### **Programmering**

Implementér optimistisk offline lock, og skriv et stykke middleware, der giver mening for jeres applikation.

1) Optimistisk offline lock er blevet implementeret

2) CookieMiddleware er blevet tilføjet i form af noget GDPR. Når hjemmesiden åbnes skal man give samtykke til, at hjemmesiden indeholder cookies og der kan gemmes oplysninger som email og kode.

### **Systemudvikling**

1)

**Beskriv de aktiviteter I har gennemført i HLD og LLD. Hvordan passer de med de teknikker I beskrev i systemudviklingsmetoden?**

Aktiviteter i HLD:

Prototyping Teoretisk:

* Vi udformer en objektmodel
* Vi udformer en domænemodel
* Vi udformer brugssituationer
* Vi udformer SSD’er til tilhørende (brugssituationer / operationer)
* Vi udformer de tilhørende OC’er
* Vi udformer Tabel Designs til databasen

Aktiviteter i LLD:

Prototyping Teoretisk:

* Vi udformer en DCD
* Vi udformer en SD

**Hvordan understøtter de kvaliteten af arkitektur og integration?**

Som det ser ud nu svarer de aktiviteter vi laver i både HLD og LLD til de teknikker der anvendes (beskrevet i sprint 3). Disse teknikker medføre sporbarhed og en rød tråd gennem udviklingsfasen, da en teknik fører til en anden. Derigennem afdækkes en masse ideer under udviklingsfasen, om hvorvidt de er hensigtsmæssig og mest optimal. Dette giver samtidigt teamet en ide om projektets arkitektur, samt hvordan den skal integreres iterativt.

**Hvordan kan I benytte dem i fokusområderne programmering og test?**

De artefakter vi har lavet under HLD og LLD ligger til grund for designet af selve hjemmesiden. Derfor kan du med fokus på fokusområdet ‘programmering’ tage alt indhold fra vores artefakter og programmere det. Det gør det nemmere for programmørerne at have et “fælles sprog” som bliver afspejlet i artefakterne. Det sikrer sporbarhed i koden at følge f.eks. samme navngivning igennem alle artefakterne. Det samme kan siges om fokusområdet ‘test’, da man ikke vil bruge alverdens tid på at rode kode igennem. Har man automatiske tests der kører jævnligt for at sikre at koden virker, er det meget vigtigt at det er samme navngivning eller datatyper i metoder mm. for at det ikke giver et forkert udslag. Så spilder man ikke tid på at prøve at rette op på en fejl, hvor logikken i virkeligheden virker!

### **Teknologi**

### **Sprint 1:**

TOPOLOGI:

* Mesh Topologi (Multipoint connection)

NETVÆRK:

* WAN
* Domæne (rettigheder)
* Server (Webhotel/ hosting)
* Database

### **Sprint 2:**

**Vi forestiller os følgende scenarie efter ARP-protokollen er taget i brug:**

* Vi har en server Host (vores hjemmeside, simply.com)
  + Enhed: Server (simply) ?
  + Server host’en vil have en default gateway (router), som server host’en kender vha. af et ARP table, hvori routerens IP-adresse og MAC-adresse er kendt.
  + Server host’en vil være koblet til en switch i sit lokale netværk (LAN),
    - Switchen vil have en tilhørende “MAC Adresse Table”, med server host’ens MAC-adresse og switch-port den er tilkoblet. Derudover vil tabellen indeholde Routerens MAC-adresse og switch-port (OSI L2).
  + Switchen er koblet til en router, for at kommunikere til clients hostnes netværk (IP-adresse).
    - Routeren vil have en tilhørende “Router ARP Table”, med server host’ens IP-adresse og tilhørende MAC-adresse (OSI L3).
    - Routeren vil derudover have en tilhørende “Routing Table”, med server host’ens IP-adresse(+ submask), samt dens tilhørende Router interface og method, som vi antager er Directly Connect. Samtidigt vil tabellen indeholde client host’nes IP-adresse(+ submask), samt dens tilhørende Router interface og method, som vi antager er Dynamic Routing (OSI L3).
* Vi har en til mange client Hosts (brugerne af hjemmesiden)
  + Enhed: Stationær computer, bærbar computer
  + En client host vil have en default gateway (router), ligesom server host’en
  + En client host vil være koblet til en switch i sit lokale netværk (LAN), ligesom det er beskrevet ovenover i forbindelsen med server host’en.
* Switchen er koblet til en router, for at kommunikere til server host’ens netværk (IP-adresse).
  + Routeren vil have en tilhørende “Router ARP Table”, med client host’ens IP-adresse og tilhørende MAC-adresse.
  + Routeren vil derudover have en tilhørende “Routing Table”, med client host’ens IP-adresse(+ submask), samt dens tilhørende Router interface og method, som vi antager er Directly Connect. Samtidigt vil tabellen indeholde server host’ens IP-adresse(+ submask), samt dens tilhørende Router interface og method, som vi antager er Dynamic Routing.

**Hvad sker der når clienten skriver vores url i deres browser:**

1. Først skriver clienten url’en for at connecte til serveren, hjemmesiden er hosted på.
2. Browseren leder efter ip-adressen af domænet i DNS(domain name server).
3. browseren starter en tcp connection med serveren.
4. browseren sender en http request til serveren.
5. serveren tager sig af http requesten og sender en http response tilbage til clienten.
6. browseren viser nu html indholdet på siden.
7. clienten er beskyttet af protokollen https. Det er en sikkerhedsprotokol der beskytter på 3 primære områder. Disse områder er kryptering, godkendelse og dataintegritet.

### Sprint 4:

**Sikkerhedsmæssige Foranstaltninger:**

* Authorization
  + Role based authorization
  + Login authorization
* Authentication
  + Identity
  + Email authentication
* Data Annotationer
  + [REQURIED], [MaxLenght] mm. Sikre at brugernes input er i rigtigt/ forventet format. Bidrager samtidigt til brugervenlighed.
* Optimistic Offline lock
  + Beskytter sårbare dele af koden (critical section), imod at f.eks to brugere (med hver sin tråd/ thread) tilgår samme data (shared data), og ændre dataen samtidigt, hvilket kan føre til Race conditions.
* Overposting
  + Stopper objekter at blive produceret gennem fx. POSTMAN
* Cookie Policy (GDPR)
  + Sikre at brugeren giver samtykke til at hjemmesiden gemmer og bruger brugerens data/ informationer. GDPR: Beskytter personfølsom data ud fra europæisk lovgivning (EU).