Dokaz koncepta - Tim 2

Makan Albert #king
Miloš Panić #baš
Luka Bjelica #idegas
Jovana Jevtić #namax

Uvod

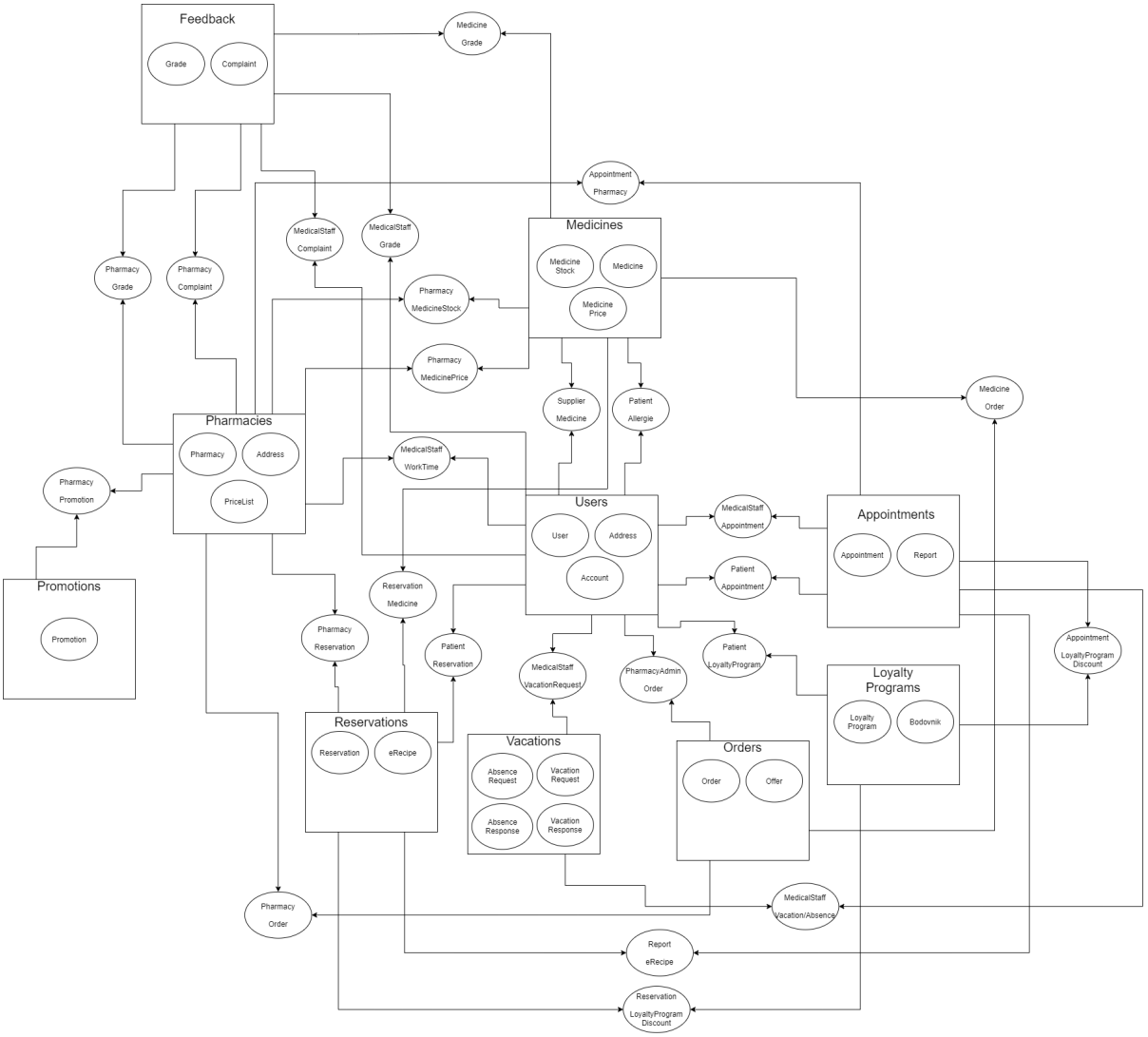
U skladu sa pretpostavkama da je sistem korišćen od strane 200 miliona korisnika i da ima milion rezervacija lekova i zakazanih pregleda na mesešnom nivou, zaključujemo da sistem mora biti skalabilan i visoko dostupan. Kako bi se ispunili prethodno navedeni ciljevi i akcenat staviti na visoke performanse sistema, odlučili smo da sistem bude razvijen u skladu sa principima mikroservisne arhitekture. Imajući u vidu da razvoj sistema baziranog na mikroservisnoj arhitekturi donosi mnoge izazove jer se radi o distribuiranom sistemu, veliki značaj imaju alatke koje olakšavaju prevazilaženje istih uz minimalnu količinu uloženog truda i maksimalan stepen automatizacije. Shodno tome, potrebne su nam alatke koje podržavaju skaliranje određenih komponenti sistema i mehanizam “load-balancing-a”, enkapsuliraju komunikaciju između njih na sinhron i asinhrnon način, pružaju mogućnost oporavka od greške i nadgledanja sistema u svakom smislu, odnosno na nivou operativnog sistema, kao i na aplikativnom nivou i automatsko ažuriranje pojedinih komponenti sistema koje može biti sprovedeno nezavisno u odnosu na ostatak sistema. Korišćenje kubernetes-a, kao orkestratora aplikativnog sloja sistema, sve prethodno navedene potrebe su u manjoj ili većoj meri zadovoljene. Dodatno, korišćenje kubernetes-a omogućava automatsko skaliranje instanci pojedinih mikroservisa u zavisnosti od potrebe sistema, odnosno količine pristiglih zahteva u datom vremenskom periodu. Svaki mikroservis se može nezavisno skalirati, čime se izbegava slučaj stvaranja uskog grla sistema time što delovi koji su pod najvećim opterećenjem bivaju skalirani na veliki broj instanci.

Pored dobre organizacije aplikativnog sloja sistema, veoma važnu ulogu preuzima način skladištenja, i još važnije, pristupanja podacima. Po principima mikroservisne arhitekture, teži se ka tome da svaki mikroservis bude u što većoj meri nezavistan. Pored nezavisnog životnog ciklusa individualnog mikroservisa, kao i prethodno spomenute mogućnosti za nezavisnom isporukom, u to spada i činjenica da on treba da “poseduje” podatke sa kojima rukuje. Ono što se dobro pokazalo u praksi jeste korišćenje “database per service” šablona, gde svaki mikroservis radi sa posebnom instancom baze podataka. Time se u toliko više podstiče ideja izolacije podataka na nivou individualnog mikroservisa. Izazov se javlja kada dva ili više mikroservisa koriste, odnosno rukuju, istim entitetima. Tada je potrebno primeniti neke od mehanizama poput keširanja ili asinhrone sinhronizacije* zavisno od konteksta. Da bi sistem bio otporan na greške i mogao da preživi čak i prirodne katastrofe, potrebno je osigurati to da su podaci uskladišteni na više različitih fizičkih lokacija i da su redovno sinhronizovani.

Da bi se prethodno navedeni zahtevi ispunili, potrebno je podatke skladištiti u SUBP koji podržava rad sa više instanci, odnosno skaliranje baze podataka, replikaciju i distribuirane transakcije. Takođe, da bi se omogućilo keširanje i asinhrona sinhronizacija podataka deljenih entiteta, potrebno je koristiti tehnologije koje podržavaju takve mehanizme. RavenDB je distribuirana NoSQL baza podataka koja podržava ACID transakcije i mehanizme optimističkog i pesimističkog zaključavanja pojedinih resursa. Više instanci

RavenDB baze podataka može biti smešteno u jedan klaster u kojem svaka instanca sadrži sve podatke. Bilo koja instanca može izvršavati operacije upisa i čitanja, nezavisno od uloge. Prilikom svakog upita, automatski se kreira Index (B+ stablo pretrage), čime se postižu veoma dobre performanse. Pored toga što RavenDB podržava dispečovanje događaja (eng. Event dispatching), za potrebe asinhrone sinhronizacije pogodno je koristiti neki od “MessageBroker-a” poput RabbitMQ i Redis-a koji ujedno pruža pogodan mehanizam keširanja.

Predlog podele entiteta sistema u izolovane kontekste (eng. Bounded Contexts) i prikaz deljenih entiteta



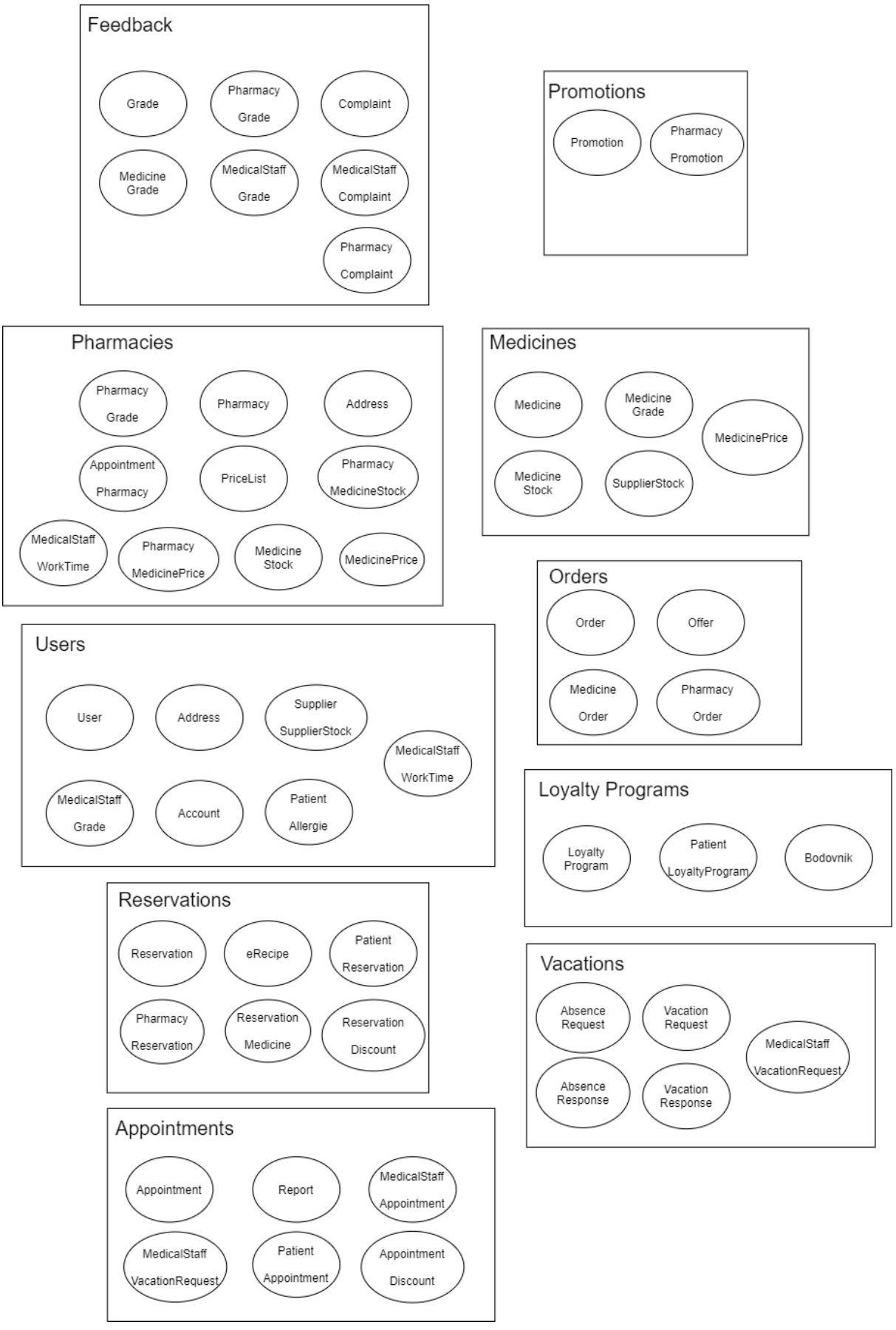
Na slici je prikazana šema izolovanih konteksta koja oslikava entitete koje svaki kontekst u potpunosti poseduje, kao i one koje deli sa drugim kontekstima. Iz date šeme je proistekla podela sistema na mikroservise.

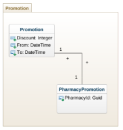
Može se videti da je sistem podeljen na sledeće “podsisteme”, odnosno kontekste:

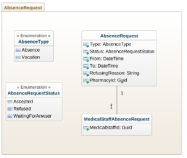
1. Podsistem za rad sa korisnicima

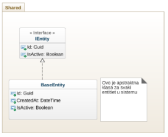
2. Podsistem za rad sa apotekama
3. Podsistem za rad sa lekovima
4. Podsistem za rad sa pregledima
5. Podsistem za rad sa porudžbinama
6. Podsistem za rad sa rezervacijama
7. Podsistem za rad sa povratnim informacijama
8. Podsistem za rad sa godišnjim odmorima i odsustvima
9. Podsistem za rad sa promocijama
10. Podsistem za rad sa programima lojalnosti

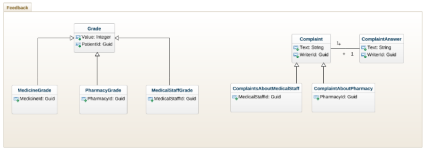
Na sledećoj slici je takođe prikazana šema izolovanih konteksta, ali bez veza između istih. Akcenat je u ovom slučaju bačen na sve entitete kojima pojedinačni konteksti rukuju, među kojima se nalaze oni koji su isključivo u njegovom “vlasništvu” i oni deljeni.

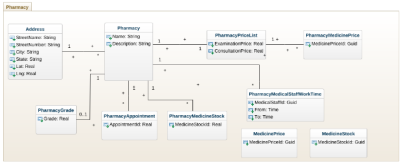

Feedback
Grade
Pharmacy Grade
Complaint
Medicine Grade
MedicalStaff Grade
MedicalStaff Complaint
Pharmacy Complaint
Promotions
Promotion
Pharmacy Promotion
Pharmacies
Pharmacy Grade
Pharmacy
Address
Appointment Pharmacy
PriceList
Pharmacy MedicineStock
MedicalStaff WorkTime
Pharmacy MedicinePrice
Medicine Stock
MedicinePrice
Medicines
Medicine
Medicine Grade
MedicinePrice
Medicine Stock
SupplierStock
Orders
Order
Offer
Medicine Order
Pharmacy Order
Users
User
Address
Supplier SupplierStock
MedicalStaff WorkTime
MedicalStaff Grade
Account
Patient Allergie
Loyalty Programs
Loyalty Program
Patient LoyaltyProgram
Bodovnik
Reservations
Reservation
eRecipe
Patient Reservation
Pharmacy Reservation
Reservation Medicine
Reservation Discount
Vacations
Absence Request
Vacation Request
MedicalStaff VacationRequest
Absence Response
Vacation Response
Appointments
Appointment
Report
MedicalStaff Appointment
MedicalStaff VacationRequest
Patient Appointment
Appointment Discount

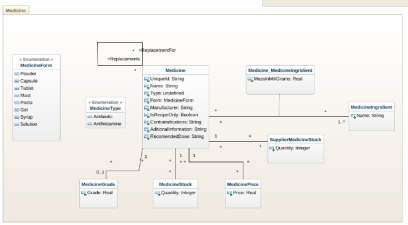

Promotion
Promotion
+Discount: Integer
+From: DateTime
+To: DateTime
PharmacyPromotion
+PharmacyId: Guid

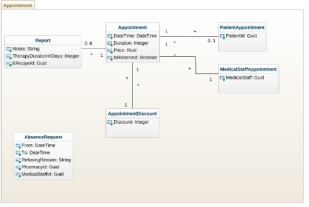

AbsenceRequest
«Enumeration»
AbsenceType
Absence
Vacation
«Enumeration»
AbsenceRequestStatus
Accepted
Refused
WaitingForAnswer
AbsenceRequest
+Type: AbsenceType
+Status: AbsenceRequestStatus
+From: DateTime
+To: DateTime
+RefusingReason: String
+PharmacyId: Guid
MedicalStaffAbsenceRequest
+MedicalStaffId: Guid

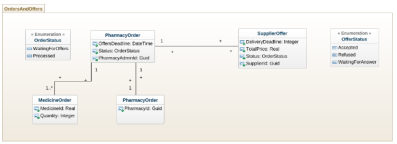

Shared
«Interface»
IEntity
+Id: Guid
+IsActive: Boolean
BaseEntity
+Id: Guid
+CreatedOn: DateTime
+IsActive: Boolean
Ovo je apstraktna klasa za bazni entitet u sistemu

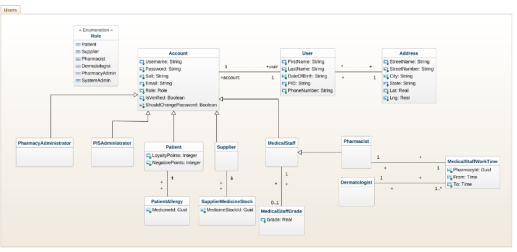

Feedback
Grade
+Value: Integer
+PatientId: Guid
MedicineGrade
+MedicineId: Guid
PharmacyGrade
+PharmacyId: Guid
MedicalStaffGrade
+MedicalStaffId: Guid
Complaint
+Text: String
+WriterId: Guid
ComplaintAnswer
+Text: String
+WriterId: Guid
ComplaintsAboutMedicalStaff
+MedicalStaffId: Guid
ComplaintAboutPharmacy
+PharmacyId: Guid

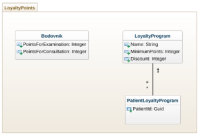

Pharmacy
Address
+StreetName: String
+StreetNumber: String
+City: String
+State: String
+Lat: Real
+Lng: Real
Pharmacy
+Name: String
+Description: String
PharmacyPriceList
+FromMedicinePrice: Real
+ConsultationPrice: Real
PharmacyMedicinePrice
+MedicinePriceId: Guid
PharmacyMedicalStaffWorkTime
+MedicalStaffId: Guid
+From: Time
+To: Time
PharmacyGrade
+Grade: Real
PharmacyAppointment
+AppointmentId: Real
PharmacyMedicineStock
+MedicineStockId: Real
MedicinePrice
+MedicinePriceId: Guid
MedicineStock
+MedicineStockId: Guid

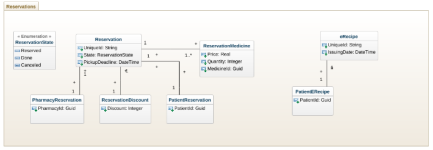

Medicine
«Enumeration»
MedicineForm
Powder
Capsule
Tablet
Must
Paste
Gel
Syrup
Solution
+ReplacementFor
+Replacements
Medicine
+UniqueId: String
+Name: String
+Type: undefined
+Form: MedicineForm
+ManufacturerName: String
+IsStrongerDrug: Boolean
+ContraIndications: String
+AdditionalInformation: String
+RecommendedDose: String
Medicine_MedicineIngridient
+MedsWithIDuans: Real
MedicineIngridient
+Name: String
«Enumeration»
MedicineType
Antibiotic
Antihistamine
SupplierMedicineStock
+Quantity: Integer
MedicineGrade
+Grade: Real
MedicineStock
+Quantity: Integer
MedicinePrice
+Price: Real


Appointment
Report
+Notes: String
+TherapyDurationInDays: Integer
+ReceiptId: Guid
Appointment
+DateTime: DateTime
+Duration: Integer
+Price: Real
+IsInterpreted: Boolean
PatientAppointment
+PatientId: Guid
MedicalStaffAppointment
+MedicalStaff: Guid
AppointmentDiscount
+Discount: Integer
AbsenceRequest
+From: DateTime
+To: DateTime
+RefusingReason: String
+PharmacyId: Guid
+MedicalStaffId: Guid

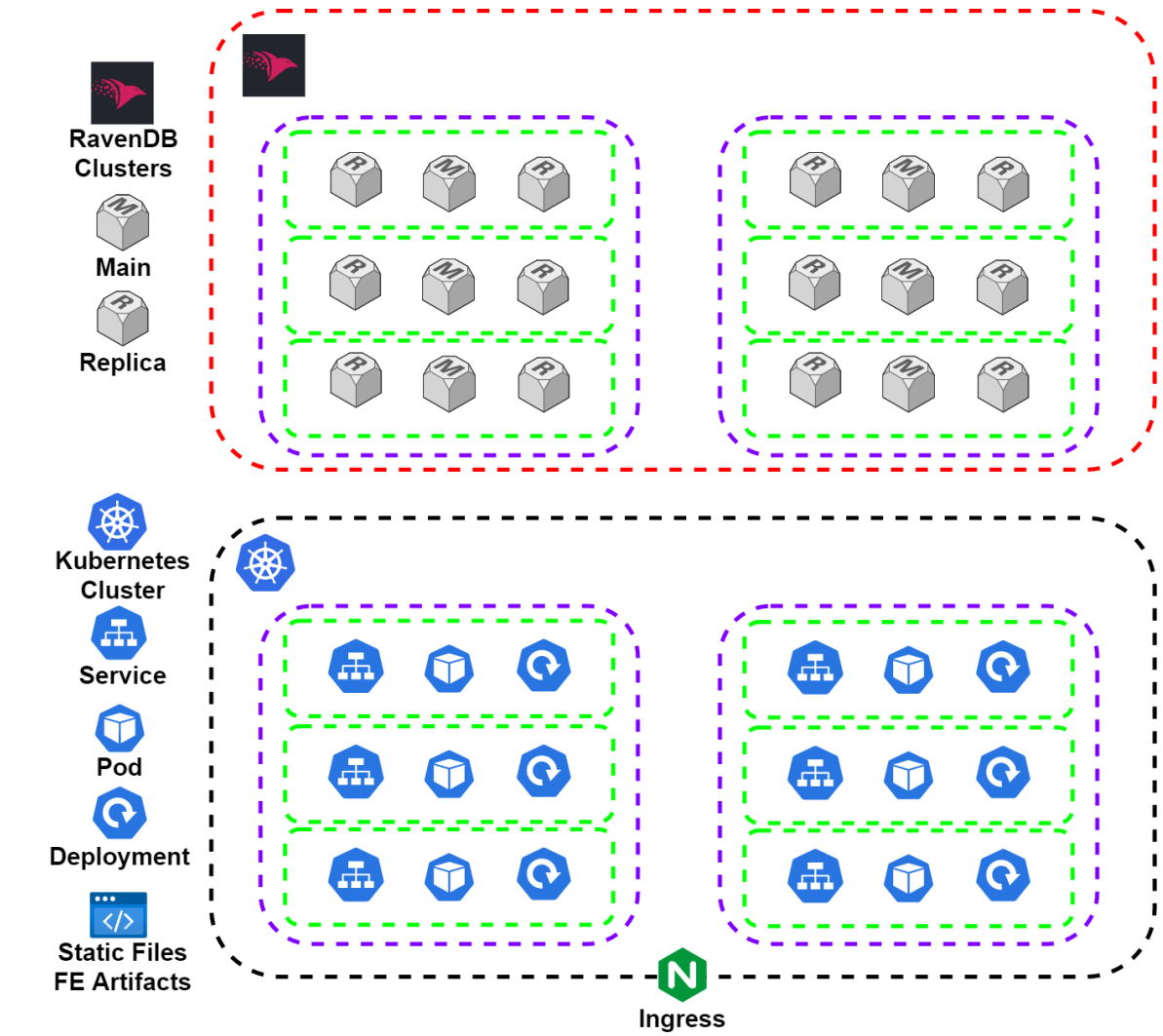

OrdersAndOffers
«Enumeration»
OrderStatus
WaitingForOffers
Processed
PharmacyOrder
+OfferDeadline: DateTime
+Status: OrderStatus
+PharmacyAdminId: Guid
SupplierOffer
+DeliveryDeadline: Integer
+TotalPrice: Real
+Status: OrderStatus
+SupplierId: Guid
«Enumeration»
OfferStatus
Accepted
Refused
WaitingForAnswer
MedicineOrder
+MedicineId: Real
+Quantity: Integer
PharmacyOrder
+PharmacyId: Guid

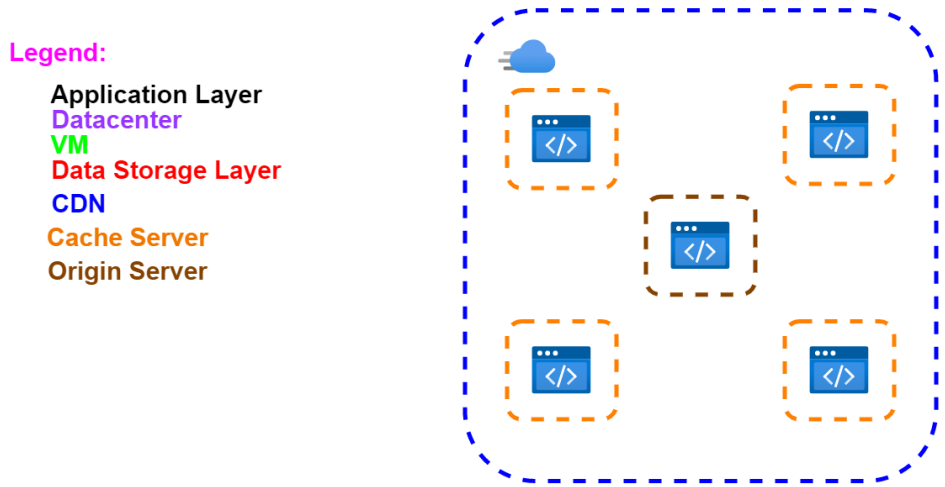

Users
«Enumeration»
Role
Patient
Supplier
PharmacyId
DermatologyId
PharmacyAdmin
SystemAdmin
Account
+Username: String
+Password: String
+Salt: String
+Email: String
+Role: Role
+IsVerified: Boolean
+ShouldChangePassword: Boolean
+account
User
+FirstName: String
+LastName: String
+DateOfBirth: String
+PID: String
+PhoneNumber: String
Address
+StreetName: String
+StreetNumber: String
+City: String
+State: String
+Lat: Real
+Lng: Real
PharmacyAdministrator
PSAdministrator
Patient
+LoyaltyPoints: Integer
+NegativePoints: Integer
Supplier
MedicalStaff
Pharmacist
+PharmacyId: Guid
+From: Time
+To: Time
Dermatologist
MedicalStaffWorkTime
PatientAllergy
+MedicineId: Guid
SupplierMedicineStock
+MedicineStockId: Guid
MedicalStaffGrade
+Grade: Real


LoyaltyPoints
Bodovnik
+PointsForExamination: Integer
+PointsForConsultation: Integer
LoyaltyProgram
+Name: String
+MaximumPoints: Integer
+Discount: Integer
PatientLoyaltyProgram
+PatientId: Guid


Reservations
«Enumeration»
ReservationState
Reserved
Done
Canceled
Reservation
+Request: String
+State: ReservationState
+PickupDeadline: DateTime
ReservationMedicine
+Price: Real
+Quantity: Integer
+MedicineId: Guid
eRecipe
+Request: String
+IssuingDate: DateTime
PharmacyReservation
+PharmacyId: Guid
ReservationDiscount
+Discount: Integer
PatientReservation
+PatientId: Guid
PatientERecipe
+PatientId: Guid

Predlog organizacije infrastrukturne komponente sistema





Na slici je prikazana šema infrastrukture sistema. Sistem je podeljen u 3 glavne celine: CDN (Content Delivery Network), aplikativni sloj i sloj za skladištenje i rad sa podacima. Slojevi su međusobno nezavisni, u smislu da bilo koji od slojeva može biti promenjen bez potrebe da se uvode bilo kakve izmene u preostala dva. Pravougaonici sa isprekidanim

linijama pored toga što definišu granice svake celine, predstavljaju neki vid privatne mreže (npr. overlay) unutar koje komponente mogu međusobno da komuniciraju.

CND ima ulogu u serviranju statičkih podataka, poput artifakta potrebnih za rad front-end aplikacije. Razlog zašto smo se odlučili za njega, umesto npr. običnog http servera, jeste u tome što odlično radi sa velikim brojem korisnika, pogotovo u slučaju da se nalaze na različitim geografskim lokacijama, što svakako jeste naš slučaj. Na dijagramu je predstavljen jedan izvorni (eng. origin) server koji sadrži “istinite” podatke, i više keš servera koji sadrže kopiju istih.

Unutar aplikativnog sloja se nalaze aplikacije koje implementiraju biznis logiku. U to spadaju svi mikroservisi i dodatne komponente potrebne za rad istih. Klaster je podeljen u dve zone otkaza (eng. failure zones) koje predstavljaju logički ekvivalent datacentra, tako da je zagarantovano da se virtuelne mašine u jednoj zoni otkaza nalaze na različitim fizičkim lokacijama u odnosu na one iz druge zone otkaza. U slučaju prirodne katastrofe (poput meteora, nuklearke, vulkana, zemljotresa, cunamija, velikog požara ili suprotno - poplave) koja bi zahvatila jedan datacentar, sistem bi bez problema nastavio sa radom koristeći instance koje se nalaze u drugom datacentru, odnosno drugoj fizičkoj lokaciji. Zona otkaza je podeljena na proizvoljno veliki broj virtuelnih mašina, zavisno od potrebe sistema. Razlog zašto se na dijagramu ne nalazi jasno definisan broj virtuelnih mašina jeste činjenica da se od sistema očekuje da bude automatski prilagodljiv količini opterećenja pod kojim se nalazi.

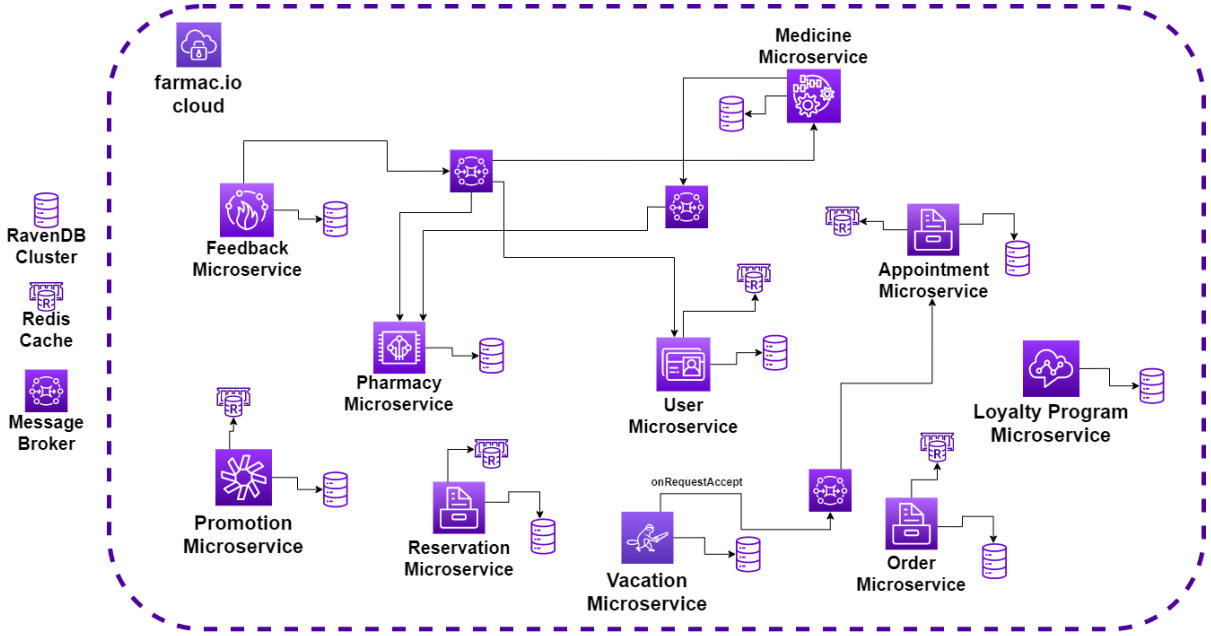
Load balancing se vrši na dva nivoa: na nivou klastera i na nivou servisa. Prvi nivo je omogućen korišćenjem kubernetes objekta “ingress”. Zamišljeno je da se load balancing na tom nivou odvija po algoritmu najbržeg odziva (eng. least response time) čime bi se maksimalno iskoristilo to što su zone otkaza na različitim geografskim lokacijama. Drugi nivo jeste load balancing zahteva između više različitih instanci istog servisa, koji je omogućen korišćenjem “service” objekta koji nudi kubernetes. Ideja je da se zahtevi ravnomerno raspodele na sve postojeće instance servisa, pa je izabran algoritam kružnog crvendaća (eng. round robin).

Sloj za perzistenciju i pristup podataka u sebi sadrži klastere instanci baza podataka za svaki servis. Za razliku od aplikativnog sloja, ovaj sloj se ne oslanja na kubernetes. Razlog tome jeste činjenica da je mnogo manji stepen mogućnosti automatizacije procesa za stateful aplikacije. Postoji veliki broj stvari koji se doslovce “mora” raditi ručno. Ipak, to ne znači da automatizacije nema, već da je u velikoj meri smanjena. Slično aplikativnom sloju, ovaj sloj je takođe podeljen na zone otkaza unutar kojeg se nalazi proizvoljan broj virtuelnih mašina. Važno je napomenuti da se za svaki servis kreiraju dva klastera od po 3 “čvora” (eng. node). To znači da u bilo kom momentu mikroservis ima dva klastera u dve različite zone otkaza i da se tri instance baze nalaze na tri različite virtuelne mašine. Time je postignuta otpornost na greške i prevencija gubljenja podataka u slučaju manjeg ili većeg otkaza. Ukoliko bi se javila potreba za tim, moguće je particionisati (eng. sharding) podatke na više različitih virtuelnih mašina. Time bi postigli poboljšanje u vidu performansi uz preduslov da se podaci mogu smisleno podeliti.

U ovom sloju je moguće, mada ne i neophodno, uvesti load balancer koji raspoređuje zahteve na više različitih zona otkaza. Zasigurno, load balancing se vrši na nivou klastera, i to u vidu algoritma najbržeg čvora (eng. fastest node) koji je ugrađen u klijent biblioteke koje

nudi RavenDB. Time su znatno povećane performanse sistema i stepen otpornosti na greške.

Predlog organizacije komponente sistema odgovorne za implementaciju biznis logike



Na slici je prikazana šema predloženog sistema koja je bazirana na mikroservisnoj arhitekturi. Može se videti da svaki servis ima odgovarajuću bazu (odnosno klaster instanci) čime je ispoštovan šablon database per service. Pored toga, servisi mogu komunicirati sa drugim servisima, odgovarajućim message brokerom ili instancom keš servisa. Servisi koji šalju podatke, odnosno dispečuju događaje imaju strelicu usmerenu ka message brokeru, dok oni koji slušaju za iste imaju strelicu usmerenu ka njima. U slučaju da dođe do disfunkcije svih instanci određenog mikroservisa (primer bi bio buba u kodu (eng. bug) koja uzrokuje prestanak rada odmah nakon pokretanja), ideja je da se izbegne dalje širenje otkaza na servise koji komuniciraju sa njim i da se samo degradira funkcionalnost sistema koju obavlja taj servis.

Okvirna procena memorije potrebne za skladištenje svih podataka u narednih 5 godina

Ukupan broj korisnika sistema je 200 miliona, a rezervacija i pregleda na mesečnom nivou je 1 milion.

Pretpostavke vezane za druge entitete koji su ušli u proračun potrebne memorije:

- Po nekim statističkim podacima, optimalan broj apoteka je 25 po 100000 stanovnika ali je Srbija pravi primer statističkog autlajera (eng. outlier) gde postoji duplo veći broj

apoteka nego što je to potrebno. Uzimajući neki broj između 25 i 50, i činjenicom da u sistemu postoje apoteke iz raznih država, dobili smo 75000 apoteka u sistemu.
- Neka pretpostavka je da apoteke generalno rade u dve smene, neke imaju isključivo farmaceute a neke i dermatologe i shodno time smo dobili broj od 8 zaposlenih u jednoj apoteci, što znači da u proseku imamo 600000 zaposlenih u sistemu.
- Ukoliko je podatak tačan, u Srbiji postoji nekih 3000 registrovanih lekova. Ukupna količina korisnika se može smestiti u oko 29 Srbija. Ako uzmemo u obzir da bi mnogo manji procenat populacije koristilo naš sistem, dolazimo do nekog broja od 100000 lekova u sistemu. Svaki lek ima u proseku 10 sastojaka i 3 zamenska leka.
- Uz pomoć statističkog alata(random.org) smo dobili da svaka apoteka u proseku ima 1500 lekova na stanju. Smatramo da se cena leka menja jednom godišnje.
- Verujemo da bi u sistemu bilo više rezervacija nego pregleda i to u odnosu 70:30. Tako na mesečnom nivou imamo 300000 pregleda, od kojih bar 70% ima preporučenih u proseku 3 leka(eRecept). Rezervacija na mesečnom nivou ima 700000 sa u proseku 3 rezervisana leka.
- Predviđen broj dostavljača u sistemu je 7500. Svaki dostavljač ima u proseku 500 lekova na stanju.

Slika ispod ilustruje količinu memorije potrebnu za svaku grupu entiteta(Korisnik ima Account, User, Address) i ukupnu količinu memorije u sistemu. Dobijeni broj je 1.1tb. Ne učestvuju svi entiteti sistema u ovom proračunu ali oni ne bi mnogo promenili rezultat. Apoteke zauzimaju najveću količinu memorije u odnosu na broj apoteka i to je mesto koje bi se zasigurno moglo bolje izmodelovati.

Naziv entiteta	Predviđena količina	Jedinično zauzeće memorije(bytes)	Ukupno(bytes)
Korisnici	199392500	1623,49	323711729825
Apoteka	75000	8255828,94	619187170500
Zaposleni	600000	3380,6	2028360000
Lekovi	100000	5948,43	594843000
Pregledi	18000000	2463,9	44350200000
ERecepti	12600000	2996,33	37753758000
Rezervacije	53970000	3361,66	181428790200
Dostavljači	7500	316182,26	2371366950

Ukupno(bytes)
1211426218475

Ukupno(kbytes)
1183033416

Ukupno(megabytes)
1155306,071

Ukupno(gigabytes)
1128,228585

Ukupno(terabytes)
1,101785727

Predlog operacija koje treba nadlegati u cilju poboljšanja sistema

Osim vertikalnog skaliranja, kao načina za opsluživanje velikog broja korisnika, takođe je potrebno i da najučestalije korišćene operacije budu optimalne po pitanju odziva. On se može poboljšati keširanjem određenih podataka, primenom asinhronih operacija ili unapređenjem određenih algoritama. Kako bismo znali učestalost korišćenja određenih operacija i prosečno vreme odziva, potrebno je raditi monitoring operacija. Kandidati za monitoring su:

- Pravljenje naloga pacijenata

- Prijava na sistem
- Rezervacija lekova
- Zakazivanje pregleda/konsultacija
- Pregled apoteka i lekova

Poželjno bi bilo, nakon implementacije svih funkcionalnosti, a pre nego što sistem postane popularan i dostigne cifru od 200 miliona korisnika raditi monitoring svih operacija u sistemu. Na taj način bismo mogli detektovati anomalije u operacijama koje nisu toliko često korišćene ali bi se svakako mogle poboljšati. Primeri takvih operacija u trenutnom sistemu su:

- Dodavanje novih lojaliti programa zato što se mora prilagoditi lojaliti program velikom broju korisnika. Poboljšavanje ove operacije je tema za sebe i mogao bi ceo seminarski biti napisan o tome, ali mi nećemo.
- Svi upiti ka bazi podataka su sinhroni što ozbiljno utiče na broj korisnika koji se mogu opslužiti u jednom trenutku i to se može poboljšati korišćenjem EF Core aync operacija.
- Takođe bi bilo poželjno napraviti poseban mikroservis za asinhrono slanje emailova.

Radi poboljšanja korisničkog dojma, sazvali bismo određenu grupu pacijenata i dermatologa/farmaceuta uz pomoć kojih bi naši inžinjeri došli do dizajna koji bi najbolje odgovarao korisnicima.

Predlog strategije za keširanje podataka

Svo keširanje u sistemu se odvija uz pomoć Redis-a. Keširanje ćemo razdvojiti na dve grupe. Keširanje koje radimo kako bismo smanjili komunikaciju između mikroservisa i keširanje koje radimo kako bi odziv na neki zahtev korisnika bio najkraći moguć.

Podaci koje treba keširali kako bi komunikacija između mikroservisa bila što ređa su:

- Trenutni popust apoteke može da se kešira unutar Appointment i Reservation mikroservisa zato što se koristi prilikom svake rezervacije lekova i zakazivanja pregleda (ukupno 1 milion mesečno). Keširanje se radi po zahtevu.
- Trenutni popust koji korisnik ostvaruje uz svoj lojaliti program u Appointment i Reservation mikroservisima zato što se koristi prilikom svake rezervacije lekova i zakazivanja pregleda (ukupno 1 milion mesečno). Keširanje se radi po zahtevu.

Pored keširanja, radi se i replikacija određenih podataka kako bi se smanjila komunikacija između mikroservisa. Asinhrona sinhronizacija se vrši uz pomoć RabbitMQ.

- Ocene apoteka, zaposlenih i lekova su potrebne prilikom svakog čitanja tih entiteta. Keširanje ovde nije dobra ideja zato što se ocene često mogu menjati pa se zato prave kopije ocena unutar Pharmacy, User, Medicine mikroservisa. Konzistentnost ovih podataka nije od ključnog značaja tako da nije problem ukoliko se desi da se pročita stara ocena.
- Godišnji odmori su potrebni unutar Appointment servisa prilikom zakazivanja pregleda. Keširanje nije dobra ideja zato što će zaposleni mnogo ređe biti na godišnjem odmoru u odnosu na to kada neće i kako bismo skroz izbegli komunikaciju

između mikroservisa, radiće se asinhrono kopiranje Vacation objekta nad kojim ne postoje operacije izmene ili brisanja.
- MedicinePrice i MedicineStock su često potrebni unutar Pharmacy mikroservisa pa se zato radi replikacija. Keširanje nije najbolje rešenje zato što nam je konzistentnost jako bitna.

Podaci koje bi trebalo keširati na nivou mikroservisa kako bismo smanjili vreme odziva:
- Apoteke za home page, može se keširati nekoliko prvih stranica unapred, dok se kasnije stranice mogu keširati po zahtevu.
- Lekovi za home page, može se keširati nekoliko prvih stranica unapred, dok se kasnije stranice mogu keširati po zahtevu.
- Specifikacija leka u vidu PDFa se generiše isključivo prilikom kreiranja ili izmene informacija o određenom leku, dok se čitanje odvija direktno sa diska pri svakom zahtevu. PDFovi najčešće preuzimanih lekova bi se mogli keširati u memoriji keš servisa.
- Podaci najposećenijih stranica apoteka se mogu keširati isključivo za pacijente neko kraće vreme zato što njima konzistentnost tih podataka nije od ključnog značaja, a operacija je vrlo ušestala.

Napomena: Ovo su samo neki od predloga keširanja/replikacije. Trenutno nismo u stanju da odredimo toliko pogodnih situacija ali bismo došli do njih prilikom implementacije ili analize nadgledanih operacija.