor's signature | |