|  |
| --- |
| universitetet i Stavanger |
| Skole App prosjektrapport |
| DAT 220 Software development |
| **12/8/2016** |
| Gruppe 5  Yngve  Torbjørn  Helge Johannessen Bjorland  Tom  Aleksander |
|  |

|  |
| --- |
| [Type the abstract of the document here. The abstract is typically a short summary of the contents of the document. Type the abstract of the document here. The abstract is typically a short summary of the contents of the document.] |

“Agile practices focus on giving teams and individuals ownership of their work and accountability to the team and organization, so that you can care to make things better.”

# Forord [Helge]

kort forklarer hvorfor en skriver rapporten og forhistorien til valg av tema.

Dersom flere forfattere står bak rapporten bør en kort utdype hver enkelt sitt bidrag. Forordet bør inkludere en takk til personer som har bidratt med hjelp og innspill underveis.

Stavanger, 8. Desember 2016

Gruppe 5: Yngve, Torbjørn, Helge, Tom og Aleksander

[sign]

# Sammendrag [Helge]

* Hensikt, problemstilling
* Omfang og begrensninger
* Metoder som er brukt
* De viktigste resultatene
* Konklusjoner og anbefalinger

Innhold

[Forord [Helge] 1](#_Toc467959168)

[Sammendrag [Helge] 2](#_Toc467959169)

[Introduksjon [Aleksander] 4](#_Toc467959170)

[Metode [Yngve/Tom] 4](#_Toc467959171)

[Arbeidsform 4](#_Toc467959172)

[Verktøy 4](#_Toc467959173)

[Oppbygning 4](#_Toc467959174)

[Resultat [Torbjørn] 4](#_Toc467959175)

[Konklusjoner og anbefalinger 5](#_Toc467959176)

[Startfasen 5](#_Toc467959177)

[Kode 5](#_Toc467959178)

[Verktøy 6](#_Toc467959179)

[Samarbeid 7](#_Toc467959180)

[Oppsummering 7](#_Toc467959181)

[Videre arbeid 7](#_Toc467959182)

[Referanser 9](#_Toc467959183)

[Vedlegg [Tom / Yngve] 10](#_Toc467959184)

[Skisser 10](#_Toc467959185)

[Kode 10](#_Toc467959186)

# Introduksjon [Aleksander]

* Bakgrunn
* Problemstilling, målformulering og omfang
* Eventuelt leserveiledning og beskrivelse av rapportens videre oppbygging

HUSK: Hovedpoeng var sammenligning

# Metode [Yngve/Tom]

## Arbeidsform

Litteratur henvisning på ting som er gjennomgått i forelesning? Ex. Waterfall prosess?

## Verktøy

## Oppbygning

Begynte med Rust/Mysql 🡪 gikk videre til php

Testing: Husk – console log, men ikke i funksjoner

# Resultat [Torbjørn]

Resultatene skal presenteres på en klar, saklig og objektiv måte i dette kapittelet, mens diskusjonen rundt dem kommer i diskusjonskapittelet.

Liste over user stories som er levert og hvilke vi ikke fikk tid til…

# Konklusjoner og anbefalinger

## Startfasen

***Hovedpoeng:*** *Sikre felles forståelse av problemet. Sett av nok tid til brainstorming i starten og bruk skisser for å diskutere løsninger.*

Det er viktig å bruke nok tid i starten på å forstå problemstillingen. Vi fikk oppgitt at vi skulle lage en applikasjon hvor en bruker kunne sammenligne fridager på ulike skoler. Vi startet naturlig nok med tanker om å lage en standard kalender, men etter en del diskusjon kom vi frem til at dette ikke løste problemet godt nok. Vi ble enige om at en vanlig kalender ikke fungerer bra nok for å sammenligne fridager på tvers av flere skoler. Etter en del brainstorming og skissering endte vi med en løsning som lignet mer på en Gantt oversikt. Under brainstorming fasen brukte vi tavle for å tegne opp mulige løsninger. Etter dette gikk vi over til en kombinasjon av skjermdump og et enkelt tegneprogram for å skissere ulike løsninger og endringer. Vi brukte en del tid i starten på å få en felles forståelse av problemet og hvordan vi ønsket å løse det. Dette var en god investering ettersom det gjorde gjennomføringen mer effektiv.

## Kode

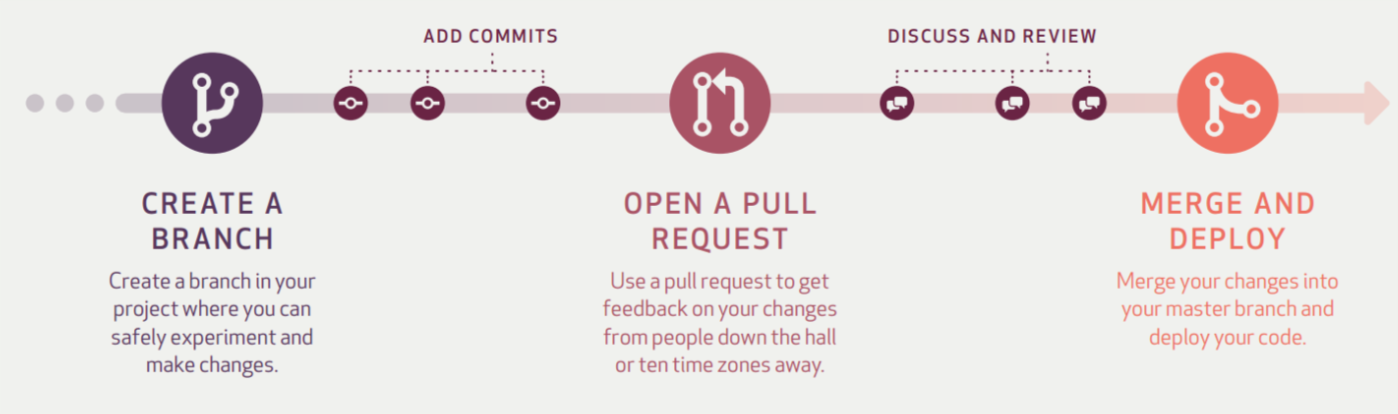
***Hovedpoeng:*** *Standardiser språk, struktur og arbeidsmetodikk.*

I et prosjekt som dette hvor flere skal bidra å jobbe med samme kode var vår erfaring at det kan være utfordrende om noen bruker programmeringsspråk som andre ikke kjenner til. Det kan være greit å bli enige om hva som er standardspråk fra starten av. Vi valgte språk som alle hadde kjennskap til slik at vi startet med samme forutsetninger.

Det er veldig nyttig å benytte SOLID prinsippene for objekt orientert programmering (Wikipedia, 2016) når man bygger opp kodebasen. Dette gjør det enklere for alle å forstå koden og feilsøking tar mindre tid. Noe av koden måtte vi omstrukturere i etterkant noe som var tidkrevende og vanskelig. Dette kunne vært unngått om vi hadde fulgt SOLID prinsippene gjennomgående.

Det var veldig nyttig å benytte Git workflow som vist i Figur 1. Vi begynte med å få på plass et skjelett som inneholdt strukturen vi hadde blitt enige om. Deretter laget man en egen branch for nye egenskaper man skulle legge til. Det er noen fallgruver som er greie å unngå. Vi erfarte tidlig at det er greit å begrense størrelsen på endring i hver branch. I starten fikk vi litt problem med å merge inn til master fordi branchen var så ulik master at det var vanskelig å se om man skrev over endringer andre hadde gjort i mellomtiden. Vi prøvde isteden å dele opp i mindre oppgaver slik at vi fikk flere og hyppigere merges til master. Det kan også være problematisk å reverte merges som er gjort dersom noen har merget inn endringer i etterkant.

Når vi skulle gjøre større struktur endringer avtalte vi dette på forhånd slik at alle merget inn det de hadde gjort først. Vi stoppet utviklings arbeid midlertidig mens omstrukturering pågikk og fortsatte deretter med en oppdatert branch fra master. En annen metode vi brukte var å lage en duplisert fil hvor omstrukturering ble gjort og henvisning ble endret til denne filen når alt var klart.



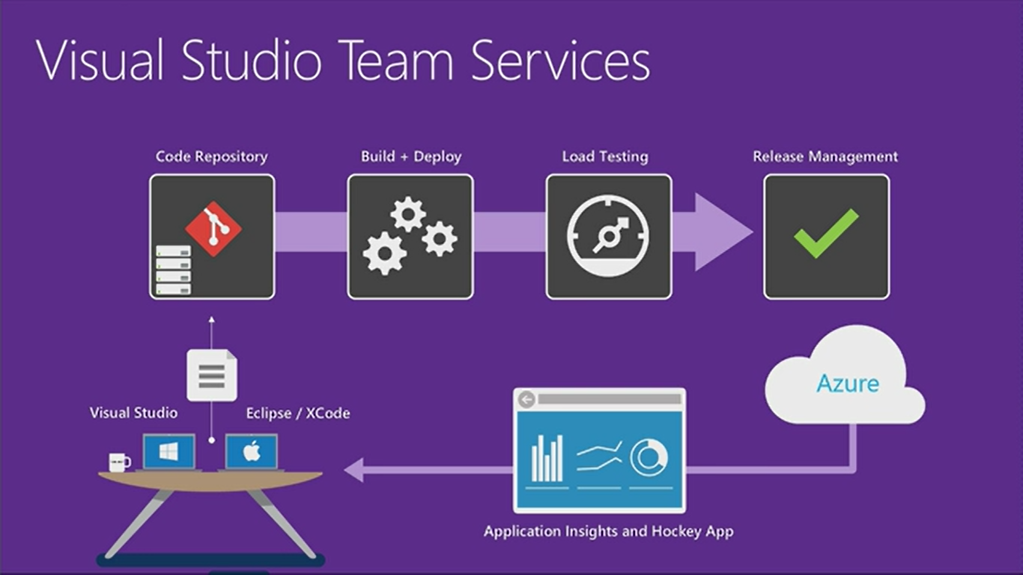
Figur 1- Github workflow (Github, 2013)

## Verktøy

***Hovedpoeng:*** *Finn et samarbeidsverktøy som har all funksjonalitet du kommer til å trenge.*

Vi startet prosjektet med å bruke Trello og Google docs som samarbeids verktøy. Trello tilbyr et enkelt kanban board, og user stories satt vi opp og prioriterte i et Google docs regneark. Dette fungerte til å begynne med, men når det ble stilt flere krav om å jobbe i henhold til Scrum metodikk så var det en del oppgaver som ble svært tungvinte eller ikke mulig å gjøre. Vi valgte derfor å bytte til Visual Studio Team Services (VSTS) som hadde de funksjonene vi manglet. En overordnet illustrasjon av funksjonene i VSTS er vist i Figur 2. De tilleggsfunksjonene vi syntes var mest nyttige var følgende:

* Timeboxing av user stories i sprinter
* Oversikt over user stories som var pågående og i backlog
* Oppdeling av user stories i underoppgaver
* Estimering av oppgaver med tid og storypoints
* Oversikt over tilgjengelig og planlagt kapasitet i teamet
* Nyttige rapporter for retrospektiv sesjonene
* Git integrasjon
* Integrert test verktøy
* Automatisk bygging og publisering basert på triggere (ved branch merge til master)
* Nyttige Add-Ons som f.eks. Arcade Hub når man trenger en pause i kodingen.



Figur 2- Visual Studio Team Services (Vunvulea, 2016)

## Samarbeid

***Hovedpoeng:*** *Sett opp korte og hyppige statusmøter samt retrospective sesjoner etter endt sprint. Ha klare ansvarsområder. Sjekk hverandres kode før man merger til master.*

Vi hadde et distribuert team, noe som ga oss noen ekstra utfordringer i forhold til samarbeid. Vi brukte Skype for å kommunisere, noe som fungerte bra. I de fleste møtene brukte vi bare lyd, men vi kunne muligens hatt enda mer effektivt samarbeid om vi hadde brukt mer video. Det var også noe problem med lydkvalitet, noe som ble løst med å kjøpe inn en konferansehøyttaler.

Vi ble tidlig enige om å fordele ansvaret for ulike deler av applikasjonen. Dette sørget for at noen alltid hadde eierskap til de ulike delene av koden. Utvikling gikk selvfølgelig ofte på tvers av de ulike delene, men om det var noe en trengte å avklare visste man hvem en skulle kontakte.

Vi hadde to status møter i uken, tirsdag etter tilbakemelding fra produkteier og på fredag. I tillegg hadde vi retrospektiv møte på mandagene slik at vi var klar for møte med produkteier på tirsdagen. Status møtene var korte statusmøter hvor vi gikk gjennom Kanban board uten å gå i detalj på sakene. Problemløsing ble håndtert i mindre grupper i etterkant. Hyppige statusmøter var en god måte å forsikre at vi hadde riktig progresjon i arbeidet. VSTS gir også gode grafer som viser burndown og kan gi en indikasjon på hvordan man ligger an i forhold til plan.

I forhold til samarbeid kan det også være nyttig å arrangere noen hackaton sesjoner hvor man sitter sammen å koder. Dette erfarte vi kunne gi god progresjon da det var svært effektivt å få gjort avklaringer med andre i teamet.

Vi hadde litt problemer i starten med kode som ble merget inn til master og førte til feil. For å unngå dette innførte vi en regel om at før man merget inn en branch til master måtte en annen på teamet gå gjennom endringene å godkjenne, en såkalt «code review». For å gjøre det enklere å teste for andre satte vi også opp manuelle tester for hver enkelt user story før vi begynte utviklingen. Personen som gjennomførte gjennomgangen gjorde også testen for å se at all funksjonalitet fungerte som planlagt. Dette var en effektiv måte å hindre at det ble merget inn kode med feil.

## Oppsummering

Totalt sett var dette en spennende reise hvor vi fikk utvikle oss både som team og enkeltpersoner. Vi gikk fra en enkel og standard tilnærming til samarbeid til et sluttpunkt hvor vi hadde innarbeidet de fleste av Scrum prinsippene. Det var oppløftende å se hvordan samarbeidet i teamet ble mer effektivt etter hvert som vi forbedret måten å jobbe på.

# Videre arbeid

Web applikasjonen er i utgangspunktet klar til å tas i bruk, men en må sette opp en server plassering. Vi satte opp produksjonsmiljø både i Azure og AWS og publiserte der. Dette var kun med prøve kontoer hvor vi gikk tom for kreditt før slutten av faget. Azure hadde god integrasjon mot VSTS som vi brukte til å sette opp automatisk bygging og publisering ved endring.

Design og brukervennlighet kunne vi nok ha jobbet videre med, gjerne med en user interface designer. Muligheten til å få varsling på ønskede hendelser var noe vi ville ha lagt til om vi skulle ha fortsatt utviklingen, men dette er rimelig omfattende å få på plass. Vi rakk heller ikke å legge til funksjonalitet for å legge til hendelser i egen kalender.

# Referanser

Github. (2013, Desember 13). *Github*. Retrieved November 2016, from Guides: https://guides.github.com/pdfs/githubflow-online.pdf

Vunvulea, R. (2016, Januar 12). *Vunvulea Radu Tech Wall*. Retrieved from Tools to migrate to/from Visual Studio Team Services Radu Vunvulea: http://vunvulearadu.blogspot.no/2016/01/tools-to-migrate-tofrom-visual-studio.html

Wikipedia. (2016, November 16). *SOLID\_(object-oriented\_design)*. Retrieved November 26, 2016, from Wikipedia: https://en.wikipedia.org/wiki/SOLID\_(object-oriented\_design)

# Vedlegg [Tom / Yngve]

## Skisser

## Kode