

Best Health Strategies

Bakir Karović

Harun Hadžić

Sara Avdagić

Sadržaj

Opis projekta	4
Scenariji	5
1. Slučaj prijave korisnika.....	5
2. Reakcija sistema na registraciju (proces kreiranja plana ishrane).....	5
3. Administratorski pristup sistemu.....	5
Scenarij 1 – registracija klijenta.....	6
Scenarij 2 – pregled preporučenog plana ishrane	7
Scenarij 3 – upravljanje uposlenicima	8
Dijagram slučajeva upotrebe	10
Opšti use-case dijagram.....	10
Registracija use-case diagram	11
Dijagram aktivnosti.....	12
Administrator	13
Login i prikaz ishrane	14
Registracija klijenta.....	15
MVC Class diagram.....	16
Dijagram objekata	17
ERD	18
Prototip.....	19
SOLID principi.....	23
Single Responsibility Principle.....	23
Open Closed Principle.....	23
Liskov Substitution Principle	23
Interface Segregation Principle	23
Dependency Inversion Principle.....	24
Dijagram klasa sa SOLID principima.....	26
Dizajn paterni	27
STRUKTURALNI PATTERNI.....	27
1. ADAPTER.....	27
2. FACADE.....	27
3. DECORATOR.....	27

4. BRIDGE	28
5. COMPOSITE	28
6. PROXY.....	29
7. FLYWEIGHT	30
KREACIJSKI PATERNI	30
1. SINGLETON	30
2. PROTOTYPE.....	30
3. FACTORY METHOD.....	31
4. ABSTRACT FACTORY.....	31
5. BUILDER.....	31
PATERNI PONAŠANJA	32
1. STRATEGY	32
2. STATE.....	32
3. TEMPLATE METHOD	32
4. OBSERVER	33
5. ITERATOR	33
6. CHAIN OF RESPONSIBILITY	33
7. MEDIATOR	34
Class diagram sa dizajn paternima	35
Dijagram sekvence	36
Login i prikaz ishrane	36
Registracija klijenta.....	37
Administrator	38
Dijagram komponenti, paketa i raspoređivanja.....	39
Dijagram komponenti	39
Dijagram raspoređivanja.....	39
Dijagram paketa	40
Link na web aplikaciju.....	41
Video prikaz rada aplikacije Best Health Strategies	41

Opis projekta

Svjedoci smo sve bržeg života koji čovjek 21. vijeka živi. Manjak osviještenosti, znanja, ali i vremena dovodi do toga da se jako malo pažnje posvećuje vlastitom zdravlju i prilagođavanju životnog stila potrebama zdravlja. BestHealthStrategies predstavlja softver, odnosno web aplikaciju koja na jednostavan i intuitivan način omogućava planiranje i prilagođavanje ishrane potrebama svakog pojedinca. Cilj sistema je osigurati individualan plan ishrane za klijente po njihovim ličnim potrebama i željama, te im pomoći u kreiranju zdravog životnog stila. Kompletan sistem kontroliše vlasnik sistema odnosno administrator. Zaposlenici (nutricionisti) su zaduženi da kreiraju posebne planove ishrane koji idu uz specifične potrebe svakog klijenta.

Scenariji

1. Slučaj prijave korisnika

Kada korisnik pristupi web aplikaciji prikazuju mu se osnovne informacije sistema kao i mogućnost prijave. Ukoliko korisnik posjeduje profil jednostavno unosi e-mail i lozinku. Ukoliko su podaci ispravni sistem dopušta pristup posebnoj stranici klijenata. Stranica posjeduje plan ishrane strogo namjenjen tom korisniku. Omogućeno je pretraživanje i sortiranje planova. Ukoliko korisnik neposjeduje profil prikazuje mu se specijalna stranica za registraciju koja se sastoji od niza koraka. Prvi korak je popunjavanje forme sa osnovnim podacima korisnika te unos BMI podataka. Drugi korak je odabir kategorije ishrane. Svaka kategorija sadrži opis. Treći korak je isključivanje održenih sastojaka ili jela iz plana. Nakon popunjavanja sistem vrši validaciju podataka te ukoliko su isti prihvati preusmjerava korisnika na login stranicu.

2. Reakcija sistema na registraciju (proces kreiranja plana ishrane)

Ukoliko korisnik neposjeduje profil prikazuje mu se specijalna stranica za registraciju koja se sastoji od niza koraka. Prvi korak je popunjavanje forme sa osnovnim podacima korisnika te unos BMI podataka (visina, težina, količina aktivnosti, mršanje/povećanje mišićne mase). Drugi korak je odabir kategorije ishrane (gluten free, ketogenic, vegetarian, lacto-vegetarian, ovo-vegetarian, vegan, pescetarian, paleo, primal, whole30). Svaka kategorija sadrži opis. Treći korak je isključivanje održenih sastojaka ili jela iz plana (odabir sastojaka na koje je klijent alergičan ili ne želi da ih konzumira). Svaki sastojak je podjeljen u posebne odjeljke kako bi korisnik što lakše odabrao navedeno (neki od odjeljaka su: meso, voće, povrće i slično). Nakon popunjavanja vanjski api koji sadrži preko 70 000 jela za sve tipove korisnika kupi podatke te kreira idealan sedmični plan ishrane. Sistem kreirani plan za datog klijenta spašava u bazu podataka.

3. Administratorski pristup sistemu

Administrator prisupa login stranici te unosi podatke (e-mail i lozinku). Sistem vrši validaciju istih, te pri pozitivnom ishodu preusmjerava ga na posebnu stranicu namjenjenu samo za administratora. Administrator vrši pregled zaposlenika (nutricionista). Odlučuje se za brisanje zaposlenika iz sistema. Nakon toga traži zamjenu za istog te unosi novog zaposlenika u sistem. Unosi njegove podatke te sistem vrši validaciju istih. Ukoliko su podaci ispravni novi zaposleni se prikazuje u listi istih. Također, salje se potvrda o kreiranju profila novom zaposlenom. Sistem očekuje potvrdu od strane zaposlenog u trajanju od 7 dana. Ukoliko potvrda ne stigne u roku sistem opet vrši slanje.

Scenarij 1 – registracija klijenta

Naziv	Registracija klijenta
Opis	Klijent unosi svoje lične podatke, podatke o načinu ishrane te BMI podatke te kao rezultat dobija korisnički račun sa sedmičnim planom ishrane.
Vezani zahtjevi	Scenarij ispunjava potrebu za korisničkim računom.
Preduvjeti	/
Posljedice - uspješan završetak	Kreiranje sedmičnog plana ishrane koji se veže za
Posljedice - neuspješan završetak	Klijent nije registrovan (odustao od registracije).
Primarni akteri	Klijent
Ostali akteri	Sistem, API za kreiranje plana ishrane

Glavni tok:

Klijent	Sistem	API
1. Unos ličnih podataka		
2. Odabir načina ishrane		
3. Unos BMI podataka	4. Spašava lične i BMI podatke te podatke o ishrani šalje API-u	5. Prijem podataka za kreiranje plana ishrane
	7. Spašava kreirani plan ishrane	6. kreiranje plana ishrane
	8. Obavještava se korisnik o kreiranom profilu	
10. Login sa kreiranim računom	9. Klijent se preusmjerava na login	
11. Pregled plana ishrane		

Alternativni tok: Odustajanje od registracije

Preduvjeti: Klijent je odustao od registracije prije nego što je potvrdio unos podataka u glavnom toku

Klijent	Sistem	API
1. Unos ličnih podataka		
2. Odabir načina ishrane		
3. Unos BMI podataka		
4. Klijent odustaje od registracije te se registracija prekida	5. Sistem odbacuje sve podatke koje je klijent unio	

Scenarij 2 – pregled preporučenog plana ishrane

Naziv	Pregled preporučenog plana ishrane
Opis	Klijent se prijavljuje na sistem te vrši pregled svog sedmičnog plana ishrane.
Vezani zahtjevi	/
Preduvjeti	Klijent posjeduje korisnički račun.
Posljedice - uspješan završetak	Klijentu se prikazuje sednični plan ishrane
Posljedice - neuspješan završetak	Korisnik je unio pogrešne podatke za prijavu.
Primarni akteri	Korisnik
Ostali akteri	Sistem

Glavni tok:

Klijent	Sistem	API
1. Unosi podatke za prijavu	2. Vrši validaciju podataka	
	3. Prikaz Sedmičnog plana ishrane	
4. Klijent nakon pregleda odabira alternativni tok 1 ili izlaz		

Alternativni tok 1: Pretraga svih planova

Preduvjeti: Klijent je odabrao pretragu svih planova u glavnom toku

Klijent	Sistem	API
1. Odabira pretragu svih planova ishrane		
2. Unosi parametre ishrane za pretragu	3. Parametre pretrage šalje API-u	4. Vrši pretragu planova na snovu primljenih parametara
	6. Prikaz rezultata pretrage	5. Vraća rezultat pretrage
7. Klijent, ako želi, sortira dobijene planove po nazivu		

Posljedice - uspješan završetak	Klijentu se prikazuje rezultat pretrage planova ishrane
Posljedice - neuspješan završetak	Pretraga je neuspješna (nije prikazan niti jedan rezultat)

Alternativni tok 2: Pogrešni login podaci

Preduvjeti: /

Klijent	Sistem	API
1. Unosi podatke za prijavu	2. Vrši validaciju podataka	
	3. Podaci nisu validni.	
	4. Omogućava korisniku ponovni unos podataka	

Posljedice - uspješan završetak	Klijent je prijavljen
Posljedice - neuspješan završetak	Klijent nije prijavljen

Alternativni tok 3: Ne posjedovanje korisničkog računa

Preduvjeti: klijent nije registrovan

Klijent	Sistem	API
1. Odabira pristup stranici za registrovanje	2. Preusmjerava korisnika na stranicu za registrovanje	

Scenarij 3 – upravljanje uposlenicima

Naziv	Pregled preporučenog plana ishrane
Opis	Administrator se prijavljuje na sistem te vrši CRUD uposlenika.
Vezani zahtjevi	/
Preduvjeti	Admin je prijavljen na sistem.
Posljedice - uspješan završetak	Admin dodaje/briše/ažurira uposlenike
Posljedice - neuspješan završetak	/
Primarni akteri	Korisnik
Ostali akteri	Sistem

Glavni tok:

Administrator	Sistem
1. Unosi podatke za prijavu	2. Vrši validaciju podataka
	3. Prikaz svih uposlenika
4. Odabira jedan od alternativnih tokova	

Alternativni tok 1: Brisanje uposlenika

Preduvjeti: Administrator je odabrao alternativni tok u glavnom toku

Administrator	Sistem
1. Odabir uposlenika kojeg želi brisati	
2. Potvrda o brisanju	3. Šalje obavijest uposleniku o prekidu radnog odnosa

Posljedice - uspješan završetak	Uposlenik je izbrisан из система
Posljedice - neuspješan završetak	/

Alternativni tok 2: Ažuriranje uposlenika

Preduvjeti: Administrator je odabrao alternativni tok u glavnom toku

Administrator	Sistem
1. Odabir uposlenika kojeg želi ažurirati	
2. Unos novih podataka	3. Validacija novih podataka
	4. Ažuriranje uposlenika
	5. Prikaz poruke o uspješnom ažuriranju

Posljedice - uspješan završetak	Podaci о упосленику су аžурирани
Posljedice - neuspješan završetak	Нови подаци нису валидни, упосленик nije аžурiran

Alternativni tok 3: Dodavanje novog uposlenika

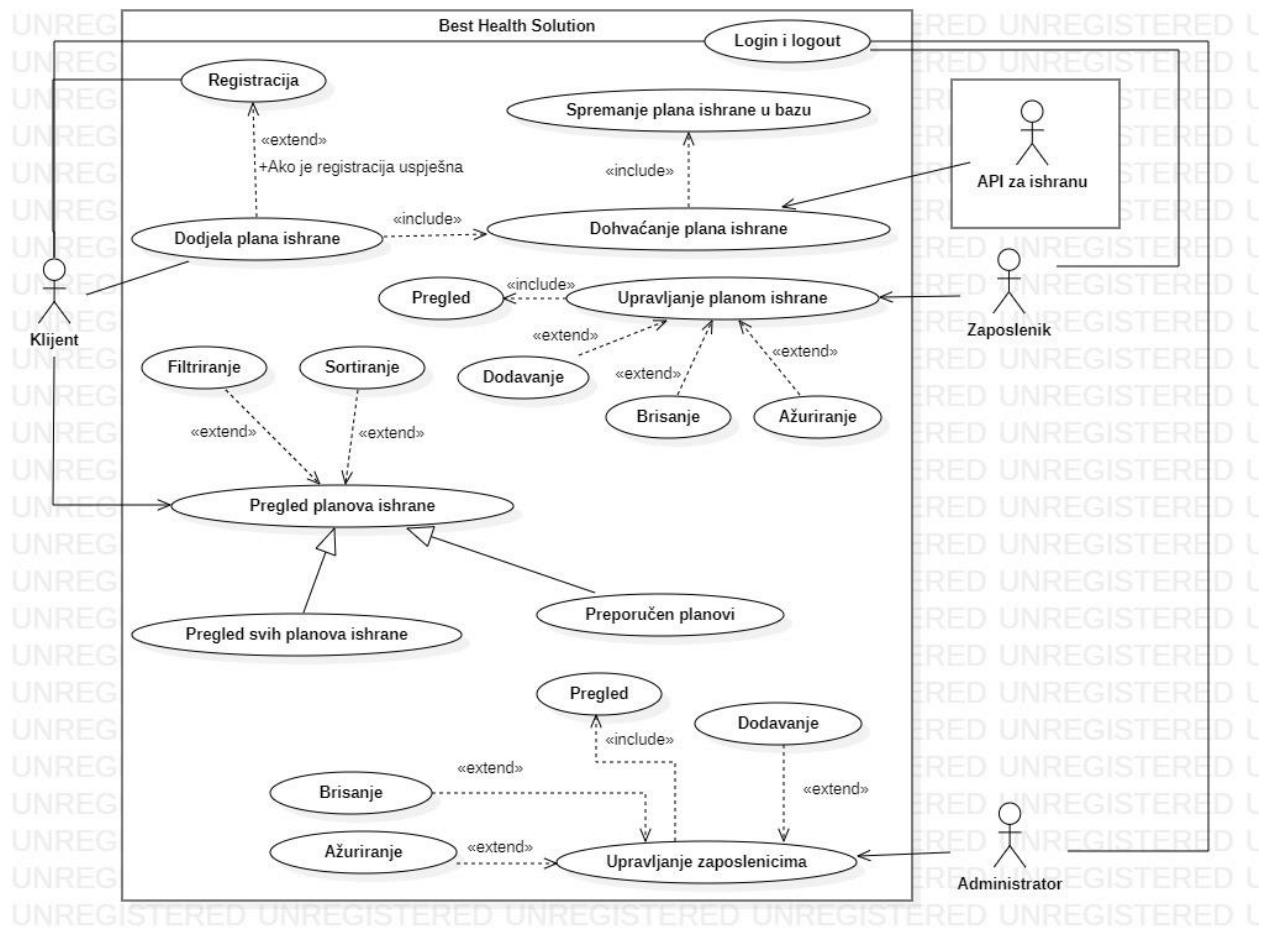
Preduvjeti: Administrator je odabrao alternativni tok u glavnom toku

Administrator	Sistem	Upozlenik
1. Unos podataka o novom upozleniku	3. Validacija novih podataka	
	4. Slanje potvrde novom upozleniku о пријему у радни однос	
		5. Prihvata posao
	6. Dodaje upozlenika у листу запослених	

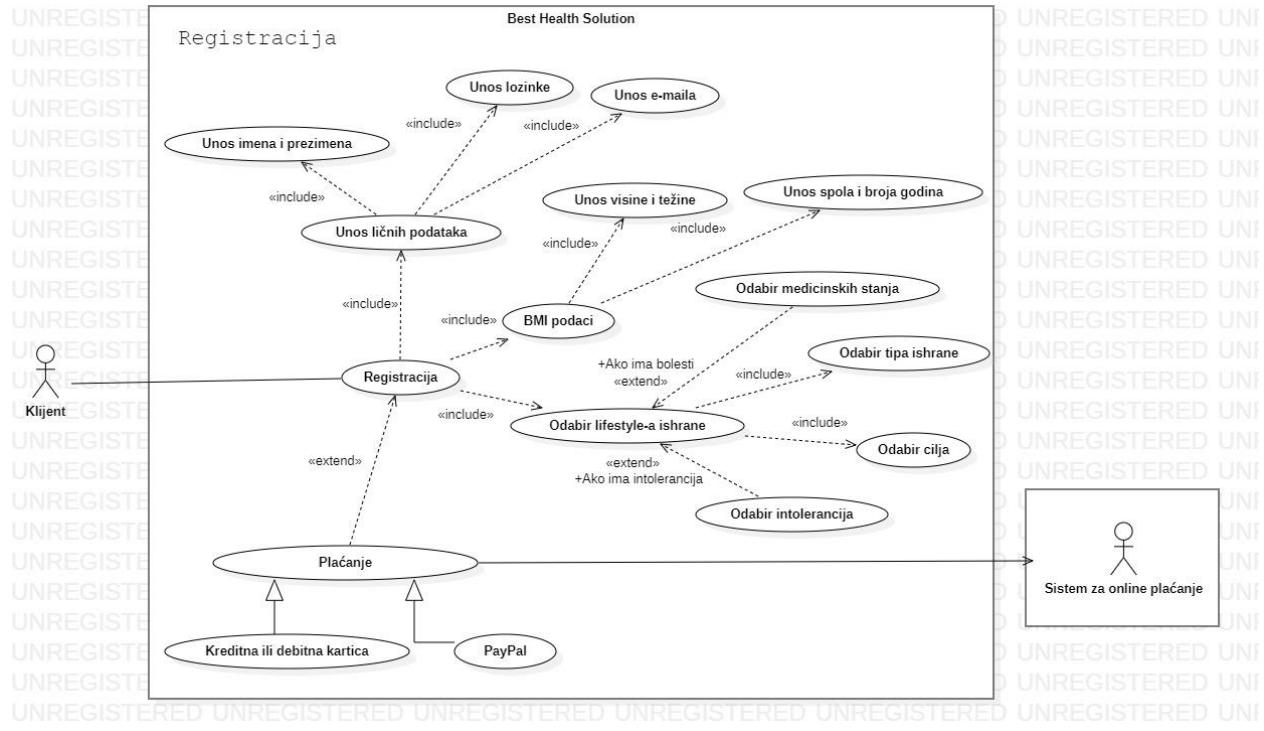
Posljedice - uspješan završetak	Upozlenik je prihvачен у радни однос.
Posljedice - neuspješan završetak	Upozlenik je odbio приступити у радни однос или nije одговорио на потврду за пријем у радни однос у roku од 7 dana

Dijagram slučajeva upotrebe

Opšti use-case dijagram

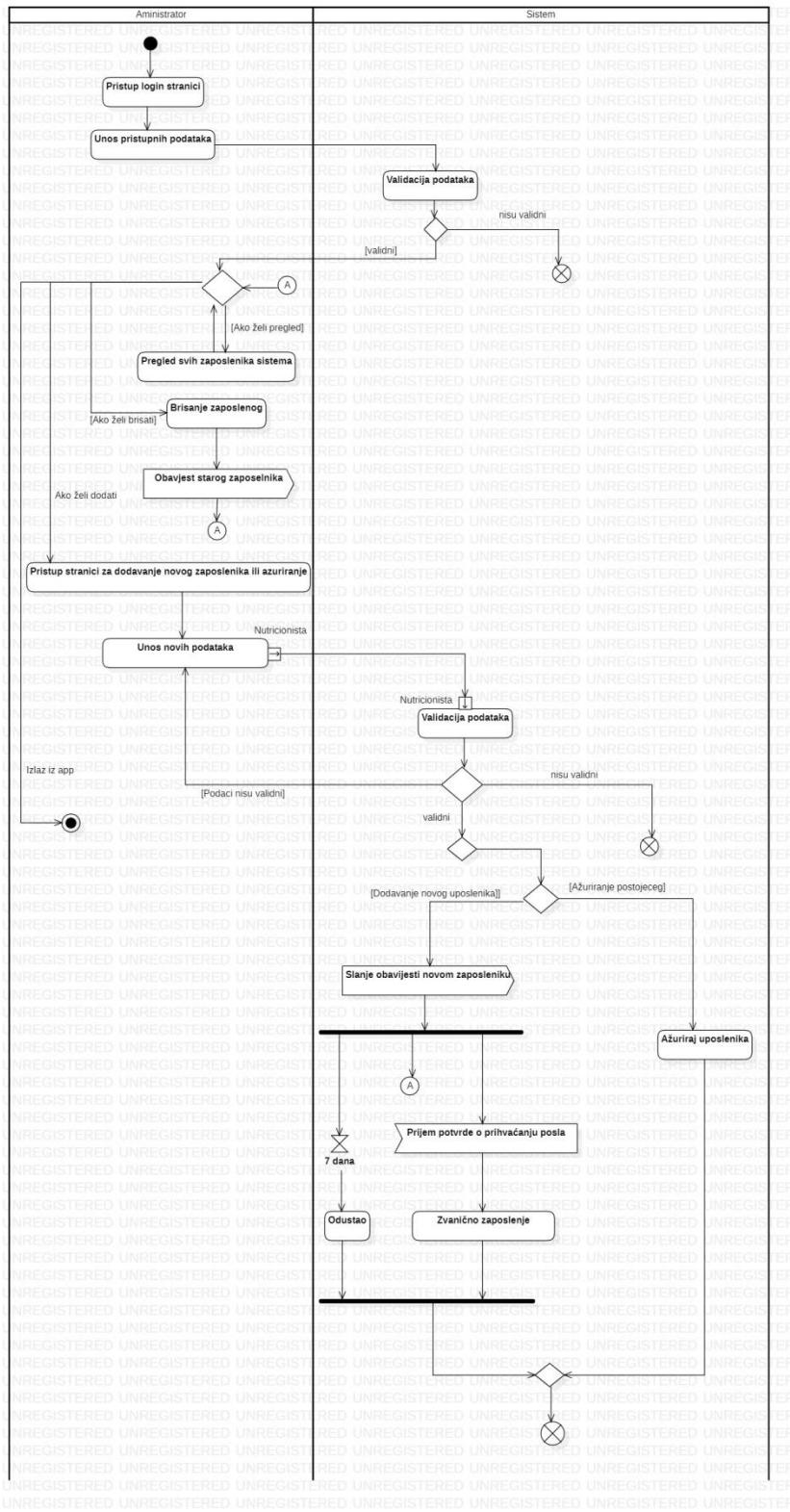


Registracija use-case diagram

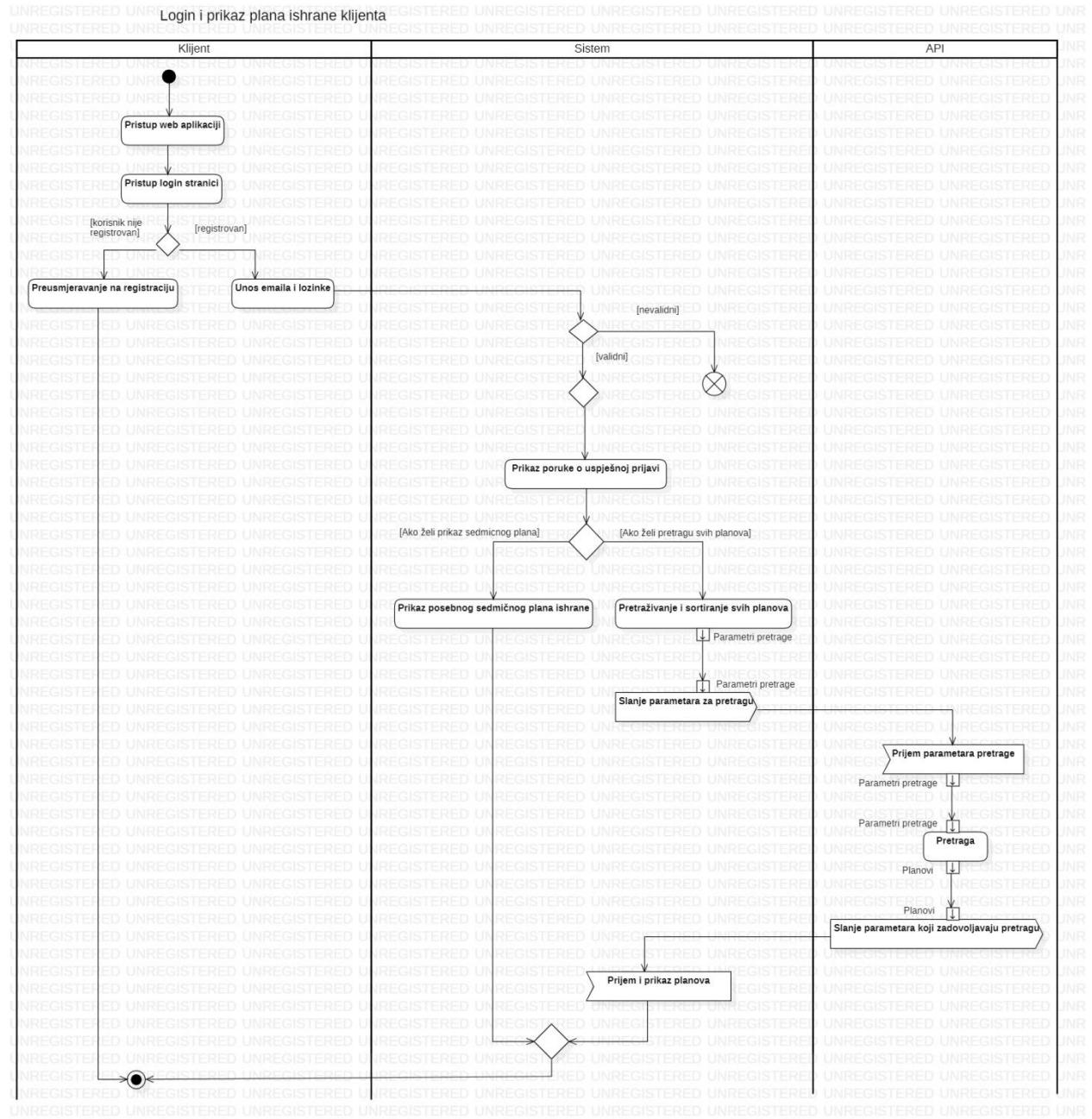


Dijagram aktivnosti

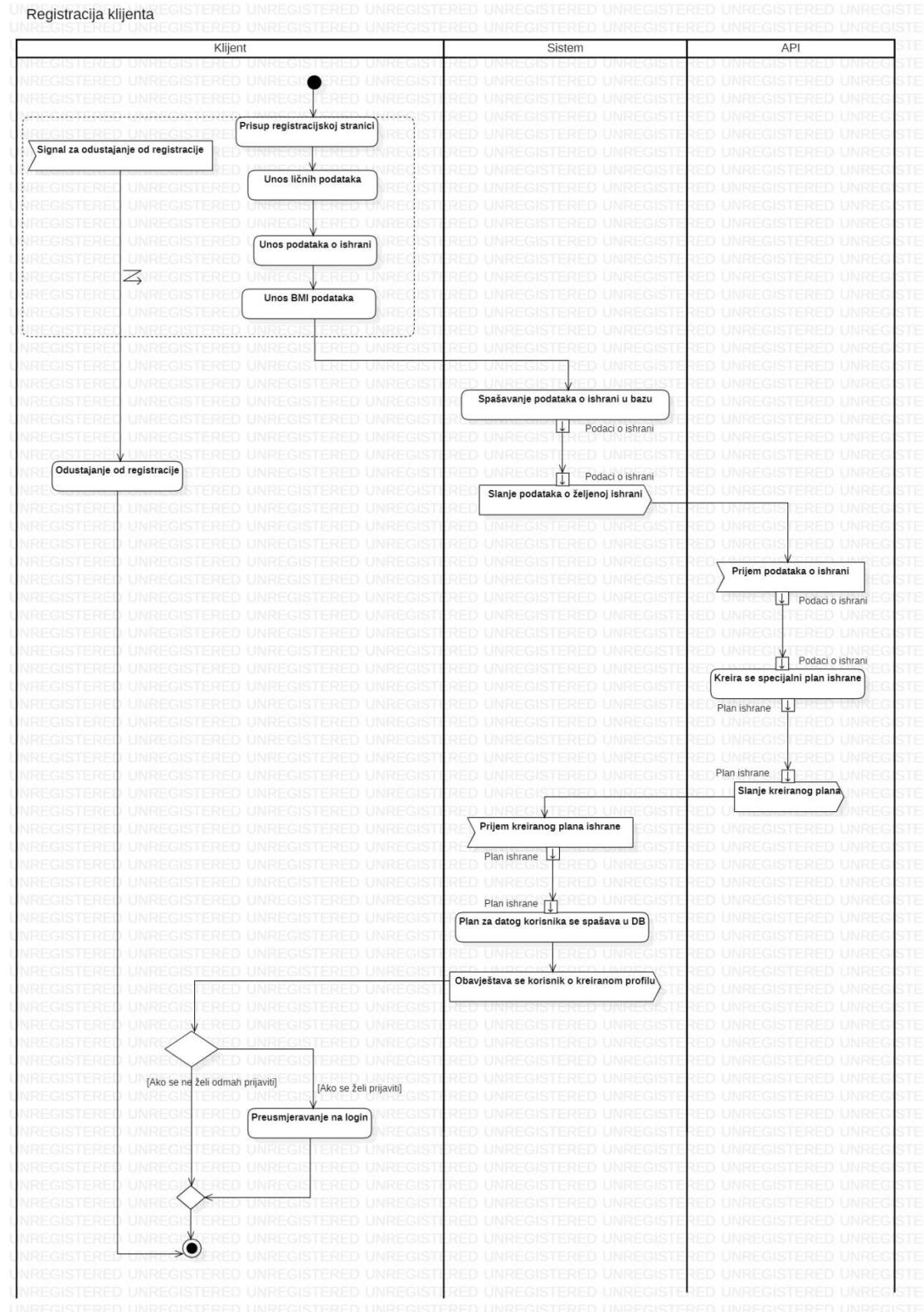
Administrator



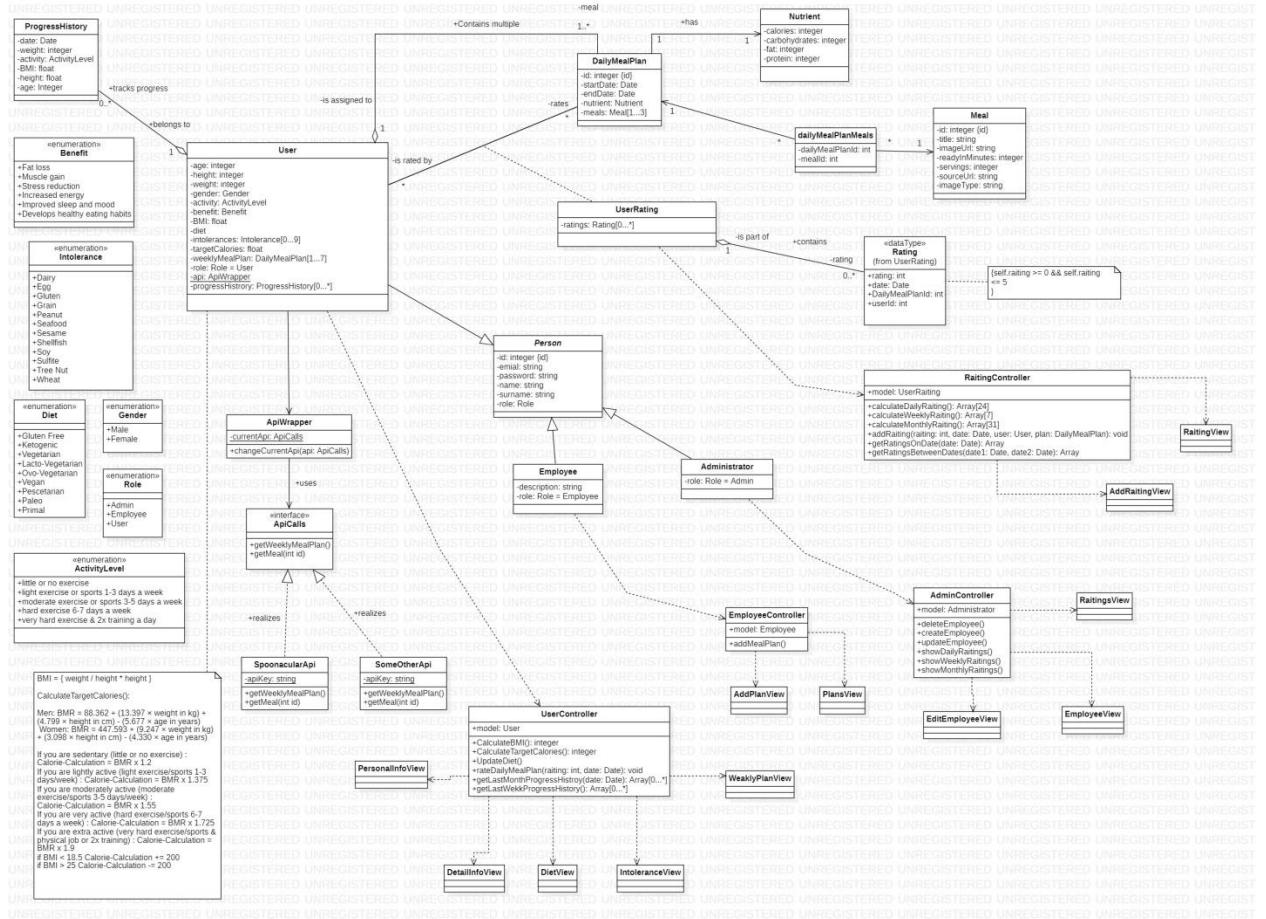
Login i prikaz ishrane



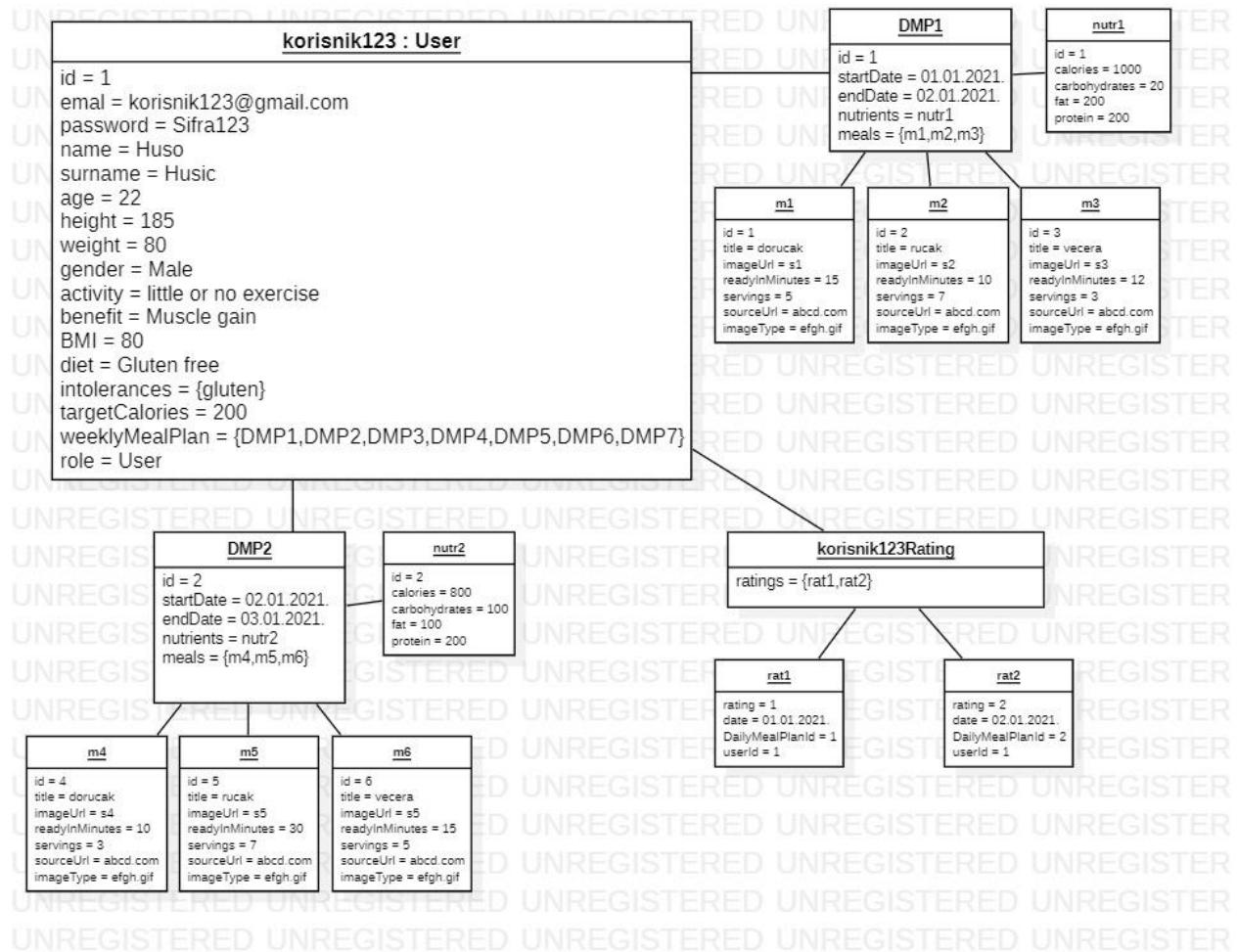
Registracija klijenta



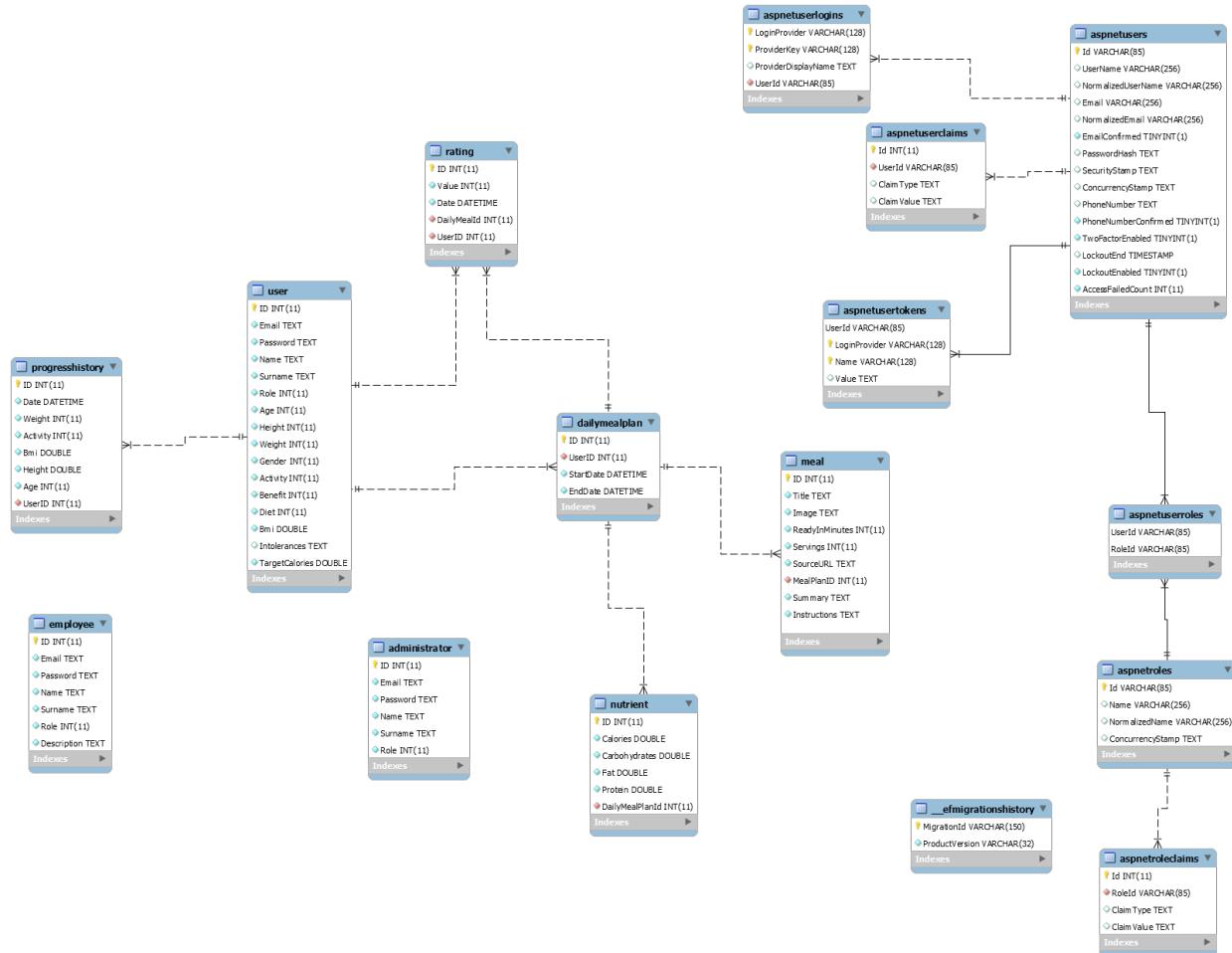
MVC Class diagram



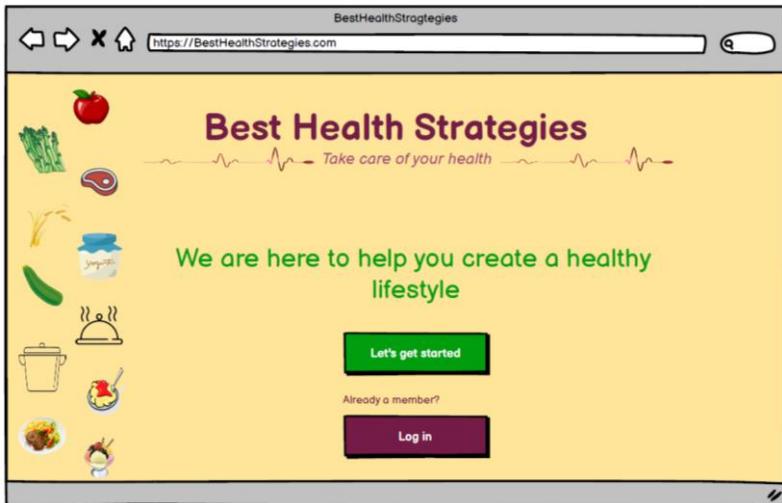
Dijagram objekata



ERD



Prototip



BestHealthStrategies
https://BestHealthStrategies.com/register

1 Tell me something about yourself

name
surname
e-mail
password
re-enter your password



BestHealthStrategies
https://BestHealthStrategies.com/register

2 I really wanna get to know you

age (years)
height in centimeters
weight in kilograms

male female

How active are you?

sedentary (little or no exercise)
 lightly active (light exercise/sports 1-3 days/week)
 moderately active (moderate exercise/sports 3-5 days/week)
 very active (hard exercise/sports 6-7 days/week)
 extra active (very hard exercise/sports & physical job or 2x training)



BestHealthStrategies
<https://BestHealthStrategies.com/register>

3 This is getting better

How would you describe your diet?



gluten free keto vegetarian lacto-vegetarian ovo-vegetarian

vegan pescatarian paleo primal whole30

gluten free: eating plan that excludes foods containing gluten; very low carb, high fat diet that shares many similarities with the Atkins and low carb diets.

keto: diet free of meat, fish, and fowl flesh.

vegetarian: variation of vegetarianism that excludes meat, poultry, seafood, and eggs.

lacto-vegetarian: excludes all animal-based foods except for eggs.

ovo-vegetarian: excludes all animal-based foods except for eggs.

vegan: contains only plants (such as vegetables, grains, nuts and fruits) and foods made from plants.

pescatarian: typically includes vegetables, grains and pulses along with fish and other seafood, but generally excludes meat and sometimes dairy.

paleo: typically includes lean meats, fish, fruits, vegetables, nuts and seeds — foods that in the past could

primal: lifestyle based on eating the foods that primitive humans would have eaten.

whole30: health movement that encourages followers to cut out alcohol.

BestHealthStrategies
<https://BestHealthStrategies.com/register>

4 Do you have anything to add?

Please let me know if there is something you are allergic to or simply don't eat.

Meat and Dairy	Fruits and Nuts	Vegetables	Grains, Beans and Legumes
beef chicken	avocado banana	zucchini broccoli	rice oats
pork turkey	orange berries	mushrooms cabbage	beans green peas
fish	apple almonds	eggplant spinach	pasta potato
eggs milk	nuts chia seeds	pepper tomato	whole grain bread
cottage cheese		asparagus	corn
greek yogurt			

BestHealthStrategies
https://BestHealthStrategies.com/mealplan

Welcome Sara!

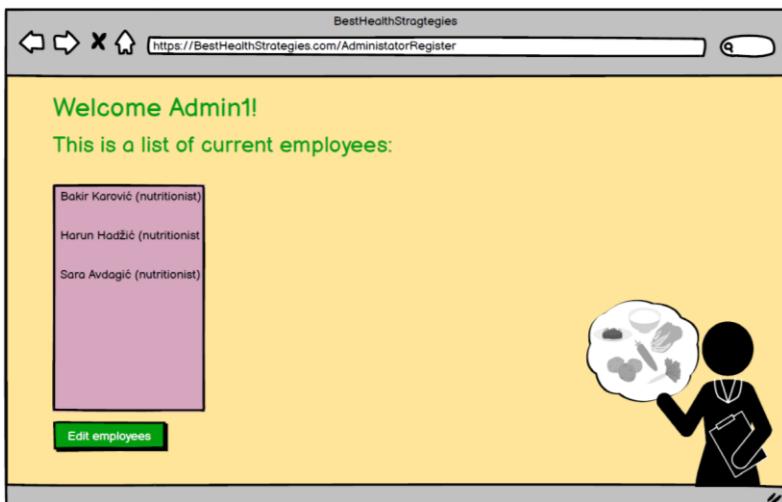
Let's see what we got for you...

A screenshot of a web browser showing a meal plan for Week 1, Day 1. The page has a yellow header and a green footer with grass. It displays three meals: Breakfast (3 egg whites with 1 yolk omelette with veggies), Lunch (Chicken breast, Quinoa & Mixed Veggies), and Dinner (Plain Greek Yogurt, topped with Berries + a dozen mixed nuts). Each meal is in a separate box with a small icon above it.

BestHealthStrategies
https://BestHealthStrategies.com/AdministratorRegister

Welcome Admin1!

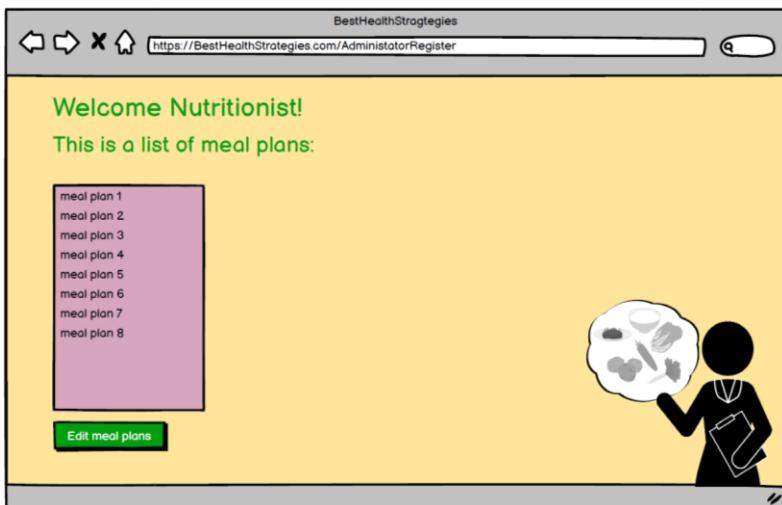
This is a list of current employees:

A screenshot of a web browser showing a list of employees. The page has a yellow header and a green footer with grass. On the left, there is a list of names: Bakir Karović (nutritionist), Harun Hadžić (nutritionist), and Sara Avdogić (nutritionist). On the right, there is a silhouette of a person holding a clipboard with a thought bubble containing a meal.

BestHealthStrategies
https://BestHealthStrategies.com/AdministratorRegister

Welcome Nutritionist!

This is a list of meal plans:

A screenshot of a web browser showing a list of meal plans. The page has a yellow header and a green footer with grass. On the left, there is a list of meal plans numbered 1 through 8. On the right, there is a silhouette of a person holding a clipboard with a thought bubble containing a meal.

SOLID principi

Single Responsibility Principle

KLASA BI TREBALA IMATI SAMO JEDAN RAZLOG ZA PROMJENU

Klasa User sadrži sve osnovne podatke o korisniku te metode koje izračunavaju BMI i TargetCalories. Ove metode služe samo klasi User.

Klasa DailyMealPlan sadrži podatke za dnevni plan ishrane. Pojedinačni podaci o nutrijentima tog dnevnog plana i podacima o jelima se nalaze u odvojenim klasama tako da plan ne ovisi o promjenitih klasa.

Klasa Raiting opisuje jedan raiting dnevnog plana ishrane. U klasi UserRaiting se nalazi svaki raiting za svaki dnevni plan zajedno sa static metodama koje ne ovise o drugim klasama.

Klasa Person opisuje svaku osobu. Ostale 3 klase Employee, Administrator i User dodaju opise na postojeću baznu klasu i ne zavise međusobno jedna od druge.

Open Closed Principle

ENTITETI SOFTVERA (KLASE, MODULI, FUNKCIJE) TREBALI BI BITI OTVORENI ZA NADOGRADNJU, ALI ZATVORENI ZA MODIFIKACIJE.

Ovaj princip je ispoštovan u svim klasama osim u klasi User jer ova klasa koristi Enumeration tipove podataka. Tako da naprimjer dodavanje nove stavke u enum tip bi za posljedicu imalo promjenu logike metoda u klasi User.

Svi ovi pobrojani tipovi ili zavise od api-a tako da se ne mogu automatski postaviti jer se mora ručno ući u dokumentaciju api-a i vidjeti koje sve pobrajene tipove nude, ili se neće mijenjati u budućnosti.

Liskov Substitution Principle

PODTIPOVI MORAJU BITI ZAMJENJIVI NJIHOVIM OSNOVNIM TIPOVIMA

Klase Administrator, Employee i User su zamjenljive njihovim podtipom Person jer sve 3 klase predstavljaju osobu.

Dodatno, sve klase koje implementiraju interfejs **ApiCalls**, mogu se koristiti na mjestima gdje se очekuje tip ApiCalls odsnosno u **ApiWrapper** klasi.

Interface Segregation Principle

KLIJENTI NE TREBA DA OVISE O METODAMA KOJE NEĆE UPOTREBLJAVATI

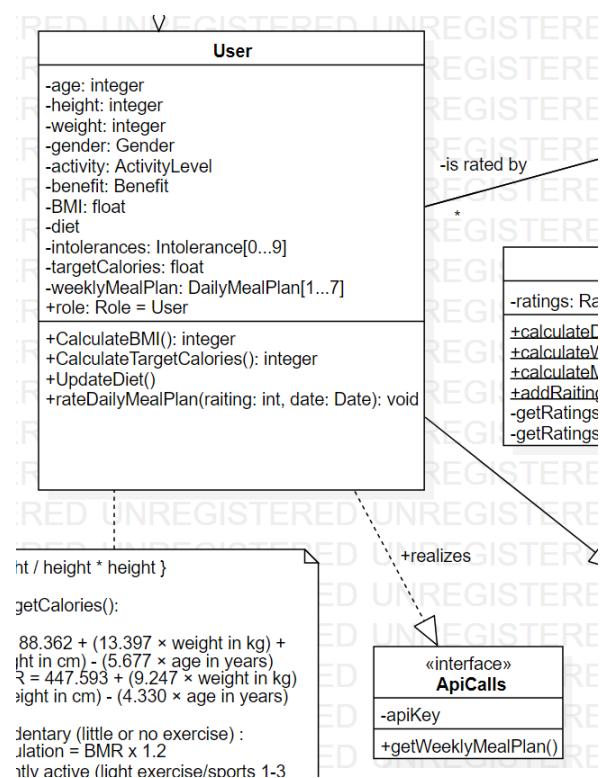
Klasa SpoonacularApi i svaka druga klasa koja možda bude dodana za drugi API implementira interfejs ApiCalls. Ova klasa koristi sve metode definisane u interfejsu tako da ga nije potrebno razdvajati na manje interfejse.

Dependency Inversion Principle

A. MODULI VISOKOG NIVOA NE BI TREBALI OVISITI OD MODULA NISKOG NIVOA. OBA BI TREBALO DA OVISE OD APSTRAKCIJA.

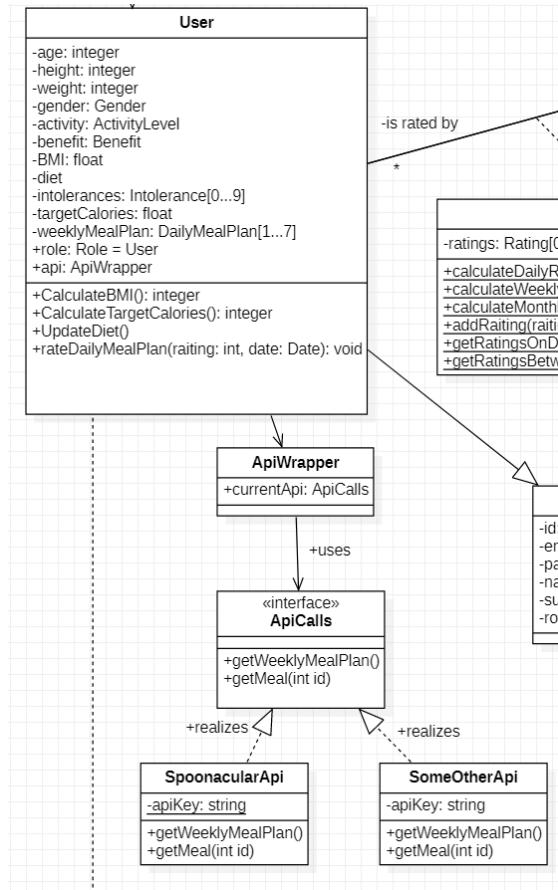
B. MODULI NE BI TREBALI OVISITI OD DETALJA. DETALJI BI TREBALI BITI OVISNI OD APSTRAKCIJA.

Ovaj princip nije bio ispoštovan. Ako bi naša aplikacija prestala koristit Spoonacular API i počela koristiti neki drugi, morali bi promijeniti implementaciju interfejsa ApiCalls što je naš high level kod.

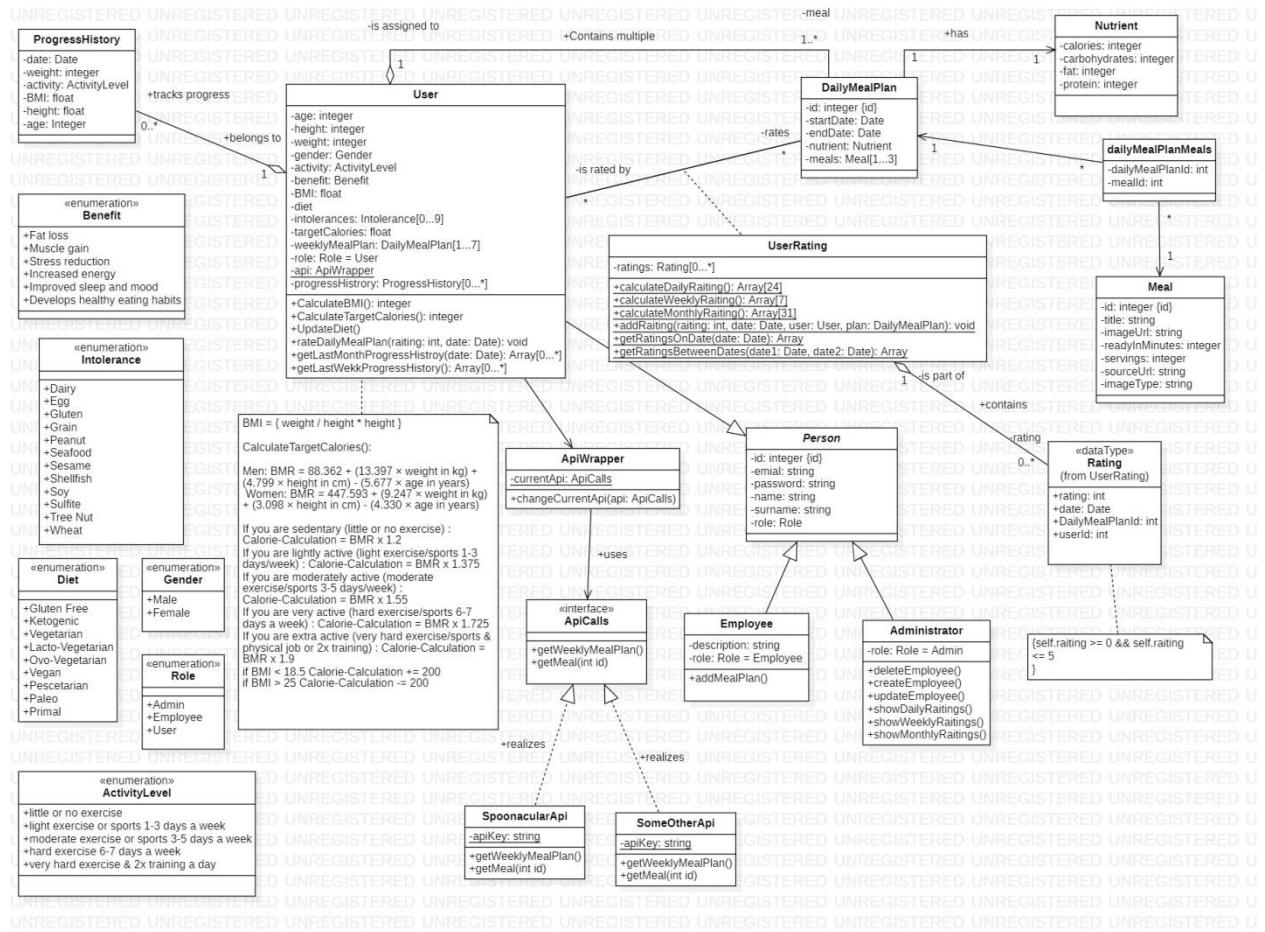


Rješenje je da koristimo neki wrapper koji će predstavljati posrednika između klase User i API-a.

Taj wrapper može biti interface koji će koristiti posebne API klase gdje će svaka klasa imati iste metode ali implementirane na različite načine, kao na slici ispod.



Dijagram klasa sa SOLID principima



Dizajn paterni

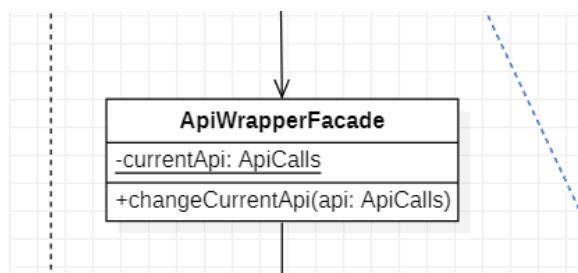
STRUKTURALNI PATERNI

1. ADAPTER

Za potrebe dodavanja ovog paterna, uvest ćemo funkcionalnost plaćanja u naš sistem. Zamislimo da se pored kartičnog plaćanja, klijentu želi omogućiti i plaćanje putem PayPal sistema. Tu bismo mogli dodati ovaj patern na način da kreiramo interfejs IPlaćanje koji će naslijediti adapter klasu , te kreiramo klasu vrstaPlaćanja, kako bismo u budućnosti mogli dodati još vrsta plaćanja (sada dodajemo PayPal plaćanje). Kreirat ćemo i VrstaPlaćanjaAdapter klasu koja će izvršiti adaptiranje vrste plaćanja iz kartičnog u PayPal (ukoliko je to ono što nam treba).

2. FACADE

Pozivom API-ja prave se jedinstveni dnevni planovi ishrane za svakog korisnika. Svaki dnevni plan se sastoji od više obroka: doručka, ručka i večere gdje svaki obrok sadrži određene vrste namirnica i nutrijenata, te također poštaje način prehrane klijenta i njegove specifične osobine. Vidimo da naša klasa *APIWrapper* već pojednostavljuje kreiranje planova ishrane nutricionisti jer komunicira sa klasama *Nutrient*, *DailyMealPlan*, *dailyMealPlanMeals*, *Meal* i *User* te tako prikuplja sve gore potrebne informacije pomoću kojih kreira jedinstveni plan ishrane, stoga ćemo klasu *APIWrapper* proglašiti fasada klasom. Nije nam potrebno da znamo internu implementaciju ili neke pogodnosti korištenog API-ja, dovoljno ga je samo “pozvati”.



3. DECORATOR

Za potrebe implementiranja ovog strukturalnog paterna, dodat ćemo novi korisnički zahtjev – svaki korisnik će imati svoju profilnu sliku, te će mu biti omogućena

modifikacija (rotacija, rezanje) iste. Cilj dodavanja ovog paterna je da se krajnjem korisniku olakša i unaprijedi korištenje sistema. Kreirat ćemo zasebnu klasu *Slika* koja će sadržavati atribute ime:String i slika:Bitmap. Potrebno je dodati interfejs *IslikaProfila*, kao i tri nove klase *SlikaUpdate*, *SlikaRezanje*, *SlikaRotacija*. U interfejs je potrebno implementirati metode uredi i dajSliku, a ostale tri klase će naslijediti interfejs.

Bitno je napomenuti da bi osnovna vrsta slike imala atribut tipa Slika, a ostale bi imale tipa ISlika, čime bi se osigurao tok akcija.

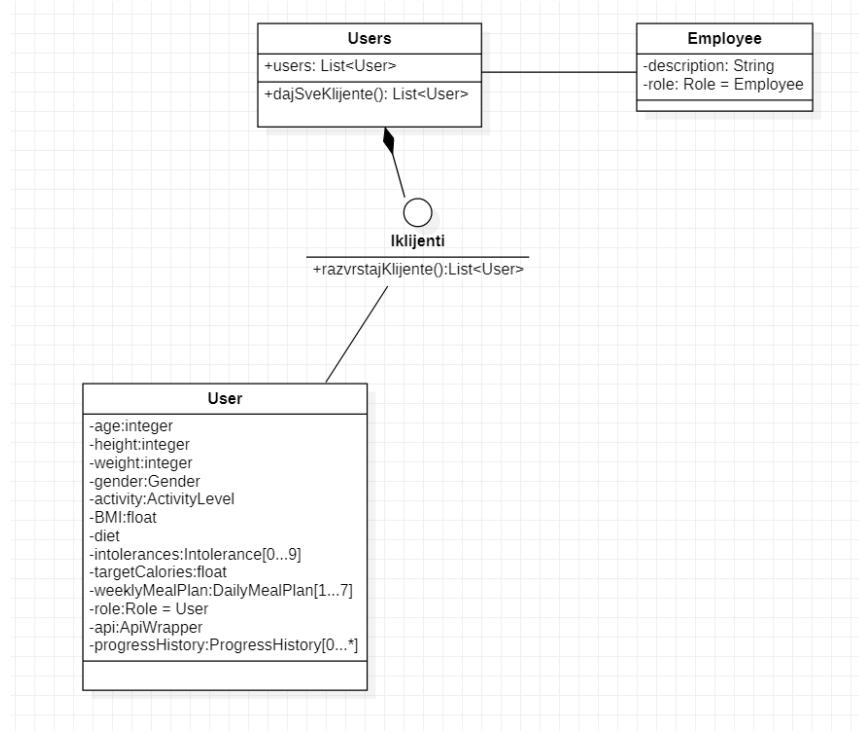
4. BRIDGE

Za potrebe implementacije ovog paterna, uvest ćemo novu funkcionalnost. U momentu registracije, klijentu se nudi mogućnost uplaćivanja jednog mjeseca ili plaćanja istovremeno za naredna 2/3 mjeseca. Oni klijenti koji se odluče za plaćanje više od jednog mjeseca istovremeno, ostvaruju određene pogodnosti u vidu popusta na cijenu koju plaćaju za korištenje sistema (za plaćanje 2 mjeseca istovremeno ostvaruje se 20% popusta na ukupnu cijenu, dok za plaćanje 3 mjeseca istovremeno ostvaruje 30% popusta na ukupnu cijenu). Tada bi bilo potrebno dodati novi interfejs *IdodatnePogodnosti*, koji će sadržavati definiciju metode za izračun cijene koju plaćaju klijenti. Također, bilo bi potrebno dodati klasu *Bridge*, koja će sadržavati apstrakciju i kojoj će klijenti jedino imati pristup, ukoliko žele pregledati cijenu pogodnosti na osnovu njihovog korisničkog računa. Te bilo bi potrebno dodati atribut koeficijent: double u, već postojeću klasu, *User*, na osnovu kojeg bi se vršio izračun cijene.

5. COMPOSITE

Ovaj patern u našem sistemu možemo implementirati na sljedeći način. Ukoliko želimo dopustiti nutricionisti pregled svih korisnika u aplikaciji podijeljenih na osnovu naziva prehrane (Gluten Free, Ketogenic, Vegetarian, Lacto-Vegetarian, Ovo-Vegetarian, Vegan, Pesceterian, Paleo, Primal), trebali bismo za potrebe ovog paterna, kreirati klasu *Users* koja će imati atribut users>List<User> i u koju ćemo dodati medotu dajSveKlijente>List<User> (*primjer vraćenog rezultata: „Gluten Free: Sara Sarić, Bakir Bakirić,... Ketogenic: Harun Harunić,...“*), koju će nutricionista moći pozvati i koja će vratiti sve klijente registrovane u sistemu sortirati po vrsti prehrane. Zatim, kreirati interfejs *IKlijenti* u koji ćemo implementirati definiciju metode razvrstajKlijente za sortiranje klijenata po vrsti prehrane. Klasa *Users* će naslijediti interfejs kako bi se kreirala hijerarhija objekata. Klasa *User* će zadržati

atribut diet (koji će čuvati informaciju o načinu prehrane) i koji će koristiti pri sortiranju.



6. PROXY

Ovaj patern ćemo u našem sistemu implementirati na sljedeći način. Prava pristupa za pregled svih planova ishrane su ograničena. Svim planovima ishrane mogu pristupiti samo zaposleni - nutricionisti, ali ne i klijent ili admin. Vrši se provjera pristupnih podataka, te se na osnovu njih određuje da li se radi o klijentu ili nutricionisti i shodno tome omogućuje pristup planovima ishrane. Potrebno je definirati interfejs **IDailyMealPlans**, te definirati novu klasu **Proxy** koja će sadržavati atribut `nivoPristupa:int` (za potrebe određivanja da li pregled zahtijeva admin, nutricionista ili klijent), te planove ishrane `dailyMealPlans:IDailyMealPlan`. Ova klasa će naslijediti interfejs i njegove metode.

7. FLYWEIGHT

Za potrebe dodavanja ovog pattern-a, uvest ćemo novu funkcionalnost. Poznato nam je da prilikom registracije klijent bira način prehrane. Mogli bismo sve načine prehrane razdvojiti u različite kategorije koje bi nutricionista mogao pregledati i uz pomoć određenih funkcionalnosti dodati/izbaciti pojedine namirnice iz plana ishrane/obroka u zavisnosti od same kategorije u kojoj se nalazi. Na taj način postigli bismo da npr. kategorija Gluten Free može sadržavati isti obrok kao i neka druga kategorija uz zamjenu običnih peciva bezglutenским i korištenje bezgluteneskog brašna u pripremi obroka. Nutricionista nakon prikupljanja informacija o klijentu može pristupiti različitoj kategoriji u zavisnosti od klijentovih želja i potreba i tako pripremiti njegov unikatni plan ishrane. Nećemo previše ulaziti u detalje, jer su isti objašnjeni u dokumentu Kreacijski Paterni, no za potrebe dodavanja ovog paterna (kao i nekih kreacijskih paterna) mogli bismo i dodati postojanje takozvanog baznog obroka koji bi nutricionisti uveliko olakšao kreiranje plana ishrane, jer bi se bazni obrok mogao lahko modificirati u zavisnosti od načina prehranepojedinog klijenta.

KREACIJSKI PATERNI

1. SINGLETON

Obzirom da naš API obavlja glavni dio posla – kreiranje planova ishrane, a on je u vezi sa jedinstvenom klasom ApiWrapper koji komunicira sa svim klasama i na taj način prikuplja sve potrebne informacije za samo kreiranje konkretnog plana ishrane, potrebno je osigurati jedinstveno instanciranje klase ApiWrapper pomoću Singleton paterna. Singleton patern će biti implementiran u klasu ApiWrapper, a sadržavat će privatnu static varijablu koja čuva jednu/jedinstvenu instancu klase, javnu static metodu (getInstance) preko koje će se pristupati Singleton klasi i privatni static objekat koji se interno instancira korištenjem privatnog konstruktora.

2. PROTOTYPE

Klijent prilikom registracije unosi svoje specifikacije koje kasnije pomažu nutricionistu da kreira jedinstven plan ishrane. Neki planovi ishrane mogu se razlikovati samo u jednoj namirnici, konkretan primjer bi bio da imamo klijenta sa alergijom na orašaste plodove te bi se njegov pojedini obrok razlikovao samo u tome da ne bi uključivao neki od plodova koji mu izazivaju alergiju. Stoga vidimo da bi se mnogi planovi ishrane razlikovali u minimalnim stvarima, tako da bi se neki planovi ishrane mogli iskoristiti za više klijenata uz neke minimalne modifikacije tj. mogao bi postojati neki osnovni obrok koji bi odgovarao svim/mnogim korisnicima, a onda bi se takav obrok nadograđivao

namirnicima koje određenom klijentu ne smetaju. Nakon kopiranja objekta (obroka), nutricionista može mijenjati njegove karakteristike bez da to utiče na originalni objekat. Mogli bismo napraviti interfejs IKopiraj sa metodom kopiraj() koji bi implementirao klasu Meal. Klasa Meal će implementirati metodu kopiraj za ranije objašnjene potrebe.

3. FACTORY METHOD

Za objašnjenje načina na koji ćemo dodati ovaj patern u naš sistem, nadovezat ću se na ideju izloženu u objašnjenju prethodnog paterna. Naime imat ćemo neki bazni obrok. Sada bismo mogli dodati klase koje dodaju/oduzimaju sastojke u zavisnosti od načina prehrane koji je klijent odabrao kao prihvatljivi. Tako bi recimo postojao bazni obrok, a onda bi se u slučaju da se radi o klijentu koji ima netoleranciju na gluten, mogle dodati još neke namirnice koje će obogatiti obrok, ali se neće kosit sa njegovom intolerancijom. Dakle ideja je da se tvorcu plana ishrane (u našem slučaju je to API pa ne moramo ulaziti u detalje načina na koji će API izvesti neke akcije) olakša kreiranje dnevnog/sedmičnog plana ishrane tako da se odvoje različite kategorije koje prate različite načine ishrane. Također dodavanje ovog paterna u naš sistem bi bilo od koristi u slučaju da se pojave neki novi načini ishrane koje je potrebno dodati u postojeći sistem.

4. ABSTRACT FACTORY

Za dodavanje ovog paterna, također ćemo se osvrnuti na različite načine prehrane i specifikacije klijenta. Iako za nas API obavlja kreiranje plana ishrane uz korištenje informacija koje mu prikupimo, možemo zamisliti jedan od mogućih scenarija kako bi to API mogao raditi. Kako smo ranije objasnili, naš klijent prilikom registracije ispunjava 5 stranica o ličnim specifikacijama – fizička spremna, težina i visina, način prehrane, BMI. Mogli bismo kreirati apstraktne fabrike – FactoryBMI, FactoryActivityLevel itd. iz kojih bi bile naslijedene konkretne fabrike npr HighActivityLevelFactory. Recimo razne klase koje bi predstavljale različite “teme” u zavisnosti od načina ishrane – Gluten Free, Ketogenic itd. bi se bavile različitom problematikom i pomagale nutricionisti da kreira plan ishrane. Ukoliko odlučimo u nekom trenu da dodamo još dodatnih specifikacija s ciljem što preciznijeg kreiranja plana ishrane, samo bismo dodali još apstraktnih fabrika te vidimo da bi se dodavanje ovog paterna moglo pokazati korisnim jer postojeće specifikacije neće biti narušene.

5. BUILDER

Ovaj patern bismo mogli iskoristiti kod kreiranja objekata tipa User jer klasa User sadrži veći broj atributa koji bi se mogli odvojeno definisati, ali idalje ne prevelik broj atributa tako da nije kompleksno ni naše postojeće rješenje. No to bi se moglo promijeniti ukoliko se klasa User nadogradi sa mnogo više informacija o korisniku i u tom slučaju bi ovaj patern bio itekako od koristi. U slučaju implementacije Builder

paterna, postojao bi interfejs IBuilder sa različitim dijelovima kreiranja korisnika, npr. metode dodajImeIPrezime(String ime, String prezime), dodajMjestoRodjenja(String drzava, String mjestoRodjenja), dodajSliku(String url) itd. Također postojali bi i različiti builderi - BuilderUser bi pozvao samo osnovne metode, dok bi BuilderUserVip pozvao i dodatne metode koje bi bile u skladu sa mogućnostima tog VIP korisnika (trenutno takav korsnik nije predviđen, no za potrebe dodavanja ovog paterna u naš sistem, ta funkcionalnost bi se mogla uvesti).

PATERNI PONAŠANJA

1. STRATEGY

Ovaj patern možemo dodati u naš sistem na sljedeći način. Poznato nam je da unutar našeg sistema imamo 3 vrste korisnika: admin-a, zaposlenog (nutricionistu) te samog klijenta. Stoga prilikom samog prijavljivanja korisnika, imamo sigurno 3 opcije prilikom registracije. Detekcija o kojem od tri nabrojana korisnika se radi, vrši se na osnovu ulaznih podataka. Na osnovu datog tipa klase tj. korisnika mogli bismo realizovati 'set-up' nakon registracije u kojem bi za svaki tip korisnika postojao drugačiji prikaz (znamo da ova tri krosnika mogu pristupiti i vidjeti različite stvari), samim tim možemo i drugačije podatke dobiti pomoću api request-a prema našoj bazi. Još jedan primjer gdje bi se mogao dodati ovaj patern odnosi se na različite planove ishrane. U zavisnosti od ulaznih podataka klijenta, nutricionista treba da izabere "strategiju" tj. tip prehrane koji odgovara ulaznim podacima klijenta.

2. STATE

Obzirom da trenutno ne vidimo neki koristan način da se ovaj patern doda u naš sistem, osmislit ćemo jednu situaciju nadovezujući se na dokument o kreacijskim paternima. U tom dokumentu je opisano postojanje takozvanog baznog obroka, čiji je cilj olakšati nutricionisti da osmisli plan ishrane. Zamislimo situaciju u kojoj postoji klijent koji ima intoleranciju na gluten i alergičan je na orašaste plodove, te klijent koji je alergičan na orašaste plodove i jagode. Mogli bismo osmisliti još nebrojeno mnogo ovakvih kombinacija. To je mjesto gdje se potencijalno javlja potreba za dodavanjem state paterna. Mogli bismo napraviti interfejs IStanjaKlijenta koji komunicira sa različitim stanjima – alergijama, preferencijama, tipovima ishrane (ovo bi sve bile klase), a klasa Meal (obrok) bi ustvari predstavljala context klasu.

3. TEMPLATE METHOD

Ovaj patern možemo dodati u naš sistem na sljedeći način. Obzirom da API za nas radi samo kreiranje plana ishrane, te sam način na koji to radi nije nam od velikog interesa bio dosad, zamislit ćemo ponovo hipotetičku situaciju i nadovezati ćemo se na prethodni patern i postojanje baznog obroka. Za dodavanje ovog paterna bit će nam potrebna apstraktna klasa BazniObrok sa metodama templateMethod, filtrirajNamirnice. U ovisnosti od toga kakav je tip ishrane određenog klijenta, ove metode će biti različito implementirane. Za konkretni tip ishrane bit će primijenjena konkretna klasa od 9 mogućih: Gluten Free, Ketogenic... U svakoj od navedenih klasa, metoda filtrirajOpcenito bi izdvajala one namirnice koje nisu podržane u toj dijeti. Na taj način bi se omogućilo olakšano kreiranje unikatnog plana ishrane za svakog klijenta.

4. OBSERVER

Ovaj patern možemo dodati u naš sistem na sljedeći način. Poznato je da sistem posjeduje funkcionalnost koja omogućava klijentu da ažurira svoj napredak. Obzirom da je nutricionista taj koji pravi planove ishrane, bilo bi korisno da može da prati i napredak svojih klijenata. Tako da ćemo observer patern dodati na način da ćemo omogućiti nutricionisti da svakoga puta kada bilo koji klijent ažurira svoj napredak, on dobije obavijest o tome. Tako će nutricionista biti u stanju da vidi kako napredak klijenata, tako i efikasnost pojedinih planova ishrane.

5. ITERATOR

Ovaj patern možemo dodati u naš sistem na sljedeći način. Recimo da nutricionista želi da izvrši pregled klijenata. Default-ni način bi bio po abecednom redu. No u slučaju da nutricionista želi da pregleda klijente sortirane po BMI – u (npr. od manjeg ka većem) ili sortirane po tipu ishrane (počevši npr. od Gluten Free). U opisanoj situaciji, ovaj patern bi moga biti od koristi.

6. CHAIN OF RESPONSIBILITY

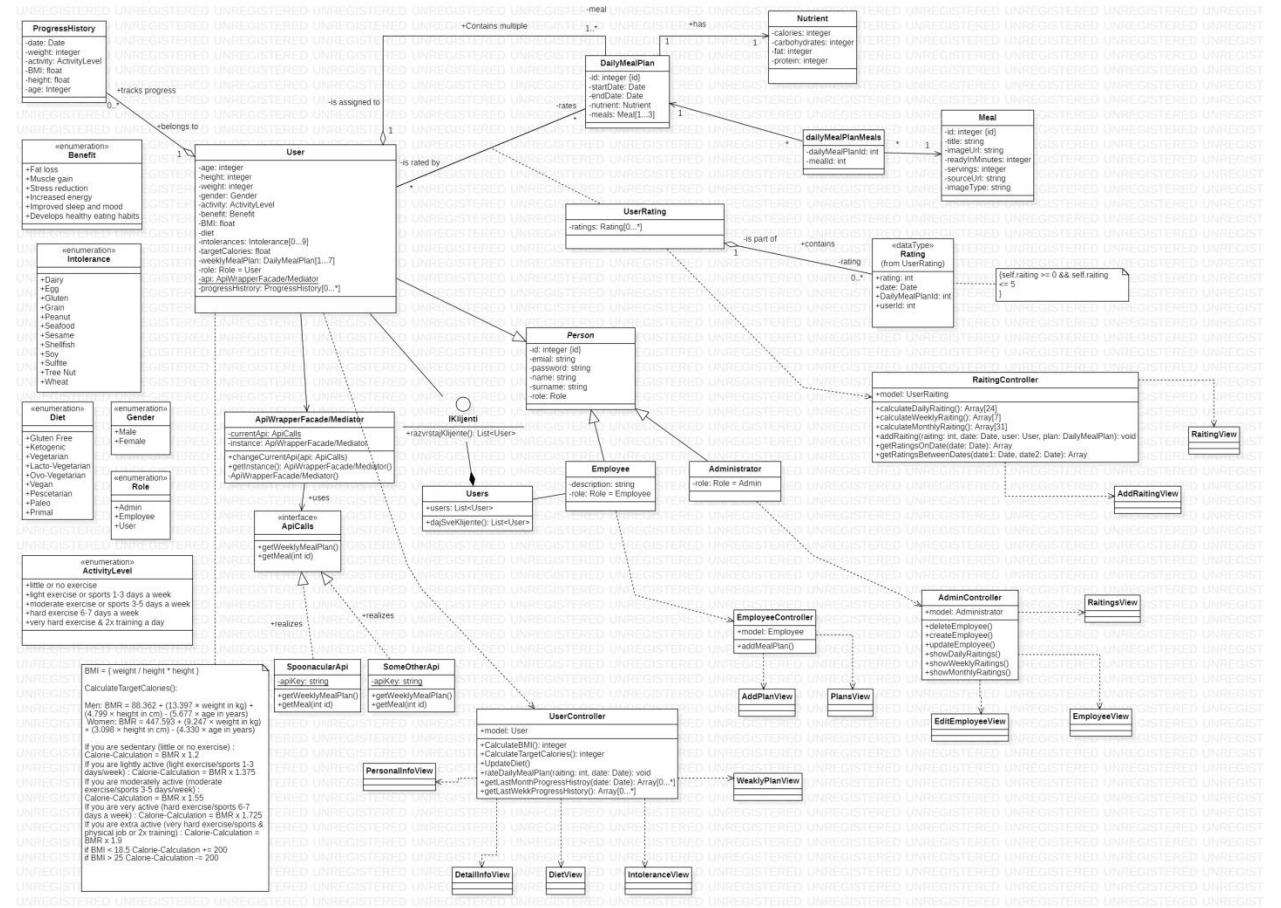
Ovaj patern možemo dodati u naš sistem uvezši opet u obzir postojanje tzv. Baznog obroka. Ukoliko nutricionista kreira plan ishrane sačinjen od 3 obroka prema potrebama svakog klijenta, desit će se da se neki obroci razlikuju u možda svega nekoliko namirnica. Tako bi uz uvažavanje klijentovih potreba, na već postojeći bazni obrok bile dodane neke namirnice, ili oduzete u slučaju da ih klijent ne podnosi. Prošlo bi se kroz tip ishrane koji je klijent označio kao željeni i izvršila detekcija onih namirnica koje nisu dopuštene u tom tipu ishrane, potom bi se provjerilo da li obrok zadovoljava nutritivne vrijednosti koje su potrebne za

konkretnog klijenta, a na samom kraju bi se izdvojile one namirnice na koje klijent posjeduje alergiju ili intoleranciju.

7. MEDIATOR

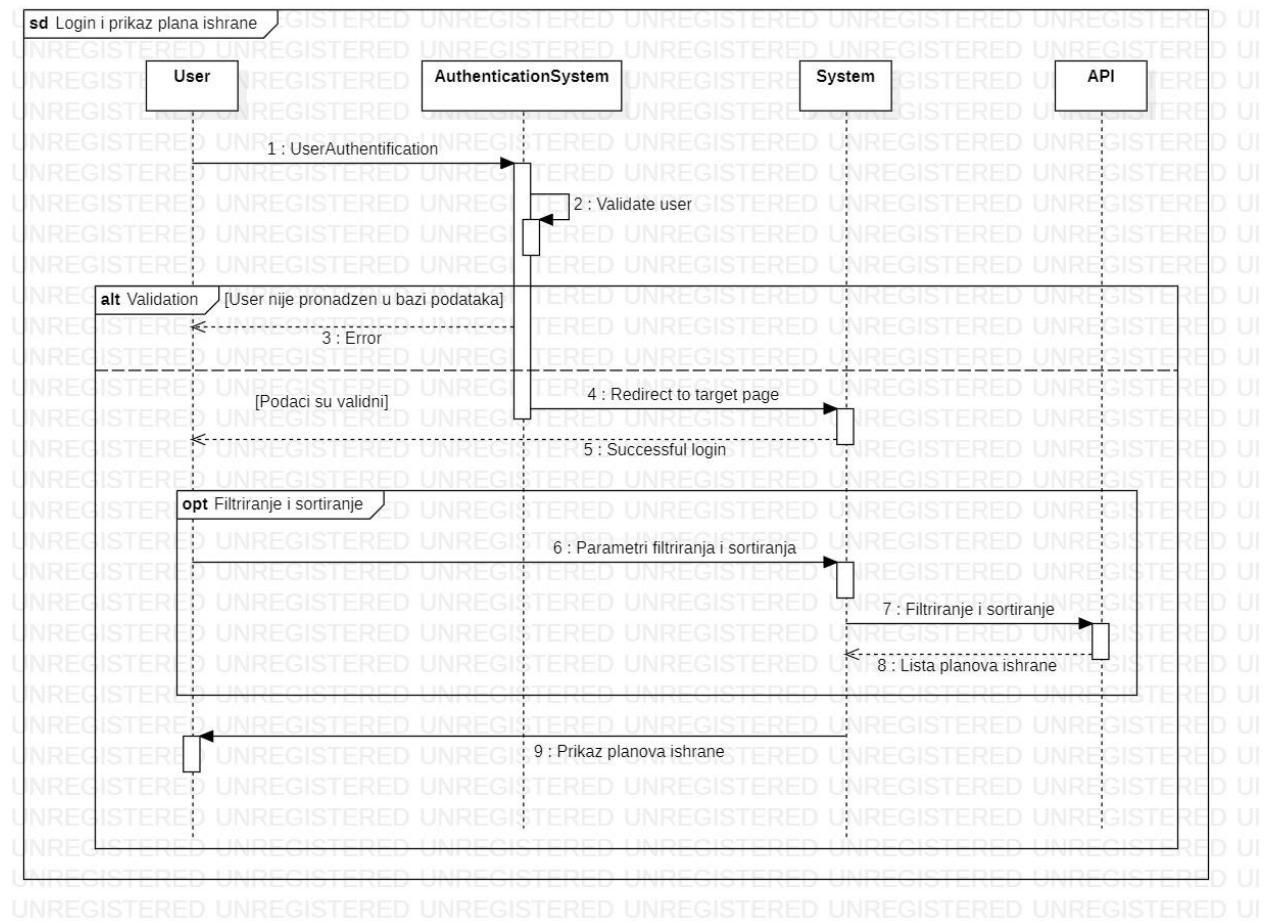
Obzirom da mediator i facade patern rade sličan posao, a mi smo facade patern dodali na način da smo ApiWrapper klasu proglasili fasadnom klasom obzirom da ona vrši komunikaciju sa svim drugim klasama i obavlja za nas posao kreiranja plana ishrane, što ujedno predstavlja srž našeg sistema, i ovaj patern ćemo dodati u naš sistem proglašavanjem ApiWrapper mediator klasom. Naime čitav proces kreiranja planova ishrane prilično je haotičan. Postoji mnogo zahtjeva koje klijent može napraviti, od tipa ishrane, BMI-a, alergija, intolerancija itd. Usaglašavanje svih tih zahtjeva bilo bi poput dodjeljivanja prednosti vozačima na raskrsnici, a policajac u našem slučaju je upravo ApiWrapper klasa koja taj haotični proces uobiči i da nam finalni product – plan ishrane za svakog klijenta.

Class diagram sa dizajn paternima

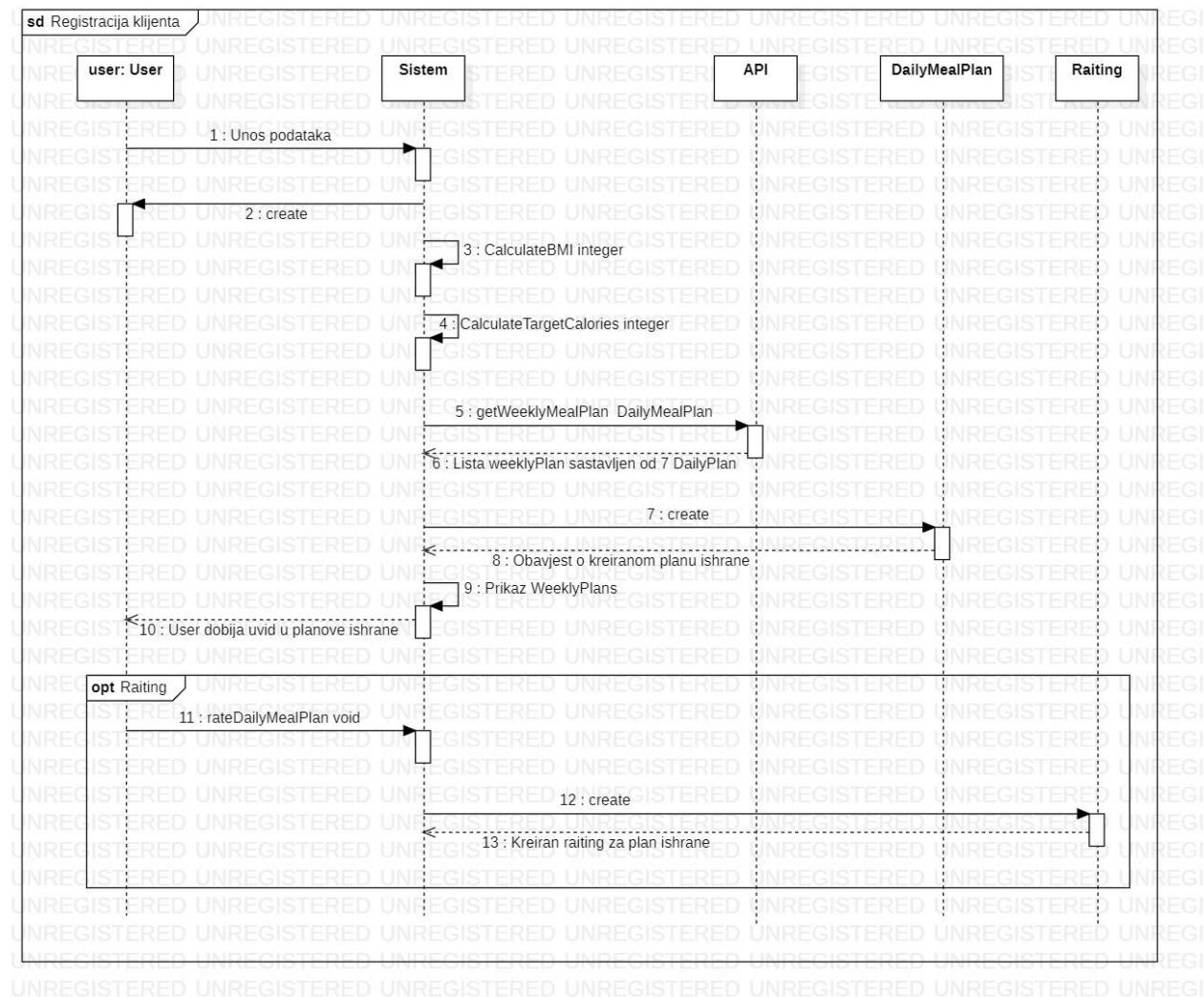


Dijagram sekvence

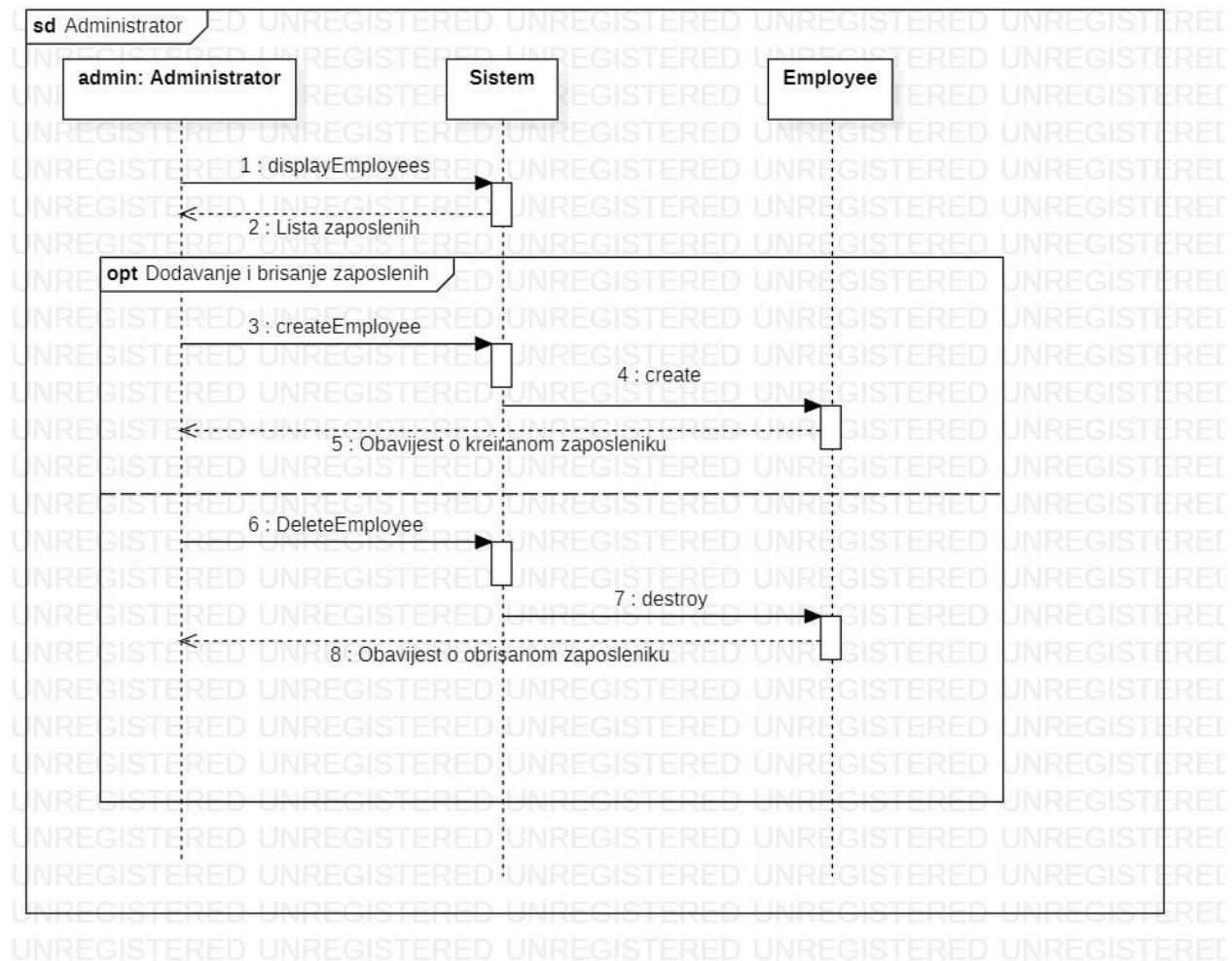
Login i prikaz ishrane



Registracija klijenta

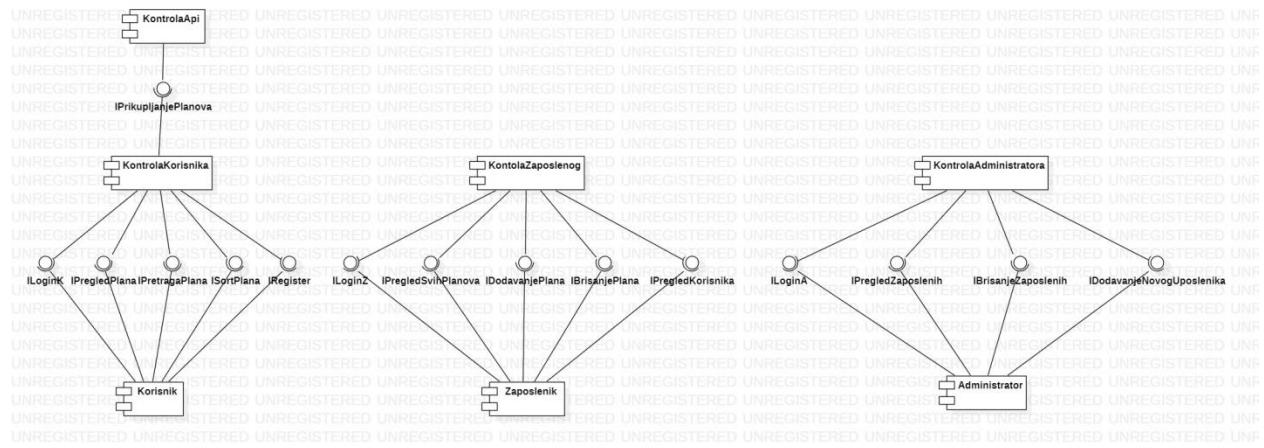


Administrator

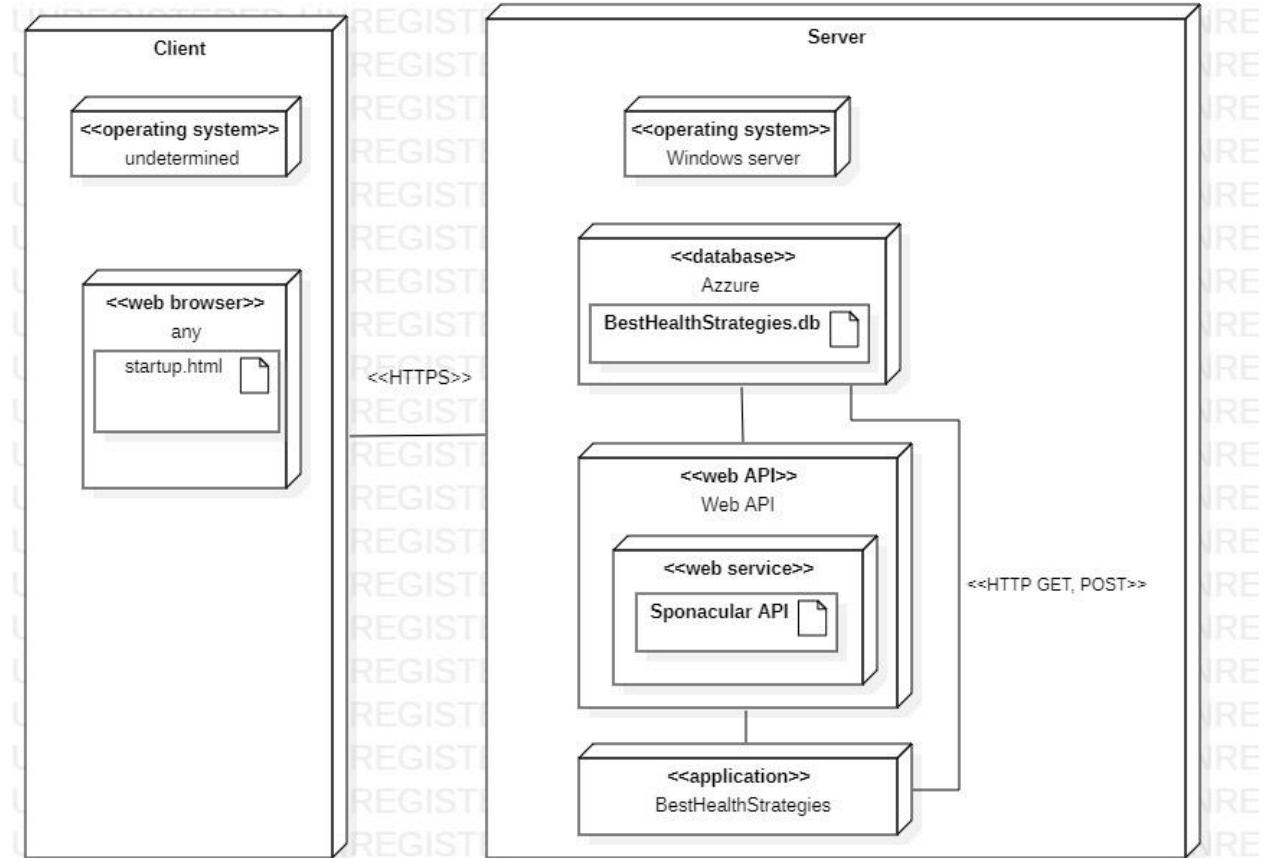


Dijagram komponenti, paketa i raspoređivanja

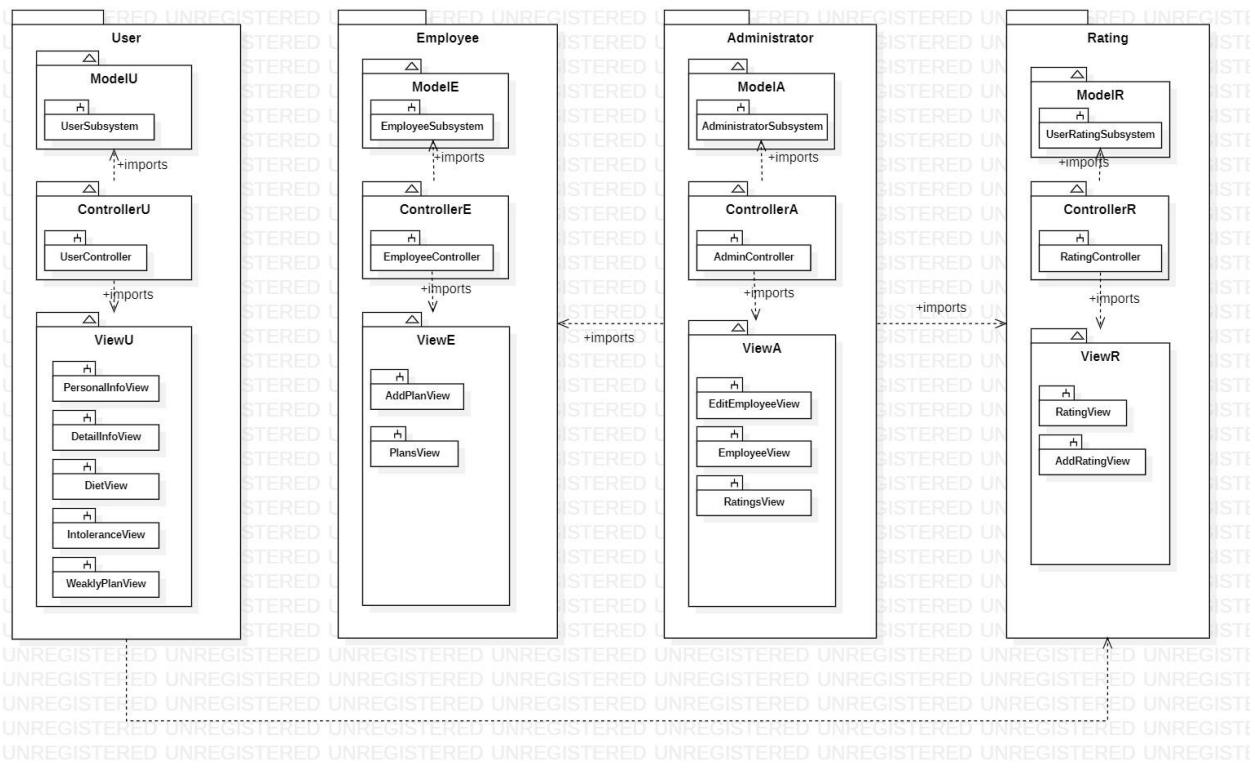
Dijagram komponenti



Dijagram raspoređivanja



Dijagram paketa



Link na web aplikaciju

https://besthealthstrategies.azurewebsites.net/?fbclid=IwAR2Gx5ECgmtA6uVDMR_QxN9t_FGZwmRTGB8yqatOz8XfQd7Aeq7Cz9NexVk

Video prikaz rada aplikacije Best Health Strategies

<https://www.youtube.com/watch?v=9zF-25usazY&t=1s>