# Fitness Junkies I/S



Lærer: Henrik Thomsen

Elever: Tobias Brammer, Steffen Thomsen og Jonas Grendslev

## Titelblad

Uddannelse: Datatekniker

Hold: HOT8\_DT6\_Programmering

HOT8\_DT6\_Infrastruktur

Emne: Motionsfællesskab

Elever: Tobias Brammer

Jonas Granslev

Steffen Thomsen

Eksamensperiode 07-01-2019 til 08-02-2019

Dokument afleveret 01-02-2019

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tobias Brammer

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Jonas Granslev Steffen Thomsen

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Henrik Thomsen

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Censor Censor

## Index

[Fitness Junkies I/S 1](#_Toc536779651)

[Titelblad 2](#_Toc536779652)

[Index 3](#_Toc536779653)

[Læsevejledning: 4](#_Toc536779654)

[Kravspecifikation 5](#_Toc536779655)

[Produkt Idé: 5](#_Toc536779656)

[Krav til produkt/Begrænsninger: 5](#_Toc536779657)

[Resident Testkonditioner: 6](#_Toc536779658)

[Brugervejledning - Backend 7](#_Toc536779659)

[Overblik 7](#_Toc536779660)

[Anvendelse 9](#_Toc536779661)

[Teknisk produktdokumentation- Backend 9](#_Toc536779662)

[Testrapport 9](#_Toc536779663)

[Brugervejledning - Frontend 10](#_Toc536779664)

[Overblik 10](#_Toc536779665)

[Brugervejledning til Netværk og Server: 12](#_Toc536779666)

[Topologi(Topologi vedhæftet i bilag, så man kan zoom): 12](#_Toc536779667)

[IP Plan: 13](#_Toc536779668)

[Password politik: 14](#_Toc536779669)

[WIFI: 14](#_Toc536779670)

[SSH: 14](#_Toc536779671)

[Netværk tests: 14](#_Toc536779672)

[Failover(HSRP): 14](#_Toc536779673)

[DMZ: 16](#_Toc536779674)

[Server tests: 18](#_Toc536779675)

[Oprettelse af ny VM: 18](#_Toc536779676)

[Failover Cluster: 18](#_Toc536779677)

[Backup: 19](#_Toc536779678)

[File Share: 19](#_Toc536779679)

[Server monitoring 19](#_Toc536779680)

## Læsevejledning:

Til det tekniske personale der skal afprøve denne proof of concept løsning

## 

## Kravspecifikation

### Produkt Idé:

Ud fra spørgsmålene i problemformulering har vi valgt at løse problemet på følgende måde. Der laves et Small Buisness virksomheds netværk samt server miljø som tilegner sig efter produktets behov.

Derudover laves en backend til fællesskabs platformen, som laves som Web API. Backend skal derudover også håndtere brugere, aktiviteter, matching og sessioner. App-delen skal laves til IOS.

### Krav til produkt/Begrænsninger:

* Brugeren skal kunne oprette en bruger.
* Brugeren skal kunne logge ind på sin bruger.
* Brugeren skal kunne logge ind med Facebook.
* Systemet skal kunne huske brugeren.
* Brugeren skal kunne ændre password på sin konto.
* Brugeren skal kunne se og ændre i sin egen information.
* Brugeren skal kunne sætte et profilbillede.
* Brugeren skal kunne sætte en primær sportsgren.
* Brugeren skal kunne sætte brugt tid per uge.
* Brugeren skal kunne angive sin position.
* Brugeren skal kunne finde en mulig træningspartner ud fra sportsgren, tid og position.
* Brugeren skal kunne udfører en aktivitet, f.eks. løb.
* Brugeren skal kunne udfører en aktivitet sammen med andre.
* Brugeren skal kunne se de udførte aktiviteter.
* Brugeren skal kunne oprette en træningsplan.
* Brugeren skal kunne dele en træningsplan med andre.
* Man skal kunne logge på et virksomheds domæne
* Domænet skal være et “Small Buisness” domæne til 30 mennesker
* Der skal være et servermiljø til at understøtte appens funktionalitet
* Der skal være et netværk til at understøtte appens funktionalitet
* Der skal være høj tilgængelighed så serverne er tilgængelige 99,9% af tiden
* Der skal være mulighed for at kunne skalere løsningen når der kommer flere ansatte i virksomheden
* Der skal være sikkerhed efter cisco best practice
* Der skal være mulighed for et storage så de ansatte har et sted de kan gemme deres filer, og et sted, hvor appens backend kan være
* Der skal være mulighed for backup så filer ikke går tabt, hvis der skulle ske en disaster
* Der skal være overvågning af serverne og netværket så man kan holde øje med, hvor meget belastning der er på det

## Resident Testkonditioner:

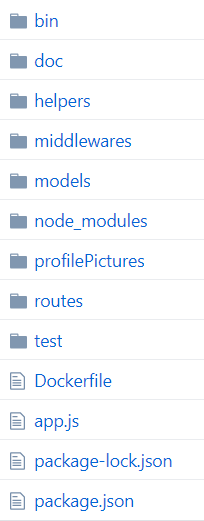
|  |  |
| --- | --- |
| **Krav:** | **Udført?** |
| Kan medarbejderne tilgå netværket? |  |
| Kan medarbejderne logge på domænet? |  |
| Har medarbejderne mulighed for at arbejde 99,9% af tiden? |  |
| Har medarbejderne fuld sikkerhed i virksomheden |  |
| Har medarbejderne tryg data? |  |
| Har medarbejderne mulighed for at gemme data? |  |
| Er det muligt at overvåge server og netværk? |  |
| Er det muligt at skalere server og netværk så det bliver større? |  |
| Kan brugerne oprette en bruger? |  |
| Kan brugerne logge ind, også med Facebook? |  |
| Bliver brugeren husket af systemet? |  |
| Bliver der fundet matches til brugerne? |  |
| Har brugerne mulighed for at skabe en relation? |  |
| Har brugerne mulighed for at starte og udfører en aktivitet? |  |
| Kan brugerne se udførte aktiviteter? |  |
| Kan brugerne oprette en træningsplan? |  |
| Kan træningsplanen deles mellem brugere? |  |
| Kan brugerne tilføje andre brugere til en aktivitet? |  |

## Brugervejledning - Backend

### Overblik

For at kører denne proof of concept løsning skal sourcekoden hentes. Dette gøres ved at klone fra <https://github.com/steffen2811/Fitness_APP>. Derefter kan sourcekoden til backend finde under src/API/Fitness\_APP/.

Her ligger en package.json file, som beskriver hvilke dependencies APIét har.

Derudover kan det ses at APIét starter fra /bin/www, som sørger for at APIét initialiseres samt åbner porten, så APIét kan kaldes.

Derudover det kort beskrives som:

API dokumentation: /Doc/index.html

MySql forbindelse: /Helpers/mysql.js

Middlewares til authentication, checks og file upload: /Middlewares

Hentet dependencies: /node\_modules

Profilbillede storage: /profilePictures

Router der bruges: /Routes

Import fil til postman (API test): /test

#### Docker

Når APIét skal op og kører skal dette ske via Docker.

Docker kan kører på et væld af miljøer, heriblandt Linux, Windows og Mac. I proof of concept løsningen er en Ubuntu server valgt som webserver.

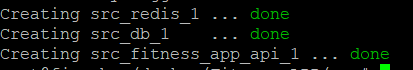
Herpå installeres Docker og git, hvorefter APIét kan kører op.

Dette sker ved at oprette en holder i roden kaldet docker. Herefter klones <https://github.com/steffen2811/Fitness_APP> indi forholderen.

Derefter åbnes /docker/Fitness\_APP/src, hvor docker-compose filen ligger.

Her køres kommandoen “docker-compose up -d --build”

Denne kommando sørger for at MySql, Redis og APIét køres op og er klar til brug.



#### Verificere installation

Opsætningen kan testen ved at kører integration test via programmet postman.

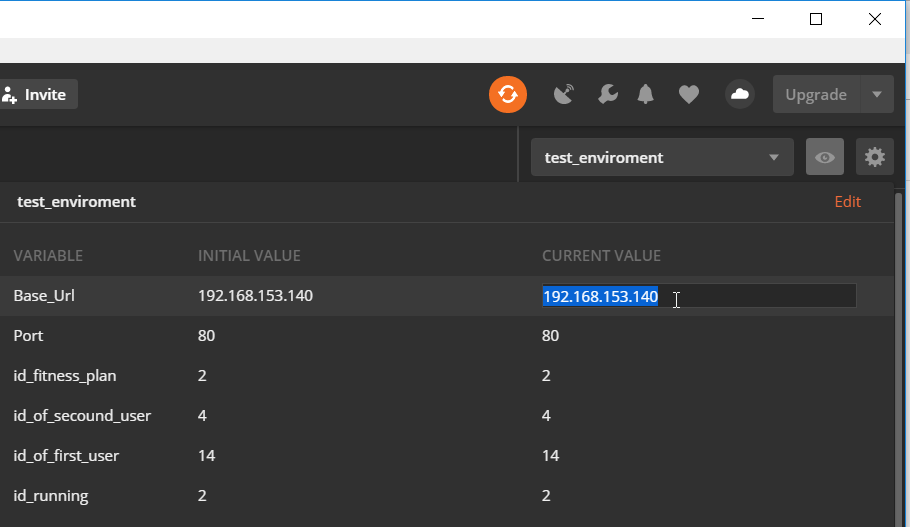
Dette gøres udefra, altså på den anden side af ASA firewall.

Postman installeres og filerne ***Simple\_test\_API.postman\_collection.json*** og ***test\_enviroment.postman\_environment.json*** importeres fra mappen ***Fitness\_APP\src\API\Fitness\_APP\test***.

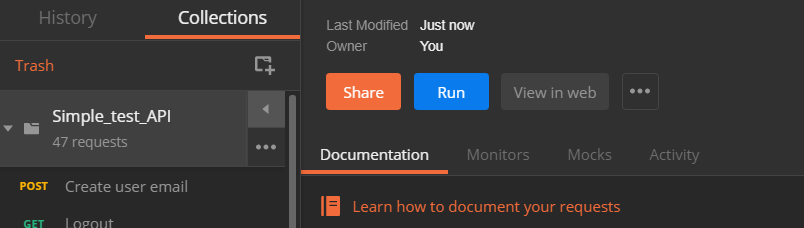
Herefter vil en collection og et environment være tilgængelig i Postman.

Først skal IP adressen ændres så den passer til det miljø APIét er kørt op i.

Dette gøres ved at vælge “test\_enviroment”, hvorefter der skal klikken på øjet. Nu kan current value for “Base\_Url” ændres, som skal sættes til den IP som APIét tilgås fra.

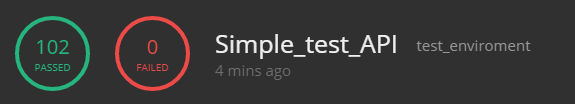


Herefter kan Simple\_test\_API kører. Dette gøres ved at gå til collections, derefter den lille pil, og run.



Derefter åbnes et nyt vindue hvor man kan kører testen. Her er det vigtigt at enviroment sættes til test\_enviroment, hvorefter testen kan køres.

Når testen er færdig skal resultatet bliver 102 passed. Nu er installationen af APIét valideret.



### Anvendelse

Al brug af APIét er beskrevet på en autogenereret webside.

#### API dokumentation

Dokumentationen kan findes i /Fitness\_APP/src/API/Fitness\_APP/doc/index.html

Derudover er en PDF fil også vedlagt i bilag som indeholder alt autogenereret dokumentation. Dokumentet hedder **FitnessAPP backend API.pdf**

## Teknisk produktdokumentation- Backend

### Testrapport

Testrapporten kan vises igennem Postman, ved at åbne filen ***Fitness\_APP\src\API\Fitness\_APP\test\Simple\_test\_API.postman\_test\_run.json***

Her kan hele integration testen ses.

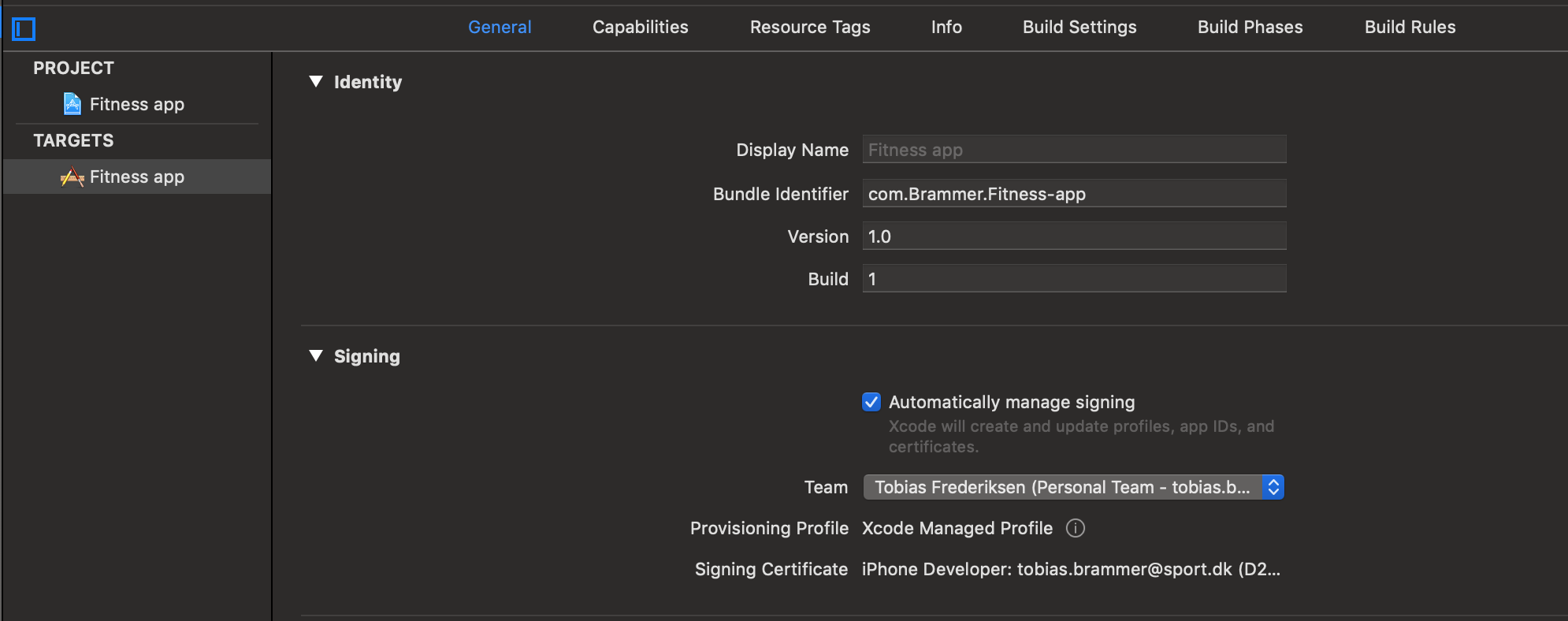
## Brugervejledning - Frontend

### Overblik

For at kører denne prove of concept løsning skal sourcekoden hentes. Dette gøres ved at klone fra <https://github.com/steffen2811/Fitness_APP>. Derefter kan sourcekoden til frontend findes under src/APP\_IOS/Fitness app.

for at køre appen, skal man åbne Fitness app.xcworkspace, for at få alle pods med og dermed alle funktionaliteter.

Derefter kan app’en bygges til en simulator eller en fysisk telefon. For at bygge appen på en fysisk telefon skal man have signet projektet først, dette gøres sådan.



under signing skal man check Automatically manage signing også skal man vælge sig selv under Team. Derefter kan man vælge sin egen telefon under target også bliver appen bygget på mobilen.

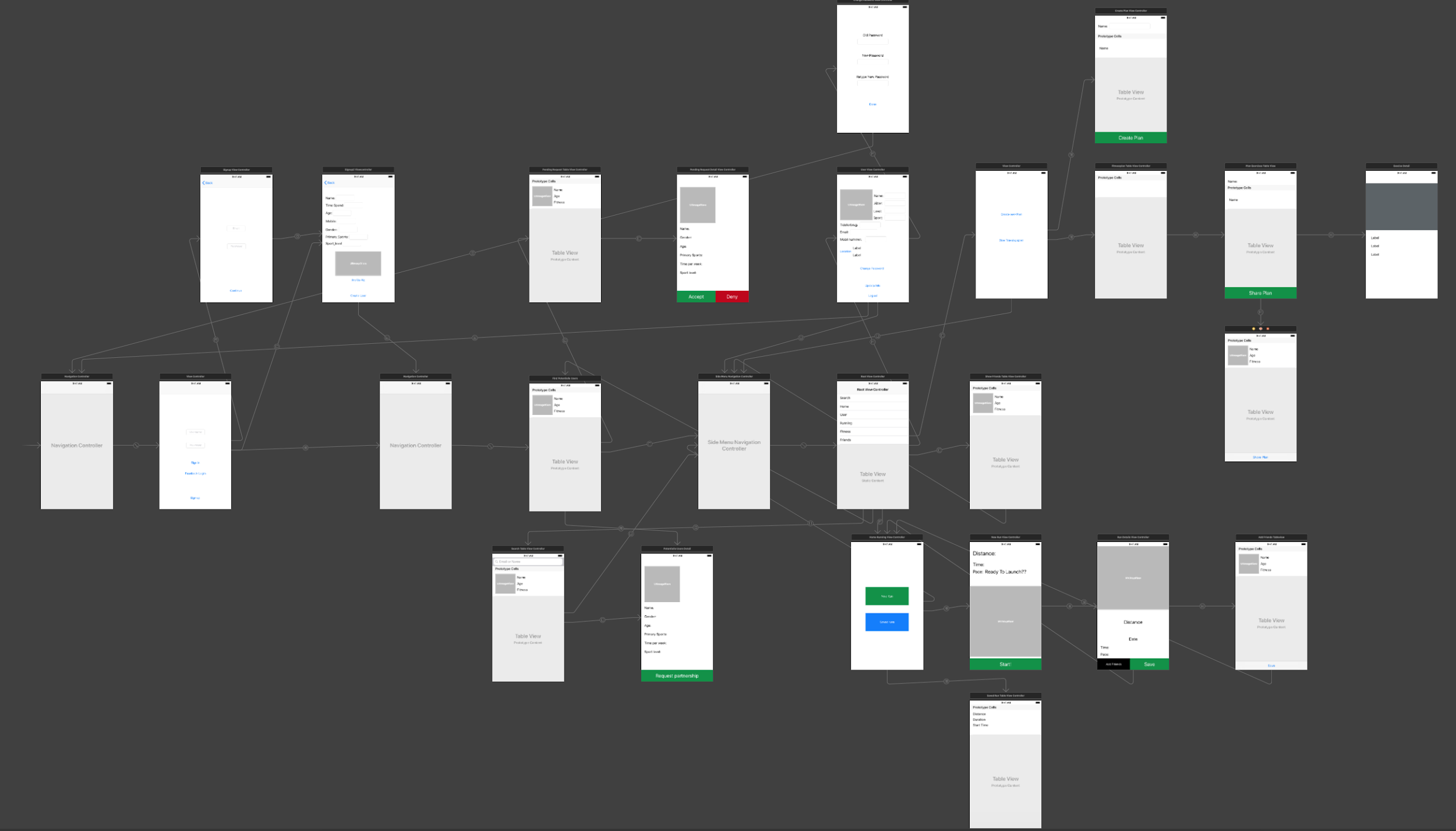
For at få appen på app store kan man følge denne step by step guide lavet af apple.

<https://help.apple.com/app-store-connect/>

hvis der ønskes at bruge et lokalt miljø af vores skal url’en ændres til localhost:3333

få at få et overblik over app og alle dens view, kan man åbne Main.storyboard i projektet.

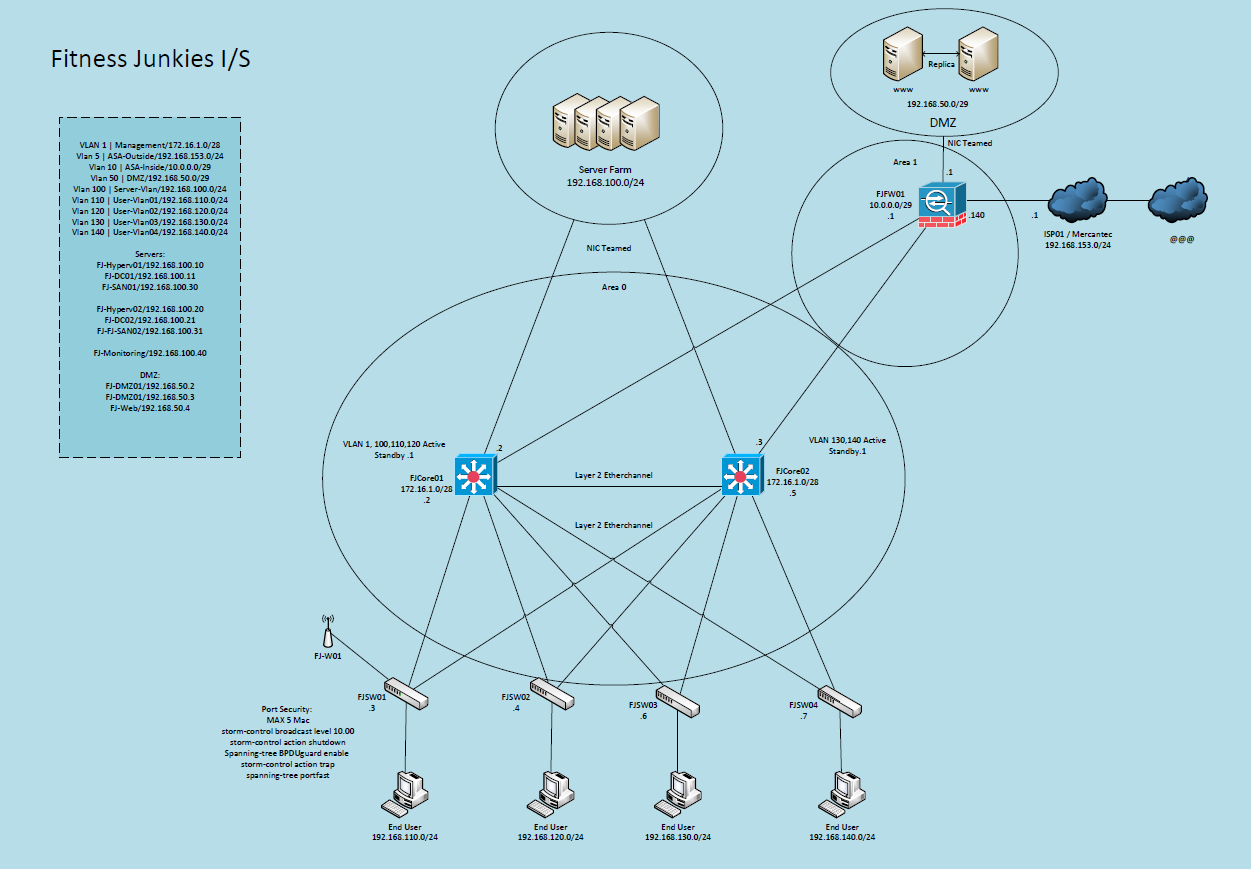
eller på billedet under (zoom in for at se detaljer)



# Brugervejledning til Netværk og Server:

For nemmere at kunne forstå og bruge netværket og server miljøet vil vi gerne lave nogle tests som viser, hvordan det virker.

### Topologi(Topologi vedhæftet i bilag, så man kan zoom):



## IP Plan:

Vlan 1

Mangagement /172.16.1.0/28

FJ-Core01 .2

FJ-SW01 .3

FJ-SW02 .4

FJ-Core02 .5

FJ-SW03 .6

FJ-SW04 .7

Vlan 5/Outside /192.168.153.0/24

FJ-FW01 .140

Vlan 10/Inside /10.0.0.0/29

FJ-FW01 .1  
FJ-Core01 .2  
FJ-Core02 .3

Vlan 50/DMZ /192.168.50.0/29

FJ-DMZ01 .1

FJ-DMZ02 .2

FJ-Web .3

Vlan 100/Server-Vlan /192.168.100.0/24

FJ-Hyperv01 .10

FJ-DC01 .11

FJ-SAN01 .30

FJ-Hyperv02 .20

FJ-DC02 .21

FJ-FJ-SAN02 .31

FJ-Cluster .15

FJ-Hyper-v Broker .16

FJ-DFS Manager .17

FJ-Monitoring .40

Vlan 110/User-Vlan02 /192.168.110.0/24

FJSW01 Access DHCP.100-200

FJ-W01 .10

Vlan 120/User-Vlan03 /192.168.120.0/24

FJSW02 Access DHCP.100-200

Vlan 130/User-Vlan04 /192.168.130.0/24

FJSW03 Access DHCP.100-200

Vlan 140/User-Vlan01 /192.168.140.0/24

FJSW04 Access DHCP.100-200

### Password politik:

Alle usernames er administrator.

I proof of concept løsningen er alle passwords “Password1”.

Inden løsningen kommer i et produktionsmiljø er det vigtigt i ændre passwords på udstyret. Vi anbefaler i i ændre passwords hver 3 måned og at der er et stor bogstav og et tal i det.

### WIFI:

SSID: FJ

Password: Password1

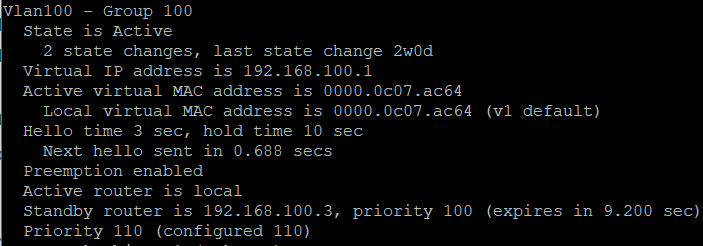
### SSH:

Der er oprettet SSH på alt netværks udstyr, hvilket vil sige i kan remote control det med software som putty.

## Netværk tests:

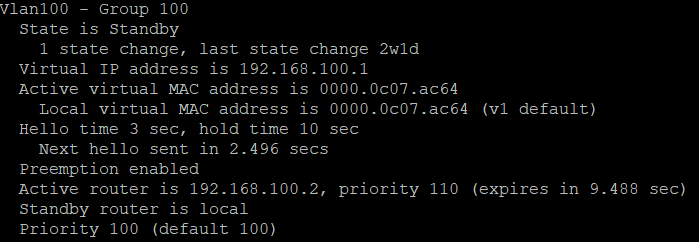
### Failover(HSRP):

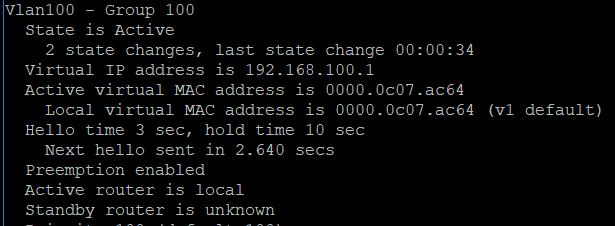
For at vise at netværket kører videre efter et nedbrud på en distributions switch vil vi gerne demonstrere, hvordan det virker.



På billedet ser FJ-Core01. Her kan vi se den er active for Vlan100.

Vi kan også se FJ-Core02 er standby for Vlan100

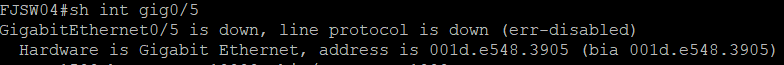


Skulle FJ-Core01 ske og slukke eller Vlan100 gå ned, ville FJ-Core02 tage over som vist. Dette sker helt automatisk da vi har konfigureret HSRP.

Her ser vi FJ-Core02 som har taget over som active router og standby er unknown da den er slukket. Når FJ-Core01 så tænder igen ville den blive active igen.

Port Security Storm Control:

For at demonstrere, hvad der sker hvis der opstår en kortslutning/loop i netværket har vi sat et ekstern switch til vores access lag som ikke deltager i spanning-tree, hvor vi efter laver en kortslutning og ser at vores netværk laver en shutdown på porten. Vi har switchen til gigabit0/5.



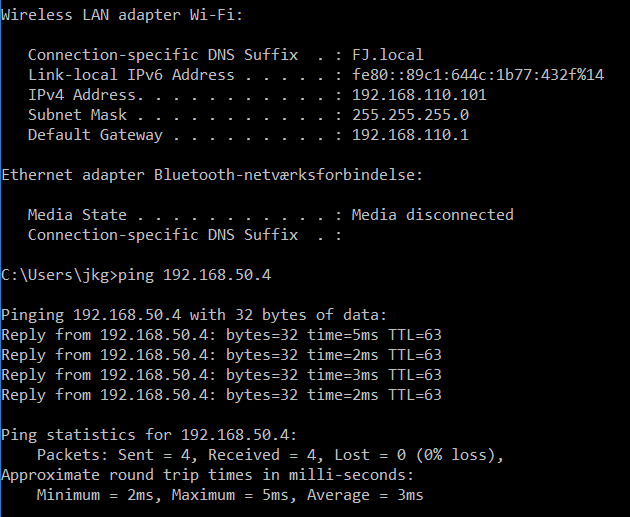
Her kan vi se den går i error disabled state og når loopet er fjernet vil den komme op igen.



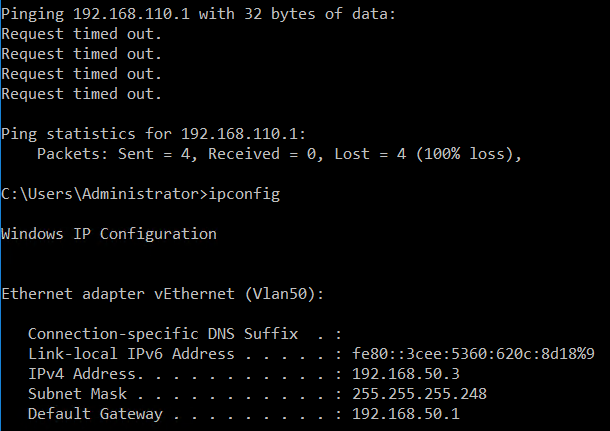
### DMZ:

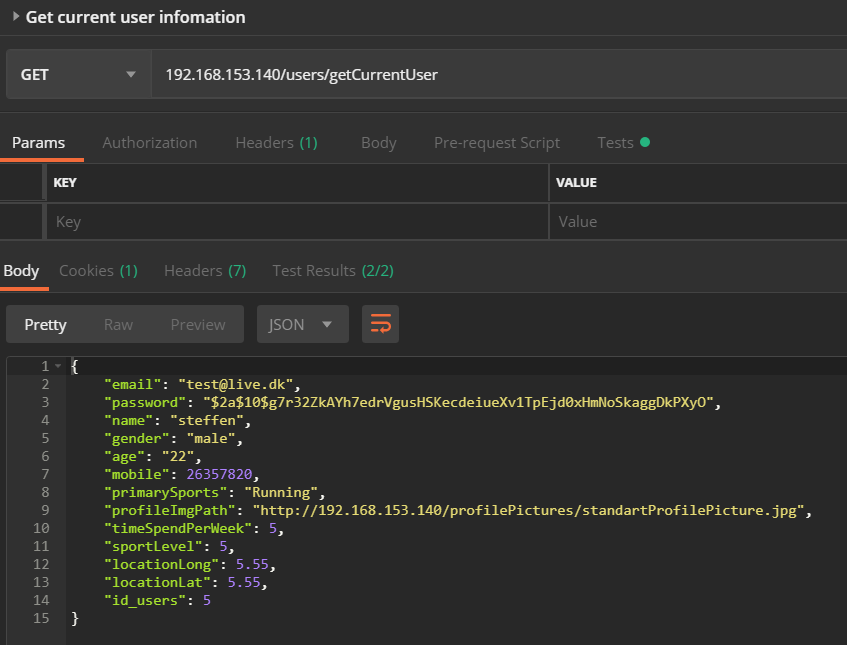
For at DMZ zonen virker skal man ikke kunne komme fra DMZ zonen til LAN, men man må godt komme fra LAN til DMZ zonen. Dette kan vi teste med en ping.

Her laver vi en ping fra vores interne netværk på Vlan 110.



Vi kan derefter gå ind på vores FJ-DMZ01 server og prøve at pinge VLAN110 som er på det interne netværk.

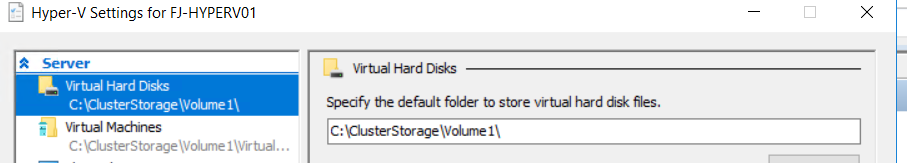


For at teste og se at brugerne med appen kan tilgå webserveren kan vi teste med at sende HTTP request i postman. Her kan vi se vi har fået reply fra Web-serveren.

## Server tests:

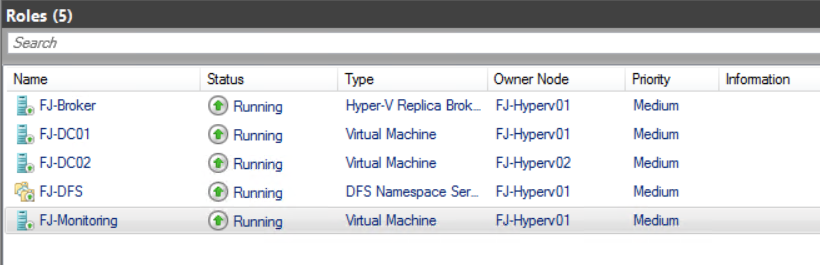
### Oprettelse af ny VM:

Når vi skal oprette en ny VM er det vigtigt den bliver gemt på clusteret, ellers vil der ikke være høj tilgængelighed. Som default har vi angivet, at det bliver gemt på clusteret, men bliver volumet nu fyldt så kan vi ændre stigen til det andet cluster volume med browse funktionen i hyper-v settings.

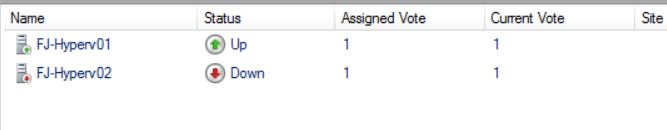


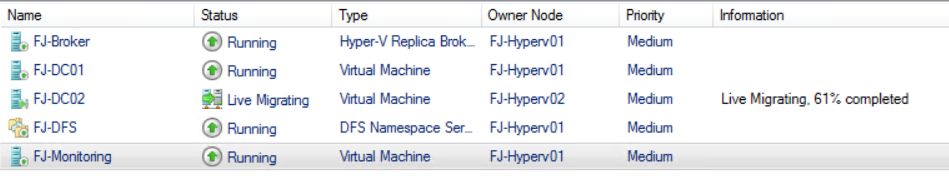
### Failover Cluster:

For at sikre os høj tilgængelig har vi vores failover cluster. Skulle en af hyper-v serverne ske at faile, så tager den anden server over og hoster alle dens VM. På billedet ser vi vores cluster settings og hvilke noder der hoster, hvilke funktioner.

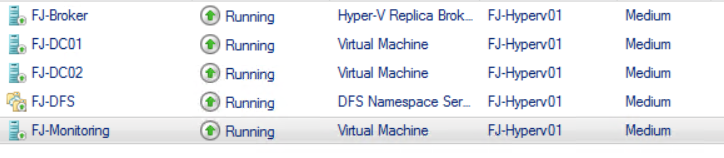


Som vi kan se hoster FJ-Hyperv02 DC02. Hvis vi slukker for FJ-Hyperv02 ville den live migrate DC serveren over på FJ-Hyperv01 noden. Dette kan vi teste ved at slukke for FJ-Hyperv02.



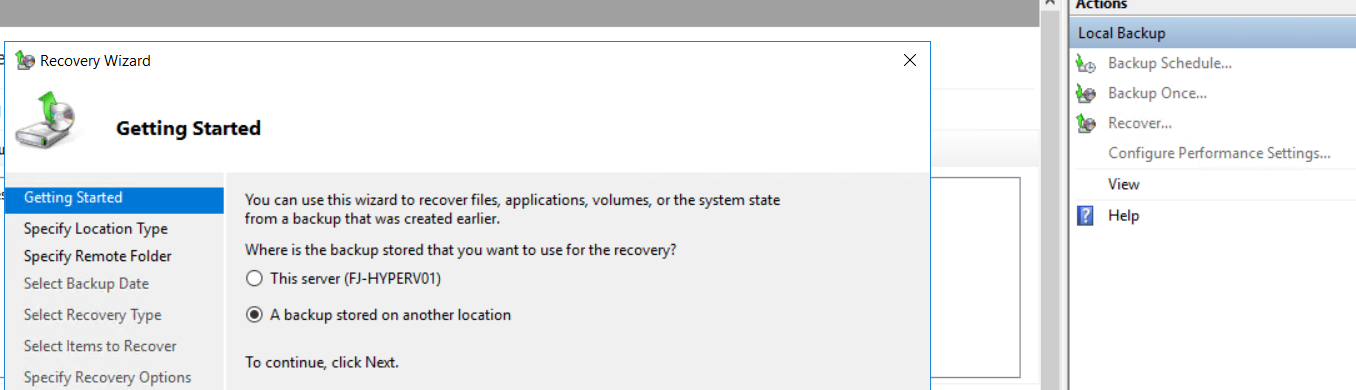


Her kan vi se den er igang med live migration. Når den er færdig med at migrate kommer den over på FJ-Hyperv-01



### Backup:

Som beskrevet i proces rapporten har vi kørt et backup schedule op der foretager backup 1 gang om ugen med full backup og sletter backuppen når den er mere end 3 måneder gammel. Hvis en disaster skulle ske er det eneste vigtige at få recovered backuppen. Dette kan man gøre via Backup manager, hvor man vælger recover… og a backup stored on another location. Derefter peger man over på QNAP serveren.

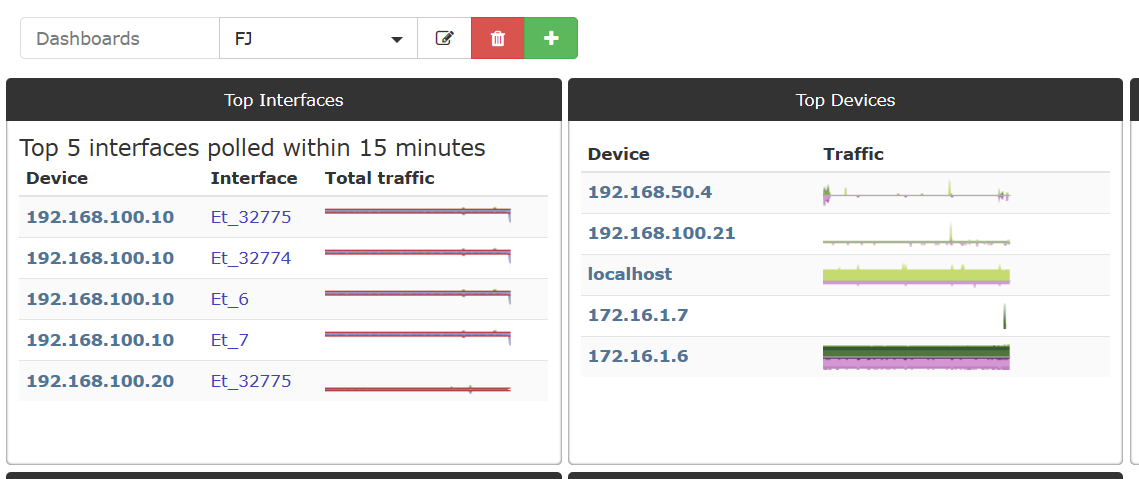


### File Share:

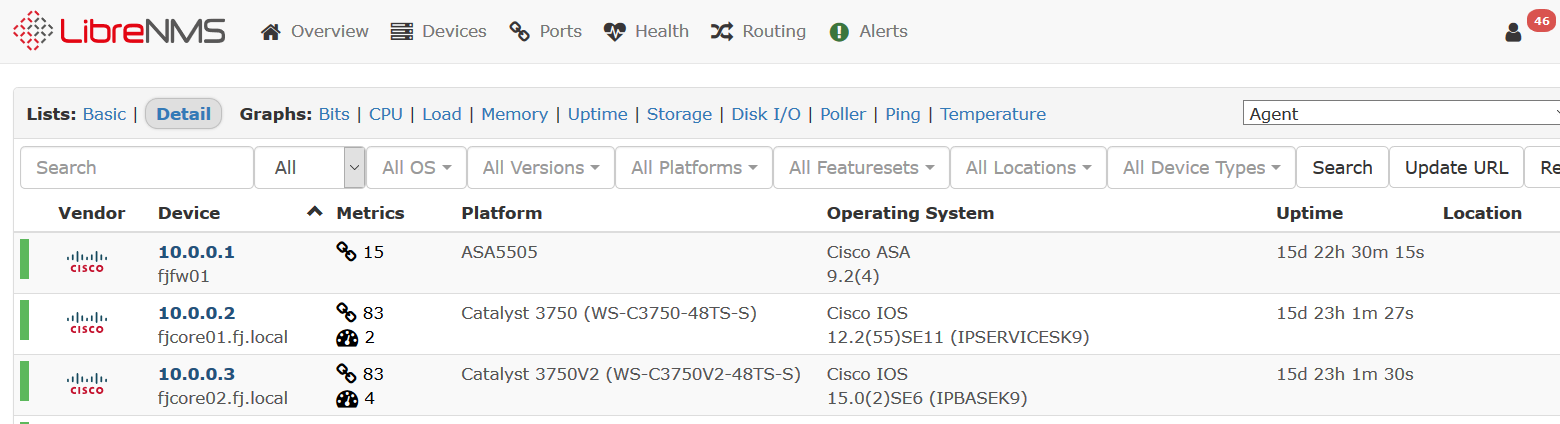
Alle ansatte vil have mappet et netværks drev(M:) via GPO, hvor de skal gemme deres filer på, da de er højt tilgængelig. Derfor er det vigtigt at de får at vide de ikke må gemme deres filer på det lokale drev.

### Server monitoring

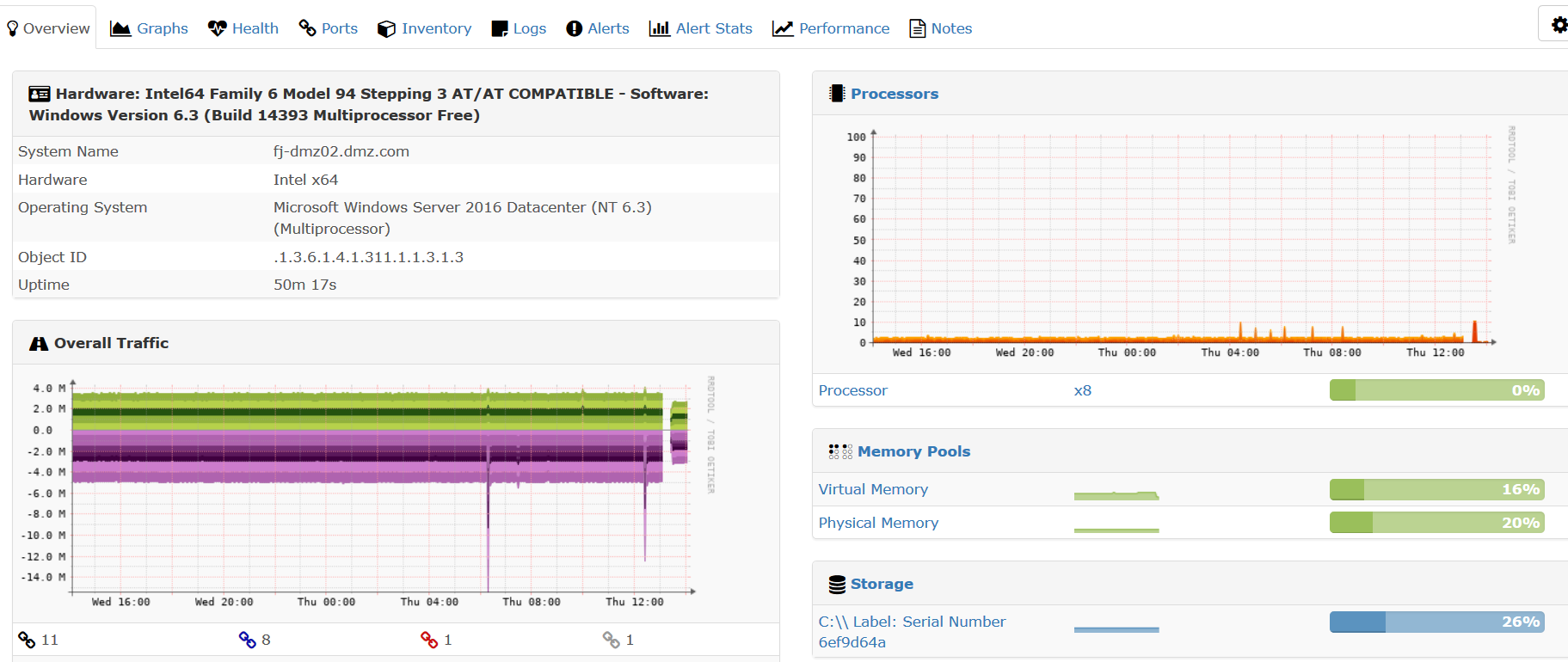
For at overvåge enhederne i netværket har vi sat en NMS server op som trækker information ud af vores devices omkring ex lagerplads og RAM forbrug. For at tilgå NMS serveren går man ind på 192.168.100.40 i sin browser og logger ind. Herefter vil man kunne se et dashboard. Dette dashboard kan man selv ændre som har har lyst til afhængig af hvad man gerne vil se, ved at trykke på (+) og tilføje en funktion.



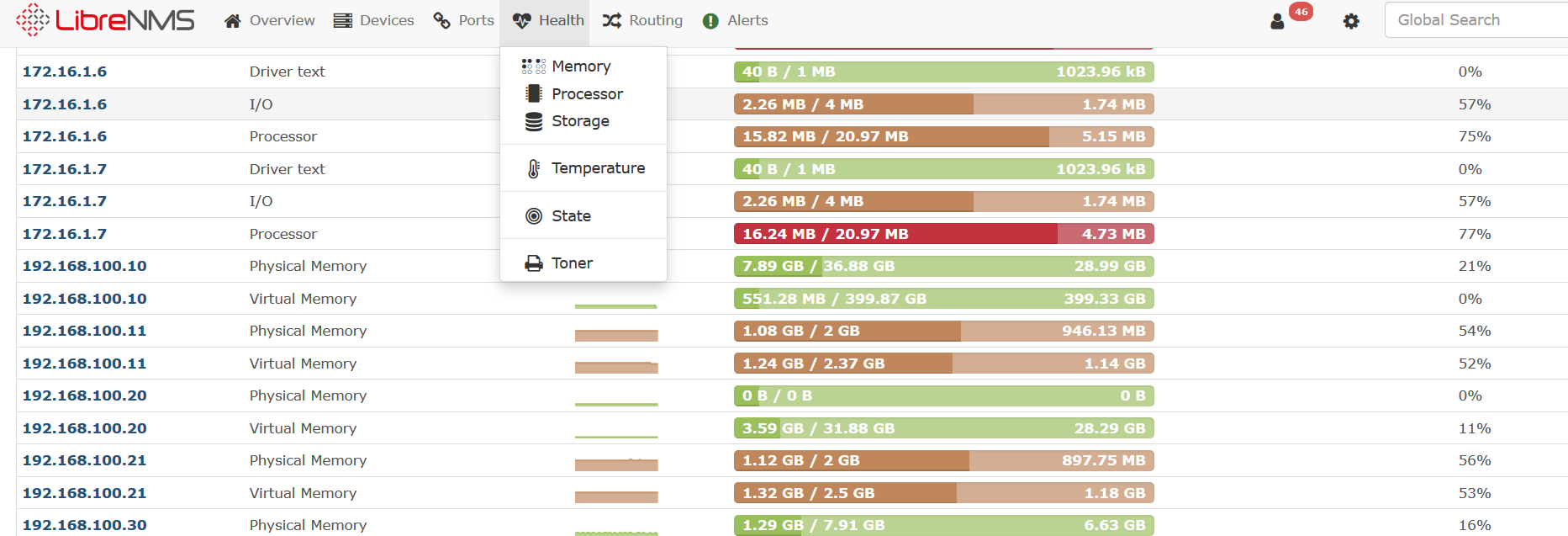
Hvis man trykker på devices i toppen og går ind under “all devices” vil man kunne se alle enhederne som NMS trækker information ud af.



Hvis vi vælger en device som vist på billedet ville vi få et overblik over, hvor meget ex ram og storage den bruger.



Hvis man vil have et overblik over, hvor meget Ram alle enheder bruger eller, hvad temperaturen er kan man i toppen vælge health og memory.



Vil vi gerne tilføje en ny enhed trykker vi på devices og add new device. Her kan vi tilføje enheden med IP adresse, SNMP version og community string.

