Teknologi

Krav

I skal konstant overveje hvilken teknologisk kontekst jeres løsning, og de enkelte produkter I udarbejder, skal fungere i.

Viden – Den studerende har:

- udviklingsbaseret viden om praksis og centralt anvendt teori inden for design og realisering af distribuerede systemer
- forståelse for fundamentale netværksbegreber

Færdigheder - Den studerende kan:

- anvende centrale og i praksis udbredte applikationsprotokoller
- vurdere relevante teknologiske aspekter i udviklingen af distribuerede systemer

Kompetencer – Den studerende kan:

- i en struktureret sammenhæng tilegne sig ny viden, færdigheder indenfor distribuerede systemer

I skal fremstille

Listen over de emner/begreber fra faget Teknologi I anvender i jeres løsning skal opdateres.

Giv en kort beskrivelse af hvorledes jeres MVC løsning *kunne* implementeres på flere (adskilte) servere. I skal altså ikke (nødvendigvis) beskrive den aktuelle implementering, men fremstille en tænkt løsning, hvor forskellige elementer af jeres løsning er placeret (implementeret) på flere individuelle servere der er fysisk helt adskilte. Overvej hvilke fordele og hvilke ulemper det kan have ud fra et rent netværksperspektiv.

OSI model

- **Application** - Grænsefladen mellem applikationer og det underliggende netværk. Indeholder protokoller som tillader applikationer at kommunikerer med netværket - fx DNS, HTTP, FTP og SMTP.
- **Presentation** - Præsenterer data modtaget fra underliggende lag til genkendelige formater, som jpeg, mpeg, mp4 osv. Laget komprimerer data for at få en hurtigere responstid. Samtidig krypteres dataen for at øge sikkerhed, ved hjælp af fx SSL eller TLS.
- **Session** - Opretter og synkroniserer med sekvensdata sessionen mellem sender- og modtager-enhederne. Sørger for, at kommunikationsmetoden er den rette til situationen.
- **Transport** - Frames --> segments. Sørger for, med den rette transport protokol (TCP/UDP), at den forbindelse oprettes. Tjekker pålideligheden af forbindelsen via flowkontrol, opdeling af packets i segmenter og fejlkontrol.
- **Network** - Frames fra Data link laget omdannes til packets. Holder styr på IP-adresser, for at kunne sende packets rundt på forskellige netværker.
- **Data link** - Holder styr på MAC-adresser, for at sikre at beskeden (frames) bliver sendt det rette sted hen. Definerer metoder til at få adgang til det delte medium før der transmitteres data.

- **Physical** - Dækker over alle fysiske dele af nettet - både serveren, hvorpå vores domæne er, samt kundens eget internetsystem. Definerer standarderne for kodning af signaler, som skal sendes over nettet for at kommunikationen domænet og brugere fungerer.

TCP/IP model

- **Application** - Dækker over OSI-modellens application-, presentation- og session-lag.
- **Transport** - Samme som transport-laget i OSI-modellen.
- **Internet** - Samme som network-laget i OSI-modellen.
- **Network access** - Dækker over Data link-laget og det fysiske lag i OSI-modellen.

Når der skal sendes data over internettet, fx hvis en kunde tilføjer et produkt til kurven, går processen gennem OSI-modellens lag, hvor det første/inderste lag er det fysiske, og det sidste/yderste lag er applikationslaget. Hvert lag benytter kun det underliggende lags funktioner, og tilbyder kun funktioner til laget over (med undtagelse af første og sidste lag).

2. Iteration:

- **Port forwarding**
  - Port forwarding er en teknik, der bruges til at give eksterne enheder adgang til computertjenester på private netværk. Port forwarding giver din router muligheden for at videresende indgående forbindelser med et matchende portnummer samt ip-adresse til en intern enhed med den pågældende adresse.
- **Domain Name System/DNS**
  - Oversætter domænenavne som "www.google.com" til IP-adresser, fordi vi mennesker bedre kan forstå ord end numre, hvor computere kører udelukkende på numre.
- **Dynamic Domain Name System/DDNS**
  - Når IP-adresser ændres af en DHCP-server, skal DNS-serveren opdateres løbende, så packets kan sendes og modtages korrekt, selvom Hosts' IP-adresser ændres.
- **Dynamic Host Configuration Protocol/DHCP**
  - Tildeler IP-adresser til enheder som forbindes til et netværk. Typisk indbygget i routeren. Holder styr på hvilke IP-adresser er tilgængelige i det scope, som er valgt under opsætningen, og hvilke enheder der har reserveret hvilke adresser.
- **Ping**
  - Ping er et internet software værktøj som bruger din internetforbindelse til at sende nogle pakker med data til en bestemt adresse. Disse pakker sendes derefter tilbage til din computer. Testen registrerer den mængde tid det tog for pakkerne at nå adressen, og hvorvidt nogen pakker gik tabt i processen.
- **TCP explained**

    - Transmission control protocol (TCP) er en netværkskommunikationsprotokol designet til at sende datapakker over internettet. TCP er en befinder sig i transportlaget i OSI-modellen og bruges til at skabe en forbindelse mellem enheder ved at transportere og sikre levering af meddelelser via understøttende netværk og Internettet.
- **UDP explained**
    - En anden form for forbindelse end TCP. Data som sendes over en UDP-forbindelse fylder mindre, og det betyder ikke noget hvilken rækkefølge dataen bliver sendt i. Bruges typisk ved telefonopkald el. lign, da enkelte manglende packets er acceptabelt så der ikke er forsinkelse i forbindelsen.
- **TCP 3-way handshake**
    - En måde at sikre forbindelsen mellem to hosts, fx client og server.
    - 1. Client sender en SYN (synchronize) til server.
    - 2. Server sender SYN-ACK(acknowledgement) tilbage til client
    - 3. Client sender ACK tilbage til server
    - 4. Forbindelsen er oprettet
- **TCP flow control and window size**
    - For at sikre, at modtageren af en fil ikke oversvømmes, bliver al data sendt igennem en buffer hos modtageren og senderen. Når modtagerens applikationen er klar til at modtage data og bearbejde det, indhentes det fra bufferen i stedet for direkte fra senderen. Hver gang modtageren modtager en fil, sendes en ACK (acknowledgement) til senderen, for at sikre at alle filerne sendes korrekt.
- **An overview of HTTP**
    - HTTP/Hypertext Transfer Protocol er en protokol som befinder sig i applikationslaget i OSI-modellen. HTTP giver muligheden for at kommunikere data på Internettet. HTTP fungerer ved, at man sender en anmodninger til en webserver om at diverse web elementer såsom billeder og webelementer (HTML m.m), som en webbrowser så kan formidle til at danne en form for brugerflade for brugeren.
- **Ports**
    - en port består af et 16 bit nummer. Porte bliver brugt fx i routers, switch og TCP. En port bruges til at kommunikere mellem maskiner.
- **Introduction to Sockets**
    - når der er en end to end connection, er det via sockets de punkter er forbundet med i et to vejs forbindelse. TCP bruger det til at sikre at data bliver sendt til den rigtige maskine når socket bliver forbundet til porte.
- **Threads**
    - Threads bliver ofte brugt til at udføre små opgaver i programmering, man bruger ofte multithreading for at programming kan udføre mange processer, så man kan spare hukommelse og få concurrency i sit program.

3. Iteration: Brug af flere servere

Der kan være flere forskellige grunde til, at man kunne overveje at implementere sin løsning.

Én grund, kunne være, at forskellige dele af ens løsning har forskellige kvalitetskriterier (tænk FURPS).

Nogle dele af ens system være meget afhængigt af behandlingstid (Performance), imens andre dele prioriterer helt anderledes.

Et eksempel på et system, hvor forskellige dele har forskellige kvalitetskriterier, kunne være online multiplayer computerspil. De dele af systemet, som håndterer selve spillet er formentligt være meget afhængig af Performance, hvorimod login/authentication-systemet er mere afhængigt af sikkerhed.

Her kunne det give mening at splitte funktionaliteten ud på forskellige servere. Den funktionalitet, som er afhængig af performance, kan lægges på flere lokale servere der er tættere på slutbrugeren, eller servere med noget bestemt hardware, for at imødekomme specifikke krav. Andre dele af systemet, som ikke kræver high-end hardware eller lav forsinkelse, kan potentielt placeres på én central server.

Det er meget muligt, at dele af vores eget system på sigt vil få forskellige krav, og man kunne overveje opdeling af systemet på flere servere, men det virker ikke relevant lige nu.

Andre grunde til at man kunne elementer af et system op kunne være sikkerhedsmæssige eller udviklingsmæssige årsager. Man sørger for at det fx kun er et team som arbejdede på model og controller, og et separat team som arbejder på views delen af koden, og tredje team som sætter det sammen. Det ville være en ulempe for vores projekt da det ikke er større end det er, men for teams som arbejder på et større projekt kunne det være oplagt for at danne mere overskuelighed for programmeringen.
Man kunne også vælge at dele programmet op, af administrative grunde. Som fx når programmet er deployed, og virksomheden ønskede at det kun var visse afdelinger som skulle have mulighed for at redigere i specifikke dele. Det kunne være en fordel for bedre sporbarhed i koden og hvem der har lavet ændringerne.
Det kunne også være fordi ens database er fyldt og derfor vil gemme dataene fremadrettede et andet sted, derfor vil det kun være nødvendigt at splitte noget af koden op.

Hvis vi generelt skal sige noget om fordele og ulemper ved opdeling, ud fra et rent netværksperspektiv, går kunne vi forestille os:
Ulemper

- Mere trafik, pga. kommunikation imellem serverene
- (Måske) mere komplekst

Fordele

- Forskellige krav til forskellige dele af system kan (måske) nemmere imødekommes

- Man kan have servere på flere fysiske lokationer, så brugere fra forskellige steder kan have en fornuftig forbindelse
- Load balancing. Måske smart med virtuelle servere?
- Server uptime - hvis én server er nede (evt. pga. angreb), kan en anden server (mirror) tage over.