SVEUČILIŠTE U ZAGREBU
Fakultet elektrotehnike i računarstva

Diplomski studij

Razvoj komunikacijske programske podrške

Ak. God. 2021/22.

Odabir arhitekture i tehnologija pri izradi programskog rješenja

Tri odluke kod razvoja programskog proizvoda

- Arhitektura
  - o čemu ovisi?
  - mogućnost evolucije proizvoda
- Integracija
  - ako se morate integrirati s nekim sustavom
  - kako najbolje riješiti integraciju – sada i u budućnosti?
  - može li se pojaviti nešto novo s čime ćete se morati integrirati?
- Tehnologija
  - čime trenutno raspolažete?
  - što će biti u budućnosti?

moguće riješiti odgovarajućom arhitekturom!

Odabir arhitekture

Definicija arhitekture

*"Architecture is the **fundamental organization** of a system, embodied in its **components**, their **relationships** to **each other** and the **environment**, and the principles governing its **design and evolution**"*

(prema ANSI/IEEE)

Ključni elementi arhitekture

- Organizacija
  - orkestracija komponenti, sustava...
- Komponente (moduli)
  - svaki modul ima specifičnu zadaću – podjela zadataka / funkcionalnosti
- Odnosi komponenti
  - hijerarhija, međuzavisnost
  - komunikacija i protokoli u homogenom okruženju
- Okolina (integracija, sučelja)
  - komunikacija i protokoli u heterogenom okruženju
  - brokeri (*message broker*)
- Evolucija
  - može li proizvod/usluga “rasti”?

Kako krenuti s definiranjem arhitekture?

- Odabir arhitekture uvelike ovisi o proizvodu / usluzi
  - uočiti sličnosti onoga što trebate s onim što već postoji
  - koristiti istražene i isprobane uzorke (*architecture/design patterns*)
  - teško je izmisliti nešto novo (često i nepotrebno)!
- Dekompozicija
  - sustav podijeliti u module sa specifičnim i jasnim funkcionalnostima
- Komunikacija modula
  - definirati sučelja
  - homogeno okruženje
- Integracija
  - heterogeno okruženje
  - kako riješiti komunikaciju?
- Dobro razraditi arhitekturu prije implementacije
  - ušteda vremena kasnije kod eventualnih proširenja

Moduli i sučelja

- Modul
  - ima specifičnu zadaću
  - funkcionira kao samostalni entitet
  - *separation of concerns*
  - na razini arhitekture – “crna kutija”
  - može se dijeliti na više pod-modula, ovisno o razini apstrakcije
- Odnosi između modula
  - modul A **koristi** modul B
  - modul A **je dio** modula B
- Sučelja modula
  - nužna za komunikaciju (moduli su “crne kutije”)
  - mora se znati koji su ulazi i izlazi modula (analogija s *firewall-om*)
  - sučelje odražava funkcionalnost
- Tek nakon definicije sučelja svih modula u sustavu može se krenuti u implementaciju!

Na što paziti kod definiranja arhitekture?

- Česte greške kod definiranja arhitekture
  - fokus isključivo na trenutno stanje i trenutne funkcionalnosti (zahtjeve)
  - čim jednostavnija i brža za implementirati u početku
  - arhitektura se gradi s programskim rješenjem
  - zanemarivanje mogućih promjena u infrastrukturi
  - zanemarivanje mogućih promjena u sustavima s kojima je proizvod integriran (npr. rezervacijski sustav)
- Na što još treba obratiti pažnju?
  - dizajn podložan promjeni
  - proizvod dio obitelji proizvoda

Dizajn podložan promjeni

- Očekivati promjene i znati kako reagirati na njih – parametrizacija?
- Što se najčešće mijenja?
- Algoritmi
  - pojavi se novi, brži, optimalniji...
- Reprezentacija podataka
  - npr. ulaz u sustav nije doc datoteka nego (noviji) docx – Rational Requisite za DZ
  - nova i brža vrsta strukture podataka
- Promjene u infrastrukturi
  - nadogradnja operacijskog sustava, novi konektori za bazu podataka...
  - novi hardver – nova vrsta modema, novi diskovni sustav...
- Promjene iz društvene/poslovne sfere
  - novi porezi – npr. PDV s 22 na 25%, fiskalizacija
  - nova vauta – euro
- Evolucija proizvoda
  - koliko će biti složeno dodavati potpuno nove funkcionalnosti?

Proizvod kao dio obitelji proizvoda

- Ne ograničiti se isključivo na primjenu koju trenutno implementirate
  - npr. implementirate rezervacijski sustav za tramvajske karte
  - koliko je složeno taj sustav jednog dana doraditi da služi i za vlakove, autobuse...?
- Pokušati identificirati skup ključnih modula neovisno o implementaciji koje ćete moći iskoristiti za svaku konkretnu primjenu
- Funkcionalnosti specifične određenoj primjeni smjestiti u posebne module
  - kasnije te module možete lakše zamijeniti s modulima za novu primjenu
  - jezgra sustava ostaje ista!

Uzorci i stilovi arhitektura (1/2)

- “recepti” - smjernice za izgradnju programskih rješenja
- razlika između uzoraka i stilova?
  - uzorci – rješavaju problem (npr. kako odvojiti sučelje i model – npr. MVC)
  - stilovi – način na koji se organizira, koncept koji se često ponavlja...
- uvijek se radi o kombinaciji uzoraka i stilova
- uzorci arhitekture
  - 3-slojna (3-tiered)
  - višeslojna arhitektura (*N-tiered*)
  - model-view-controller (MVC)
- stilovi arhitekture
  - klijent – poslužitelj
  - peer-to-peer
  - repozitorij
  - SOA (*service-oriented architecture*)
  - objavi-pretplati (*publish-subscribe*)

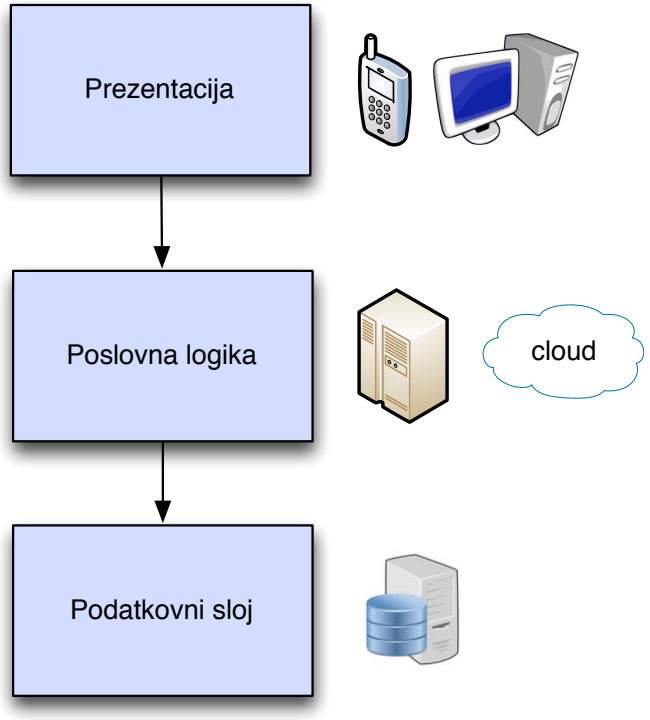
Uzorci i stilovi arhitektura (2/2)

- stilovi se često dijele prema izvedbi (i primjeni!) programskog rješenja
  - kako komponente/moduli komuniciraju?
    - brokeri, *event-driven*
  - kako se dijele resursi (npr. memorija)?
    - repozitorij,
  - kako je arhitektura strukturirana?
    - nor. komponente
  - distribuiranost arhitekture
    - npr. “monolitni sustavi”, klijent-poslužitelj...

- kombinacija stilova i uzoraka!
  - npr 3-slojni klijent-poslužitelj
  - MVC aplikacija se izvodi kao 3-slojna aplikacija u oblaku
  - peer-to-peer vs klijent-poslužitelj?

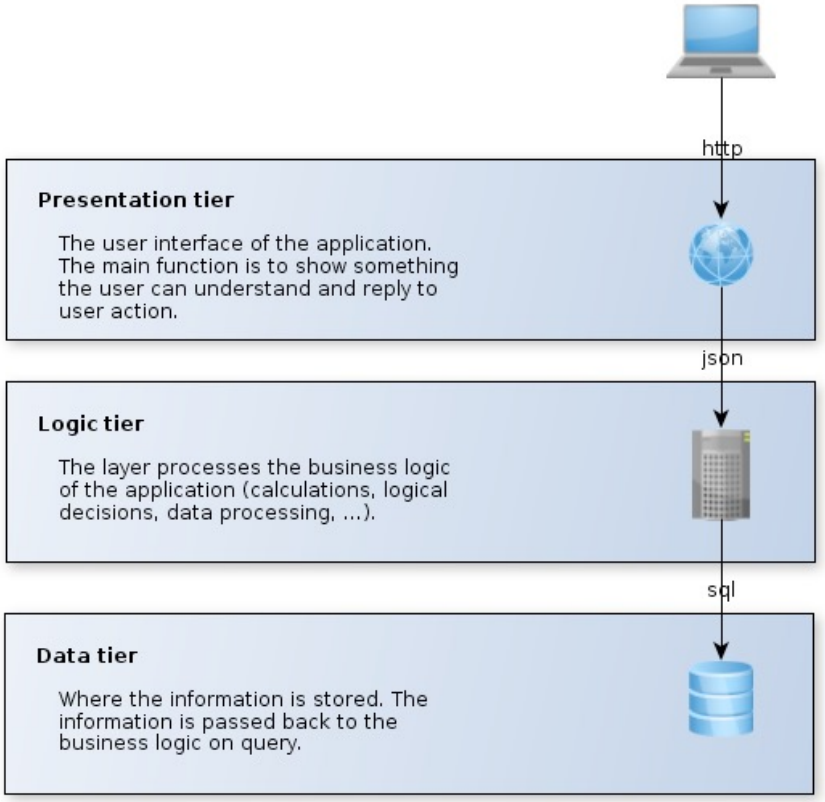
Višeslojna arhitektura (1/3)

- “Tradicionalna” arhitektura
  - *n-tiered*
- Više slojeva sa specifičnim funkcionalnostima
- Svaki sloj komunicira isključivo sa slojem ispod sebe
- Najčešći slučaj u praksi: 3-slojna arhitektura
  - prezentacijski sloj
  - sloj poslovne logike
  - podatkovni sloj
- Više slojeva ovisno o složenosti sustava
- Iskoristivost svakog sloja (*re-usability*)
- Kombinacija n-slojeva s ostalim uzorcima
  - npr. web aplikacije: klijent-poslužitelj + MVC + 3-slojna arhitektura



Višeslojna arhitektura (2/3)

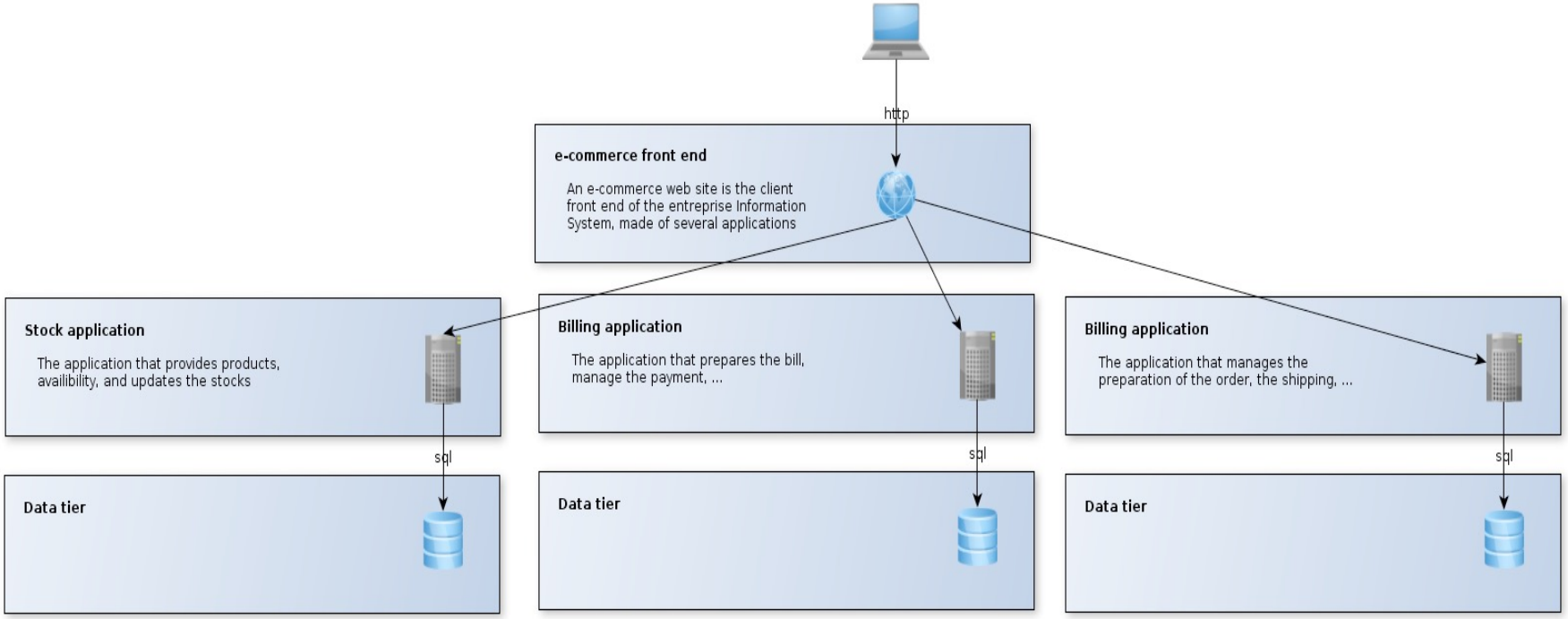
- Primjer: aplikacija za e-trgovinu temeljena na arhitekturi MVC
- zbog sigurnosti aplikacija se mora integrirati kao 3-slojna arhitektura
  - svaki sloj na jednom poslužitelju
  - kombinacija uzoraka!
  - MVC na sloju poslovne logike
  - integracija kao 3-slojna



Primjer prema: http://javathought.wordpress.com/2012/07/04/play-and-3-tier-n-tier-architecture-part-i/

Višeslojna arhitektura (3/3)

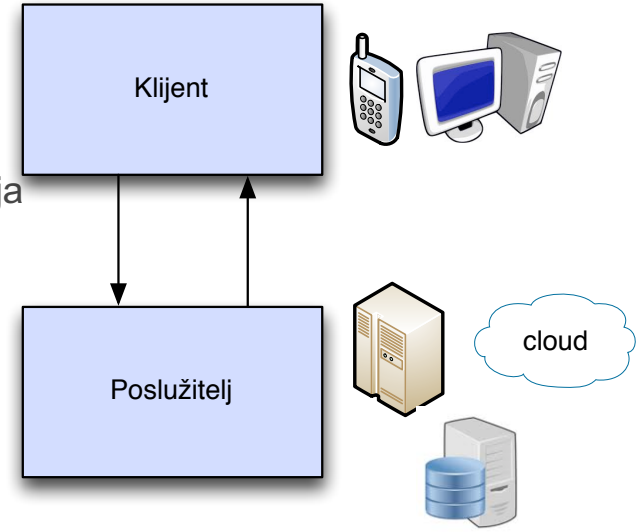
- moguća je i dodatna dekompozicija primjera aplikacije za e-trgovinu!



Primjer prema: http://javathought.wordpress.com/2012/07/04/play-and-3-tier-n-tier-architecture-part-i/

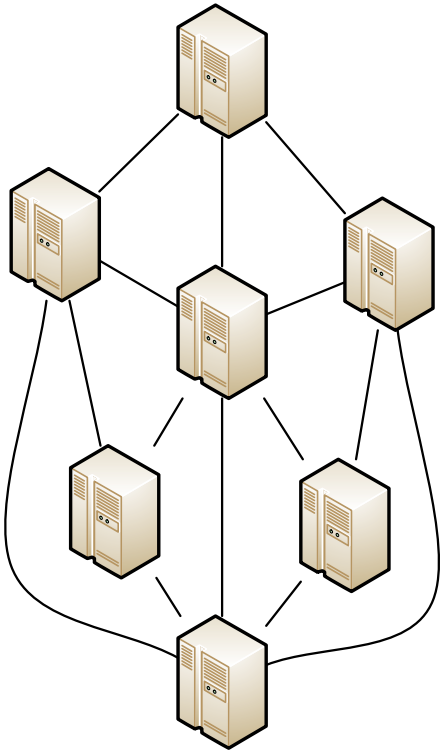
Klijent – poslužitelj

- Još jedna “tradicionalna” arhitektura (npr. *web* stranice)
- Klijent
  - prezentacija
  - pred-obrada podataka
  - komunikacija s bazom podataka posredstvom poslužitelja
- Poslužitelj
  - obrada zahtjeva
  - upravljanje klijentima
  - sigurnost
  - konzistentnost baze podataka
- Otvorena pitanja
  - distribuiranost poslužitelja
  - čim manje složene obrade na klijentu (npr. Javascript?)
  - poslužitelj u oblaku – jedna logička usluga, više instanci



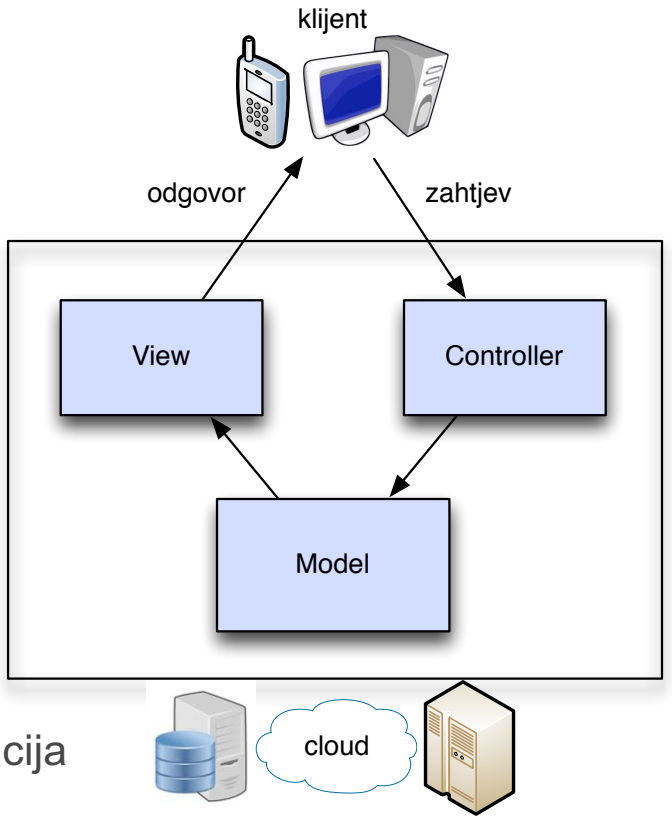
Peer-to-peer

- Generalizirani klijent-poslužitelj
  - klijent je ujedno i poslužitelj
  - poslužitelj je ujedno i klijent
- Naglasak na distribuiranosti
  - distribucija procesiranja
- Mreža ravnopravnih čvorova
  - svaki čvor obavlja dio procesiranja
- Što je sve P2P?
  - SETI@HOME – distribuirano procesiranje
  - Skype – centalna kontrola, tok podataka P2P (nekada!)
  - torrenti? – P2P koncept, ali nije arhitektura aplikacije
- Kontrola?
  - centralizirani
  - decentralizirani



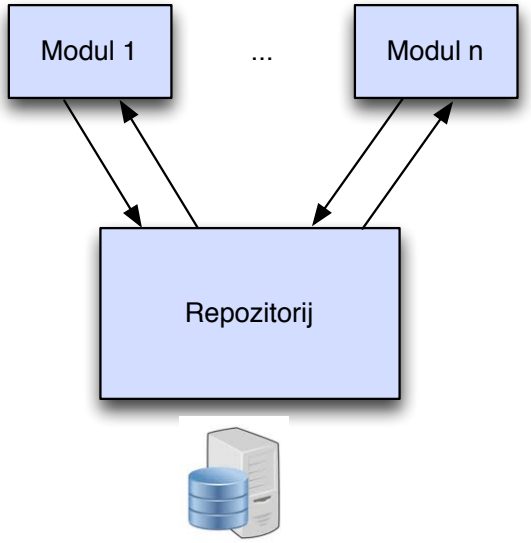
Model-View-Control (MVC)

- Strogo odvaja model, akcije i prezentaciju
- Kontroler (*Controller*)
  - presreće zahtjeve klijenta i prilagođava model
  - obavještava model o promjeni zahtjeva korisnika
- Pogled (*View*)
  - ono što klijent “vidi”
  - mijenja se prema modelu
- Model
  - logika
  - mijenja pogled na zahtjev kontrolera
  - “zna” kako promijeniti pogled
- Elementi su crne kutije
  - nisu (ne moraju biti) povezani kodom
  - poruke preko sučelja
- Osnova većine današnjih složenijih web aplikacija



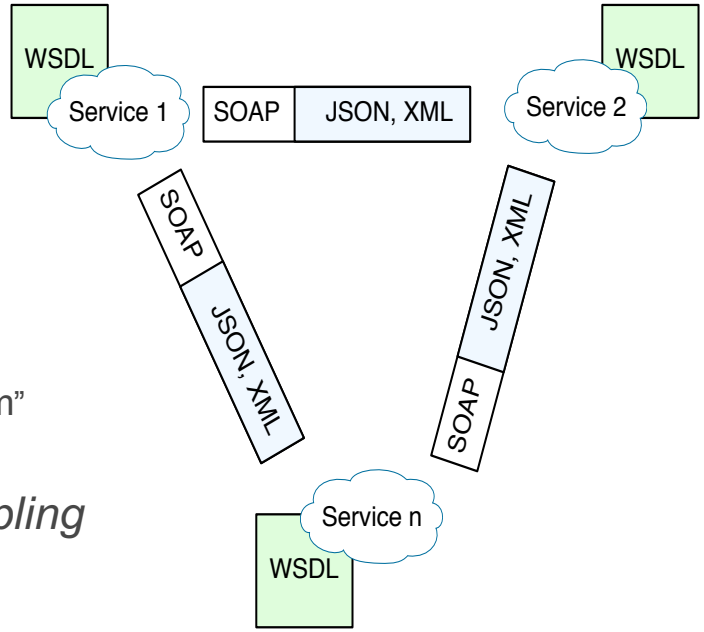
Repozitorij

- Centralni repozitorij kao resurs koji koriste ostali elementi
  - moduli, komponente, web servisi, klijenti...
- Ostali elementi arhitekture ne komuniciraju međusobno
  - isključivo kroz repozitorij
- Naglasak na sinkronizaciji pristupa – konkurentnost
  - kontrola pristupa
  - zaključavanje resursa
  - ....
- Repozitorij upravlja tokom usluge
- Najčešće baza podataka sa dijelom poslovne logike
  - distribuirana
  - centralizirana



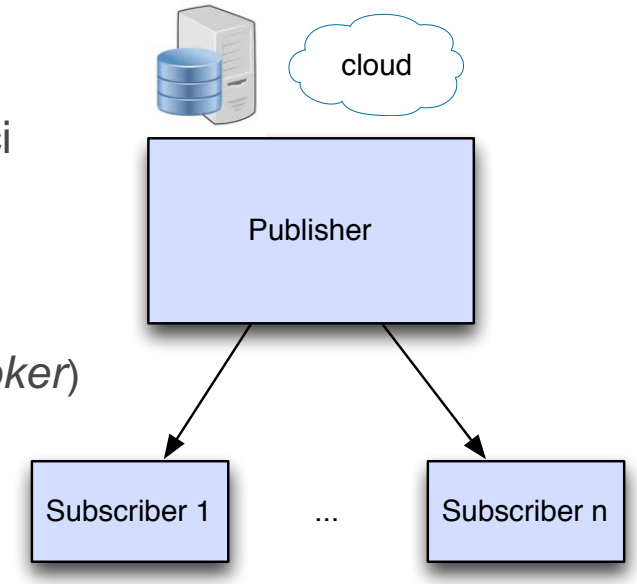
SOA (*Service Oriented Architecture*)

- Arhitektura usmjerena uslugama
- “Klasičan” princip – moduli i sučelja
  - kod SOA-e – usluge i komunikacija između usluga i klijenata
- Opis usluge (servisa)
  - WSDL (*Web Service Description Language*)
- Komunikacijski protokol
  - SOAP (*Simple Object Access Protocol*)
- Zapis podataka
  - XML, JSON
- Integracija
  - jednostavnija jer sve usluge “pričaju istim jezikom”
  - ne ovisi o platformi, infrastrukturi...
- Međusobna neovisnost usluga - *loose coupling*
- Iskoristivost (*re-usability*)



Objavi-pretplati

- Model komunikacije
- Jedan entitet objavljuje, više entiteta je pretplaćeno na objavu
  - *consumer-producer*
- Entitet koji objavljuje ne zna tko su pretplatnici
  - *loose coupling*!
  - ne šalje im “poruke” nego samo objavljuje
  - pretplatnici “uzimaju” poruke (objave)
- Obično se koristi broker poruka (*Message broker*)
  - pretplaćuje se na brokera koji prosljeđuje objave
  - broker za svaki entitet koji objavljuje



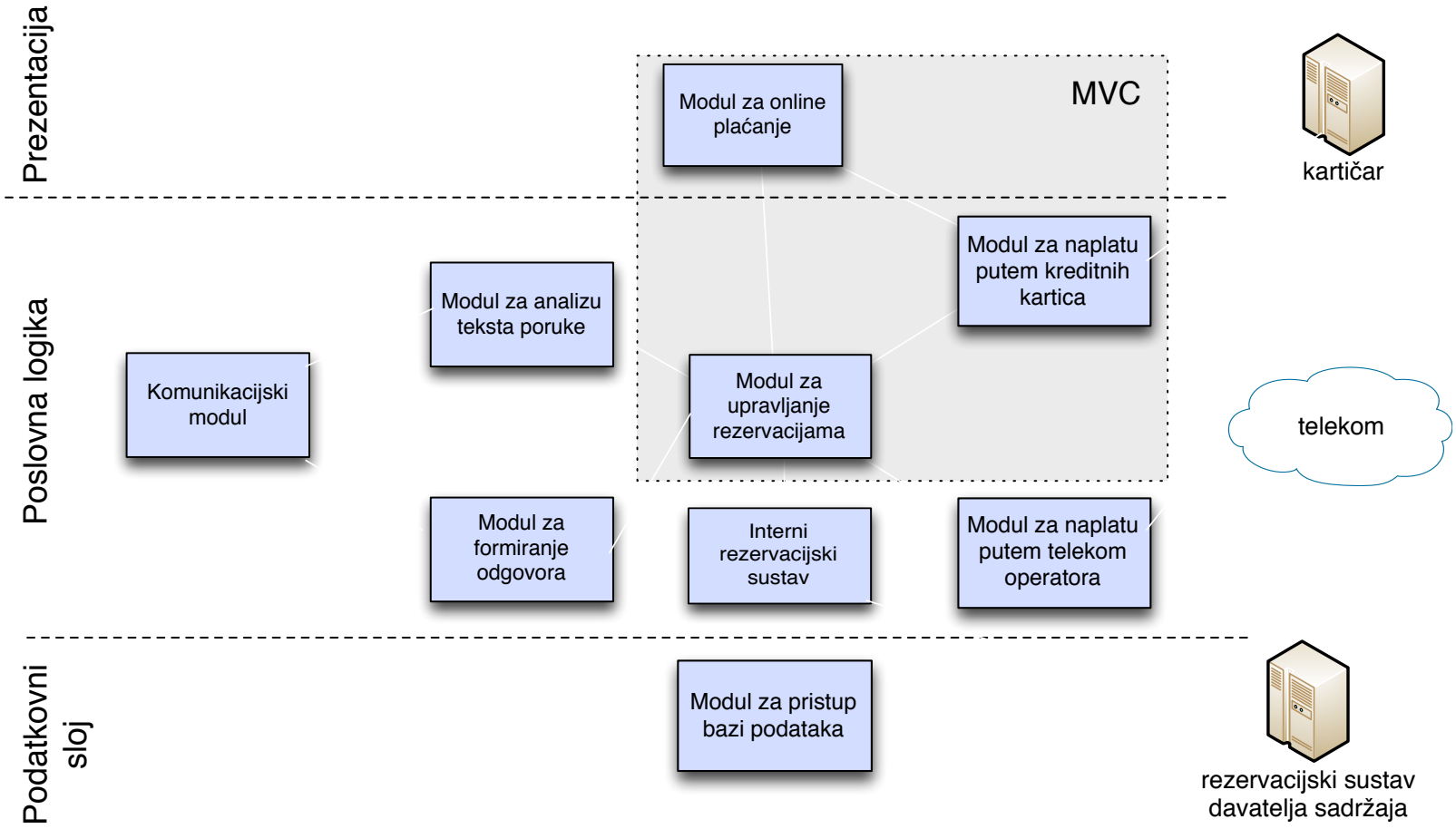
Pogledi na arhitekturu

- Logički
  - raspored modula i odnosi između njih
- Procesni
  - komunikacija, konkurentnost u okviru arhitekture
- Fizički
  - na kojim fizičkim elementima (hardware) se izvode komponente/moduli
- Razvojni
  - pregled softverskih komponenti (npr. Java paketi i klase, klasni dijagrami)
- Slučajevi uporabe
  - raspisivanje slučajeva uporabe koji se izravno odnose na arhitekturu

Parametri kvalitete arhitekture (podsjetnik)

- Pouzdanost
- Raspoloživost
- Portabilnost
- Skalabilnost
- Performanse
- Sigurnost

Primjer definiranja arhitekture - aplikacija za kupnju ulaznica za kino putem SMSa



Odabir tehnologije

Kako odabrati tehnologiju? (1/2)

- Tehnologija nužno ne ovisi o arhitekturi
  - izbor optimalne tehnologije nakon definicije arhitekture
- Koristite tehnologiju s kojom imate najviše iskustva
- Otvoreni kod (*open source*) naspram vlasničkih (*vendor*) rješenja?
  - licenciranje i “zamke” – predzadnje predavanje
- Otvoreni kod
  - besplatan
  - naplaćuje se podrška
  - velika zajednica razvijatelja (*community*)
- Vlasnička rješenja
  - skupo na početku, godišnje licence
  - dobra podrška
  - obično stabilnija opcija

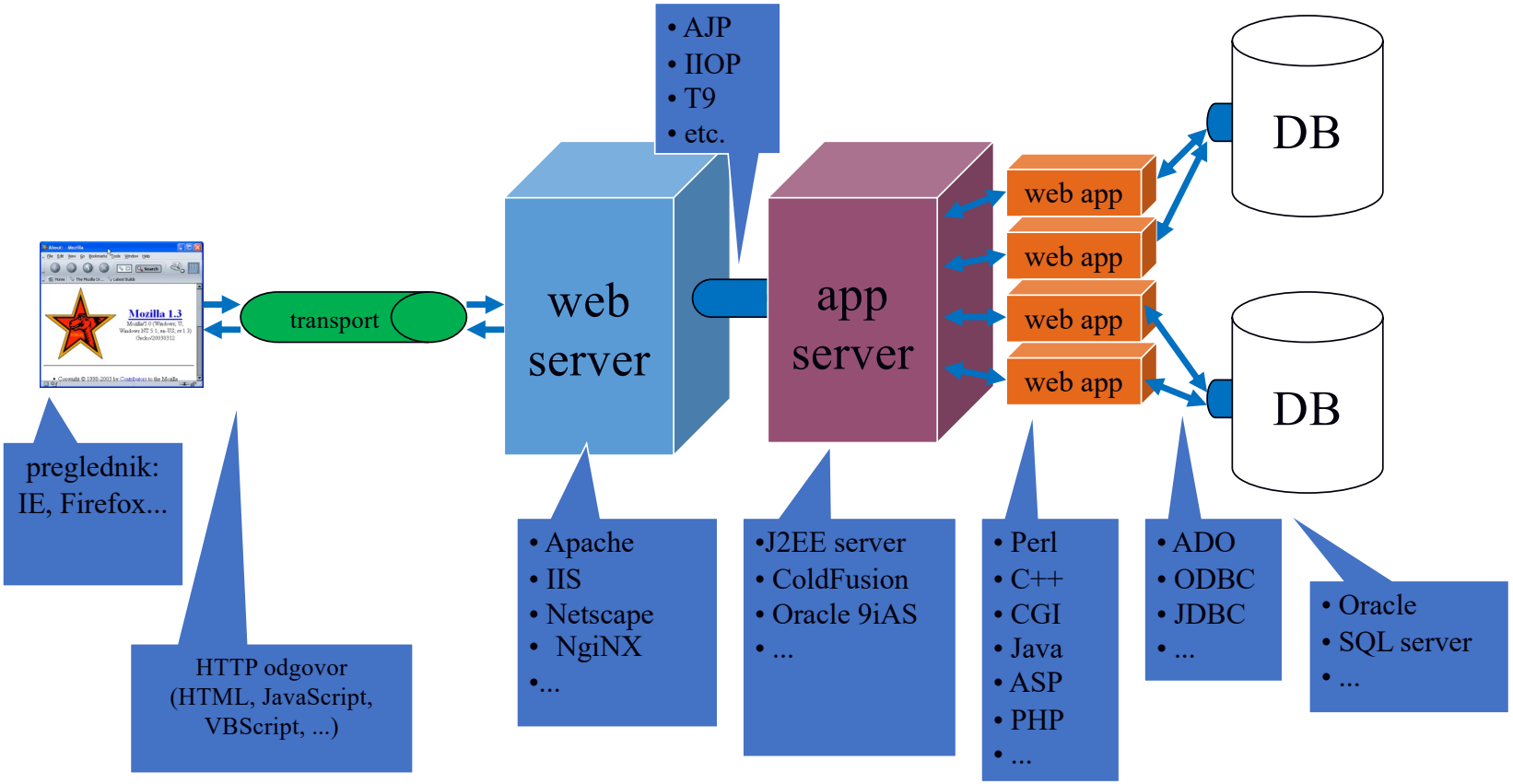
Kako odabrati tehnologiju? (2/2)

- Razvoj vlastitog rješenja ili korištenje gotovih radnih okvira (*framework*)?
- Razvoj vlastitog rješenja
  - dugotrajno
  - znate točno kako sve funkcionira - lakše ispravljanje grešaka
  - dobro za manje projekte ili kada imate vremena
- Korištenje radnih okvira
  - može drastično ubrzati razvoj
  - problem ako dođe do greške - morate znati što se događa unutar “crne kutije”
  - čest slučaj - brzi razvoj ali sporo pronalaženje i ispravljanje grešaka
  - najbolji izbor ako znate kako radni okvir funkcionira!

Primjeri tehnologija – web aplikacije/servisi (1/2)

- Sustavi za upravljanje sadržajem
  - ograničena programabilnost ali brzi razvoj sjedišta (sadržaja)
  - širok raspon programskih jezika (PHP, Python, Ruby, Perl, ASP...)
- Radni okviri za izradu web aplikacija/servisa
  - brzi razvoj jednostavnih do srednje složenih web aplikacija
  - npr. CodeIgniter, Ruby on Rails, J2EE obitelj, Tapestry..
- MVC radni okviri za izradu web aplikacija/servisa
  - namijenjeni složenjim web aplikacijama i servisima
  - dobra podrška za puno konkurentnih korisnika, distribuciju resursa...
  - Spring MVC, Play!...
- Radni okviri za razvoj društvenih mreža
  - podrška za mehanizme društvenih mreža – brži razvoj
  - Elgg, Anahita...

Primjeri tehnologija – web aplikacije/servisi (2/2)

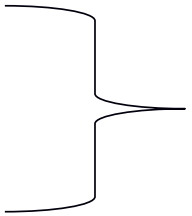


Primjeri tehnologija – nativni razvoj

- Razvoj samostalnih aplikacija za računala
  - J2SE (više ili manje neovisno o platformi)
  - C obitelj, Ptyhon...
  - nekada puno zastupljenije – danas veće usmjeravanje na web i *cloud* rješenja
- Razvoj samostalnih aplikacija za mobitele
  - Android, iOS, Windows Phone, Blackberry...
  - prilična ograničenja kod razvoja (posebno iOS)
- Specifični uređaji – specifični radni okviri / jezici
  - senzori
  - mikroračunala (rapsberry pi, arduino)
  - RFID/NFC sustavi
  - u pravilu postoje vendor API-ji
- Problem razvoja za više platformi - višeplatformska rješenja
  - npr. React Native, Xamarine…

Primjeri tehnologija – domene

- *Cloud*
  - puno novih koncepata i tehnologija za elemente oblaka
  - djelom nasljeđe farma poslužitelja
  - distribuirane baze podataka
  - migracija sjednica
  - mehanizmi repova (*queue*)
  - sigurnosni mehanizmi



veliki izbor radnih okvira za svaki problem

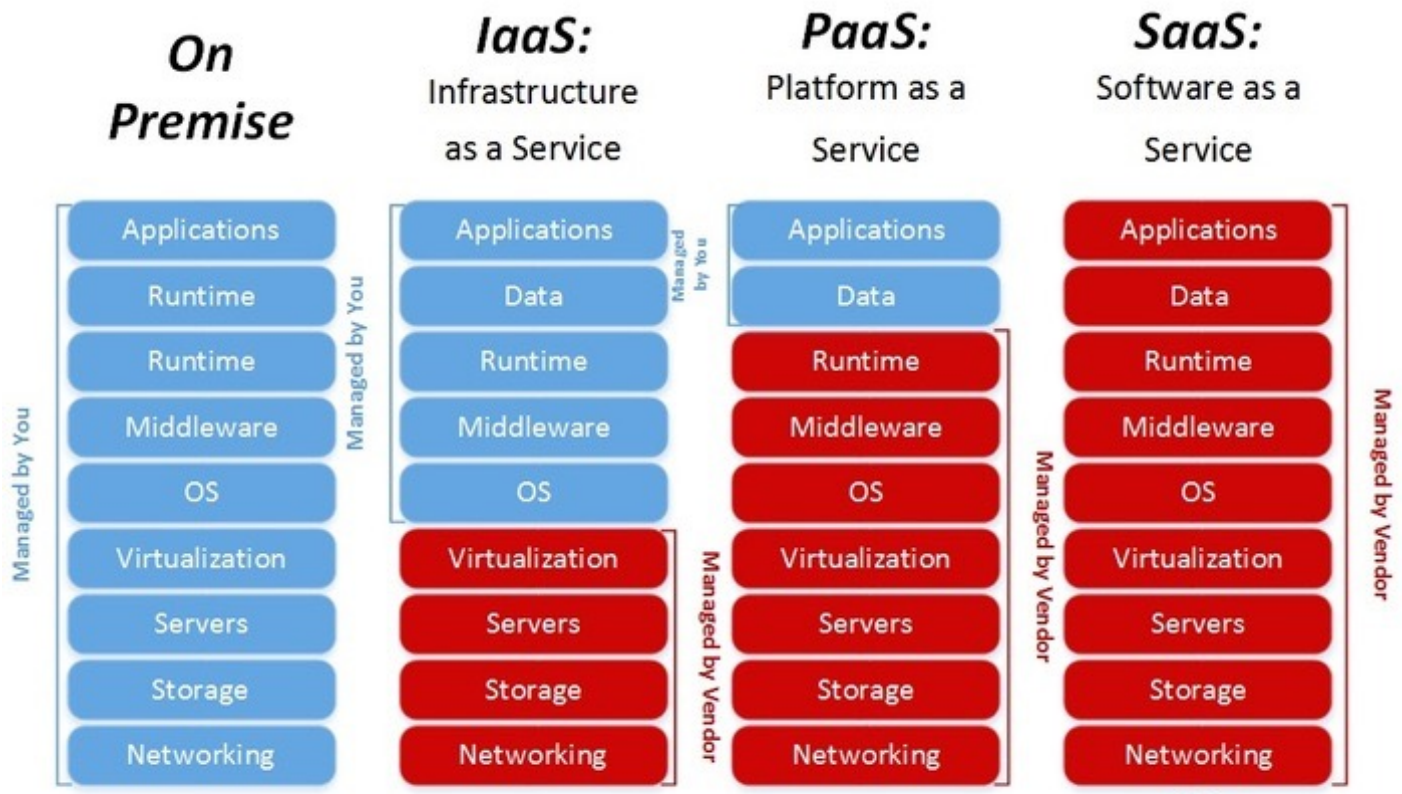
  - Popularne *cloud* platforme
    - Amazon Web Services (https://aws.amazon.com/)
    - Google Cloud Services (https://cloud.google.com/)
    - Microsoft Azure (https://azure.microsoft.com/en-us/)

  - npr. tko upravlja sjednicama?
    - novi koncepti nastali zbog *clouda* (u velikoj mjeri)

Savjeti

- Očekujte promjene u tehnologijama
  - predvidite i pripremite arhitekturu i tehnologije
- Sve “ozbiljnije” aplikacije se prebacuju u *cloud*!
- Pokušajte se odmaknuti od konkretne platforme
  - ako je moguće (problem *look&feel*-a)
  - ubrzanje razvoja za više platformi
- Tržište se brzo mijenja
  - učite nove tehnologije

Kako odabrati *cloud*



https://dzone.com/articles/avoid-being-locked-your-cloud

Cloud tehnologije / tvrtke

http://www.forbes.com/sites/kevinjackson/2012/08/12/cloud-management-broker-the-next-wave-in-cloud-computing/#2e191e563190

Literatura

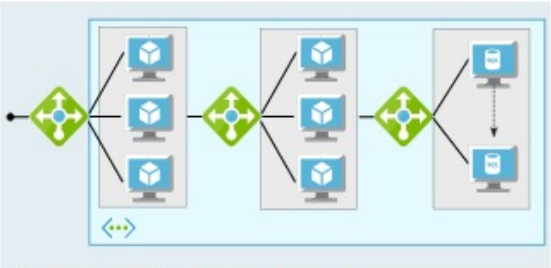
- http://www.codeproject.com/Articles/430014/N-Tier-Architecture-and-Tips - višeslojne arhitekture, članak
- http://jeromejaglale.com/doc/java/spring/mvc - Spring MVC tutorijal
- http://javathought.wordpress.com/2012/07/04/play-and-3-tier-n-tier-architecture-part-i/ - primjer arhitekture Play! web aplikacije
- http://ellislab.com/codeigniter - radni okvir CodeIgniter
- http://www.springsource.org/ - radni okvir Spring
- http://elgg.org/ - platforma za razvoj društvenih mreža
- http://aws.amazon.com/ - cloud platforma
- https://www.playframework.com/ - radni okvir Play!
- http://www.forbes.com/sites/kevinjackson/2012/08/12/cloud-management-broker-the-next-wave-in-cloud-computing/#2e191e563190 - Cloud

Usluge u oblaku

Usluge u oblaku

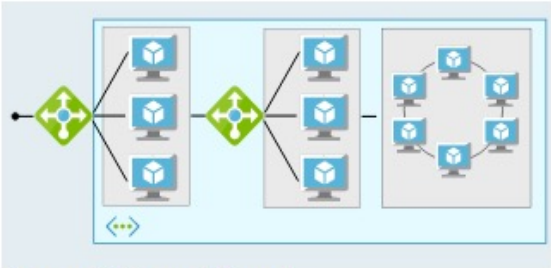
- “Preporučene” arhitekture u oblaku
  - Kako podijeliti opterećenje, ostvariti komunikaciju I integraciju…
- Uzorci dizajna u oblaku (*design patterns*)
  - Kako riješiti neki (tehnički, organizacijski) problem
- Primjeri na *Azure cloud*-u
  - primjenjivo na ostale

Osnovne arhitekture



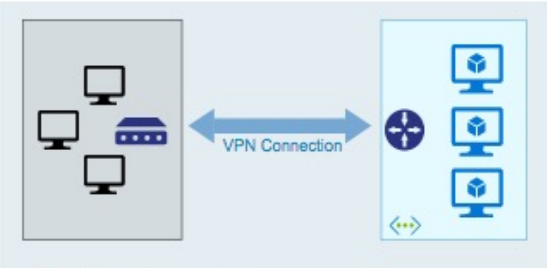
Windows VM workloads

This series starts with best practices for running a single Windows VM, then multiple load-balanced VMs, and finally a multi-region N-tier application.



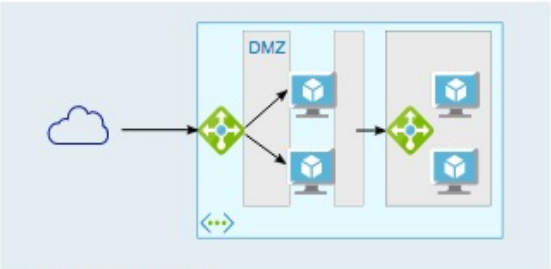
Linux VM workloads

This series starts with best practices for running a single Linux VM, then multiple load-balanced VMs, and finally a multi-region N-tier application.



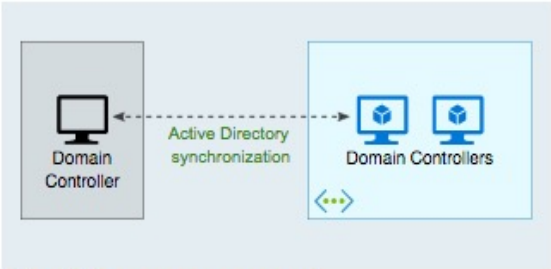
Hybrid network

This series shows options for creating a network connection between an on-premises network and Azure.



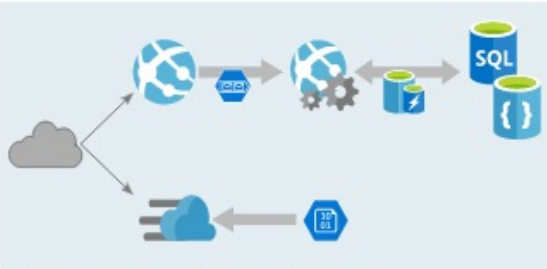
Network DMZ

This series shows how to create a network DMZ to protect the boundary between an Azure virtual network and an on-premises network or the Internet.



Identity management

This series show options for integrating your on-premises Active Directory (AD) environment with an Azure network.



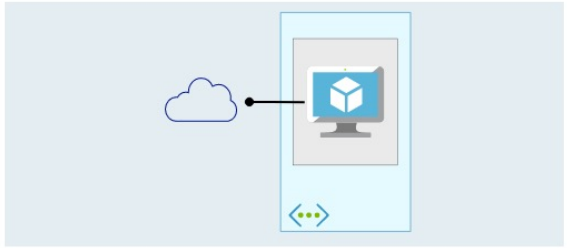
Managed web application

This series shows best practices for web applications that use Azure App Service.

https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/

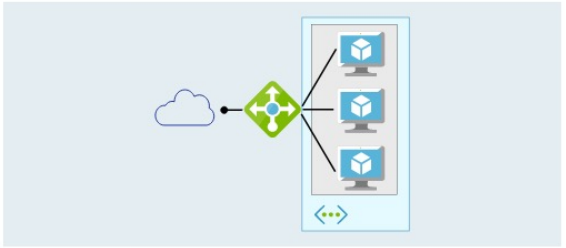
1. Windows / Linux VM

- Više arhitektura ovisno o očekivanom opterećenju
  - jedan virtualni stroj
  - više strojeva s balansiranjem opterećenja
    - N-slojeva s balansiranjem opterećenja između njih
    - Podjela po regijama (zemljopisnim)



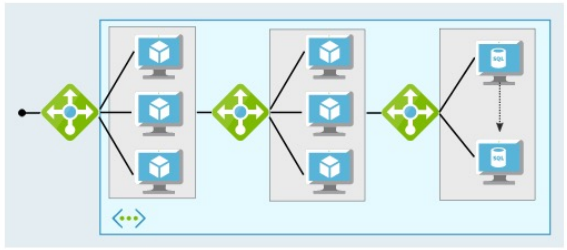
Single VM

Baseline recommendations for running any Windows VM in Azure.



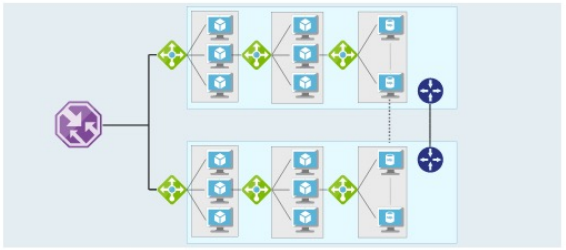
Load balanced VMs

Multiple VMs placed behind a load balancer for scalability and availability.



N-tier application

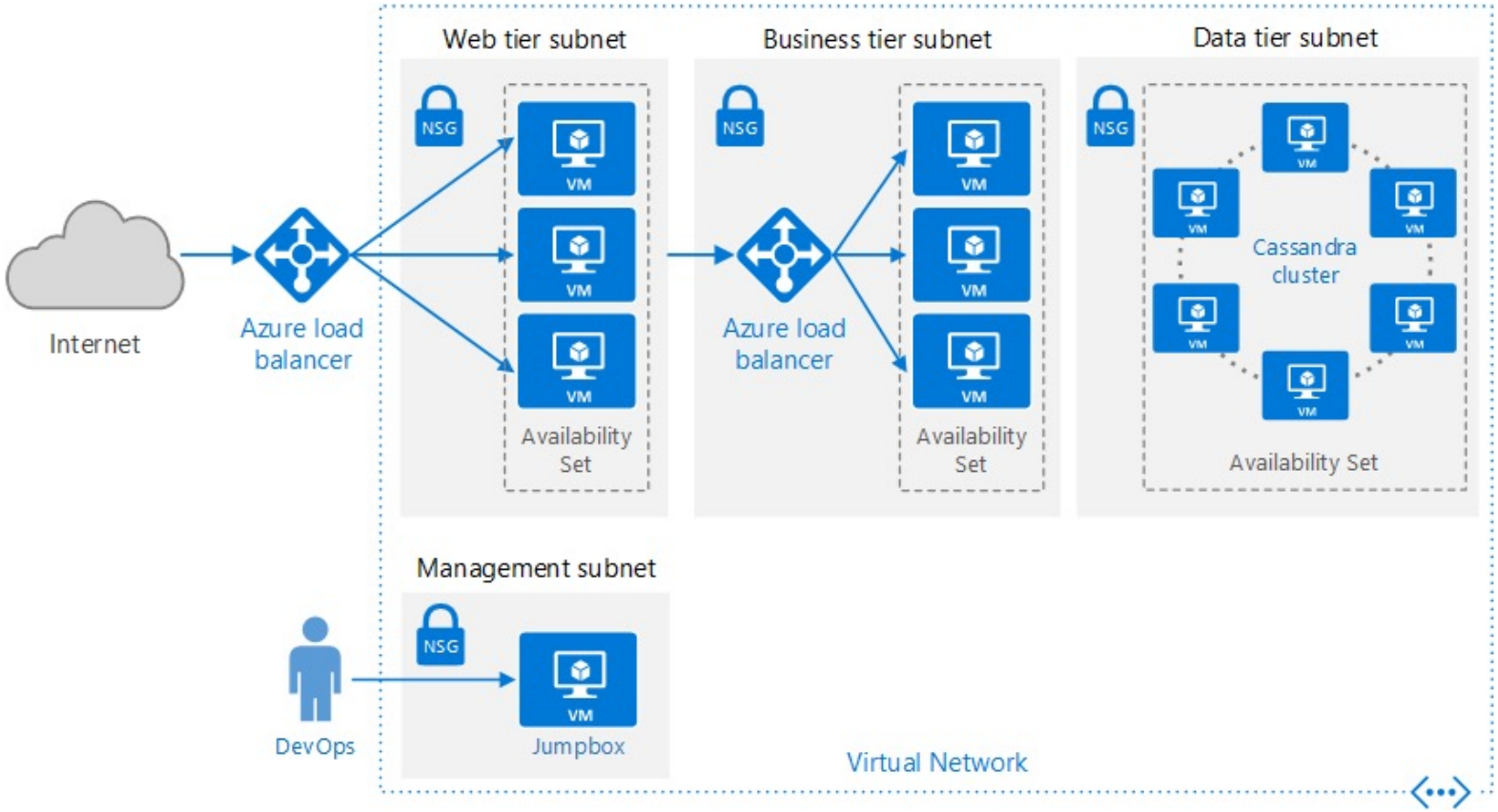
VMs configured for an N-tier application with SQL Server.



Multi-region application

N-tier application deployed to two regions for high availability.

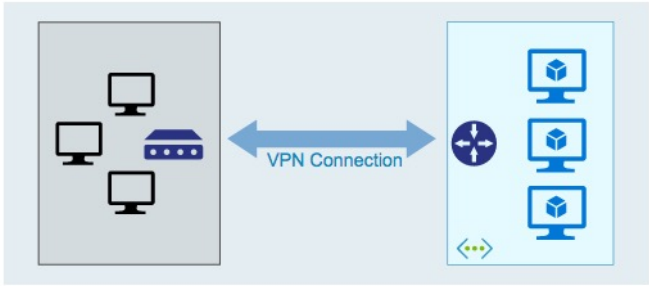
1. Windows / Linux VM – primjer N-slojeva



https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/virtual-machines-linux/n-tier

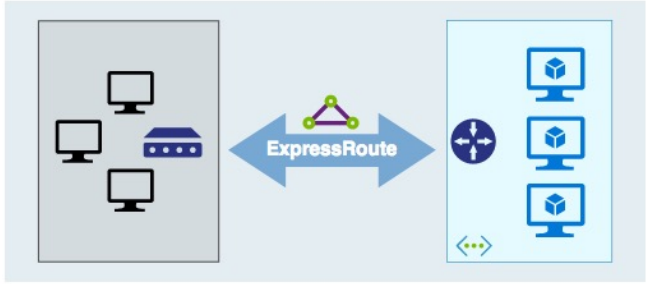
2. Hibridna arhitektura

- Dio programske logike izvodi se izvan oblaka – kako povezati s ostatkom koji je u oblaku?
  - VPN
  - *ExpressRoute* (pouzdaniji VPN)
  - *ExpressRoute* i VPN



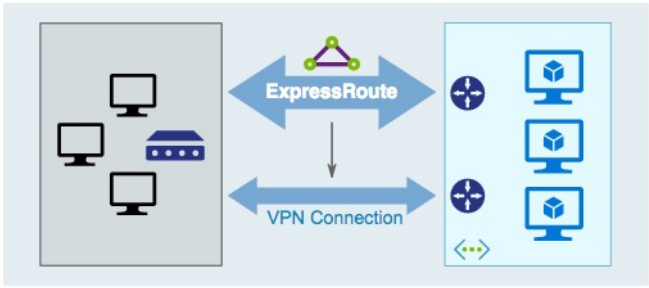
VPN

Extend an on-premises network to Azure using a site-to-site virtual private network (VPN).



ExpressRoute

Extend an on-premises network to Azure using Azure ExpressRoute

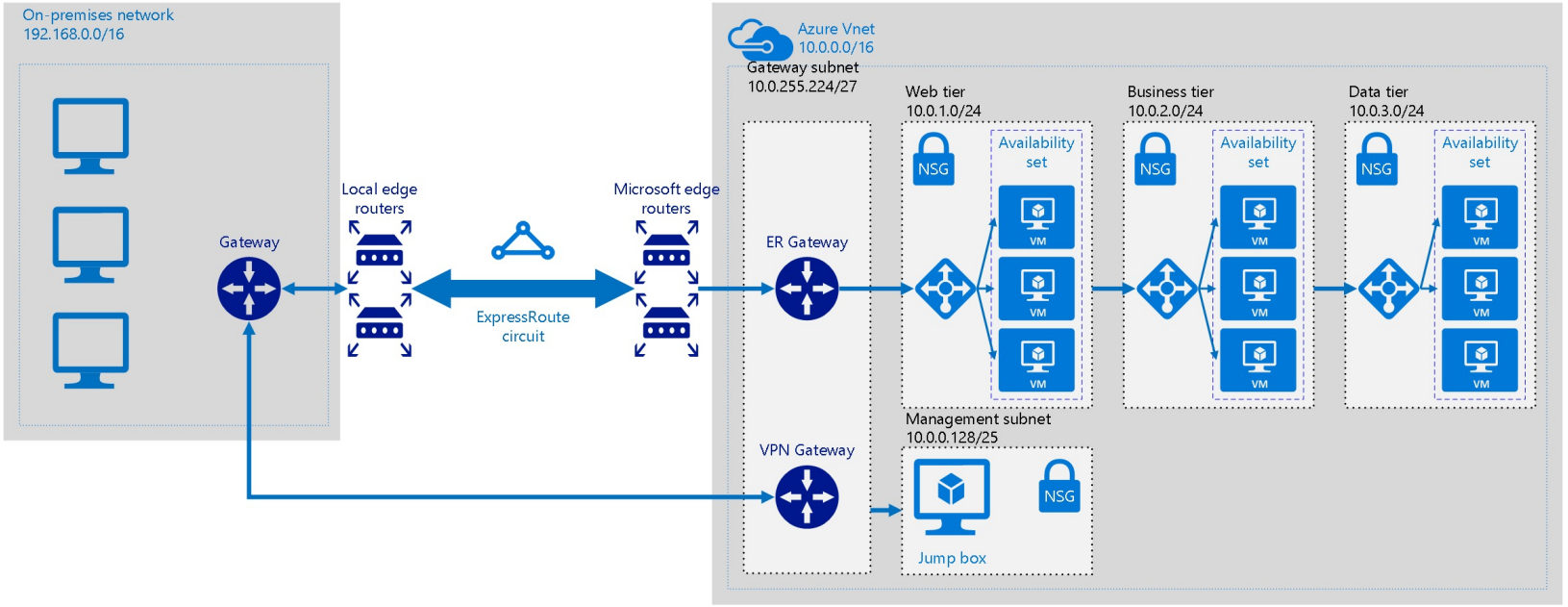


ExpressRoute with VPN failover

Extend an on-premises network to Azure using Azure ExpressRoute, with a VPN as a failover connection.

2. Hibridna arhitektura – primjer VPN i *ExpressRoute*

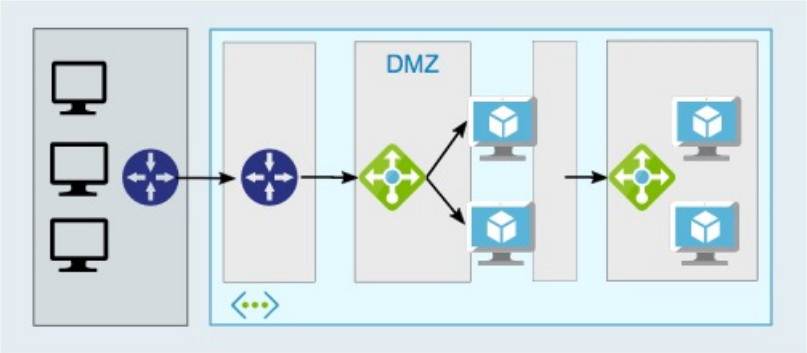
- Važno je uočiti: aplikacija u cloudu je izvedena kao Win / Linux VM u N slojeva
  - kombinacija arhitektura



https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/hybrid-networking/expressroute-vpn-failover

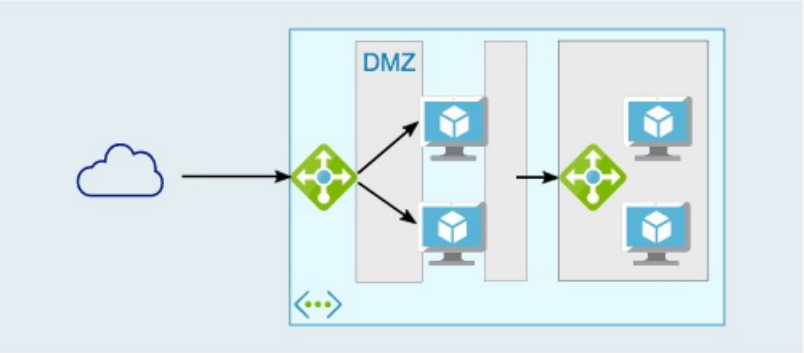
3. Arhitektura s DMZ

- Arhitektura s demilitariziranom zonom (DMZ)
  - DMZ je dio mreže koji je otvoren prema Internetu tj. vanjskim servisima, klijentima i slično
  - Važne funkcije servisa / aplikacije nalaze se u zaštićenoj mreži (npr. baza podataka, poslovna logika…)
  - U DMZ-u tipično poslužitelji web-a, elektroničke pošte, FTP… (Sigurnost u Internetu!)



DMZ between Azure and on-premises

Implements a secure hybrid network that extends an on-premises network to Azure.

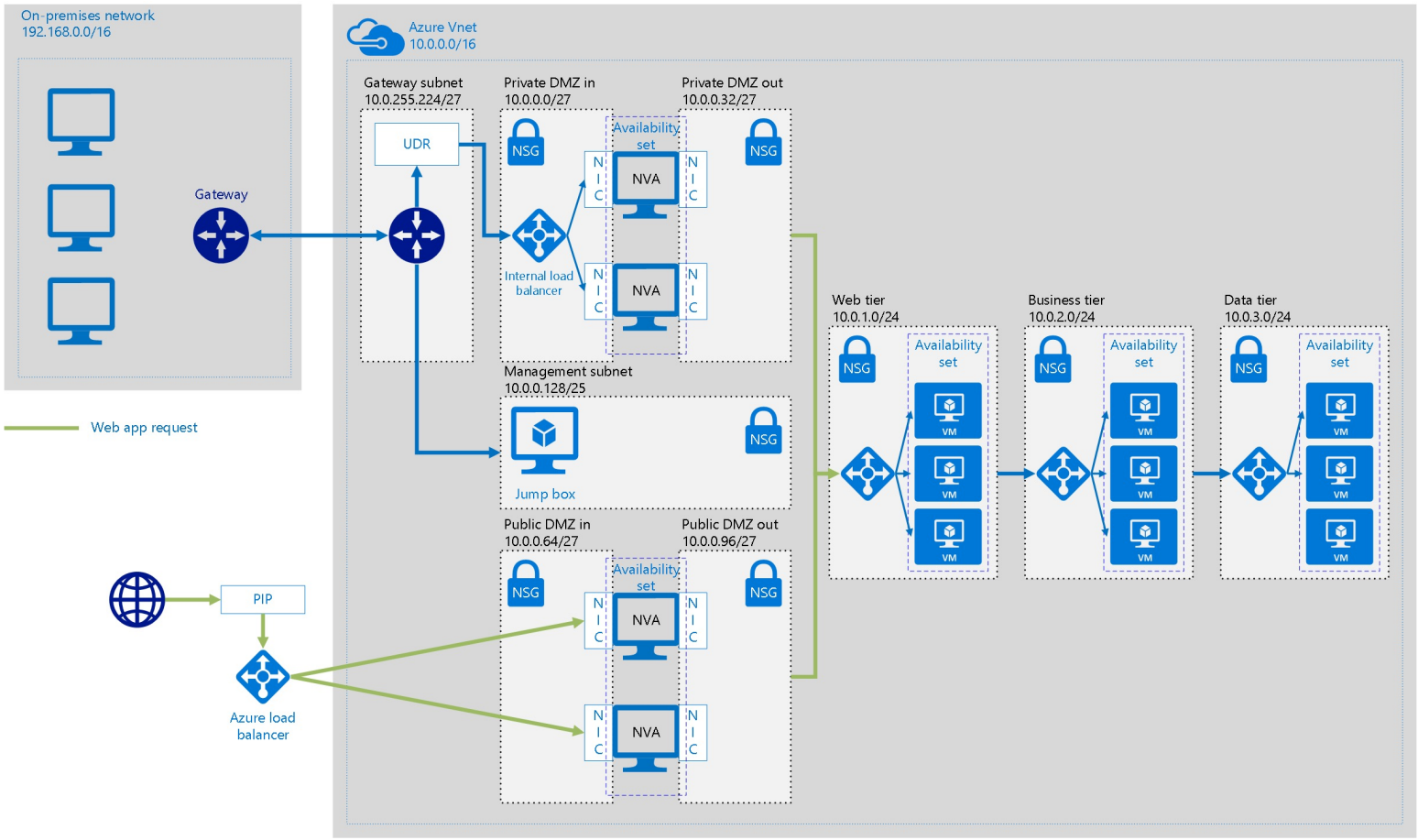


DMZ between Azure and the Internet

Implements a secure network that accepts Internet traffic to Azure.

https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/dmz/index

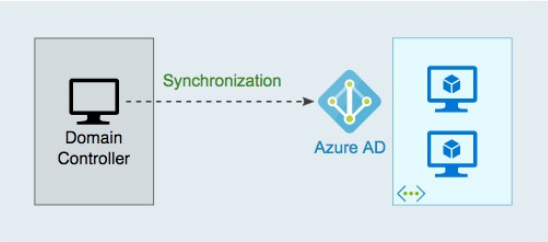
3. Arhitektura s DMZ - primjer



https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/dmz/secure-vnet-dmz

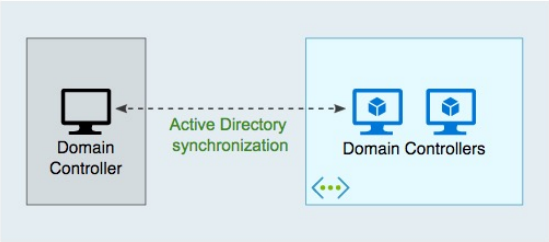
4. Arhitektura za upravljanje identitetima

- Čest problem: upravljanje identitetima, autentifikacija i autorizacija
- Što kada imamo jedno "tijelo" za upravljanje (npr. AD, *active directory*)?
- Što kada ih imamo više (*AD forest*)?
- Što kada ih trebamo više?



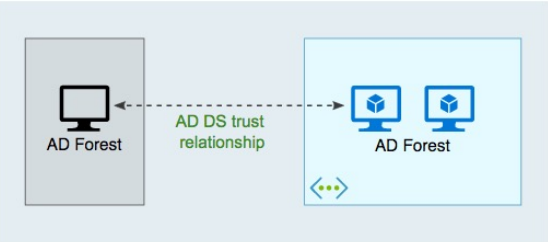
Integrate with Azure Active Directory (Azure AD)

Integrate on-premises Active Directory domains and forests with Azure AD.



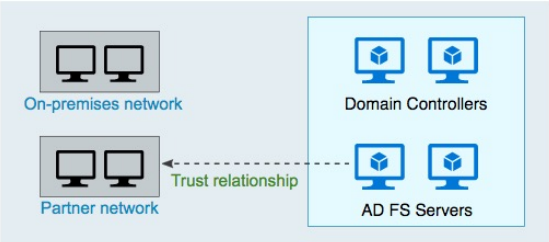
Extend Active Directory Domain Services (AD DS) to Azure

Extend your Active Directory environment to Azure to provide distributed authentication using AD DS.



Create an AD DS forest in Azure

Create a separate Active Directory domain in Azure that is trusted by domains in your on-premises forest.



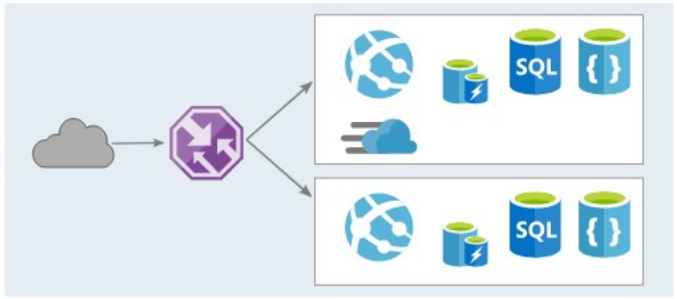
Extend Active Directory Federation Services (AD FS) to Azure

Use AD FS to perform federated authentication and authorization for components running in Azure.

https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/identity/index

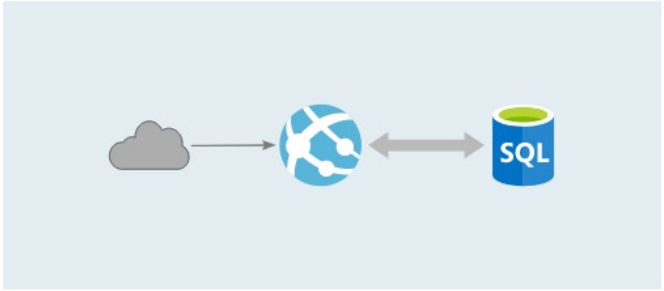
5. Arhitektura za web aplikacije

- Arhitektura osnovne web aplikacije
- Arhitektura web aplikacije s dodanom skalabilnošću
  - privremena pohrana (cache)
  - *Content Delivery Network CDN*
    - povećava dostupnost čestih (statičkih) resursa
  - web poslovi za pozadinske zadaće
- Arhitektura web aplikacije za više regija



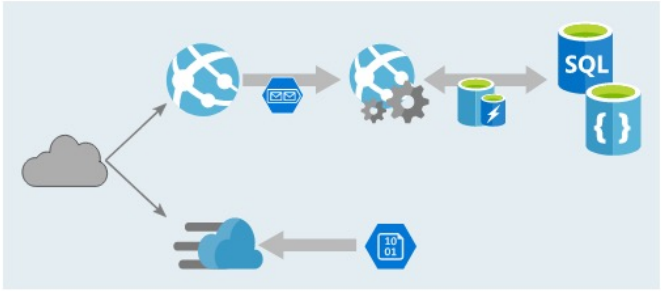
Run in multiple regions

Run a web application in multiple regions to achieve high availability.



Basic web application

A basic web application that uses Azure App Service and Azure SQL Database.

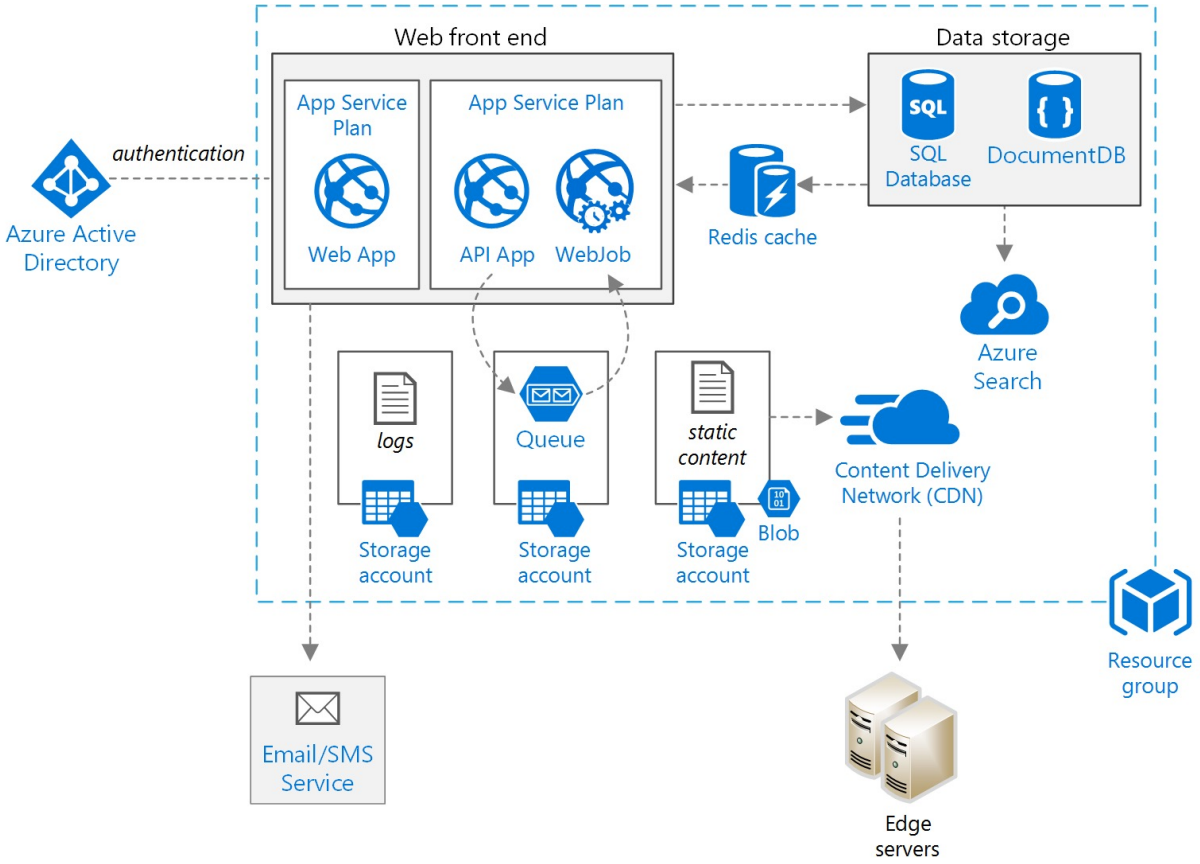


Improved scalability

Improve the scalability and performance of a web application by adding cache, CDN, and WebJobs for background tasks.

https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/managed-web-app/index

5. Arhitektura za web aplikacije – primjer skalabilnosti



https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/managed-web-app/scalable-web-app

Uzorci dizajna u oblaku

- Organizacija prema aspektima arhitektura i tehničkim/organizacijskim problemima koje je potrebno riješiti
  - Raspoloživost (*Availability*)
  - Upravljanje podacima (*Data Management*)
  - Dizajn i implementacija (*Design and Implementation*)
  - Poručivanje (*Messaging*)
  - Upravljanje i nadzor (*Management and Monitoring*)
  - Performanse i skalabilnost (*Performance and Scalability*)
  - Otpornost (*Resiliency*)
  - Sigurnost (*Security*)

Raspoloživost

- Mjera: vrijeme dostupnosti usluge (*uptime*) u postotcima
  - primjer Amazon WS – 99.95% vremena za svaki (mikro)servis – dovoljno?
  - Definirano *Service Level Agreement*-om (SLA)

Pattern	Summary
Health Endpoint Monitoring	Implement functional checks in an application that external tools can access through exposed endpoints at regular intervals.
Queue-Based Load Leveling	Use a queue that acts as a buffer between a task and a service that it invokes in order to smooth intermittent heavy loads.
Throttling	Control the consumption of resources used by an instance of an application, an individual tenant, or an entire service.

- *Throttling*
  - Degradacija usluge / prometa / propusnosti kako bi se osigurao rad i izbjegli prekidi
  - “poznato” iz telekomunikacijskih mreža

Upravljanje podacima

- Podaci su centralni element u većini sustava
  - Migracija, replikacija, konzistetnost, problemi privatnosti i legislative…

Pattern	Summary
Cache-Aside	Load data on demand into a cache from a data store
CQRS	Segregate operations that read data from operations that update data by using separate interfaces.
Event Sourcing	Use an append-only store to record the full series of events that describe actions taken on data in a domain.
Index Table	Create indexes over the fields in data stores that are frequently referenced by queries.
Materialized View	Generate prepopulated views over the data in one or more data stores when the data isn't ideally formatted for required query operations.
Sharding	Divide a data store into a set of horizontal partitions or shards.
Static Content Hosting	Deploy static content to a cloud-based storage service that can deliver them directly to the client.
Valet Key	Use a token or key that provides clients with restricted direct access to a specific resource or service.

Dizajn i implementacija

- Dobar dizajn je ključ dugoročne evolucije usluga
  - Jednostavnost, iskoristivost, konzistetnost komponenti…

Pattern	Summary
CQRS	Segregate operations that read data from operations that update data by using separate interfaces.
Compute Resource Consolidation	Consolidate multiple tasks or operations into a single computational unit
External Configuration Store	Move configuration information out of the application deployment package to a centralized location.
Leader Election	Coordinate the actions performed by a collection of collaborating task instances in a distributed application by electing one instance as the leader that assumes responsibility for managing the other instances.
Pipes and Filters	Break down a task that performs complex processing into a series of separate elements that can be reused.
Runtime Reconfiguration	Design an application so that it can be reconfigured without requiring redeployment or restarting the application.
Static Content Hosting	Deploy static content to a cloud-based storage service that can deliver them directly to the client.

Poručivanje

- Kako komponente i (mikro)servisi komuniciraju?
  - Asinkrono, slaba međuovisnost (*loosely coupled*), idempotentnost (!)…

Pattern	Summary
Competing Consumers	Enable multiple concurrent consumers to process messages received on the same messaging channel.
Pipes and Filters	Break down a task that performs complex processing into a series of separate elements that can be reused.
Priority Queue	Prioritize requests sent to services so that requests with a higher priority are received and processed more quickly than those with a lower priority.
Queue-Based Load Leveling	Use a queue that acts as a buffer between a task and a service that it invokes in order to smooth intermittent heavy loads.
Scheduler Agent Supervisor	Coordinate a set of actions across a distributed set of services and other remote resources.

Upravljanje i nadzor

- Pregled stanja sustava
- Mogućnost promjene konfiguracije bez zaustavljanja usluge

Pattern	Summary
External Configuration Store	Move configuration information out of the application deployment package to a centralized location.
Health Endpoint Monitoring	Implement functional checks in an application that external tools can access through exposed endpoints at regular intervals.
Runtime Reconfiguration	Design an application so that it can be reconfigured without requiring redeployment or restarting the application.

Performanse i skalabilnost

Pattern	Summary
Cache-Aside	Load data on demand into a cache from a data store
CQRS	Segregate operations that read data from operations that update data by using separate interfaces.
Event Sourcing	Use an append-only store to record the full series of events that describe actions taken on data in a domain.
Index Table	Create indexes over the fields in data stores that are frequently referenced by queries.
Materialized View	Generate prepopulated views over the data in one or more data stores when the data isn't ideally formatted for required query operations.
Priority Queue	Prioritize requests sent to services so that requests with a higher priority are received and processed more quickly than those with a lower priority.
Queue-Based Load Leveling	Use a queue that acts as a buffer between a task and a service that it invokes in order to smooth intermittent heavy loads.
Sharding	Divide a data store into a set of horizontal partitions or shards.
Static Content Hosting	Deploy static content to a cloud-based storage service that can deliver them directly to the client.
Throttling	Control the consumption of resources used by an instance of an application, an individual tenant, or an entire service.

Otpornost

- Oporavak od grešaka

Pattern	Summary
Circuit Breaker	Handle faults that might take a variable amount of time to fix when connecting to a remote service or resource.
Compensating Transaction	Undo the work performed by a series of steps, which together define an eventually consistent operation.
Health Endpoint Monitoring	Implement functional checks in an application that external tools can access through exposed endpoints at regular intervals.
Leader Election	Coordinate the actions performed by a collection of collaborating task instances in a distributed application by electing one instance as the leader that assumes responsibility for managing the other instances.
Queue-Based Load Leveling	Use a queue that acts as a buffer between a task and a service that it invokes in order to smooth intermittent heavy loads.
Retry	Enable an application to handle anticipated, temporary failures when it tries to connect to a service or network resource by transparently retrying an operation that's previously failed.
Scheduler Agent Supervisor	Coordinate a set of actions across a distributed set of services and other remote resources.

Sigurnost

- Otpornost na napade i zaštita podataka

Pattern	Summary
Federated Identity	Delegate authentication to an external identity provider.
Gatekeeper	Protect applications and services by using a dedicated host instance that acts as a broker between clients and the application or service, validates and sanitizes requests, and passes requests and data between them.
Valet Key	Use a token or key that provides clients with restricted direct access to a specific resource or service.

Literatura

- Azure – Cloud design patterns
  - https://docs.microsoft.com/en-us/azure/architecture/patterns/
- Azure – Reference Architectures
  - https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/
- AWS Architecture Center
  - https://aws.amazon.com/architecture/