## производителност

Времето за отговор е важен показател в сферата на софтуера и системното инженерство, представляващ продължителността между заявката на потребителя и последващия отговор на системата. Той служи като осезаема мярка за ефективността на системата и често пряко се свързва с удовлетвореността на потребителите. Бързото време за реакция обикновено означава оптимална производителност на системата, което води до положително потребителско изживяване, докато продължителната реакция може да е показателна за неефективност в системата.

Следното уравнение е общ метод за концептуализиране на производителността:

*Време за отговор = Време за обработка + Време на изчакване*

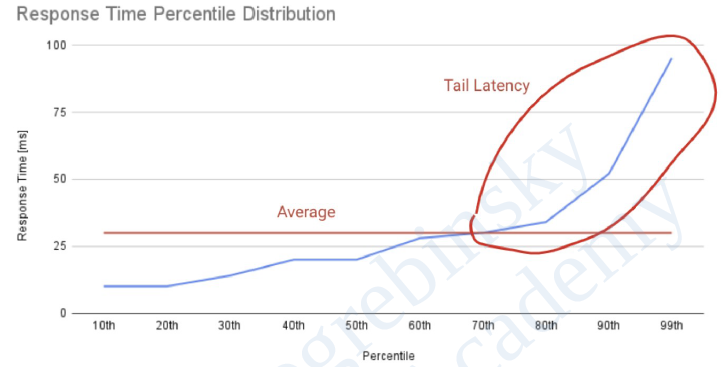
Време за отговор: Това е общото време, необходимо от момента, в който потребителят изпрати заявка до момента, в който получи отговор. По същество това е времето, което потребителят чака да види резултат след започване на действие.

Време за обработка: Това е действителното време, необходимо на системата за изчисляване на резултата след получаване на заявката. Той включва задачи като изчисляване на данни, запитване към бази данни и всяка друга работа, която системата извършва, за да изпълни заявката.

Време на изчакване: Това представлява времето, което заявката прекарва в чакане в опашки, преди да бъде обработена. В система с голям трафик могат да постъпят няколко заявки едновременно. Ако системата не може да ги обработи наведнъж, някои заявки трябва да изчакат, като по този начин се увеличава времето за изчакване.

Чрез разбирането и разделянето на времето за реакция на неговите компоненти, системните администратори и разработчиците могат да определят областите за подобрение. Например, ако времето за обработка е дълго, може да е необходима оптимизация на алгоритми или код. Ако времето за изчакване е дълго, това може да е знак, че системата се нуждае от по-добро балансиране на натоварването или увеличен капацитет за обработка.

Ниво на латентност е от съществено значение, тък като гарантира, че преобладаващата част от клиентите получават последователно и отзивчиво обслужване.



Вместо да разчитат единствено на средни стойности, крива на разпределение предоставя по-изчерпателна картина на ефективността. Примерът на фиг.1.Х показва, че 95% от заявките се обработват в сравнително оптимално време, докато 5% отнемат повече. Често процентилите се използват за определяне на цели за ниво на обслужване (SLO) и споразумения за ниво на обслужване (SLA) за производителност на системата. Това е така, защото няколко изключително дълги или кратки времена за реакция могат да изкривят средната стойност, докато крива на разпределение осигурява по-стабилна метрика, която е по-малко податлива на изкривяване от извънредни стойности.

Крайната латентност е ефикасността на заявките в горните процентили (като P95, P99 и P99.9). Акцентът е върху най-слабите заявки. Голямото забавяне може да означава проблеми, които възникват само при определени условия, като конкуренция за ресурси, забавяне на рамка за изпълнение, хардуерни проблеми и т.н.

[Google: 53% of mobile users abandon sites that take over 3 seconds to load | Marketing Dive](https://www.marketingdive.com/news/google-53-of-mobile-users-abandon-sites-that-take-over-3-seconds-to-load/426070/)

Google: 53% of mobile users abandon sites that take over 3 seconds to load

Async Operations:Relevant mainly for IO operations (files, database access,networking, etc.), Caching • Store frequently accessed data close to the API • Usually in Memory • Set expiration and invalidation , Rate Limit • Limit the maximum concurrent requests the API handles API

## Мащабируемост

Трафикът и натоварването са динамични, влияещи пряко върху производителността на системата и възможността за разширяване на бизнеса. Мащабируемостта се отнася до способността на системата да управлява ефективно увеличеното работно натоварване. Познати са три измерения на мащабируемост:

Вертикална мащабируемост се случва чрез надграждане на хардуер като процесор, памет или честотна лента на мрежата, позволявайки да обработва повече трафик. Въпреки че осигурява повишаване на производителността, съществуват ограничения за това колко може да се „надстрои“.

Хоризонтална мащабируемост се постига чрез добавяне на ресурсни единици. Вместо да се подобрява един сървър, множество сървъри се създават, за да разпределите натоварването. Този подход може да предложи висока достъпност и толерантност към грешки. Това въвежда сложности, особено при координацията между ресурсите.

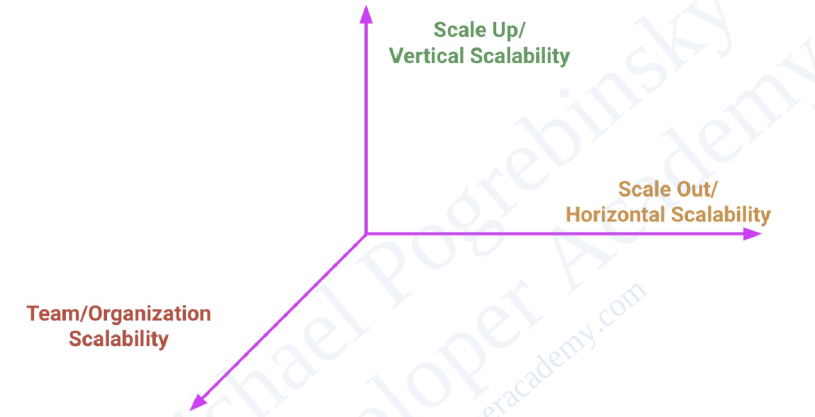
Фигурата показва пример.



Фиг.1.

Екипна или организационна мащабируемост: От гледна точка на разработчиците, мащабирането не е свързано само с производителността на системата, но и със скоростта на разработка на софтуер. С нарастването на инженерните екипи производителността може да намалее поради фактори като по-чести срещи, конфликти в кода и други.

Обща визуализация на типовете мащабируемост:



## Висока наличност

Наличността е първостепенен атрибут на качеството. Наличността на системата оказва влияние както върху потребителите, така и върху бизнеса: потенциална загуба на приходи и загуба на клиенти.

Наличността е частта от времето, през което дадена услуга е функционална и достъпна. Тя може да бъде изразена като процент от времето на работа спрямо сумата от времето на работа и времето на в застой:

*Availability = uptime / (uptime + downtime)*

Също така може да се оцени с помощта на показатели като MTBF (средно време между отказите) и MTTR (средно време до възстановяване), с формулата:

*Availability = MTBF / (MTBF + MTTR)*

Тази формула подчертава, че намаляването на времето за възстановяване (MTTR) може да подобри наличността.

Абсолютната 100% наличност е нереалистична поради необходимостта от поддръжка и надстройки. 90% наличност се равнява на над 2 часа ежедневен престой или 36 дни годишно. 95% се равнява на около час дневно или 18 дни годишно. Индустриалните стандарти, особено определени от облачните доставчици, се движат около 99% до обикновено 99,9% (наричани „три деветки“). Това представлява престой от по-малко от 1,5 минути дневно.

## Толерантност към грешки

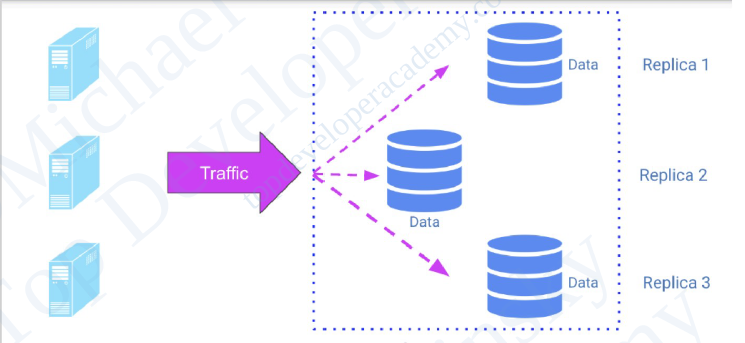
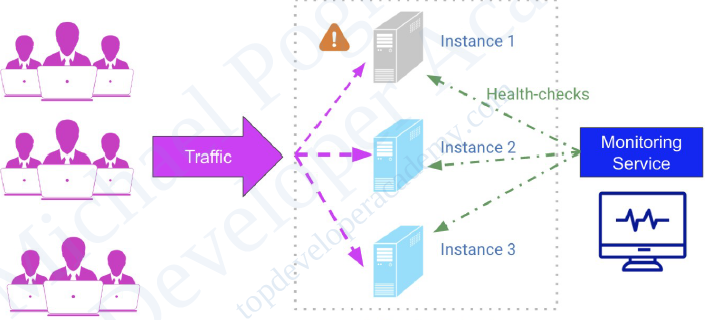
Постигането на висока наличност осигурява поддържането на непрекъснати бизнес операции. Източници на грешки могат да бъдат различни, като някои от тях са:

Човешка грешка: внедряване на неизтестван софтуер, неправилни конфигурации и изпълнение на неправилни команди.

Софтуерни грешки: проблеми като застой, получен от работните рамки, сривове поради липса на памет, изключения за нулев указател и грешки в сегментирането.

Хардуерни повреди: повреди на сървъри, рутери или устройства за съхранение поради ограничен срок на годност, природни бедствия, причиняващи прекъсване на захранването, и мрежови повреди поради инфраструктурни проблеми.

Тактики за толерантност към грешки:

* Предотвратяване на повреда: Елиминиране на единични точки на повреда: Чрез излишък и репликация. Техниките включват изпълнение на реплики на приложения на множество сървъри и реплики на бази данни на различни компютри. 
* Откриване и изолиране на повреда: Мониторингът се свързва с Инструменти, които изпращат периодични проверки, вземайки предвид процентите на грешки, времето за реакция и други показатели за ефективност, за да открие потенциални повреди. 
* За да се поддържат системните операции, като след откриването на дефектен екземпляр трафикът трябва да бъде пренасочен. Също така дефектна система може да бъде рестартирана за потенциално разрешаване на определен проблем. Като алтернатива, връщането към предишна версия може да отстрани проблеми със софтуера.

## Споразумение за ниво на обслужване, цели и индикатори в системния дизайн

Споразумение за ниво на обслужване (**SLA**): договор между доставчик на услуги и неговите потребители или клиенти. Той въплъщава обещанията за качествени услуги, като наличност, производителност и време за реакция . Ако не се спазват, SLA могат да доведат до санкции като възстановяване на средства, кредити за услуги и разширени лицензи.

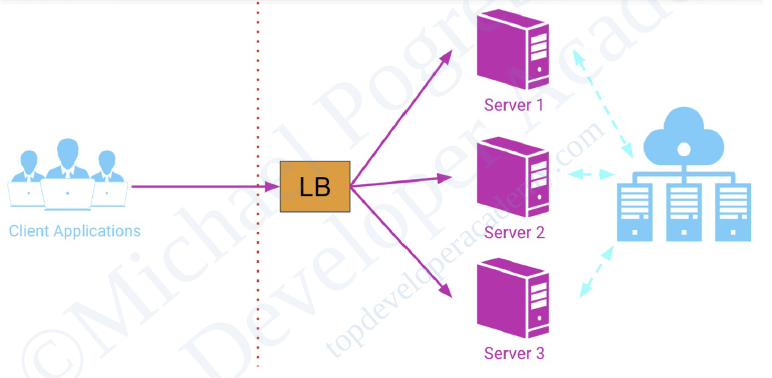
Цел за ниво на обслужване (**SLO**): Индивидуалните цели, определени за една система. Всеки SLO показва целева стойност или диапазон за специфични системни аспекти, като наличност на три деветки или време за реакция под 100 ms на 90-ия процентил.

Индикатор за ниво на обслужване (**SLI**): Количествена мярка за определяне на спазването на SLO. Той представлява данните за ефективността в реално време, които се събират и оценяват дали се постигат SLO.

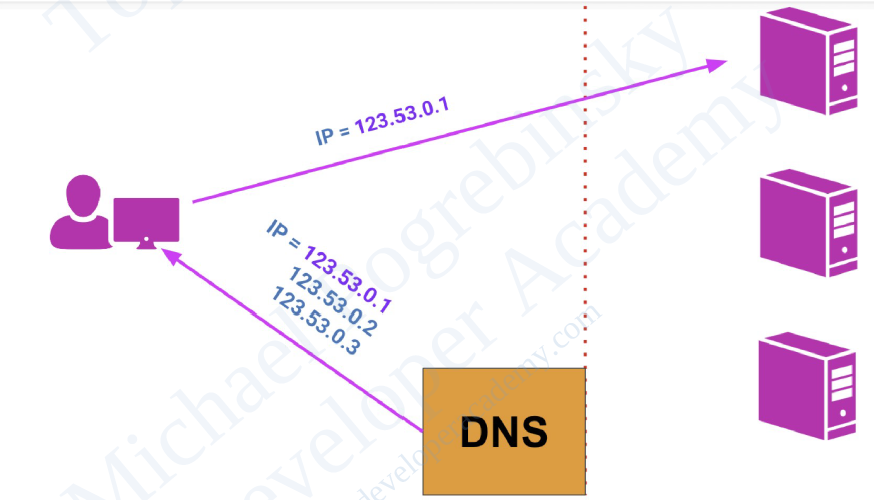
SLA, SLO и SLI са основни за осигуряване на качество на услуга. Докато SLA често се определят от бизнес и юридически екипи, SLO и SLI са повече в областта на софтуерните архитекти и инженери.

## Балансиране на натоварването

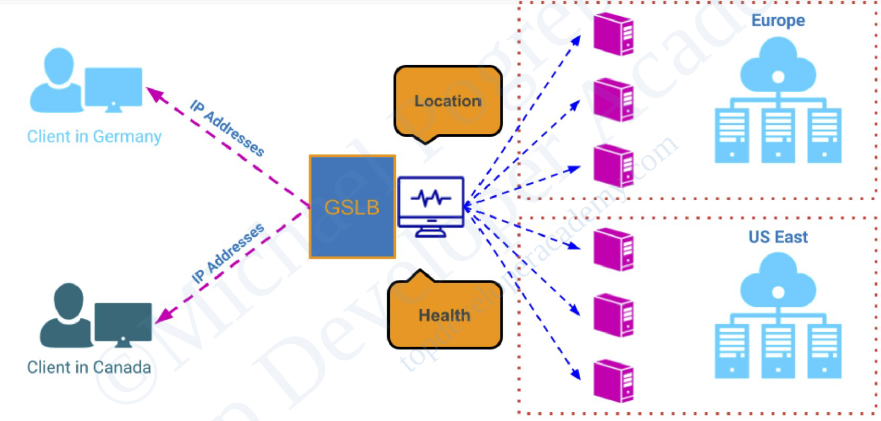
Балансирането играе основна роля в осигуряването на мащабируемост, висока наличност и поддръжка. Балансирането става чрез инструменти разпределящи входящия трафик между множество сървъри, като гарантират, че нито един сървър не е претоварен.



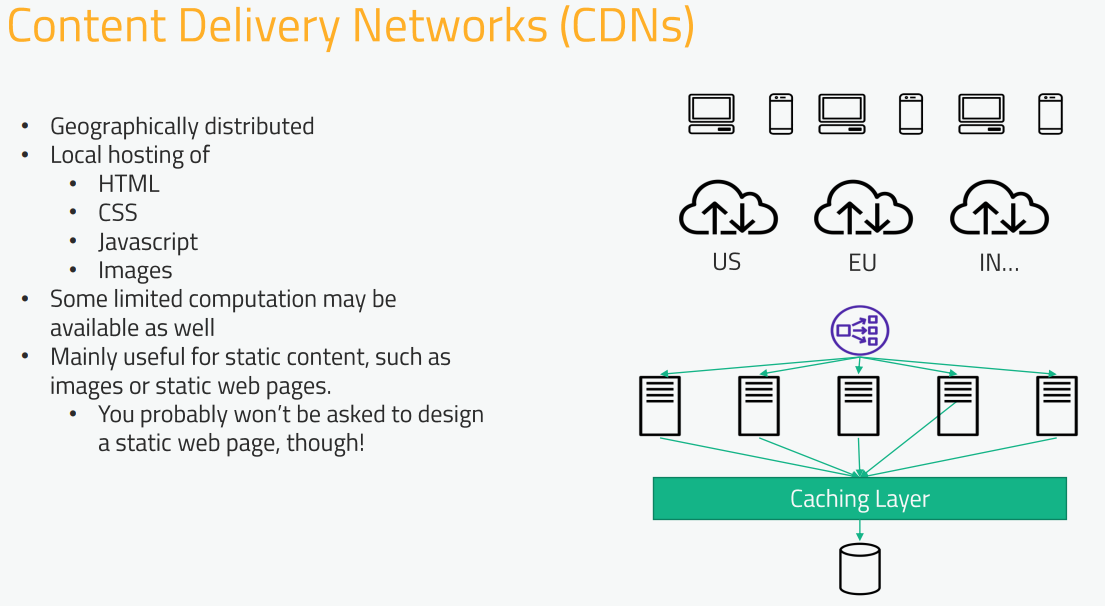
Абстракционен слой, представя множество сървъри като един обект, като по този начин прикрива вътрешността на системата. Това улеснява хоризонталното мащабиране, което позволява добавянето или премахването на сървъри, също така предоставя възможност за извършване на поддръжка, без прекъсване на услугата.

Друг вид балансиране на натоварването може да бъде на DNS. Използва системата за имена на домейни за разпределяне на трафика, но е ограничена в проверките за изправност и излага IP адресите на сървъра, като по този начин създава потенциални рискове за сигурността. 

Глобално балансиране на натоварването на сървъра (GSLB): Хибридно решение, съчетаващо функциите на DNS с интелигентността на балансиращите натоварването на хардуера или софтуера (Специализирани устройства, оптимизирани за разпределяне на трафик или програми, които се изпълняват на компютри с общо предназначение за управление). GSLB може интелигентно да насочва трафика въз основа на местоположение, изправност на сървъра и други параметри, което го прави полезен за възстановяване след бедствие.



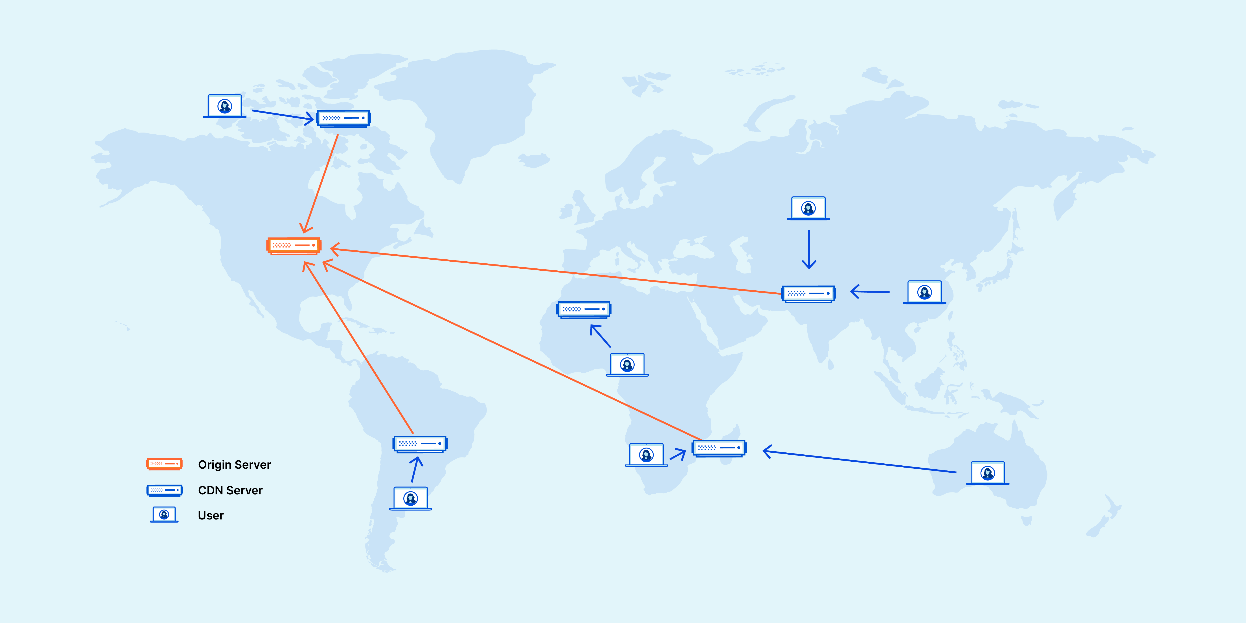
мрежата за доставка на съдържание (CDN)



CDN е система от стратегически разположени сървъри по целия свят. Той кешира и доставя съдържание на потребителите въз основа на тяхното географско местоположение. Чрез обслужване на съдържание от близко местоположение забавянето се намалява, което кара уебсайтовете да се зареждат по-бързо. Разликата с уеб хостът е, че той съхранява съдържанието на уебсайта, докато CDN само го кешира и доставя. Докато CDN не заместват традиционния хостинг, те го допълват, като подобряват производителността и намаляват разходите за честотна лента на първоначалния сървър.

Предимства от използването на CDN:

* Подобрено време за зареждане на уебсайта: Близък CDN сървър гарантира по-бързо време за зареждане на страницата.
* Намалени разходи за честотна лента: CDN използват кеширане и други оптимизации, за да минимизират данните, предоставяни от първоначалния сървър, което намалява разходите за хостинг.
* Повишена наличност на съдържание и излишък: CDN могат да се справят с големи скокове на трафик и са проектирани да бъдат по-устойчиви от традиционните хостинг настройки.
* Подобрена сигурност: CDN могат да предложат защита срещу DDoS атаки, да предоставят сигурни SSL/TLS сертификати и други функции за сигурност.



CDN са стратегически разположени в точките за обмен на интернет (IXP) за оптимална свързаност и намалени транзитни времена. Те използват различни оптимизации, включително хардуерни и софтуерни подобрения, за да осигурят бързо и ефективно доставяне на съдържание. Разпределеният характер на CDN гарантира намалено забавяне, тъй като потребителите се свързват към географски по-близък сървър.

CDN използват балансиране на натоварването, за да разпределят равномерно мрежовия трафик между множество сървъри. Те използват интелигентни системи за преодоляване при отказ, за да се справят с прекъсванията на сървъра или центъра за данни, пренасочвайки трафика към работещи сървъри или центрове за данни. CDN често използват актуални TLS/SSL сертификати, за да гарантират сигурни връзки и трансфер на данни. Те могат също така да предлагат допълнителни функции за сигурност като защитни стени за уеб приложения (WAF) и DDoS защита. CDN намаляват броя на заявките към първоначалния сървър, като обслужват кеширано съдържание, което води до спестяване на честотна лента и намалени разходи.

## Кеширане

Кеширането е техника за запазване на подмножество от данни в слой от високоскоростно съхранение, за да се ускори бъдещо извличане на данни. Кешовете временно съхраняват данни в хардуер с бърз достъп, намалявайки необходимостта от достъп до бавно основно хранилище. В сравнение с конвенционалните бази данни, RAM и базирани на памет машини: позволяват по-висока скорост на заявките. Кеширането се използва в множество технологични слоеве, включително операционни системи, мрежи за доставка на съдържание, уеб приложения и бази данни. Те са от съществено значение за приложения, които изискват достъп в реално време до огромни колекции от данни в множество възли. От решаващо значение е да разберете валидността на кешираните данни.

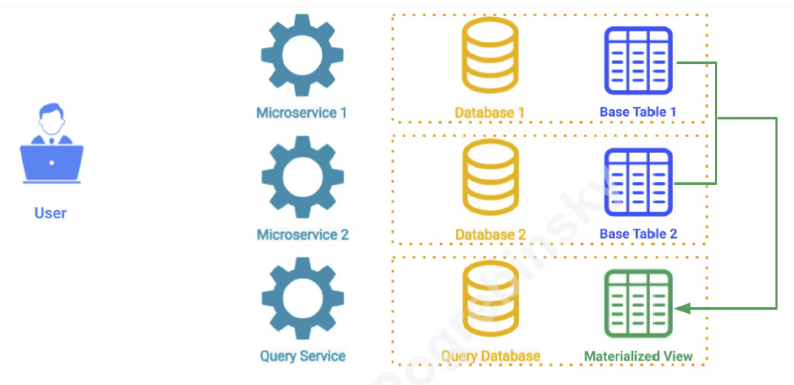
Кеширането е неразделен елемент от съвременните компютри, чиято цел е да осигури по-бърз достъп до данни чрез временно съхраняването им на високоскоростен носител за съхранение. Кеширането може значително да подобри производителността на приложението, да намали разходите и да осигури по-безпроблемно потребителско изживяване, когато се внедри правилно.

Caching Strategies

|  |  |  |  |
| --- | --- | --- | --- |
| Име | How it works | Advantages | Disadvantages |
| Lazy Loading |  |  |  |
| Write-through |  |  |  |
| Adding TTL (Time To Live) |  |  |  |

## Materialized views pattern

Материализираните изгледи са основополагащи за дизайна, управляван от домейн, разделянето на отговорността за запитване на команди и източника на събития. В контекста на DDD сложността на даден домейн е разделена на множество агрегати и обекти, всеки от които капсулира отделна бизнес способност. Когато се внедри CQRS, отговорностите за модификация на данни (команда) и извличане на данни (запитване) са разделени и оптимизирани за съответните им предизвикателства. Следователно страната на заявката може да изисква различно представяне на данни от командната страна, което може да бъде приспособено от материализирани изгледи. Тези изгледи са резултати от заявки, които са предварително изчислени и оптимизирани за операции за четене. В системи, използващи източник на събития, където промените в състоянието се съхраняват като поредица от събития, за разлика от директните модификации на базата данни, материализираните изгледи са незаменими. Те улесняват реконструкцията на текущото състояние на даден обект чрез възпроизвеждане на свързаните с него събития. Това не само осигурява ефективно запитване, но също така се подравнява добре с модела на CQRS, като отделя проблемите и позволява на страната за четене да се мащабира и оптимизира независимо от страната на запис.



## Solid

Single Responsibility Principle (SRP): Принципът на единната отговорност (SRP) е фундаментална концепция в обектно-ориентираното програмиране и дизайн. Тази лекция предостави цялостно разбиране на SRP, неговото значение и приложението му в сценарии за кодиране в реалния свят. Фокусирането върху съставянето на разбираем за човека код е от решаващо значение, защото, въпреки че компютрите са умели в интерпретирането на всеки логически код, хората трябва да го поддържат, поправят и разширяват. Индивидуална дефиниция: Клас или метод трябва да изпълнява само една задача и трябва да изпълнява тази задача изключително добре. По-малко трудно за разбиране и четене. Тъй като трябва да тествате само една конкретна функционалност, тестването е опростено. По-лесно за модифициране и поддръжка. Основната цел на SRP е да насърчи изричното разделяне на отговорностите, което води до по-ясен, по-тестваем и по-лесен за поддръжка код. Разбирането и използването на SRP може значително да подобри качеството на софтуера. Когато се изискват промени в бъдеще, добре разделеният код е по-малко вероятно да съдържа грешки, тъй като е по-малко вероятно модификациите на една част от системата да засегнат други части.