Hovedopgave

Mikkel Bendix Munk
mikk2385@edu.eal.dk

Christoffer Frydkjær Kristensen
chri635r@edu.eal.dk

13. november 2015

Indhold

Gruppekontrakt

- Vi arbejder efter SCRUM
- Sprint varighed: 14 dage
- Mødetid: senest kl. 8.00
- Daily standup: kl. 8.15
- Arbejdsugen er 37 timer inkl. frokost
- Kodekoordinering, gør opmærksom hvis man tager en ny opgave
- Spørg om hjælp
- Frihed under ansvar, det forventes af gruppens medlemmer at man overholder aftaler og gør alt hvad man kan for at overholde deadlines
- Hvis der bliver konflikter, skal vi turde tale om det og finde en løsning
- Hvis man skal gå før, gør man opmærksom på det til daily standup
- Hvis man er forhindret i at møde, kontaktes gruppen hurtigst muligt

Projektstyring

Scrum

Vi har anvendt *Scrum* som værktøj til projektstyring igennem hele forløbet med hovedopgave. Det giver nogle gode retningslinjer for hvordan man skal planlægge et projekt, og giver samtidig mulighed for nemmere at forhindre at ens projekt går i den forkerte retning. I forlængelse af at man typisk vil prøve at undgå fejl eller dårligere perioder samt dårlig arbejdsgang, giver *Scrum* også mulighed for at rette op på de ting der går skidt eller hvis man er utilfreds med noget.

Planlægning

Vores projekt er delt op i nogle perioder, som kaldes for et *sprint*. Et sprint varer for vores vedkommende 2 uger, og består af nogle forskellige faser. Det starter med *sprint planning*, som er den fase hvor vi planlægger hvad vi skal lave af opgaver inden for sprintet. Vores overordnede opgaver for projektet er beskrevet i en *backlog*, som er defineret ud fra use cases. Fra *backlog'en* trækker vi de opgaver ind i det kommende *sprint*, som vi mener er mulige at lave færdige inden tidsperioden på to uger er gået. Efter de overordnede opgaver er trukket ind i *sprintet*, skal de deles op i mindre og mere håndtèrbare opgaver. Når dette er gjort og vi er enige om hvad opgaven indeholder, kaster

vi nogle point på de opdelte opgaver som definerer hvor lang tid vi tror det tager at løse det givne problem. Så snart vi er enige om hvor tidskrævende alle opgaver er, kan *sprintets* største fase begynde hvor vi løser opgaver.

Hver dag starter med et *daily stand-up* møde, hvor hver medlem af gruppen fortæller om hvad personen lavede dagen før, hvad der skal laves den pågældende dag og eventuelt hvilke udfordringer der har været eller hvis man har noget vigtigt der skal drøftes.

Når et *sprint* er overstået holdes der *sprint review*, hvor vi inviterer interessenter til et møde der går ud på at fremvise hvad vi har lavet i løbet af det forgange *sprint*. På mødet er det også muligt for andre at komme med feedback til produktet eller arbejdsgangen.

Til *sprint retrospective* reflekterer vi blandt andet over *sprintets* forløb og konkluderer på feedback. Herefter nedskriver hvert medlem af gruppen et tal fra 1-5, hvor 1 er lavest og 5 er højest. Dette tal summerer op hvor godt man personligt har haft det i løbet af sprintet. Til tallet følger der nogle positive og negative kommentarer, som er en forklaring på det tal hver person er endt med. Vi taler herefter frit fra leveren om de gode og dårlige oplevelser i løbet af sprintet, og så sørger gruppens *scrum master* for at eventuelle bliver løst.

Vi har tilføjet en graf der viser det gennemsnitlige humør-tal uge for uge. Den afspejler hvor godt gruppen synes forløbet er gået.

TODO: Indsæt happiness graf (sprint 1: 4, sprint 2:)

Kvalitetssikring

Vi vil kvalitetssikre vores produkt ved at teste alt det der giver mening at teste, så vi sikrer at funktionaliteten fungerer efter hensigten. Derudover laver vi reviews på hinandens kode inden det bliver godkendt og merges til master branchen i git. For at sikre læsbarhed og konsistens i koden, følger vi nogle specifikke PHP kode standarder som er beskrevet i afsnittet omkring kode standarder. Efter hvert sprint holder vi et review møde hvor interessenter fra Ordbogen A/S vil deltage og komme med feedback. Så vi på den måde hele tiden sikrer os at det vi laver, er det Ordbogen gerne vil have. For at sikre kvaliteten i projektet yderligere vil vi bruge FURPS som kvalitetsmodel og hele tiden veje projektet op imod FURPS.

Produktbeskrivelse

Systemet kan læse, gemme og vise statistik på en præsentabel måde. Systemet er delt op i to subsystemer, hvoraf det ene tager sig af modtagelsen af data og det andet tager sig af visningen af den data. Grunden til vi har to subsystemer er at vi tager udgangspunkt i to typer af brugere, hhv. brugeren der indlæser data og brugeren der aflæser data. Vi har designet systemet efter brugerens behov, som har været:

- bla bla bla

Vi har valgt at bruge AngularJS i frontend og derfra kalde RESTful APIs i backenden. I figur 1 kan man se hvordan det samlede system kommer til at se ud, hvor vores middleware/routing lag kalder backenden, som returnere et view, med de dataer, der skal bruges. Vi lavede denne arkitektur ud fra GRASP principperne og imødekomme object orienteret programmering

Database

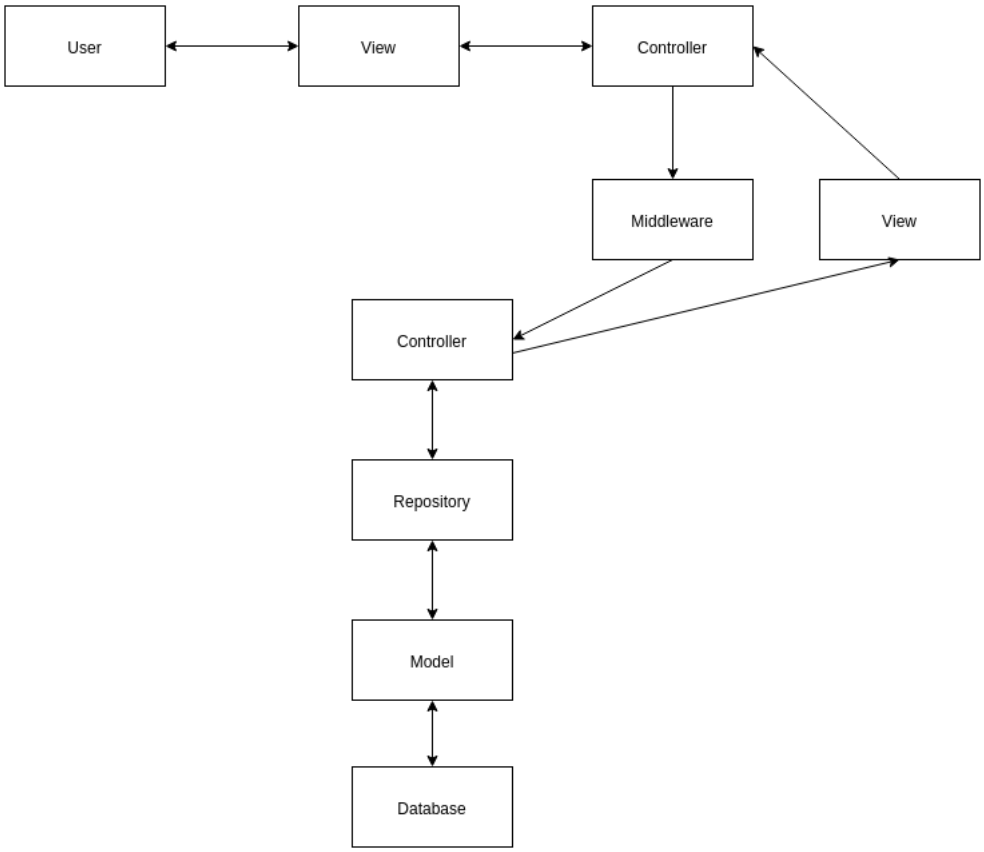
Valg af database

Vores valg stod imellem MySQL/MariaDB, MongoDB og ElasticSearch. Vores valg faldt på ElasticSearch. Grunden til vi valgte at bruge ElasticSearch og ikke de to andre var at da vi laver et statistik program, som skal hente data på tværs af forskellige tabeller, ville vi undgå at lave en masse kald til databasen og joine data sammen. Det er ElasticSearch lavet til og vinder klart på performance. Der hvor ElasticSearch taber sammenlignet med de andre databaser, er at det oprindeligt er lavet som en search engine og ikke en decideret database. Dvs. man får ikke den samme sikkerhed, som de andre tilbyder.

ElasticSearch

ElasticSearch er en næsten real-time search engine som er bygget på Apache Lucene, som er et søge API. ElasticSearch er bygget op omkring et Cluster, som er en liste af nodes, hvor en node er en server der gemmer data. Clustered indexere så dataen og gør det søgbart[1]. ElasticSearch gør også brug af index, som er en liste af dokumenter, der har samme karakteristika og er skrevet i JSON format. Ved større index kan de dele et index op i shards og et shard er et fuldt funktionelt index. Det gør man kan kan skalere data

[1] https://www.elastic.co/guide/en/elasticsearch/reference/current/setup-repositories.html



Figur 1: Vores valgte arkitektur for både frontend og backend

horisontalt.

TODO: Tilføj billede af opbygning af ElasticSearch