**QA og Test**

* Testdriven development

**Risiko og Risikostyring**En ”Risiko” definerés som: ”En mulig negativ planafvigelse”. Altså, en hændelse der kan bringe vores tidsplan, rygte, og i sidste ende, vores job i farezonen.  
For bedst muligt, at kunne ruste teamet imod konsekvenserne af en opfyldt risiko, er det vigtigt, at kunne komme disse risici i forkøbet.   
Risikostyring/analyse dækker over den valgte metode til at identificere, og håndtere, de problemer der eventuelt måtte opstå i forbindelse med afviklingen af et projekt. Selv i mindre projekter er der er en lang række ting der kan gå skævt, og derfor er det også vigtigt, at kunne prioritere de hændelser man forudser.

For effektivt at kunne håndtere risiko i vores projekt, har vi valgt   
\* Hvad er den mulige risiko  
\* Hvad er den vurderede sandsynlighed for at det hænder  
\* Hvad er konsekvensen hvis det sker  
\* Hvordan kan vi håndtere/undgå risikoen

Risikostyring i Scrum  
I et klassisk plandrevet scenarie udfører man en grundig risikoanalyse i starten af et projekt, eventuelt med tilhørende prototyper etc. Det kan man naturligvis også gøre i et Scrum-projekt, og i så fald foregår det i det såkaldte ”Sprint 0”. En alternativ måde at tilgå risikostyringen på er ved, at tage det på et sprint til sprint basis. Hvor man for hver afsat timebox i projektet vurderer hvilke risici der lurer i horisonten, og så håndtere dem som en naturlig del af det kommende sprint.   
En tredje måde at håndtere risiko i et projekt, er at vælge et trin imellem disse to yderligheder. Det hovedsaglige, at man tilpasser sig til situationen.

<http://www.scrumalliance.org/community/articles/2013/2013-may/risk-management-in-agile>

**Arkitektur**  
Scrum foreskriver, at man laver lige præcist så meget design forarbejde som er påkrævet. Hvad et udviklingshold så føler der er ”påkrævet” vil givetvis være helt forskelligt fra virksomhed til virksomhed. Det er dog forbundet med en række fordele, at man overvejer en grundlæggende struktur fra starten af et projekt. Som i et traditionelt plandrevet projekt, kan man med fordel tale med kunden. Det produkt som denne ønsker, bør lægge til grund for den grundlæggende arkitektur.   
Der er flere praktiske grunde til, at det kan være en fordel at have en basal arkitektur fra starten. Det kan hjælpe udviklerne til, at estimere projekts størrelse, både i tid og penge. Typisk er der nogle interessenter, være det så banken, product owner, eller virksomhedens ledelsesechelon, der ikke har kapacitet til, at forstå de udfordringer et projekt består af. Hvis man har et tentativt design, så har man også bedre muligheden for at kommunikere disse udfordringer igennem fra start. Derudover kan det også vise sig, at være et nyttigt holdepunkt senere i projektet hvis der opstår tvivl om retningen.

**Vores perspektiv**

I vores tilfælde ønsker product owner et produkt der kan holde dynamisk og persistent da, og som kan tilgåes via en hjemmeside. På længere sigt skal systemet kunne udvides til, at omfatte apps til android, iOs og Windows Phone. Ud fra de kriterier kan vi så som udviklere skimte omridset af en arkitektur, der har en database, og et interface som kan tilgåes fra disse platforme.   
Almindelig sund fornuft siger os, at vi bør sigte efter en lagdelt (omend, det ikke fra starten er præcist besluttet hvor mange lag vi taler om) arkitektur.   
For product owner er udviklingstiden en væsentlig faktor, derfor anser vi det som fornuftigt, at lægge vægt på en slank klient, og en fed backend. Det er ikke en beslutning uden ulemper, men det betyder, at jo mere logik der ligger i backend, jo mindre skal skrives igen når nye klienter skal udvikles.

Med disse overvejelser i hånden, og formaningen fra XP paradigmet om hyppig refaktorering, har vi det arkitektur fra start som vi har til hensigt, at have.