Skalabilnost, predlog implementacije i arhitekture u slučaju

ZAMISR, je aplikacija razvijena od strane studenata računarstva i automatike i njen osnovni zadatak je da omogući njenim korisnicima da lako izlože svoje vile, brodove ili instruktorske veštine ili da iste rezervišu. Celokupna aplikacija je razvijena koristeći Dotnet 5.0, vuejs kao i mssql. Sve od navedenih tehnologija su invarijantne na platformu.

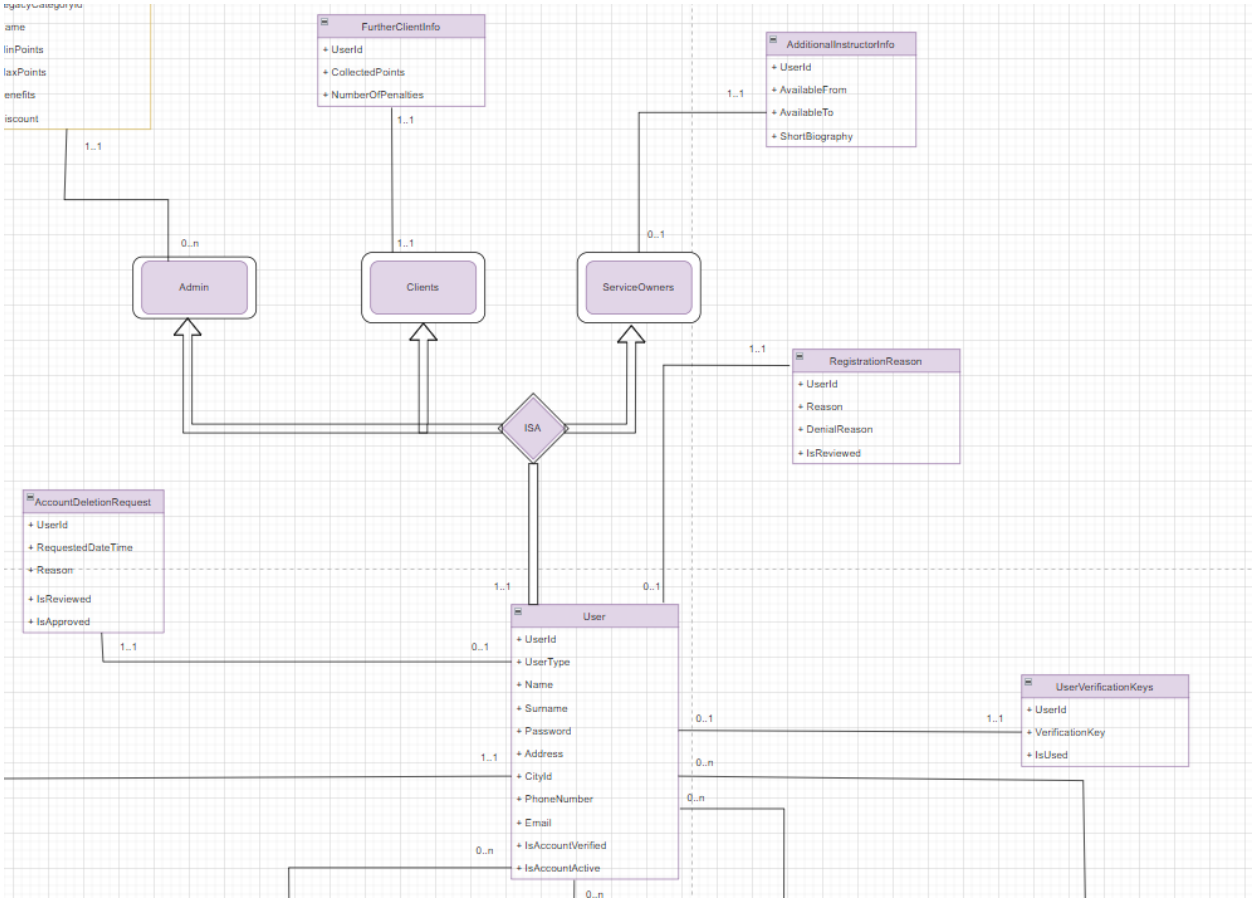
Osnovne pretpostavke:

- Ukupan broj korisnika aplikacije je 100 milion.
- Broj rezervacija svih entiteta na mesečnom nivou je milion.
- Sistem mora biti skalabilan i visoko dostupan.

1. Šema relacije i particionisanje

U sistemu koji smo koristili se jasno vidi nekoliko celina od kojih se sistem sastoji. Zbog uočenih celina predlaže se funkcionalno particionisanje baze na kontekste.

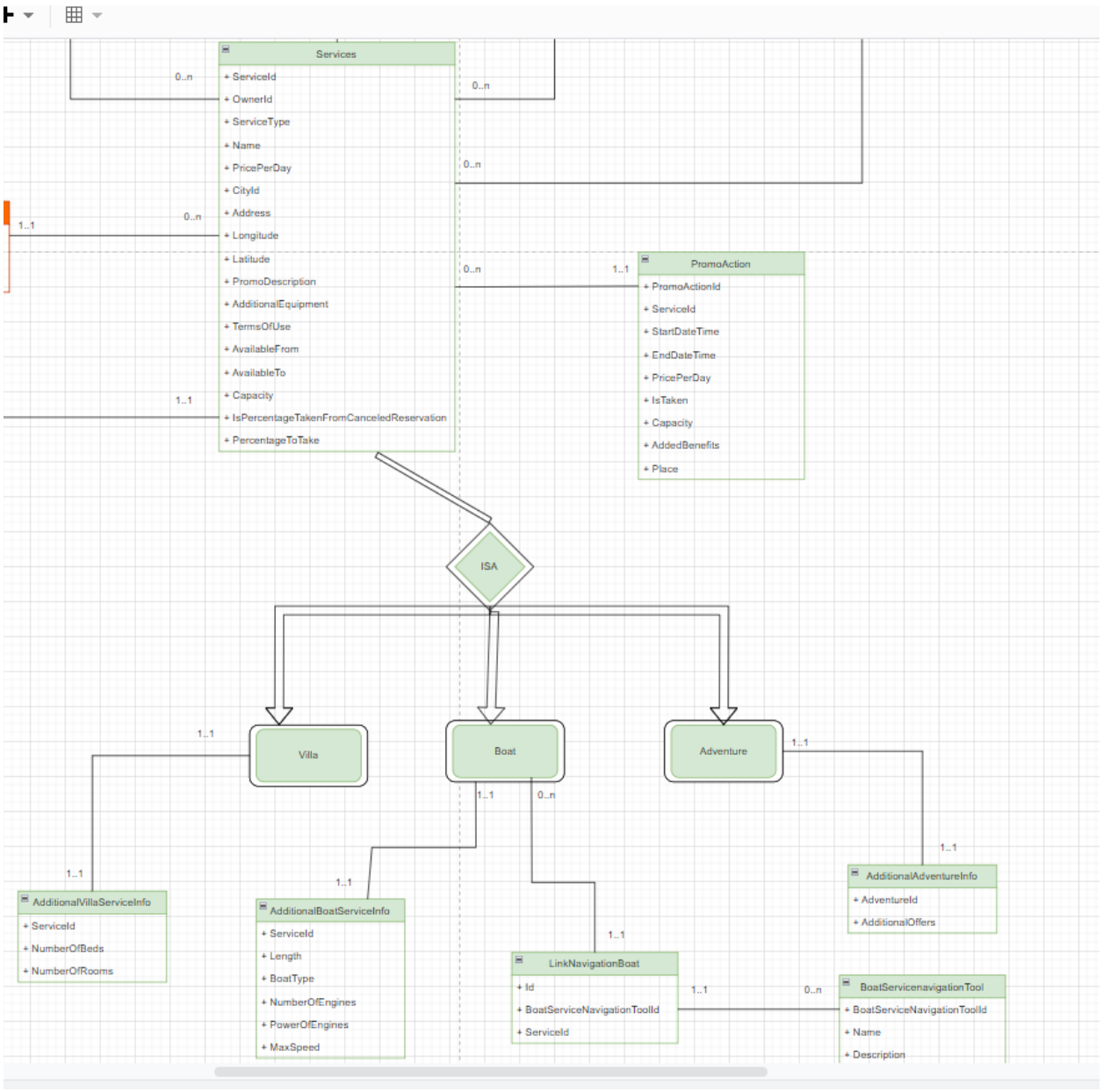
Prva celina koju ćemo razmatrati je celina vezana za korisnike.



Posmatrani deo šeme relacije, iako bitan, zbog celokupne poente aplikacije, ne bi trebalo da bude pod nekakvim velikim opterećenjem. Menjanje ličnih podataka nije nešto što bi većina klijenata radila na

dnevnom nivou ili čak mesečnom nivou. Svakako, ovaj deo baze predstavlja jedan vezani kontekst koji se kao takav treba particionisati u zasebnu bazu.

Sledeća celina o kojoj će biti reči je vezana za servise.

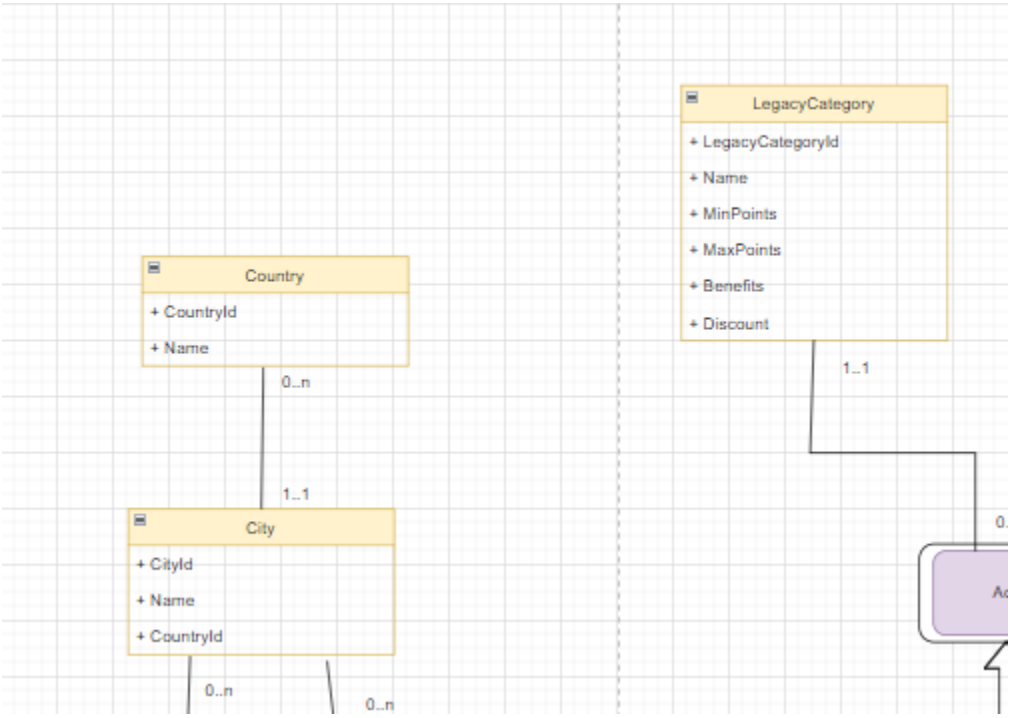


Iako negde na prelazu između dela za koji se očekuje da će biti velikog broja zahteva i onog za koji se to ne misli, zeleni deo ipak predstavlja nešto o čemu treba povesti računa. Tu se nalaze i brze akcije (Promo actions) za koje se očekuje da bi na 100 miliona rezervacija mesečno bio kritičan prostor u ovoj podšemi.

Ako bismo rekli da imamo od 100 miliona korisnika koje aplikacija ima svaki od njih jednom mesečno uđe da nešto uradi na aplikaciji, imali bismo oko 100 miliona korisnika mesečno. Neka je petina od njih vlasnik nekog servisa, što bi značilo oko 20 miliona. Kada ovu mesečnu svotu prebacimo u dnevnu dobijemo da oko 670 hiljada korisnika koji su vlasnici servisa dnevno urade nešto sa svojim servisom. Od

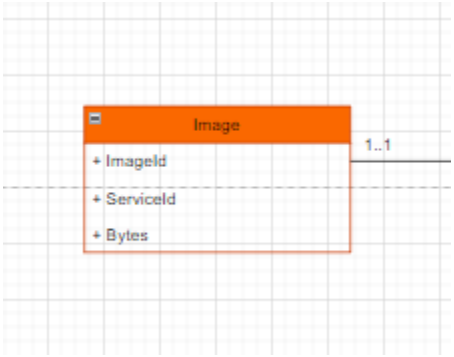
svih tih, zapravo najočekivanija operacija bi bila dodavanje novih promo akcija kako bi privukli klijente da iznajme njihove usluge. Ovo svakako jeste mesto gde se očekuje veliki broj zahteva i jeste nešto što bi se na nivou baze moglo optimizovati. Jedan od načina za to jeste uvođenje i horizontalnog particionisanja između baza jer bi jedna baza za ovaj deo bila malo. Horizontalnim particionisanjem na 5 instanci dobili bismo oko 120 hiljada zahteva dnevno ka ovom delu baze što je prihvatljivo.

Treća celina se odnosi na deo aplikacije za koji se očekuje najmanja upotreba i donekle i predstavlja opšte mesto u aplikaciji koje se nije moglo svrstati ni u jednu drugu celinu. Dodatno predstavlja nešto što bi se jednom napravilo i verovatno ne bi dugo menjalo.



Za ovaj deo sistema će uvek biti dovoljna jedna instaca baze na jednom serveru jer je očekivana dnevna interakcija najčešće samo čitanje za koje bi se moglo implementirati keširanje na nivou aplikacije.

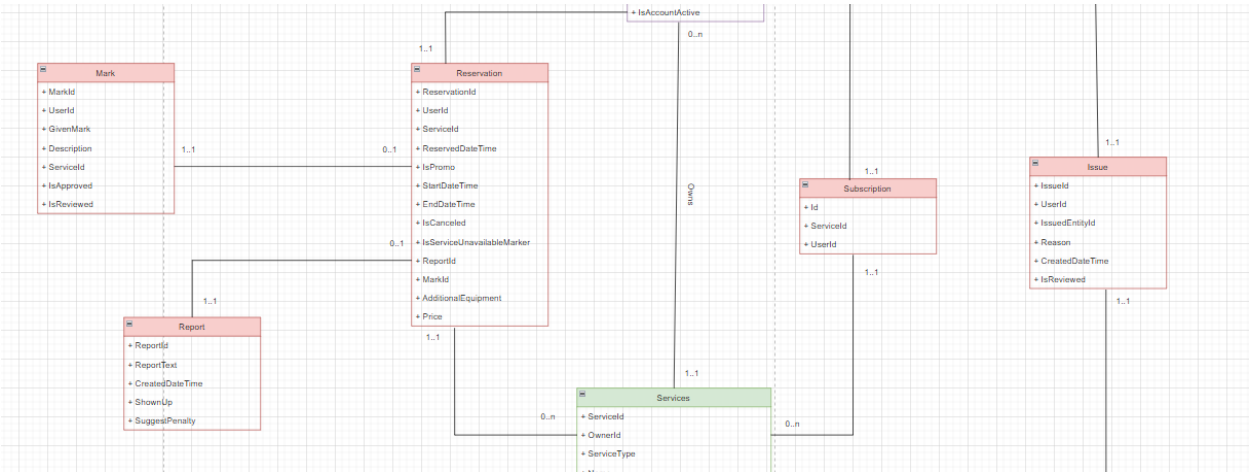
Četvrta celina ima samo jednu tabelu i to je šema sa slikama.



Iako smo se mi na ovom projektu odlučili da slike čuvamo kao bajtove, brojni članci i forumi govore da je čuvanje slika u bazi kao samo putanje do nekog skladišta sa dokumentima optimalno, postoje određeni

DBMS-ovi optimizovani za rad sa bazama koje čuvaju BLOB podatke. Slike se smatraju kao jedan od takvih podataka i njihova pretraga dovodi do degradacije te bi ove optimizacije svakako dobro došle. Kako bi se performanse očuvale, slike su nešto što se najčešće čita i samim tim keširanje na nivou baze i/ili aplikacije je svakako preporučeno.

Poslednja podšema koja predstavlja i srce same aplikacije jeste šema sa rezervacijama.



Iz pretpostavke da je mesečni broj rezervacija milion dobijamo da je dnevni prosek negde iznad 30 000. Uzevši u obzir veličinu aplikacije kad bi ona otišla na 100 miliona i pogledom na ovu podšemu vidimo da je ovo svakako mesto na kojem treba uvesti neki vid zaštite. Kako se većina ovih podataka i čita i piše na dnevnom nivou horizontalno skaliranje na 4 instance bi dovelo brojke u normalu (negde oko 7500 dnevno). Ovim bismo obezbedili najbolje performanse najbitnijeg dela aplikacije.

2. Replikacija baze i obezbeđivanje otpornosti na greške

Za samu strategiju replikacije baze neophodno je razmotriti nekoliko faktora. Ti faktori su, koliko su korisnici fizički udaljeni jedni od drugih, koliko se novih podataka piše na nekom nivou i koliko je ceo sistem brz. Kako 100 miliona nije toliko velik broj korisnika (Evropski deo Rusije, Turske i cela Nemačka zajedno imaju dobrano preko te brojke) šanse su da replike ne treba postavljati preterano gusto jedna blizu drugoj. Nakon kratkog vremena bi se primetilo gde bi bilo potrebno ubaciti replike kako bi podaci bili pristupačniji svima.

Dodatna stvar koju dobijamo replikama jeste osiguranje da ako jedna baza padne imamo vid bekapa na drugim lokacijama i one bi mogle da je zamene dok se ona ne podigne ponovo, opet na osnovu neke od postojećih instanci. Zbog ovog razloga predlaže se „full table replication“ strategija kako bi se očuvao sveukupni integritet iako bi se žrtvovala moć procesiranja. (https://www.stitchdata.com/resources/data-replication/)

3. Strategije kešitanja

Kako bi dobili najviše od keširanja trebalo bi iskoristiti nekoliko mogućnosti kešitanja. Kako su slike podaci koji se najčešće samo čitaju one bi trebalo da budu keširane na duže periode (na 72h optimalno), preporučena strategija bi bila read-through cahce strategija gde bi se za kešitanje koristila dodatna memorija bliže bazi. Slično kao i slike, stvari kao što su LegacyCategory, City i Country takođe mogu da budu keširane na sličan način jer se očekuje da će se retko menjati.

Većina ostalih podataka predstavlja nešto što je sklono promenama, dodavanjima i brisanjima i bolje bi bilo da se keširanje na tim delovima izostavi. Izuzetak je zeleni deo podšema relacije za koji bi bilo moguće ubaciti keširanje na aplikacijskom nivou jer prilikom kreiranja rezervacija se čitaju podaci o korisnicima te bismo i tu dobili dodatno poboljšanje. (https://apachebooster.com/blog/5-different-types-of-server-caching-strategies-and-how-to-choose-the-right-one/)

4. Procena za hardverske resurse potrebne za skladištenje svih podataka u narednih 5 godina

4.1 Prosečne veličine po podatku

LegacyCategory =>

Id[Int] = 4 bajta

Name[Nvarchar] = prosek 10 Chars = 10 bajta

MinPoints[Int] = 4 bajta

MaxPoints[Int] = 4 bajta

Benefits[Nvarchar] = prosek 50 Chars = 50 bajta

Discount[Float] = 4 bajta

>Total: za jednu torku = 76 bajta

Country =>

Id[Int] = 4 bajta

Name[Nvarchar] = prosek 10 Chars = 10 bajta

>Total: za jednu torku = 14 bajta

City =>

Id[Int] = 4 bajta

Name[Nvarchar] = prosek 10 Chars = 10 bajta

Country[Id] = 4 bajta

>Total: za jednu torku = 18 bajta

FurtherClientInfo =>

UserId[Int] = 4 bajta

CollectedPoints[Int] = 4 bajta

NumberOfPenalties[Int] = 4 bajta

>Total: za jednu torku = 12 bajta

AdditionalInstructorInfo =>

UserId[Int] = 4 bajta

AvailableFrom[Date] = 8 bajta

AvailableTo[Date] = 8 bajta

ShortBiography[Nvarchar] = u proseku 300 Chars = 300 bajta

>Total: za jednu torku = 316 bajta

RegistrationReason =>

UserId[Int] = 4 bajta

Reason[Nvarchar] = u proseku 150 Chars = 150 bajta

DenialReason[Nvarchar] = retko odbijeni (ovo polje 0) = 0 bajta

IsReviewed[Bool] = 1 bit

>Total: za jednu torka = 154 bajta 1 bit

AccountDeletionReason =>

UserId[Int] = 4 bajta

RequestedDateTime[Date] = 8 bajta

Reason[Nvarchar] = u proseku 250 Chars = 250 bajta

IsReviewed[Bool] = 1 bit

IsApproved[Bool] = 1 bit

>Total: za jednu torku = 262 bajta 2 bita

UserVerificationKeys =>

UserId[Int] = 4 bajta

VerificationKey[Guid] = 16 bajta

IsUsed[Bool] = 1 bit

>Total: za jednu torku = 20 bajta 1 bit

Mark =>

MarkId[Int] = 4 bajta

UserId[Int] = 4 bajta

GivenMark[float] = 4 bajta

Description[Nvarchar] = u proseku 250 Chars = 250 bajta

ServiceId[Int] = 4 bajta

IsApproved[Bool] = 1 bit

IsReviewed[Bool] = 1 bit

>Total: za jednu torku = 266 bajta 2 bita

Report =>

ReportId[Int] = 4 bajta

ReportText[Nvarchar] = u proseku 250 Chars = 250 bajta

CreatedDateTime[Date] = 8 bajta

ShownUp[Bool] = 1 bit

SuggestPenalty[Bool] = 1 bit

>Total: za jednu torku = 262 Bajta 2 bita

Reservation =>

ReservationId[Int] = 4 bajta

UserId[Int] = 4 bajta

ServiceId[Int] = 4 bajta

ReservedDateTime[Date] = 8 bajta

IsPromo[Bool] = 1 bit

StartDateTime[Date] = 8 bajta

EndDateTime[Date] = 8 bajta

IsCanceled[Bool] = 1 bit

IsServiceUnavailableMarker[Bool] = 1 bit

ReportId[Int] = 4 bajta

MarkId[Int] = 4 bajta

AdditionalEquipment[Nvarchar] = u proseku 150 Chars = 150 bajta

Price[Float] = 4 bajta

>Total: za jednu torku = 198 bajta

Subscription =>

SubscriptionId[Int] = 4 bajta

ServiceId[Int] = 4 bajta

UserId[Int] = 4 bajta

>Total: za jednu torku = 12 bajta

Issue =>

IssueId[Int] = 4 bajta

UserId[Int] = 4 bajta

IssuedEntityId[Int] = 4 bajta

Reason[Nvarchar] = u proseku 400 Chars = 400 bajta

CreatedDateTime[Date] = 8 bajta

IsReviewed[Bool] = 1 bit

>Total: za jednu torku = 420 bajta 1 bit

Image =>

ImageId[Int] = 4 bajta

ServiceId[Int] = 4 bajta

Bytes[Varbinary] = u proseku 200 000 bajta

>Total: za jednu torku = 200 008 bajta

Service =>

ServiceId[Int] = 4 bajta

OwnerId[Int] = 4 bajta

ServiceType[Int] = 1 bajt (enumeracija)

Name[Nvarchar] = u proseku 80 Chars = 80 bajta

PricePerDay[Float] = 4 bajta

CityId[Int] = 4 bajta

Address[Nvarchar] = u proseku 30 Chars = 30 bajta

Longitude[Float] = 6 bajta

Latitude[Float] = 6 bajta

PromoDescription[Nvarchar] = u proseku 300 Chars = 300 bajta

AdditionalEquipment[Nvarchar] = u proseku 200 Chars = 200 bajta

TermsOfUse[Nvarchar] = u proseku 150 Chars = 150 bajta

AvailableFrom[Date] = 8 bajta

AvailableTo[Date] = 8 bajta

Capacity[Int] = 4 bajta

IsPercentageTakenFromCanceledReservations[Bool] = 1 bit

PercentageToTake[Float] = 4 bajta

>Total: za jednu torku = 813 bajta

PromoAction =>

PromoActionId[Int] = 4 bajta

ServiceId[Int] = 4 bajta

StartDateTime[Date] = 8 bajta

EndDateTime[Date] = 8 bajta

PricePerDay[Float] = 4 bajta

IsTaken[Bool] = 1 bit

Capacity[Int] = 4 bajta

AddedBenefits[Nvarchar] = u proseku 200 Chars = 200 bajta

Place[Nvarchar] = u proseku 50 Chars = 50 bajta

>Total: za jednu torku = 282 bajta 1 bit

AdditionalVillaInfo =>

ServiceId[Int] = 4 bajta

NumberOfBeds[Int] = 4 bajta

NumberOfRooms[Int] = 4 bajta

>Total: za jednu torku = 12 bajta

LinkNavigationToolBoat =>

Id[Int] = 4 bajta

BoatServiceid[Int] = 4 bajta

Serviceid[Int] = 4 bajta

>Total: za jednu torku = 12 bajta

BoatServiceNavigationTool =>

Id[Int] = 4 bajta

Name[Nvarchar] = u proseku 30 Chars = 30 bajta

Description[Nvarchar] = u proseku 100 Chars = 100 bajta

>Total: za jednu torku = 134 bajta

AdditionalBoatServiceInfo =>

ServiceId[Int] = 4 bajta

Length[Float] = 4 bajta

BoatType[Int] = 1 bajt (enumeracija)

NumberOfEngines[Int] = 4 bajta

PowerOfEngines[Float] = 4 bajta

MaxSpeed[Float] = 4 bajta

>Total: za jednu torku = 21 bajt

AdditionalAdventureInfo =>

AdventureId[Int] = 4 bajta

AdditionalOffers[Nvarchar] = u proseku 100 Chars = 100 bajta

>Total: za jednu torku = 104 bajta

Users =>

UserId[Int] = 4 bajta

UserType[Int] = 1 bajt (enumeracija)

Name[Nvarchar] = u proseku 11 Chars = 11 bajta

Surname[Nvarchar] = u proseku 20 Chars = 20 bajta

Password[Nvarchar] = u proseku 9 Chars = 9 bajta

Address[Nvarchar] = u proseku 30 Chars = 30 bajta

CityId[Int] = 4 bajta

PhoneNumber[Nvarchar] = 10 Chars = 10 bajta

Email[Nvarchar] = u proseku 40 Chars = 40 bajta

IsAccountVerified[Bool] = 1 bit

IsAccountActive[Bool] = 1 bit

>Total: za jednu torku = 129 bajta 2 bita

4.2 Estimacije

LegacyCategory => retko korišćena mogućnost dodavanja, pretpostavlja se da se svake godine definišu tri kategorije

Country => retko korišćena mogućnost dodavanja, za broj korisnika koji se posmatra je pretpostavka da se nalaze iz 30 najvećih država.

City => retko korišćena mogućnost dodavanja, za broj korisnika koji se posmatra i za 30 država koje se pretpostavljaju da postoje, 180 gradova.

FurtherClientInfo => od 100 miliona korisnika, neka je 70% običnih korisnika (70 000 000)

AdditionalInstructorInfo => od ostalih 30 000 000 korisnika, trećina su instruktori (10 000 000)

RegistrationReason => svi korisnici su morali pri registraciji da imaju ovu torku (100 000 000)

AccountDeletionRequest => retko korišćena funkcionalnost, najčešće korisnici samo prestanu da koriste aplikaciju. U proseku 5 mesečno.

UserVerificationKeys => pri registraciji svi su morali biti upisanu u ovu relaciju (100 000 000)

Mark => poprilično korišćena funkcionalnost, svaka druga rezervacija biva ocenjena (500 000 mesečno)

Report => poprilično korišćena funkcionalnost, 2/3 rezervacija dobiju svoj izveštaj (670 000 mesečno)

Reservation => poprilično korišćena funkcionalnost, 1 milion mesečno

Subscription => poprilično korišćena funkcionalnost, petina običnih korisnika se pretplati na servis na svaka 3 meseca (56 000 000 godišnje)

Issue => svaki 20 izveštaj je negativan i otvara se žalba za njega, takođe deo korisnika se žali na usluge vezane za servise (45 000 mesečno)

Image => svaki servis ima u proseku 10 slika, svaki vlasnik servisa ima u proseku 4 servisa (1 200 000 000)

Service => svaki vlasnik u proseku 4 servisa (120 000 000)

AdditionalVillaInfo => trećina svih servisa su vile (40 000 000)

AdditionalBoatInfo => trećina svih servisa su brodovi (40 000 000)

AdditionalAdventureInfo => trećina svih servisa su avanture (40 000 000)

LinkNavigationToolBoat => svaki brod ima u proseku 2 navigacione naprave (80 000 000)

BoatNavigationTools => praktično zanemarljivo, za sada ima 4, verovatno se neće menjati značajno pa će se zanemariti

PromoAction => često korišćena, svaki 5 vlasnik servisa jednom mesečno pravi akciju (6 miliona mesečno)

4.3 Računica

LegacyCategory = 3 * 5 * 76 bajta = 1140 bajta

Country = 30 * 14 bajta = 420 bajta

City = 180 * 18 bajta = 3240 bajta

FutherClientInfo = 70 mil * 12 bajta = 8.4 * 10^8

AdditionalInstructorInfo = 10 mil * 316 bajta = 31.6 * 10^8

RegistrationReason = 100 mil * 150 bajta = 154 * 10^8

AccountDeletionRequest = 5 * 12 * 5 * 262 bajta = 76.8 * 10^3

UserVerificationKeys = 100 mil * 20 bajta = 20 * 10^8

Mark = 500 000 * 12 * 5 * 266 bajta = 79.8 * 10^8

Report = 670 000 * 12 * 5 *262 bajta = 105,32 * 10^8

Subscription = 70 mil / 5 * 4 * 5 * 12 bajta = 33,6 * 10^8

Issue = 45 000 * 12 * 5 * 420 bajta = 11,34 * 10^8

Image = 10 * 30 mil * 4 * 200 008 bajta = 2,4 * 10^14

Service = 4 * 30 mil * 813 bajta = 975,6 * 10^8

AdditionalVillaInfo = 30 mil * 4 / 3 * 12 bajta = 4.8 * 10^8

AdditionalBoatInfo = 30 mil * 4 / 3 * 21 bajt = 8,4 * 10^8

AdditioanlAdventureInfo = 30 mil * 4 / 3 * 104 bajta = 41,6 * 10^8

LinkNavigationToolBoat = 2 * 40 mil * 12 bajta = 9,6 * 10^8

BoatNavigationTools = 536 bajta

PromoActions = 30 mil / 5 * 12 * 5 * 282 bajta = 1015,2 * 10^8

Reservation = 1 mil * 12 * 5 * 198 bajta = 1188 * 10^8

Sve ukupno => 2402,4 * 10^11 bajta što se svede na oko 218 TB ili 0.218 PB. Uz strategije za repliciranje baze na više instanci ovaj broj bi se skalirao sa brojem replikacija.

5. Predlog strategije za postavljanje load balancera

Load balancer, kako bi bio iskorišćen najbolje moguće i kako bi se najviše izvuklo iz njega, treba da bude zaseban server. Same adrese servera na kojima radi aplikacija mogu biti nezavisni i trebalo bi da se

nalaze na privatnim adresama i da se cela komunikacija vrši preko load balancera. Takođe, zbog načina na koji je implementirano praćenje sesije, nije neophodno da load balancer bude „sticky“.

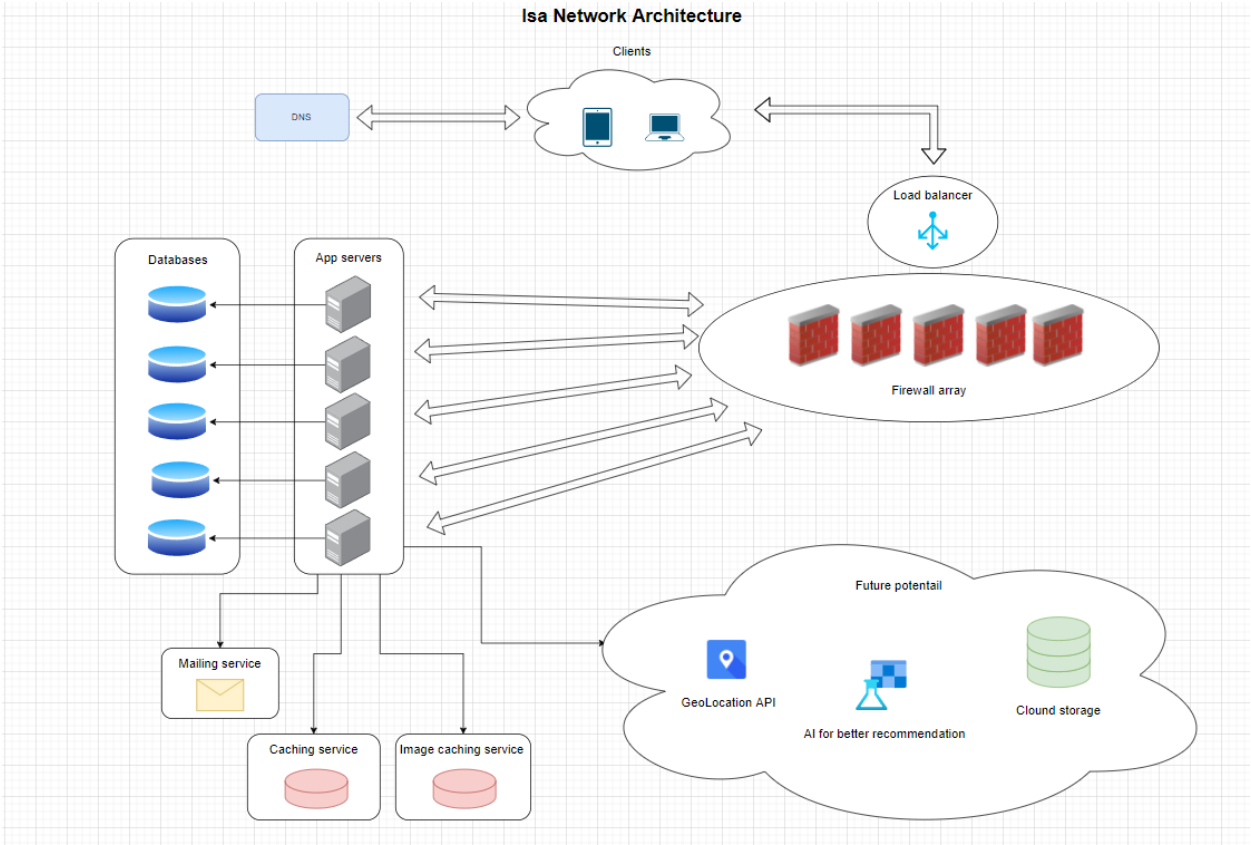
Dodatno, u koliko bi jedan server trebalo restartovati, ili ugasiti zbog bilo kakvog održavanja ili otkaza, aplikacija bi i dalje živela na drugim serverima što je benefit posedovanja dodatnog servera za load balancer.

6. Predlog operacija koje treba nadgledati u cilju poboljšanja sistema

Kako je cilj aplikacije što veći komfor pri rezervisanju i što veća interaktivnost sajta, frontend je deo aplikacije na kojem treba raditi. Kako same operacije trenutno ne predstavljaju veliki napor za sistem, predlaže se implementacija event-sourcing mehanizama kako bi se vremenom utvrdilo koje su operacije najzahtevanije, koje oduzimaju najviše vremena korisnicima i raditi na tome da se to vreme smanji, forme olakšaju i dobije bolji user-experience u budućnosti.

Deo sajta kod kojeg se očekuje mogući zastoj, jeste najveći deo memorije koji će se crpeti, slike. Trenutno na serveru ne postoji ni jedno ograničenje veličine slike, formata, ili bilo kakav vid kompresije istih. Vremenom dobiće se velika baza sa velikim slikama, jer razvoj tehnologije dovodi do poboljšanja kamera i samim tim su slike veće. Kako bi dobili optimum, pretpostavlja se da će u budućnosti biti potrebno ubaciti neko ograničenje pri dodavanju slika i neki vid kompresije istih.

7. Crtež dizajna predložene arhitekture



Pored same arhitekture, dodat je i predlog za za integraciju sa već postojećim sistemima predviđenim za budućnost aplikacije.

GeoLocation Api bi se koristili za bolje pozicioniranje servisa i samim tim bi se manje resursa u bazama koristilo za čuvanje tačnih adresa, gradova i država sistema i čuvale bi se samo koordinate dok bi se preko api-ja dobavljaje ostale bitne informacije.

Ubacivanjem cloud storage omogućio bi se još jedan vid zaštite od propadanja baza. Takođe, njihova pristupačnost danas je poprilično velika i poseduje auto sync i updating mehanizme. Naravno, uz sve to poseduje visok stepen bezbednosti. Na cloud bi se mogle prebaciti slike kako bismo rasteretili sistem od toga.

Poslednje, AI je nešto što bi se moglo ubaciti nakon nekog vremena, kada se prikupi dosta informacija o korisnicima. AI bi se koristio kao dobar recommendation service korisnicima. Uzimao bi u obzir njihovu lokaciju i njihova prethodna interesovanja i na osnovu toga dao najbolje preporuke, što bi dovelo do sveukupno lepšeg iskustva upotrebe servisa.