Rapport

*Fellesprosjekt 2015: Kalender*

Dette dokumentet er et resultat av arbeidet vårt i faget TDT4140 dette semesteret, og fellesprosjektet mellom fagene TDT4140 og TDT4145 som har gått over de siste fire ukene. Her finnes dokumentasjon til systemet vi har laget, og mye rundt selve prosessen

Buadu, Lange, Lundeby, Opheim, Tufteland

20.03.2015

Rapport

Fellesprosjekt 2015: Kalender

Innhold

[Beskrivelse av systemet og problemet - Ferdig 2](#_Toc413412676)

[Beskrivelse av utviklingsmetodikk -ferdig 2](#_Toc413412677)

[Krav -påbegynt 3](#_Toc413412678)

[Generelle krav 3](#_Toc413412679)

[Avtaler 3](#_Toc413412680)

[Bookingsystem 3](#_Toc413412681)

[Use cases 3](#_Toc413412682)

[Aktivitetsdiagram 3](#_Toc413412683)

[Klassediagram 3](#_Toc413412684)

[Testing - påbegynt 3](#_Toc413412685)

[Riskikoanalyse -ferdig 3](#_Toc413412686)

[Change management - påbegynt 3](#_Toc413412687)

[Pakker 3](#_Toc413412688)

[Klassene 3](#_Toc413412689)

[Tidsestimat og tid brukt 4](#_Toc413412690)

[Gjenstående arbeid 4](#_Toc413412691)

[Refleksjon om prosjektet - påbegynt 4](#_Toc413412692)

[Kommentarer til gjennomføring av faget 4](#_Toc413412693)

[Timeliste 5](#_Toc413412694)

[Møtereferater - påbegynt 5](#_Toc413412695)

[Referat 23.02 5](#_Toc413412696)

[Referat 24.02 7](#_Toc413412697)

[Referat 27.02 8](#_Toc413412698)

[Referat 03.03 10](#_Toc413412699)

# Beskrivelse av systemet og problemet - Ferdig

Kalendersystemet er laget for å hjelpe bedrifter med å avtale møter og organisere møterom. Systemet fungerer slik at hver ansatt har en personlig kalender og hver gruppe (eller subgruppe) har en delt gruppekalender. Enhver bruker av systemet kan opprette og invitere andre til avtaler, samt knytte avtalen opp mot et ledig møterom. De inviterte deltakerne får så mulighet til å godta eller avslå invitasjonen.

# Beskrivelse av utviklingsmetodikk -ferdig

Scrum er en metode som beskriver et prosjektledelserammeverk. Den bygger på at man har en serie med «sprints» som er fiksert på perioder der en økning i systemet er utviklet. Planlegging er gjort ved at man sjekker loggen og velger de høyest prioriterte gjenstående oppgaver til en sprint.

Scrum tas i bruk når man har en smidig utviklingsprosess. Og deles som sagt opp i sprinter. Lengden på en sprint vil variere fra prosjekt til prosjekt ut i fra størrelsen på prosjektet og organiseringen av teamet, men standarden er to uker lange sprinter.

Scrum er velutviklet og er i stor grad den utviklingsprosessen som brukes ute i arbeidslivet. Det at scrum er mye brukt tyder jo på at det funker og at det kanskje er fordelaktig framfor andre metoder. Det finnes flere fordeler og ulemper ved scrum. Vi har listet opp noen av dem.

* I smidige metoder som Scrum er det lettere å tilpasse seg endrede krav fra kunden. Et prinsipp for smidige metoder er å forvente forandringer i krav. I skiftende businessmiljøer skjer endringer raskt, og det er viktig å være i stand til å takle endrede krav.
* I Scrum leveres programvaren i flere inkrementer. Dermed får kunden mulighet til å prøve ut programvaren, og gi tilbakemelding på hva som fungerer og hva som ikke gjør det.
* Scrum er generelt mer kostnadseffektivt enn f.eks. vannfallsmodellen, grunnet fokuset på å redusere "overhead" i prosessen. Eksempelvis er det lite fokus på lage omfattende dokumentasjon.
* I Scrum er utviklingsprosessen i stor grad gjennomsiktig - alle teammedlemmene vet gjennom hele prosessen hva som skjer. Samtidig er hele teamet aktivt med i beslutningsprosesser. På denne måten føler utviklere et større ansvar for programvare som lages.
* Samtidig krever Scrum mer av hvert enkelt teammedlem. Det har blitt hevdet at utviklere må ha høyere faglig nivå i smidige metoder enn i plandreven utvikling. De må også ha en personlighet som passer til den intensive arbeidsmetoden, og være villige til å involvere seg i prosjektet. De må også ha anledning til å møte opp til de fastsatte møtene.
* Scrum krever samtidig at kunden har en representant som er tilgjengelig gjennom hele prosessen. I praksis er ikke dette alltid like enkelt å få til.
* Det mindre fokuset på dokumentasjon i Scrum kan gjøre vedlikeholdsarbeid vanskeligere.

Ulike teammedlemmer har ulike roller i scrum-teamet. Her har vi listet opp tre av de g hva deres oppgave er.

* Product owner: Skal være stemmen til kunden, og hele tiden passe på at utviklingen tilfredstiller kundens krav best mulig. Bør ikke kombineres med “Scrum master”
* Development team: Dette er utviklerene som gjør den faktiske jobben.
* Scrum master: Skal hindre at utviklingsgruppen blir distrahert fra andre ting, arrangere daglige møter, opprettholde Scrum-reglene.

Vi er 5 stk i vårt team. Daniel, Joanna, Stephanie, Synne og Tormod. Vi har selv listet opp våre styrker og svakheter og slik ble listen seende ut.

Tabell 1 Styrker og svakheter til gruppens medlemmer

|  |  |  |
| --- | --- | --- |
| Navn | Styrker | Svakheter |
| Synne | Jobber effektiv, flink til å holde oversikt og samarbeide og formidle | Utålmodig |
| Stephanie | Jobber effektivt | Liker å ha kontroll |
| Joanna | Vant med å jobbe i grupper, liker å få ting gjort | Utålmodig, vant til å ha all kontroll selv |
| Daniel | Systematisk  Utholdende | Detaljert |
| Tormod | Effektiv | Utsetter ting |

# Krav -påbegynt

## Generelle krav

* Både personer og grupper skal kunne ha kalendere
* Grupper skal kunne ha subgrupper
* En person skal kunne være med i flere grupper
* Alle kalendere skal lagres i en kalendertjener
* Alle ansatte skal kunne ha tilgang til kalenderne sin gjennom en kalenderklient
* Mange kalenderklienter skal kunne være koblet opp mot kalendertjeneren samtidig

## Avtaler

* Det skal være mulig å legge inn avtaler i kalendere. Disse skal inneholde
  + Tidspunkt
  + Plassering
  + Beskrivelse
  + Prioritet
  + Deltakere
* Det skal være mulig å endre informasjonen i avtaler, dvs. tidspunkt, plassering, deltakere og prioritet
  + Endringer i tidspunkt skal varsles til deltakere, med eventuell informasjon om overlapp
  + Dersom avtalt rom ikke er tilgjengelig ved endring av tidspunkt, skal systemet foreslå et annet rom. Deltakerne skal varsles om rombytte.
* Brukere skal kunne se avtalene i sin egen kalender
* Møteinnkaller skal kunne se ledige tidspunkter for møtedeltakere
* Deltakelse
  + Møtedeltaker skal kunne bekrefte eller avkrefte deltakelse.
  + Det skal kunne gis varsel om en invitasjon overlapper med en tidligere innlagt avtale, og informasjon om hvilken prioritet de aktuelle aktivitetene har.
  + Det skal være mulig å melde avbud etter bekreftelse.
  + Det skal kunne gis varsel til møteansvarlig om at deltakelse har bitt avkreftet eller endret.

## Bookingsystem

* Systemet skal støtte booking av møterom. Dette skal kunne gjøres samtidig med at en avtale opprettes.
* Systemet skal finne et passende møterom med utgangspunkt i følgende informasjon
  + Møtetidspunkt
  + Antall møtedeltakere

# Use cases

# Aktivitetsdiagram

# Klassediagram

# Testing - påbegynt

Teststrategien vår har foregått slik at enhetstestene ble gjort med white box testing. De enkleste klassene var det den som programmerte som selv testet, mens de litt større klassene (‘user’, ‘group ‘og ‘room’) er blitt testet av en som ikke selv har skrevet koden.

Flertallet av enhtstestene er blitt skrevet som J-unit tester.

Komponenttestene er blitt foretatt med blackbokstesting. I og med at vi alle har vært med på å utvikle systemet, og selv har ansvar for testingen, så er det ikke denne testfasen blitt foretatt av noe eksternt testteam.

# Riskikoanalyse -ferdig

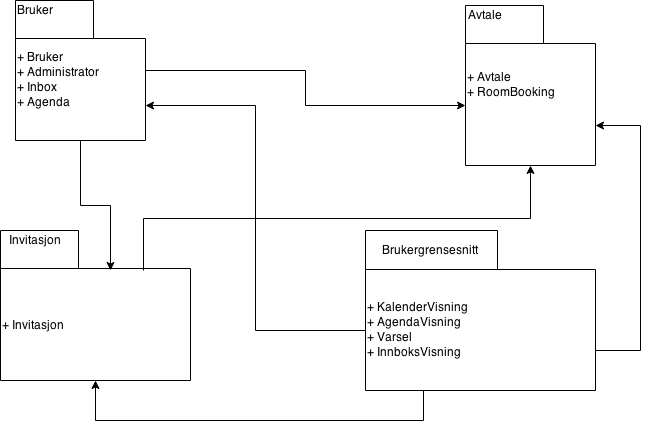
Tabell 2 – Risikoanalyse fra øving 4

|  |  |  |  |
| --- | --- | --- | --- |
| Risiko | Sannsynlighet | Konsekvens | Løsning |
| Ett medlem må pga. sykdom eller lignende være borte fra prosjektet i store deler av tiden | Liten til middels | Noe av arbeidet kan bli utsatt. Flere av de andre teammedlemmene kan komme til å måtte arbeide mer for å klare å bli ferdig. I verste fall har personen som blir borte kompetanse som de andre ikke har, og det kan bli umulig å gjøre oppgaven før | Scrum master fordeler arbeidet til noen andre dersom det er mulig. Evt. må vi få hjelp fra eksterne personer til å løse oppgaven. |
| Programmet, og backup av dette blir slettet, eller korrupt. | Liten | Vi taper mye tid, og må kanskje droppe noen av funksjonalitetene for å klare å levere innen fristen. Hvis dette skulle skje vil det gå litt fortere å gjøre det på nytt, siden vi da vet hvordan vi skal gjøre det. | Man starter på nytt, eller fra siste versjon. |
| Det viser seg at vi ikke kan løse oppgaven på den fremgangsmøten vi har valgt. | Liten til middels | Endel av arbeidet vi har gjort må forkastes, og vi taper endel tid. | Man skjekker hva som mangler for at oppgaven kan løses, og legger til disse, evt. bytter det ut med det som viser seg å være unødvendige deler |
| Databaseoppsettet vi laget i første omgang viser seg å være for lite kompleks for å løse oppgaven, eller unødvendig kompleks for å løse oppgaven. | Middels | Dette vil gå ut over de funksjonelle kravene, og det vil ta endel tid å endre på dette. | Lage en ny database med de nødvendige relasjonstabellene, og oppdatere koden til denne. |
| Vi underestimerer størrelse og kompleksitet på programvare, og hvor lang tid vi bruker på det | Middels | Vi bruker mer tid enn vi antok på enkelte oppgaver, som igjen vil gå utover andre oppgaver som er avhengige. Dette kan i verste fall gjøre at vi ikke blir ferdige. | Vi må kanskje planlegge å legge ned ekstra tid i arbeidet, og vi bør se på tidsplanen for når oppgavene skal bli ferdige og mest sannsynlig gjøre endringer. Kanskje må vi droppe mindre viktig funksjonalitet. |
| Ikke god nok opplæring/kompetanse i programmer/ språk/ verktøy vi bruker | Middels til høy | Arbeidet vil ta mer tid fordi vi vil gjøre endel feil. Disse feilene kan være vanskelige å oppdage og rette opp. | Vi spør noen som kan det bedre enn oss om hjelp, og justerer litt på forventningene våre. Kanskje vi også må kutte ut mindre viktig funksjonalitet for å klare å bli ferdige med de viktigste. |
| Folk yter ikke så mye som forventet | Middels | Vi vil merke at noen oppgaver ikke blir ferdige i tide, og det vil utsette andre oppgaver. Dette vil føre til ekstra belastning på de andre teammedlemmene. | Scrum master setter seg ned og snakker med aktulle person. Hvis det ikke kommer noe ut av dette må arbeidet delegeres til en annen person. |
| Medlemmer av teamet blir uenige eller begynner å krangle | Middels | Dårlig arbeidsmiljø og samarbeid innad i teamet som vil gjøre til at arbeidet tar lengre tid. Det kan også bli lite samsvar mellom de forskjellige delene av systemene, som kan gjøre at det ikke fungerer. | Scrum master planlegger et møte for å løse konfliktene. Hvis den ikke blir løst gjennom dialog mellom de involverte medlemmene bestemmer Scrum master hva som bør gjøres. |

# Change management - påbegynt

## Pakker

Pakkene er blitt endret i forhold til den planen vi la fram i øving 5. Denne endringen gikk vi for tidlig i prosjektet da vi innså at pakkene vi hadde laget ikke var hensiktsmessig med tanke på hvordan vi hadde lyst til å jobbe videre. Lagt ved er originalt pakkediagram og det modifiserte (og gyldige) pakkediagrammet.



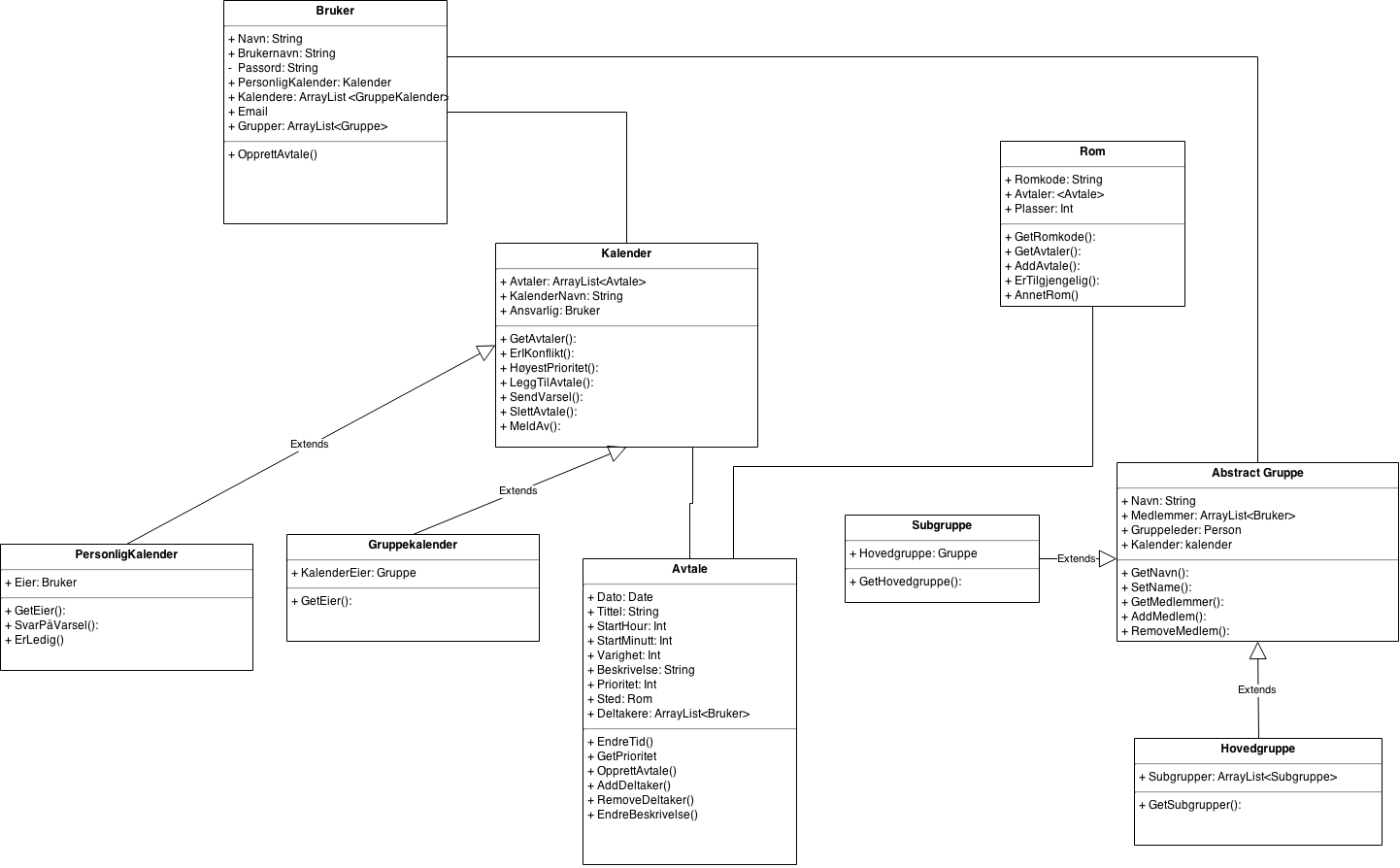
Figur 1 – Originalt pakkediagram, tatt fra øving 5



Figur 2 – Plassholder for gjeldende pakkediagram

## Klassene

Først og fremst har vi gått fra å i øvingene skrive det meste av klasser og metoder på norsk til å konsekvent, under prosjektet, bruke engelsk. Utenom den åpenbare endringene grunnet oversettelse er det også en del metoder og klasser som har blitt endret. Som eksempel kan man jo nevne at attributtet ‘name’ under ‘user’ -klassen er blitt delt opp i ‘first name’ og ‘last name’ for å gjøre det enklere å samhandle med databasen.



Figur 3 – Originalt klassediagram, tatt fra øving 5

# Tidsestimat og tid brukt

# Gjenstående arbeid

# Refleksjon om prosjektet - påbegynt

Vi fikk en litt tøff start på fellesprosjektet ved at vi fikk en utrolig kjedelig beskjed. Daniel var på fjellet på skitur helgen før prosjektstart, og var så utrolig uheldig å pådra seg brudd i kragebeinet. Han måtte vente på operasjon, og visste ikke når han kom tilbake. To uker ut i prosjektperioden hadde vi fremdeles ikke hørt noe fra han. Dette hadde vi tatt høyde for i risikoanalysen, hvor vi hadde skrevet at dersom noen ble borte ville vi løse dette ved å delegere oppgaver videre til et annet gruppemedlem, og eventuelt kutte ut mindre viktig funksjonalitet. Dette har til en viss grad løst seg selv, på den måten at vi uansett måtte prioritere de taskene som vi hadde prioritert høyest i planleggingsfasen.

Til tross for at Daniel var borte, var vi veldig ivrige i begynnelsen av prosjektperioden. Vi tenkte at det ikke gjorde noe at vi begynte på forskjellige deler, for vi hadde jo planlagt dette tidligere. Dette gjorde at vi i senere tid måtte endre en del på det noen gjorde i begynnelsen, fordi vi hadde litt forskjellige ideer om hvordan pakkene og klassene i systemet skulle kommunisere. I tillegg hadde vi litt problemer i begynnelsen med at vi brukte mye tid på ting som ikke tok oss videre i arbeidet. Et eksempel på dette var problemer med Git, som gjorde at vi til tider kunne sitte lenge å prøve å finne ut av hva som var feil, og dermed uten mulighet til å komme videre i arbeidet.

Vi har vært flinke til å være så effektive som mulig når vi har vært samlet for å jobbe, men har hatt litt problemer med at ikke alle på gruppen møter opp presis. Dette har vært frustrerende, men man har sagt hva man har ment og kommet i gang med jobbingen.

Kunnskapene våre rundt programmering og databaser har vært varierende, og de som har sittet med mye kunnskap har vært flinke til å være tålmodige og lære videre sine kunnskaper slik at alle på gruppen har følt at de kan bidra til å bringe prosjektet framover. Også på de områdene der man ikke nødvendigvis er sterkest.

Planlegging av sprinter har vært litt så som så, og måten vi har fordelt oppgaven på kunne ha vært mye bedre. Det har vært stunder med mye venting fordi en oppgave må gjøres ferdig før man kan komme videre med noe annet. At en person deler tiden sin mellom flere klasser, men noen ikke har noe å gjøre er dårlig planlagt fra vår side. Bedre bruk av gant-diagram kunne f.eks. vært løsningen på dette.

# Kommentarer til gjennomføring av faget

­fra oppgaveteksten: Hva synes dere om øvingsopplegget og fellesprosjektet (bra/dårlig)? Hva synes dere om gjennomføringen av forelesningene (øvingsforelesningene og forelesningene). Hva var bra og hva kan vi gjøre anerledes?

Appendix

# Timeliste

# Møtereferater - påbegynt

## Referat 23.02

**Dato:** 23.02

**Tid:** 12.15 - 14.00

**Sted:** R80

**Ordstyrer:** Synne

**Referent:** Stephanie

**Tilstede:** Synne, tormod, Stephanie

**Ikke tilstede:** Daniel, grunnet operasjon

|  |  |
| --- | --- |
| Sak (Kopiert saksliste) | Referat |
| Valg av ordstyrer og referent | Synne velges til ordstyrer og Stephanie som referent. |
| Felles arbeidstid? | Tirsdager, 8 - 12 (Tormod jobber fra 12) Torsdager, 10-12  Fredag: 10 - 14  Trengs det mer tid til fellesarbeid skal dette orinært legge til PU og D&D forelesning/øvingstimer slik de er satt opp på timeplanen. |
| Daniel er syk | God bedring. Vi får fordele arbeidsmengden mellom de som er igjen, og håpe at han er tilbake så fort som mulig. |
| Logging av timer | Bruker skjemaet som ligger på drive. Fylles ut fortløpende på ukebasis. Telling starter fra prosjektuke1 (uke 9) |
| Sprintmeetings med TT | Førstkommende mte er satt til 24.02 kl 11.45 - 12.15 på P15 Prøver å få booket møtetid etter kl 10 på tirsdager, gjerne slik at gruppen rekker å møtes i forkant. |
| Bruke trello? | Ser ikke at dette er nødvendig på dette tidspunktet. Men kan ta det opp til vurdering igjen om situasjonen/behovet endrer seg. |
| Installasjon av ting? | De som kan hjelper de andre. GIT og mysql bør være i boks til morgendagens møte. |

Til neste gang vi møtes:

|  |  |
| --- | --- |
| Hva | Hvem |
| FInne en ordrning for å registrere hvor lang tid som brukes pr oppgave. | Tormod |
| Sende ut innkalling til sprint meeting # 1 | Stephanie |
| Installere programmer som kreves; GIT, MYSWL | Synne, Joanna |

Eventuelt

## Referat 24.02

**Dato:** 24.02.

**Tid:** 11.45-12.15

**Sted:** P15

**Ordstyrer:** Stephanie

**Referent:** Synne

|  |  |
| --- | --- |
| Hva | Referat |
| Valg av ordstyrer og referent | Stephanie velges som ordstyrer og Synne som referent |
| Gjennomgang av sprint #1 | Stephanie forklarer hva vi har tenkt å gjøre. Vi har blitt enige om å bruke substantivmetoden til å sette opp database. Kommentarer: vi bør begynne å implementere de funksjonelle kravene, og begynne å skrive rapport. Hvis endringer blir gjort må dette dokumenteres, og grunnene må forklaring. Vi må hele tiden huske timelogging. Taskene i sprinten skal utføres utifra prioritet, og det som ikke blir gjort kan bli dyttet til neste sprint. Det er lurt å ha et lite oppdateringsmøte i begynnelsen hver gang vi møtes, i tillegg til møte på slutten av hver sprint.  Til neste gang bør vi vise hvor lang man har kommet - man bør ha kommet godt i gang med programmering.  Vi bytter tid for møtene til 10.45 på tirsdager. |
| Planlagt utførelse av sprinten |  |

Hva er gjort i dag:

Startet i implementere systemet i Eclipse.

Klasser:

Avtale (ikke ferdig), DeltagerListe, Rom, RomOversikt, Tid, Bruker

## Referat 27.02

**Dato:** 27.02

**Tid:** 10.15

**Sted:** R80

**Ordstyrer:** Synne

**Referent:**Stephanie

**Tilstede:** Synne, Tormod, Stephanie

**Ikke tilstede:** Daniel, Joanna

|  |  |
| --- | --- |
| Sak (Kopiert saksliste) | Referat |
| Booking av rom. hvem tar ansvar? | Viktig at vi har litt struktur på hvor vi er.  Må passe på at noen booker.  Hvis dette viser seg å være vanskelig satser vi på r80.  Oppsatt møtetid uke 2:  Mandag: 12 - 14  Tirsdag: 8 - 12  Fredag: 10 - 14 |
| Planlegging av sprintene om igjen. | Gjort |
| Fordeling av oppgaver som skal jobbes med i dag. | Tormod: connection pooling  Synne: fikse klasser og pakker så det passer til tidligere laget diagrammer  Stephanie: rapporten |
| Endringer som blir gjort må registreres | Går over prosjektet til slutt og oppdatere diagrammer ++ |
| Respekter andres tid! | Tatt på facebookchaten da ikke alle var på møtet i dag. |

Til neste gang vi møtes:

Hva må gjøres og hvem skal gjøre det?

## Referat 03.03

**Dato:** 03.03

**Tid:** 1.45

**Sted:** p15

**Ordstyrer:** Tormod

**Referent:** Stephanie

|  |  |
| --- | --- |
| Sak (Kopiert saksliste) | Referat |
| Vise fram databasen, og forklare hva vi tenkte da vi modellerte den | **til neste gang. ha avtale på plass. mulig å booke rom. sjekke at man ikke godtar to overlappende avtaler.** |
| Vise det vi har gjort i klassene user, time og room. (Hva vi ikke har fått gjort, og hva som skal gjøres videre) | Room er en viktig klasse som vi må få orden på. understreker at denne henger litt etter pga sykdom og at vi akkurat har lært å la klasser kommunisere med databaser. |
| TT | Til neste gang vil hun ha appointment klassen klar. Viktig at man ikke har overlappende avtaler |
|  |  |

Til neste gang vi møtes:

Hva må gjøres og hvem skal gjøre det?

Eventuelt