Pager Projekt

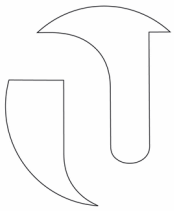
Systemarkitektur
Diagrambeskrivning

Abstrakt

Dokumentet beskriver det diagram som är en överblick över hela systemets arkitektur. Varför man bör använda sig av det och på vilket sätt man kan gå till väga för att skapa det. För och nackdelar med olika sätt att skapa det på.

Versionshistorik

Date	Version	Author	Description
24/05/2020	1.0.0	Elias Johansson	



IVAR JACOBSON CONSULTING

An Essential Unified Process Document

Innehållsförteckning

1 Introduktion

1.1 Dokumentets syfte

Det huvudsakliga syftet med detta dokument är:

- Ge en god överblick över systemets arkitektur för den produkt som grupp 8 utvecklade i kursen II1302 på KTH Kista.
- Ge en överblick över systemets olika komponenter som ingår i produkten.
- Förklara arkitekturens ursprung och varför den ser ut som den gör.

1.2 Avgränsning

Dokumentet begränsas till följande:

- Diskussioner och förtydligande kring aktuell design av systemarkitektur.
- Förändringar och förbättringar under utvecklingen.

Detta dokument kommer INTE behandla:

- Andra typer av arkitekturdiagram.
- Specifikationer eller diskussioner kring själva systemet och dess arkitektur.

1.3 Överblick

Dokumentet innehåller följande sektioner:

- **Bakgrund arkitektur** – Behandlar bakgrunden till val av arkitekturtyp och vilken modell som denna utgår ifrån.
- **Metodanpassning** – Behandlar vilka förändringar som har gjorts från den modell som arkitekturen utgår.
- **Resulterande arkitektur**– En beskrivning över hur den slutgiltiga och resulterande arkitekturen blev.
- **Lärdomar** – En beskrivning av de lärdomar som finns att göra utifrån projektets resultat och arbete.

2 Bakgrund Arkitektur

Följande del ger en bakgrund i arkitekturen och den metod som användes.

2.1 Arkitektur

För utveckling i små grupper kan det vara all idé att man som grupp tillsammans börjar med att sätta sig ner i ett möte tillsammans för att gemensamt säkerställa att man har samma vision för den produkt man ska utveckla. Av denna anledning kan det därför vara en god idé att börja med att diskutera den övergripande arkitekturen över projektet så fort som kravlistan och produktens specifikation är bestämd. Genom att tillsammans hitta en arkitektur kan man försäkra sig om att alla har samma

utgångspunkt och mål för projektet. För större projekt är det klart inte möjligt att involvera alla i arkitekturprocessen men det är fortfarande en vital del som måste finnas med. Kanske än viktigare vid större projekt eftersom det inte bara hjälper utvecklarna mot ett gemensamt och tydligt mål utan även för att bibehålla kontroll över projektet. Genom en tydlig indelning så kan man relativt enkelt se huruvida det arbete som sker är inom ramarna för det utsatta målet. Först när den övergripande arkitekturen är bestämd kan utvecklarna börja sitt arbete och därefter kommer respektiva del i system utvecklas. Då dessa är individuella komponenter av systemet i sin helhet så kan dessa utvecklas helt individuellt mot varandra utfrån förutbestämda krav och mål.

2.2 Metod

Den modell som tillslut valdes för att ta fram en systemarkitektur är en modell framtagen av Simon Brown och kallas för C4 modellen [C4]. C4 syftar på att modellen utgår från fyra delar som alla blivit namngivna så att de börjar på bokstaven C. De fyra delarna är Context, Containers, Components och Code. Dessa fyra delar agerar som olika lager eller abstraktionsnivåer.

Context är den översta nivån och har till syfte att visa produkten i sin helhet från ett ytterst övergripande perspektiv. Målet här är att vara en startpunkt och visa hur produkten passar in i sin omgivning.

Container går ner en nivå och börjar titta på hur systemet faktiskt ser ut och är uppbyggt. Det är en samling block som på en hög nivå beskriver deras respektive syfte och relationen mellan dessa. Detta är första abstraktionsnivån för det faktiska utvecklingsarbetet.

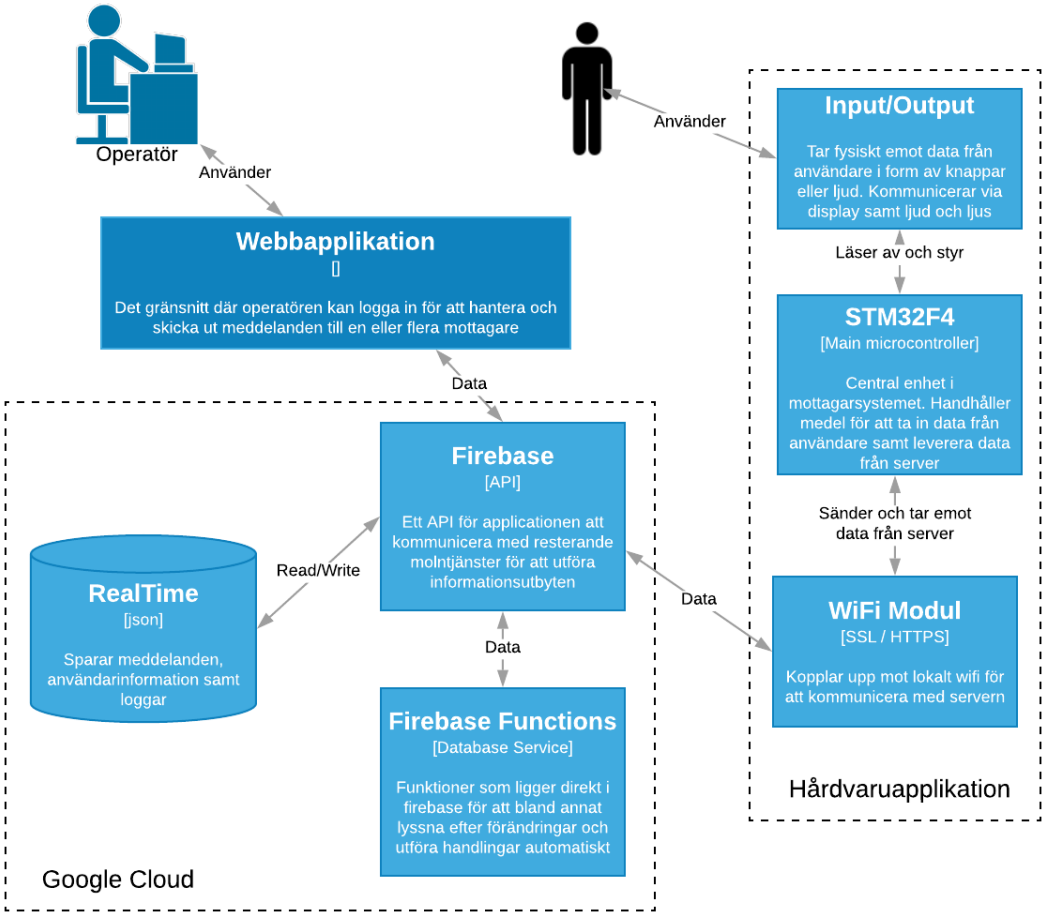
Components är idealt en individuell beskrivning av varje block från ovan abstraktionsnivå. Dvs varje container har en egen arkitektur på komponentnivån. Komponenter är här alla de delar som krävs för att utveckla den delen av produkten som denna beskriver. En komponent är oftast en relativt fristående del av ett större system som kan utvecklas helt individuellt. I de flesta fall får denna komponent in någon form av data från andra komponenter och resultatet i sin tur vidarebefodras till annan eller andra komponenter.

Code är precis som det låter en ingående struktur över hur koden ser ut. I detta fallet har vi på samma sätt gått ner en abstraktionsnivå och idealt ska det även här finnas en individuell kodnivåbeskrivning för varje komponent från nivån ovanför.

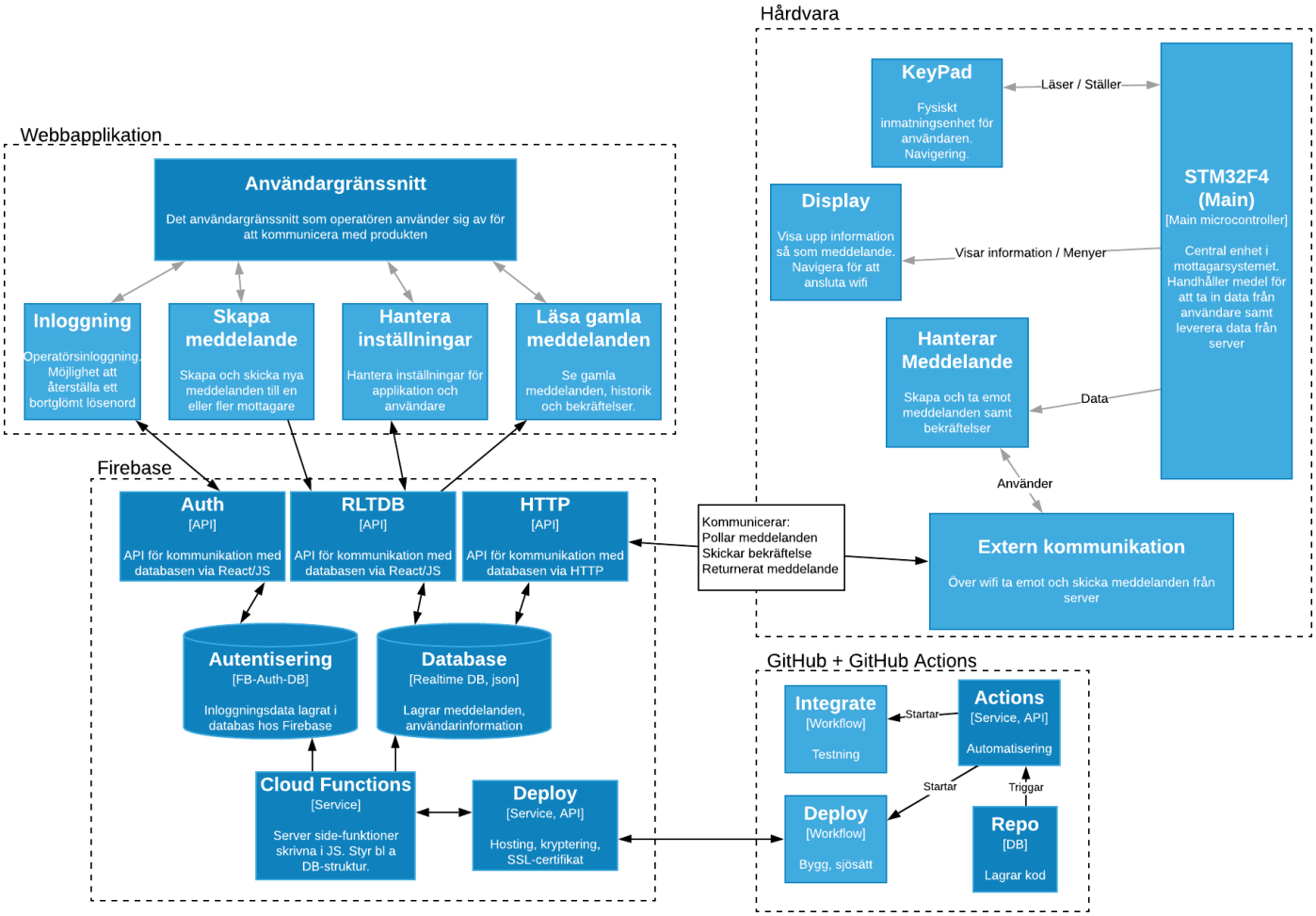
3 Metodanpassning

Det är uppenbart att modellen är uppbyggd med ett stort och mer komplext system i åtanke. Man kan ganska enkelt inse att eftersom komplexiteten och tidsåtgången för att ta fram en komplett arkitektur med denna modellen ökar exponentiellt då varje block från nivån ovan i sin tur ska ha egna block som i sin tur ska ha egna block osv. Det innebär att för varje block som läggs till på en hög abstraktionsnivå kommer det genereras ett väldigt stort antal block som ska illustreras längs kedjan. Med detta och detta projektets storlek i åtanke så var det viktigt att i ett så tidigt skede som möjligt skala ner komplexiteten för att inte ödsla för mycket tid på sånt som inte tillför något värde. I första steget togs beslutet att inte bygga ett arkitekturdiagram på den högsta abstraktionsnivån Context utan istället låta nivån under, Container, vara den yttersta överblicken för systemets helhet. Detta var möjligt eftersom storleken och komplexiteten av systemet var så pass liten att den kunde enkelt illustreras på denna

nivå utan att strukturen blir överväldigande eller svårläst. I bilden nedan kan slutresultatet av arkitekturen för Containernivån ses.



Denna fungerar endast som en kort sammanfattning av arkitekturen och det är först på nästa nivå ner i hierarkin som utvecklingen kan dra nytta av dokumentet. Även här anpassades metoden något utifrån den ursprungliga C4 modellen. Skulle man strikt följa denna så ska varje block i ovan bild ha en egen komponentbeskrivning. Komplexiteten i detta projekt är dock väldigt låg och det finns ingen anledning att göra detta här. Det skulle dessutom bli svårare att utläsa relationer mellan dessa om allting var uppdelat i små filer som innehåller väldigt lite information vardera. Istället så gjordes det bara en enda stor komponentbeskrivning som innehåller samtliga komponentindelningar som projektet består av och istället har komponenterna grupperats utefter grupperingen från överblicksnivån ovan. På så vis kan komponentbeskrivning fungera som en mer detaljerad överblick över systemets helhet och det blir därmed betydligt enklare för utvecklare att snabbt se vad som behövs göras. Översta nivån finns där som en hjälp för utomstående eller annan mindre insatt part att sätta sig in i och förstå projektet. Nedan visas komponentarkitekturen.



I ovan bild kan vi se uppdelningen där samtliga block har grupperats in i en av fyra olika grupper. Webbapplikation, hårdvara, databas och GitHub. Detta diagram har också förändrats under utvecklingens gång då den första iterationen som ett exempel även innehöll samtliga önskemål som fanns i kravspecifikationen. Mot slutet av produktens deadline så hölls det ett möte där gruppen fastställde den slutgiltiga listan. Dvs att gruppen checkade av på listan vad som var gjort, markerade det som skulle finnas med i slutprodukten och markerade de önskemål som inte skulle implementeras. Därefter kunde också arkitekturen uppdateras så att den var helt i enighet med den faktiska slutprodukten.

Det finns ytterligare ett steg ner i abstraktionsnivån. Det innebär att varje komponent i ovan bild ska i sin tur ha en beskrivning på kodnivå. Ett diagram som beskriver uppbyggnad och struktur över den kod som ska utföra den uppgift som föreligger komponenten. Det arbetet påbörjades och för någon enstaka komponent så blev det även beprövat. Gruppen kunde dock snabbt inse att för arbetslag av denna storlek på små projekt som detta så var det inte fördelaktig att göra diagram på såhär låg nivå. Dels var det väldigt kostsamt i form av tid men det användes inte heller av någon utvecklare i något av utvecklingens olika skede. Det kostade med andra ord mer än vad det tillförda värdet var.

4 Resulterande Arkitektur

Som resultat av metodanpassningen i föregående kapitel fick gruppen ut en systemarkitektur som i slutändan var en förkortad version av C4 modellen av Simon Brown. Istället för att använda sig av fyra C så användes bara två, dvs Container och Component där den förstnämnda var en enkel överblick och den sista en mer komplett arkitektur och struktur. Man skulle kunna kalla den för C2 modellen. Genom komponentdiagrammet fick gruppen en bra arbetsindelning då samtliga komponenter är individuella block i det totala systemet och kan därmed också utvecklas helt individuellt och behöver inte knytas samman förrän i ett senare skede. I vissa fall kunde hela block översättas direkt till *Stories* på arbetstavlan och därmed tilldelas en utvecklare medans i andra fall var det enklare att dela upp en komponent i två eller flera *Stories*. I detta fall föll dock samtliga *Stories* från en och samma komponent till en och samma utvecklare. Motiveringen till att dela upp komponenten var snarare det att tidsåtgången i den sprint man just då befann sig i var för stor. Om tidsåtgången för en hel komponent var större än en iteration eller sprint så delades den in i mindre delmål.

5 Lärdomar

Flera lärdomar finns att ta efter utfört projekt för både författare och grupp. I första hand så inses vikten av en bra struktur på arbete från början. Det både underlättar och sparar mycket tid av att samtliga utvecklare direkt från första knapptryckningen jobbar i samma spår mot samma mål. En väl beskrivande arkitektur hjälper med detta samtidigt som en väl indelad arkitektur hjälper i arbetsfördelningen. Det är helt enkelt värt att ta den tiden extra i början för att tjäna igen det i ett senare skede. Andra lärdomar är till exempel att vet var gränsen går. Det är viktigt att arkitekturen inte blir onödigt komplicerad eller innehåller onödiga detaljer. Det minskar möjligheten för den snabba överblicken och riskerar missförstånd. Det är också viktigt att lägga ner tiden på det som betyder något och som är viktigt för att arbetet framåt. Samtliga beslut måste ställas med frågan *hjälper detta gruppen i vårt arbete?* i baktanke.

Referenser

C4: Simon Brown, The C4 model architecture, http://www.c4model.com, 30/05-2020