# Proof of Concept

Студент 1 – Ања Максимовић PA114/2020

Студент 2 – Биљана Мијић PA124/2020

Студент 3 – Ивана Илијин PA104/2020

Студент 4 – Наталија Поповић PA117/2020

## Дизајн шеме базе података

Дизајн шеме базе података креира у Drawio-у се налази у фолдеру под називом ‘Скалабилност’.

## Предлог стратегије за партиционисање података

У нашој апликацији, главни фокус је на резервацији различитих врста медицинске опреме, што може брзо довести до прелаза лимита за податке јер се резервације никада не бришу, већ се само могу отказати (што их неће уклонити из базе). У случају великих софтверских система са обиљем података, обично се користи партиционисање података. У нашем случају, најбоље би било размислити о вертикалном и/или хоризонталном партиционисању података.

Хоризонтално партиционисање нам омогућава да разбијемо табеле са великим бројем објеката на више мањих. У нашем случају, табеле са терминима су табеле које вероватно имају највише података. Предлажемо да хоризонтално партиционишемо табеле термина по id-у власника термина. Тиме би се подаци делили на више мањих делова, што би олакшало менаџмент и приступ подацима. Међутим, треба бити свести о потенцијалним недостацима, као што је неуравнотеженост оптерећења сервера ако неки власници имају значајно више термина него други. Једно могуће решење за ово би било да се подаци додатно партиционишу по типу термина, а затим и по власнику/инструктору.

Други приступ би био партиционисање термина по времену почетка или краја. Ово би омогућило да се термини који су у далекој будућности, као што су слободни термини, чувају независно од резервација и термина у ближој будућности или садашњости. Такође, термине и резервације мањег значаја можемо држати на серверима са нижим перформансама.

Партиционисање података такође може у великој мери повећати сигурност наше апликације. Осетљиве податке бисмо могли да чувамо на посебним серверима код којих се могу применити веће сигурносне мере. Мејлове и лозинке корисника бисмо чували на посебним серверима. Величина упита би се смањила јер се ови подаци ређе користе (само при аутентификацији корисника), док другим информацијама о корисницима чешће приступамо.

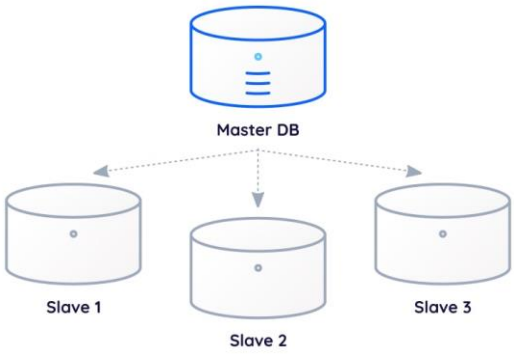
Поред овога, нешто што бисмо могли применити при имлементацији је вертикално партиционисање табела са великим бројем података. Поделили бисмо иницијалну табелу User по улози и самим тим избегли смо пренатрпаност једне табеле и честе приступе истој ради измени или уписивању.

## Предлог стратегије за репликацију базе и обезбеђивање отпорности на грешке

Предлог за репликацију базе подразумева имплементацију slave-master архитектуре. У овом моделу, секундарне (slave) базе би преузеле одговорност за читање података, попут слободних термина и ентитета, док би примарна (master) база обрађивала захтеве за уписивање. Мастер база би садржала све податке, док би славе базе имале копије истих. Овај приступ има за циљ да растерети примарне базе од захтева за читање података, што доприноси побољшању перформанси и очувању интегритета података. У случају преоптерећености или квара мастер базе, једна од славе база би преузела улогу мастера док се проблем не реши. С друге стране, ако дође до пада неке од славе база, мастер база би преузела захтеве те базе како би осигурала непрекидност рада апликације. Ова архитектура обезбеђује отпорност на грешке и боље перформансе, уз вишеструку редундантност података.

За стратегију репликације базе важно је узети у обзир факторе попут физичке удаљености корисника, брзине система и обима нових података. Предлаже се да се реплике поставе на локацијама које оптимизују доступност података корисницима, уз евентуално додавање реплика како би се побољшала доступност података у времену.

Једна од техника за повећање брзине апликације је географско размештање сервера близу група корисника, што омогућава бржи приступ подацима. Комбинујући ову технику са репликацијом података путем master-slave архитектуре, обезбеђује се ефикасан приступ подацима уз минимално оптерећење мреже. У случају преоптерећења или квара мастер базе, славе базе преузимају улогу како би одржале континуитет рада апликације, пружајући тако отпорност на грешке и безбедност података у случају природних катастрофа.

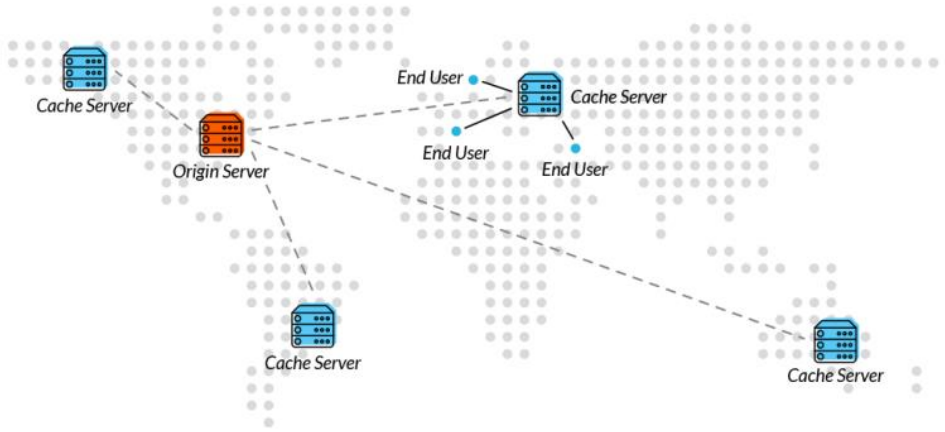



## Предлог стратегије за кеширање података

Кеширање података је кључни елемент у развоју високо растућих система. Велика количина података и чест приступ бази могу значајно успорити систем, нарушавајући корисничко искуство и одвраћајући клијенте од коришћења веб апликације. Имплементација овог процеса најефикаснија је за кеширање претежно статичких података, као што су информације о компанијама. У ситуацијама када број резервација достиже милионе на месечном нивоу, подаци о доступности ентитета постају високо променљиви. Међутим, и ове податке можемо чувати у кеш меморији, ажурирајући их сваки пут када се промене у бази.

Овај процес кеширања можемо додатно оптимизовати комбинујући га са Event Sourcing-ом, који нам омогућава да пратимо догађаје у систему. У ограниченим кеш меморијама, најбоље је кеширати податке који се често користе, као што су најпопуларнији ентитети или периоди заузетости, што је релевантно за тренутни датум. Анализом записаних догађаја можемо одредити које податке треба чувати у кешу, узимајући у обзир навике корисника при резервацијама.

Када кеш меморија достигне свој капацитет, користимо стратегију LRU (Last Recently Used), где се најстарији подаци избацују из кеша. Имплементација CDN (Content Delivery Network) кеширања такође била би корисна, посебно за брже добављање статичких података. Коришћењем CDN-а, подаци се кеширају на серверима proxy близу корисника, што омогућава бржи приступ подацима, уз смањење оптерећења главног серверског система.




## Оквирна процена за хардверске ресурсе потребне за складишење свих података у наредних 5 година

Претпоставке:

- укупан број корисника апликације је 100 милиона
- број резервација свих ентитета на месечном нивоу је милион
- систем мора бити скалабилан и високо доступан
- У просеку је резервисано 60% слободних термина
- У систему ће постојати по милион термина годишње

Величина потребна за складиштење најчешћих ентитета:

- User ~ 0.33 kB
- Employee ~ 0.32 kB
- Address ~ 0.15 kB
- WorkingHours ~ 0.0166kB

- Appointment ~ 1.75kB
- Company ~ 8.91kB
- Equipment ~ 0.35kB
- HospitalContract ~ 0.12kB

Меморија потребна за претпоставке:

- 100 милиона корисника = 33GB
- милион резервација месечно је 60 милиона резервација = 105GB
- уколико постоји милион резервација, посотји око 40 милиона слободних термина = 70GB
- слободна процена за остале ентитете ~ 500GB
  **УКУПНО ЗА 5 ГОДИНА ~ 0.7 TB**

## Предлог стратегије за постављање Load balancera

Када се не придржавамо принципа правилног расподеле оптерећења сервера, врло лако можемо доћи до проблема преоптерећености. Управљање оптерећењем, кроз алат попут **Load balancera**, постаје кључно за скалирање апликације и унапређење њених перформанси. Предлаже се концепт *главног* сервера који би дистрибуирао захтеве према другим серверима на основу њиховог тренутног оптерећења. Међутим, битно је осигурати да сви сервери обраде захтеве на конзистентан начин, без обзира на то који сервер извршава одређени захтев.

Постоји више стратегија за постављање Load balancera. Први предлог стратегије – ***Round-robin*****,** један од најједноставнијих и најчешће коришћених алгоритама. Овај алгоритам би се могао користити у случају да су сви сервери истих или приближних карактеристика. Ово је алгоритам где се захтеви клијената дистрибуирају серверима у ротацији, те се на тај начин релативно равномерно распоређује оптерећење на све сервере. Уколико имамо два сервера, први захтев се прослеђује првом серверу, други другом, трећи првом серверу апликације и тако даље.

Поред тога, истраживања показују да се алгоритам ***Least pending requests*** показао као ефикасан за балансирање оптерећења. Овај алгоритам следи број захтева који чекају на сваком серверу и усмерава нови захтев серверу који има најмање чекајућих захтева. На овај начин се осигурава бржа обрада захтева и смањује се време чекања, што доприноси бољем корисничком искуству. Овај приступ омогућава аутоматско прилагођавање променама у оптерећењу и оптимално управљање ресурсима сервера.

# Предлог које операције корисника треба надгледати у циљу побољшања система

Како бисмо креирали висококвалитетан софтвер, неопходно је да осим техничке и визуелне исправности, обратимо пажњу и на задовољство корисника. Крајњи корисници су кључни за успех власника, инструктора и саме компаније која је развила софтвер, стога је од изузетне важности прилагодити им апликацију што је више могуће.

Једна од могућности је праћење периода у дану или години када се бележи највећи број резервација, како бисмо прилагодили промоције и услуге апликације према тим информацијама. Такође, можемо узети у обзир који ентитети су најчешће резервисани и више их препоручивати корисницима, или истражити шта успешне компаније имају, а што немају мање популарне.

Поред тога, вредне повратне информације које добијамо од корисника су оцене и притужбе. На основу њих, можемо препоручивати добро оцењене компаније корисницима. Студије су показале да корисници више верују компанијама чија је опрема добро оцењена, па бисмо могли у апликацији истакнути "популарне" ентитете на основу оцена и броја рецензија, како би корисници имали веће поверење у компаније чију опрему резервишу.

# Комплетан цртеж дизајна предложене архитектуре