Institutt for datateknologi og informatikk, NTNU

SU1

# Dokumentasjon

Nils Tesdal, 2017-2019

NTNU
Kunnskap for en bedre verden

# Liker du å dokumentere?

Hva og hvor mye synes du vi bør dokumentere?


NTNU
Kunnskap for en bedre verden

# Hvorfor dokumenterer vi?

Kommunikasjon i et prosjekt?

- “Documentation is the least effective means of communication” (http://www.agilemodeling.com/essays/agileDocumentation.htm)

Bevaring/overføring av kunnskap?

- Hvis et annet team skal ta over prosjektet
- Hvis et nytt team-medlem kommer inn
- Hvis noen skal bruke/installere/videreutvikle noe vi har laget

Kontrakt?

- Vi må konkretisere hva som skal gjøres i et prosjekt (krav) for å sette en pris

# Hvordan dokumenterer vi (smidig)?

- Kortfattet og presist
- Modeller fremfor tekst
- Visuelt fremfor tekst
- Ryddig
- Må være lett tilgjengelig - WIKI fremfor Word-dokument.

Det er ingenting som er kjedeligere å lese enn omfattende (og dårlig) dokumentasjon. Derfor blir omfattende (og dårlig) dokumentasjon sjelden lest og derfor er det (nesten) bortkastet å lage det!!

# Hva dokumenterer vi?

**Visjon**

- hva er målet med prosjektet?
- hvem er interessentene og hva forventer de?

**Krav**

- mer detaljerte beskrivelser av hva vi skal lage
- fungerer gjerne som kontrakt mellom oppdragsgiver og utviklere.

**System**

- hvilke viktige designbeslutninger har vi tatt?
- hvordan er systemet bygd opp på et overordnet nivå?
- API'er og annet som andre trenger

# Hva dokumenterer vi (2)?

- **API'er**. Dette er kanskje det viktigste…
  - JavaDoc. Ikke viktig at ALL javakode dokumenteres, men alt som skal brukes som bibliotek av andre enn de som har utviklet dem MÅ dokumenters godt.
  - REST-grensesnitt. (f.eks. Swagger. SU2!)
- **Installasjonsmanual**. Hvis det vi lager må installeres og kjøres av andre enn oss selv må alle manuelle trinn som ikke forklares i selve installasjons-skriptet dokumenteres.
- **Brukermanual**? Ideelt er alt så selvforklarende at man ikke trenger det, men det får vi sjelden til. Så derfor bruker vi:
  - Dokument? (litt gammeldags)
  - Websider (evt WIKI)
  - Context-aware hjelp i appliklasjonen!

# Dokumentere kode?

Vær kritisk også her – unngå selvfølgeligheter men dokumenter det som ikke er åpenbart.

```
/* The public class Attendant was implemented
 * to model an attendant.
 * Attendant implements the interface
 * Runnable.
 *
 * List of methods:
 *       Attendant
 *       login
 *       run
 */
public class Attendant implements Runnable{
    private Socket s; // This is a class attribute;

    /* This public method is the class constructor.
       * @param (Socket) s
       *
       */
    public Attendant(Socket s)
    {
         this.s = s;
    }
```

NTNU
Kunnskap for en bedre verden

# Må vi dokumentere?

- Vær kritisk til hva som skal dokumenteres.
- Lean sier "**eliminate waste**" - så ikke lag dokumenter ingen (eller få) leser.
- Jeg har opplevd som utvikler å få kjeft for å ikke dokumentere. Men når jeg deretter dokumenterer så leser de det ikke likevel. Alle har dårlig tid.
- Dokumentasjon er dyrt – både ift produksjon og vedlikehold.
- Avstem tidlig i prosjektet hva som forventes av dokumentasjon (ikke i dette faget – her gjør dere som dere blir fortalt…)
- Skriv den dokumentasjonen dere har blitt enige om på en god, kortfattet, presis, visuell måte og gjør den lett tilgjengelig!

NTNU
Kunnskap for en bedre verden

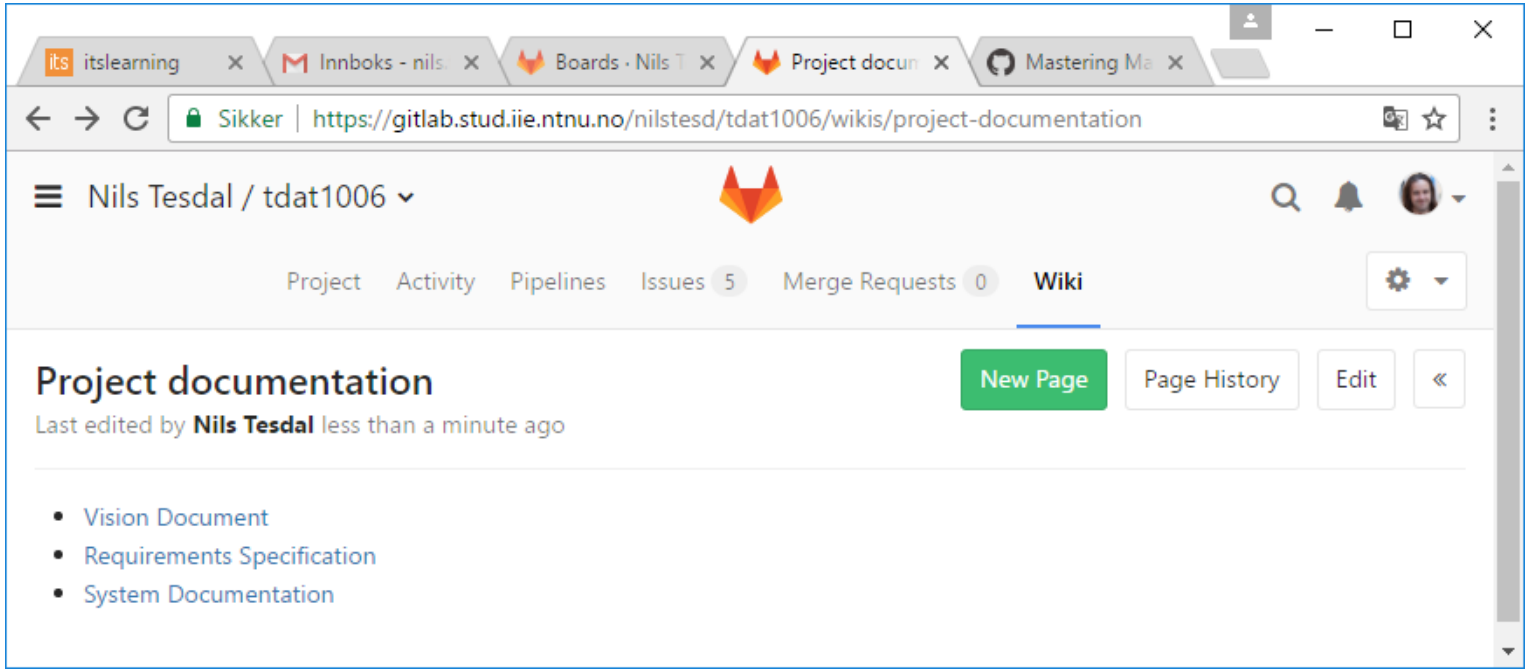
# Kunnskaps- og delingskultur

- Mye kunnskap kan også deles og bevares gjennom en god kunnskaps- og delingskultur.
- Ofte er det slik at men BARE VET hvordan ting gjøres i en bedrift og det må derfor ikke nødvendigvis dokumenteres.
- Par(mob)-programmering, designmøter, presentasjoner
- Deltakelse på konferanser
- Faglige diskusjoner
- Ikke bli en code-monkey

NTNU
Kunnskap for en bedre verden

# SU1 Prosjekt - Dokumentasjon

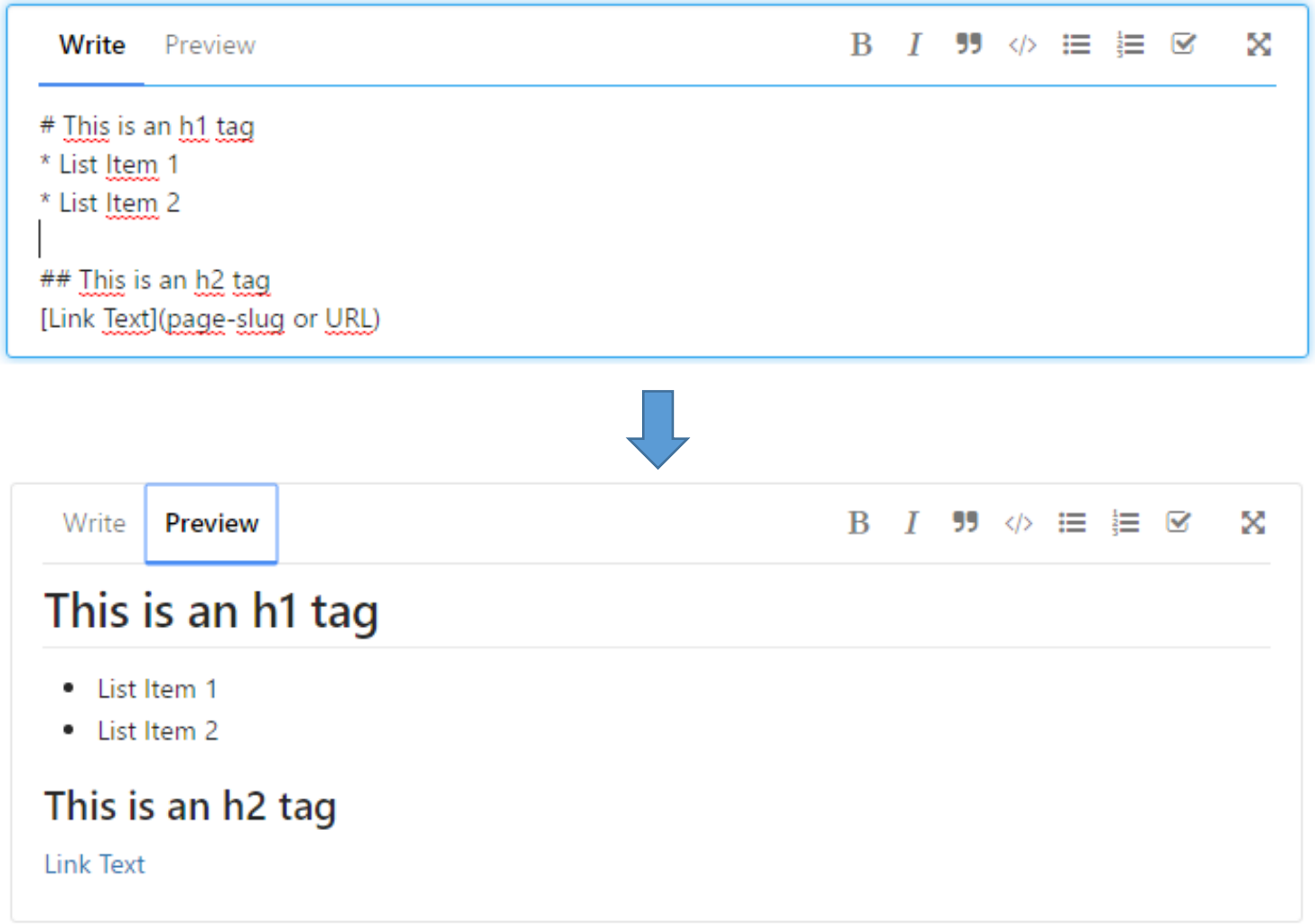
- Vi skal lage en WIKI på GitLab
- Vi bruker «Markdown»-syntaks:
  https://guides.github.com/features/mastering-markdown/






NTNU
Kunnskap for en bedre verden

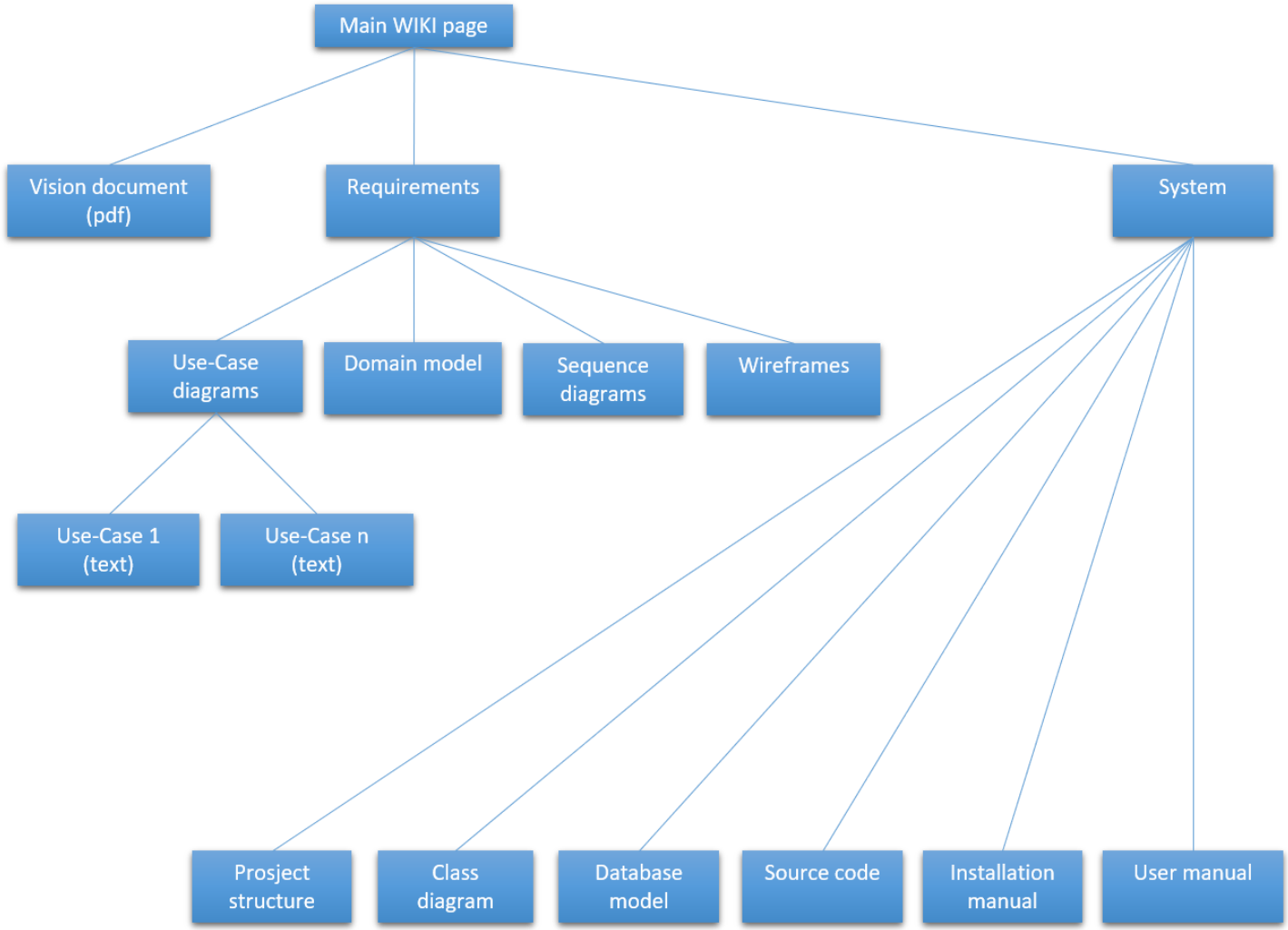
# Prosjekt WIKI - Markdown






NTNU
Kunnskap for en bedre verden

# Prosjekt-WIKI: Hva skal vi dokumentere




NTNU
Kunnskap for en bedre verden

# Hva er et visjonsdokument?

- Et dokument som ferdigstilles før man begynner med utviklingen
- Har opprinnelse fra UP (Unified Process)
- Dokumentet fokuserer på hva vi skal lage:
  - Hvilket behov man har.
  - Hva målet med utviklingen er.
- Brukes for at den som skal gjøre jobben (prosjektgruppen) og den som betaler for jobben (oppdragsgiver) skal få en samstemt oppfatning av hva dette prosjektet dreier seg om og hva som kommer til å skje fremover.
- Kan (kanskje) ses på som en kontrakt eller en avtale mellom oppdragsgiver og den som skal gjøre jobben, og må da sørge for at det inneholder de opplysningene som er viktige for at begge parter er trygge på hva de har avtalt.

# Omfang

Kan variere veldig i detaljeringsgrad og omfang ut ifra

- Hvor detaljerte krav oppdragsgiver har
- Hvor mye man ønsker å bestemme tidlig

Hvis oppdragsgiver krav til kostnader, leveringsdato og funksjonalitet er veldig rigide vil omfanget nødvendigvis bli større.

Detaljerte krav beskrives helst senere i prosessen!

# Innholdet i visjonsdokumentet

- Hva er bakgrunnen for at prosjektet er startet?
- Hvem er interessentene for prosjektet og hvilke suksesskriterier har disse?
- Hva er målet med dette prosjektet (mer enn hva vi konkret skal lage)?
- Hvilke rammebetingelser er prosjektet underlagt?
- Hvilke risikoer er involvert?
- Hvor mye vil det koste (vi vil helst slippe å ta stilling til dette…)?
- Hvilke ikke-funksjonelle krav må tilfredsstilles?
- Hvilke krav stilles til dokumentasjon?

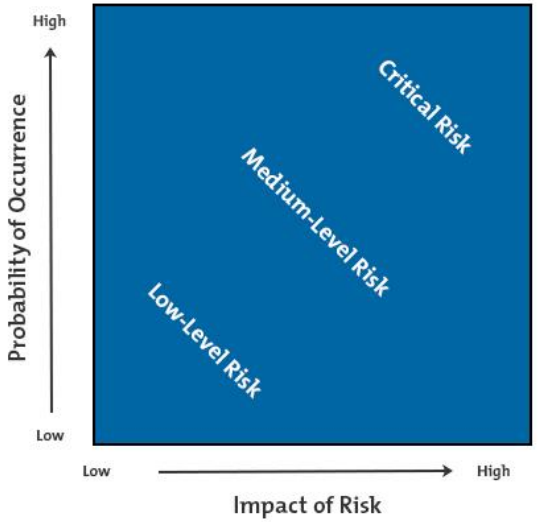
# Vi bruker en mal

- Denne bruker vi som utgangspunkt og som hjelpemiddel. BlackBoard: *Undervisning > Su11 – Prosjekt > Templates > Software Vision Documents*
- Vi tilpasser så dette i forhold prosjektet vi jobber i.
- Dette betyr at vi forenkler og varierer detaljeringsgrad i hvert avsnitt. Vi tar bort avsnitt som ikke er relevante for prosjektet.
- Ikke fyll inn avsnitt bare for å gjøre det hvis dere ikke har noe relevant å skrive. Det er støyende og mot dokumentets hensikt.
- Jeg har også lagt ut et eksempel på et enkelt visjonsdokument (på norsk) på itslearning som jeg skrev for Scrum-prosjektet for SU2 i 2015.
  - *… > Templates > Example vision document (scrum-prosjekt 2015)*
  - Merk at dette dokumentet mangler risikoanalyse som dere skal ta med

# Enkel risikoanalyse

- Risikoanalyse er et fagfelt i seg selv.
- For visjonsdokumentet skal vi bare gjøre en enkel øvelse.
- Vi lister opp mulige problemer, og setter opp en matrise med en score på:
  - Sannsynlighet (probability) for at det inntreffer
  - Konsekvens (impact) hvis det inntreffer
- Vi multipliserer disse sammen og får en score (significance). Høy score betyr at vi må prøve å adressere problemet tidlig.

| Risk Short Description | Impact | Probability | Significance |
|---|---|---|---|
| Team not staffed in time | 4 | 5 | 20 |
| Language misunderstandings | 3 | 5 | 15 |
| Team not experienced | 3 | 5 | 15 |
| Too many conflicting interests | 4 | 2 | 8 |
| Project manager overwhelmed | 4 | 2 | 8 |
| Available resources | 2 | 2 | 4 |
| Testers not available | 2 | 2 | 4 |






NTNU
Kunnskap for en bedre verden

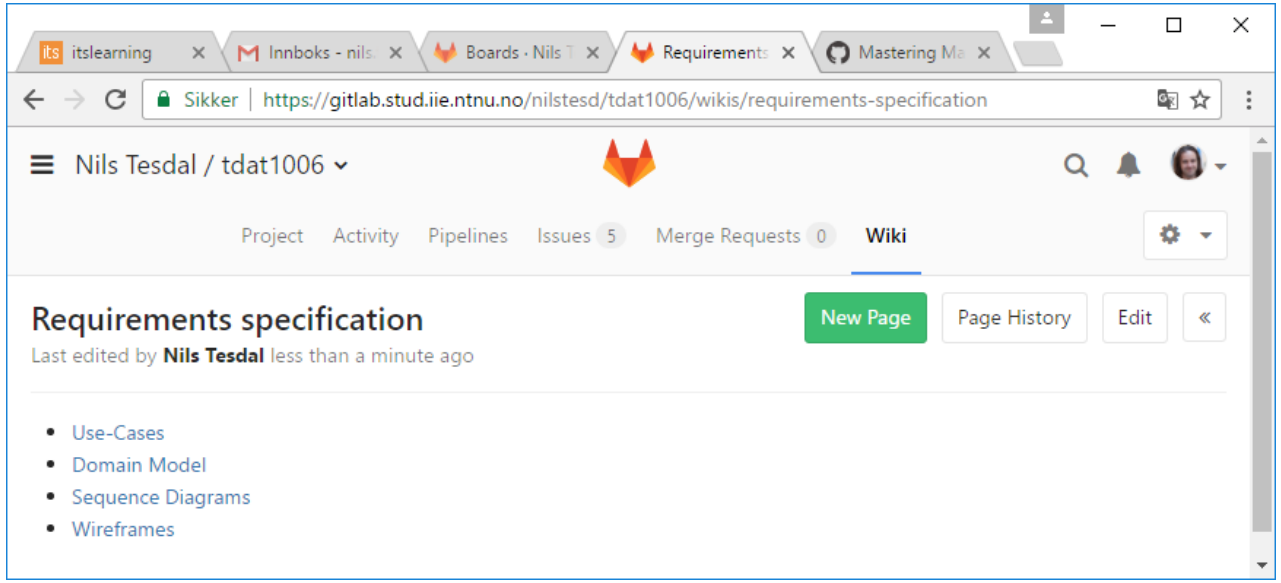
# Risk poker

- I smidige prosjekter er det vanlig å utføre såkalt risk-poker.
- Det betyr i praksis at alle team-medlemmene presenterer et tall (legger ned et kort) ***samtidig***, slik at valget ikke blir påvirket av andre.
- Det finnes app'er for Scrum planning poker som også kan brukes.

# Dokumentasjon av krav

I prosjektet skal vi lage:

- Use case diagram med tekstlige beskrivelser
- Domenemodell
- Sekvensdiagrammer for 2 sentrale use case (med domeneobjekter)
- Wireframes for de viktigste skjermbildene i systemet



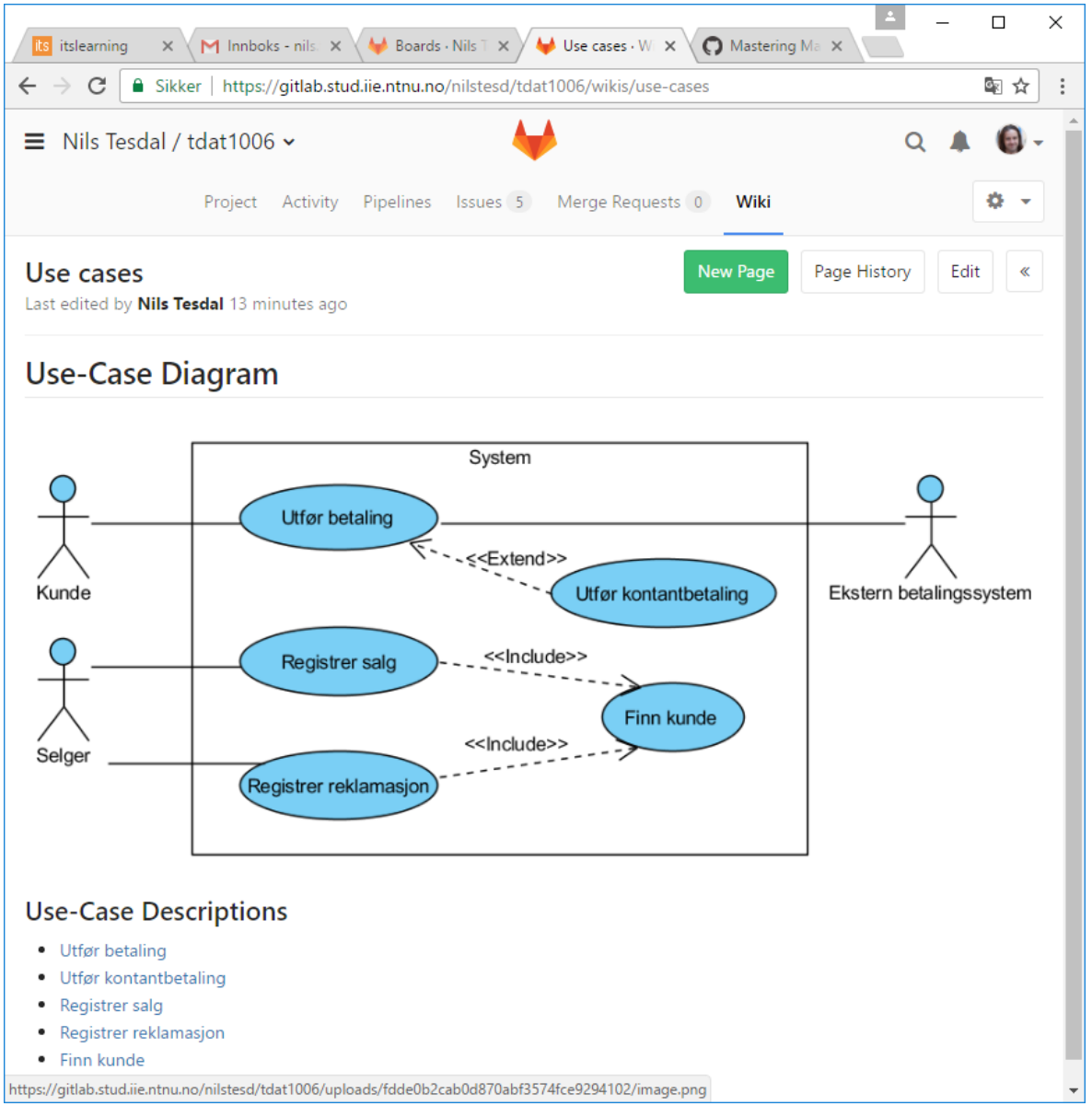



NTNU
Kunnskap for en bedre verden

# Use-Case

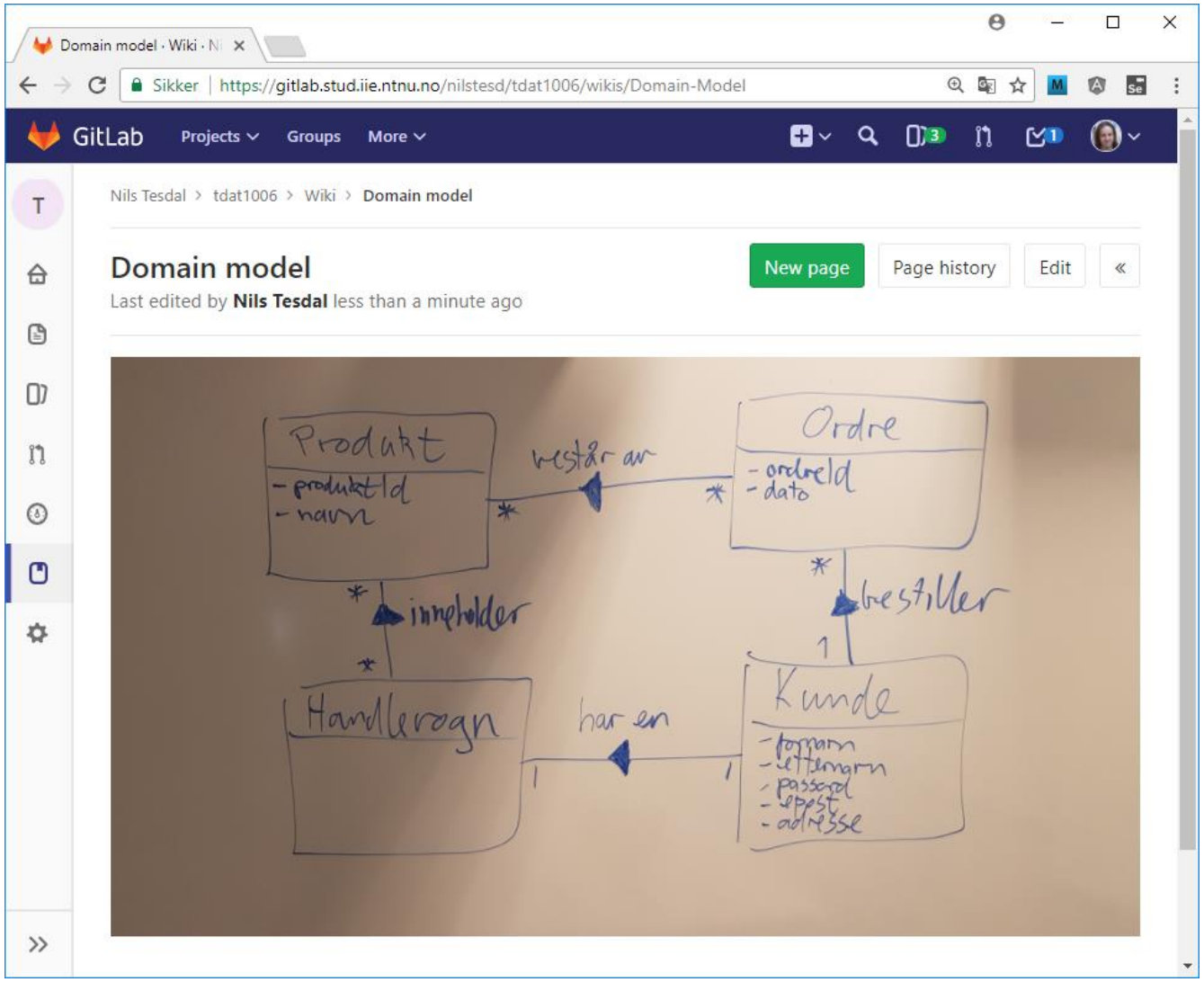
Vi lager ett diagram på en WIKI-side og linker til en tekstlig beskrivelse for hvert av user-casene.

(Beklager norsk språk i modellene. Dere skal bruke engelsk i prosjektet!)



# Domenemodell

Vi lager en domenemodell som beskriver problem-domenet i prosjektet:

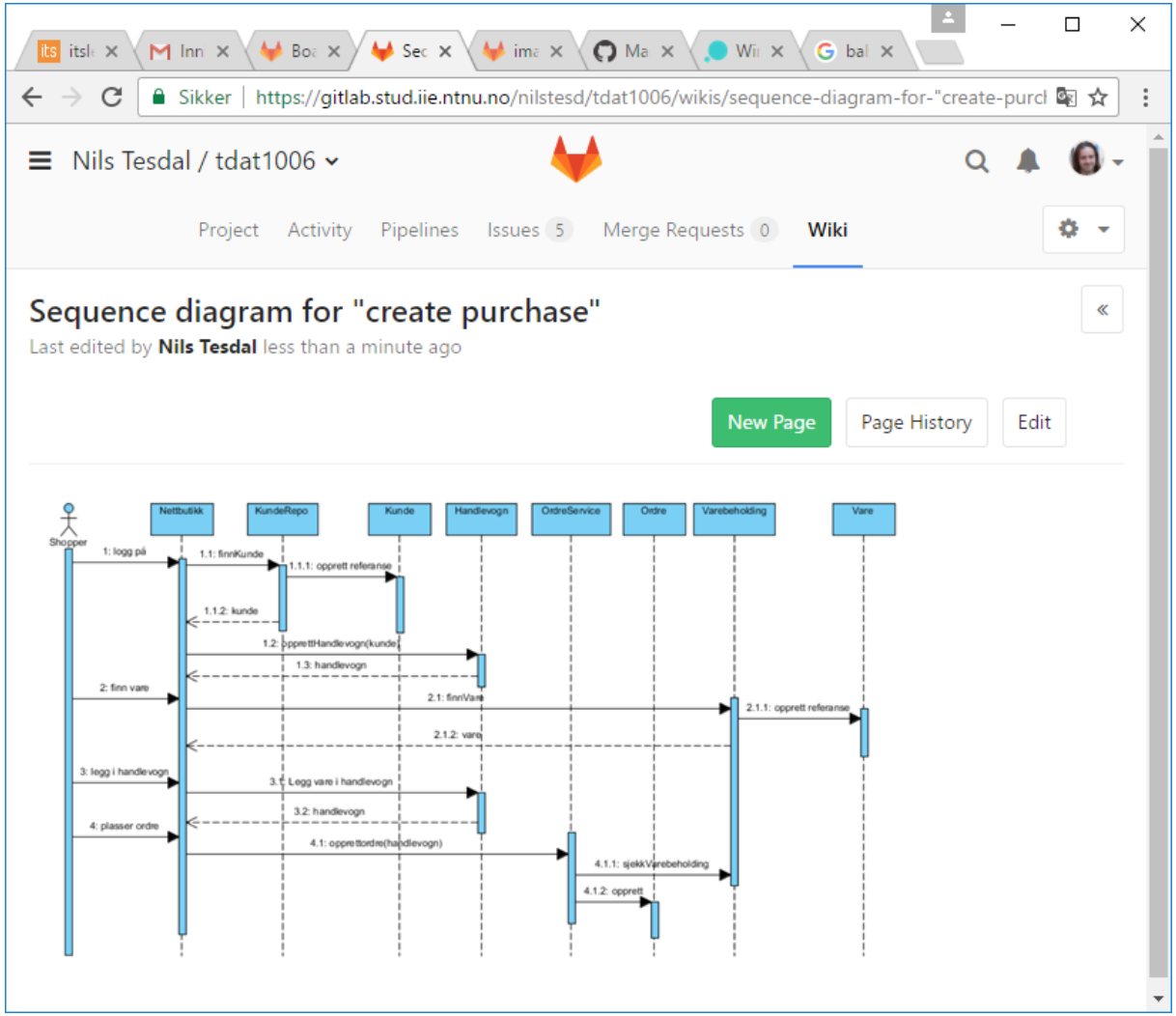





NTNU
Kunnskap for en bedre verden

# Sekvensdiagrammer

Vi velger ut to sentrale use-case og lager et sekvensdiagram hvor entitetene fra domenemodellen helst er med:



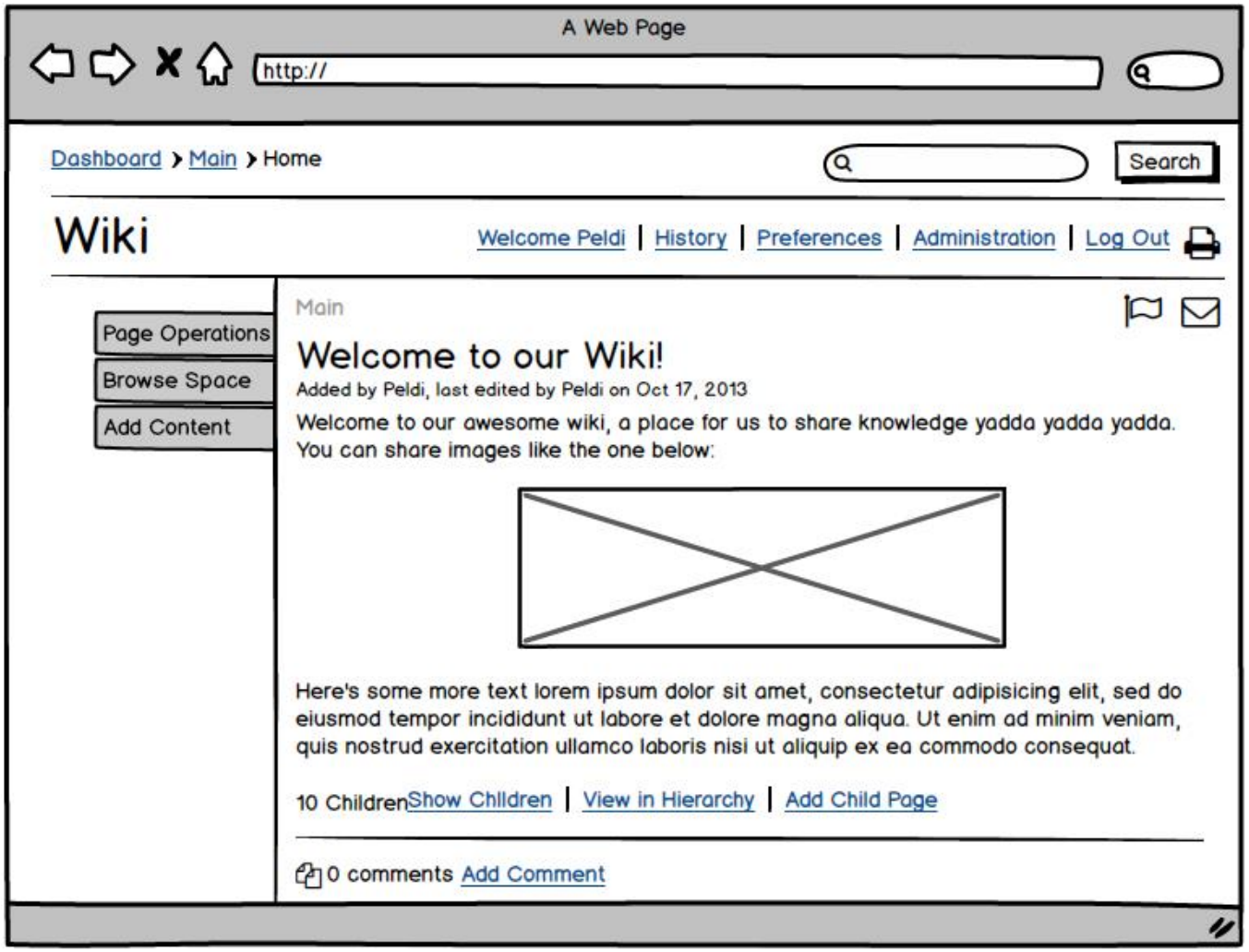



NTNU
Kunnskap for en bedre verden

# Wireframes

- Dere skal lage wireframes for de viktigste skjermbildene i applikasjonen med Balsamiq.
- Dette gjør vi for å visualisere den ferdige løsningen raskest mulig, slik at alle raskt kan bli enige om hva som skal lages.
- Verktøyet er såpass enkelt å bruke at dere bruker minimalt med tid på det tekniske så dere heller kan bruke tiden på å diskutere hva som skal lages.
- Bruk «linking» fra knapper til nye skjermbilder. På den måten kan dere lage en veldig enkel klikkbar prototype som dere faktisk kan bruke for å teste brukergrensesnittet på en person som ikke er en del av prosjektet.
- Eksporter resultatet til en pdf og legg denne inn på WIKI'en.
- Balsamiq lisens ligger under:
  *Undervisning > Su11 – Prosjekt > Balsamiq lisens*

# Wireframes - Example



# Wireframes – Brukertest

- Test wireframene ved å la en bruker (utenfor prosjektet) utføre noen oppgaver som dere spesifiserer.
- Observer og noter hva de gjør (feil) og gjør endringer i brukergrensesnittet hvis dere opplever at de har problemer med noe.
- Spør dem gjerne også om tips til forbedringer, men legg større vekt på det dere observerer. Testbrukeren deres er sannsynligvis ikke GUI-ekspert.

# Systemdokumentasjon

**Prosjektstruktur**: Beskriv hvilke biblioteker og rammeverk dere bruker og forklar fil- og katalogstrukturen for prosjektet; kildekode, pakker og hvordan disse er organisert.

**Klassediagram**: Lag et overordnet klassediagram som viser de viktigste (5-10) klassene involvert i ett (sentralt) Use-Case.

**Databasemodell**: Vis ER-modellen for databasen.

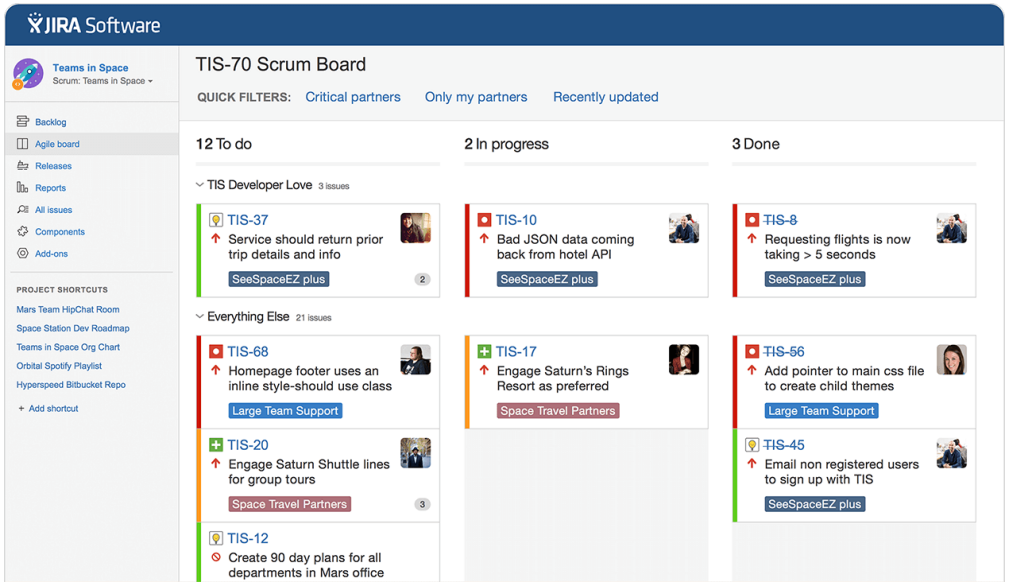
**Kildekode**: JavaDoc.

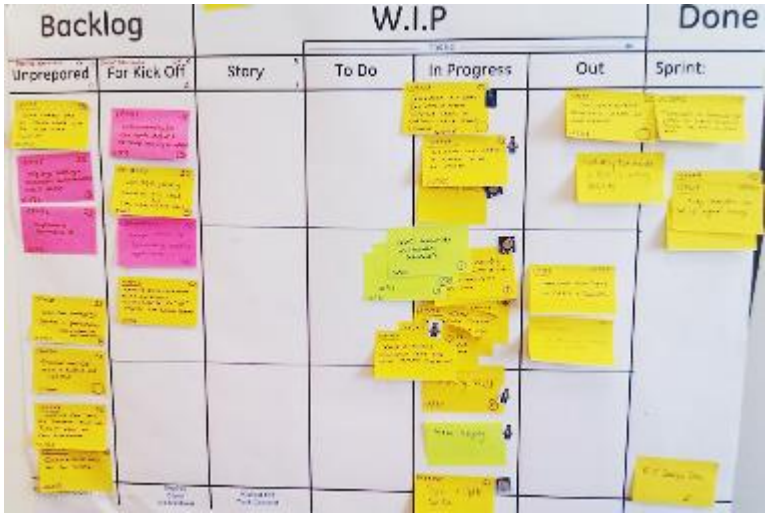
**Installasjon og kjøring**: Beskriv trinnvis hvordan løsningen installeres og kjøres og hvilken konfigurasjon som må utføres.

**Brukermanual**: Lag en enkel brukermanual på WIKI'en eller forklar hvordan context-aware hjelp og annen hjelpefunksjonalitet er inkludert i applikasjonen. Link gjerne til WIKI'en fra applikasjonen.

# Dokumentere fremdrift

- Vi visualiserer fremdrift på fysiske tavler, eller virtuelt med Trello, Jira, GitLab eller GitHub.
- For å dokumentere fremdrift kan vi ta snapshots av tavla periodevis (f.eks. hver uke) og legge ved bilder i hovedrapport.



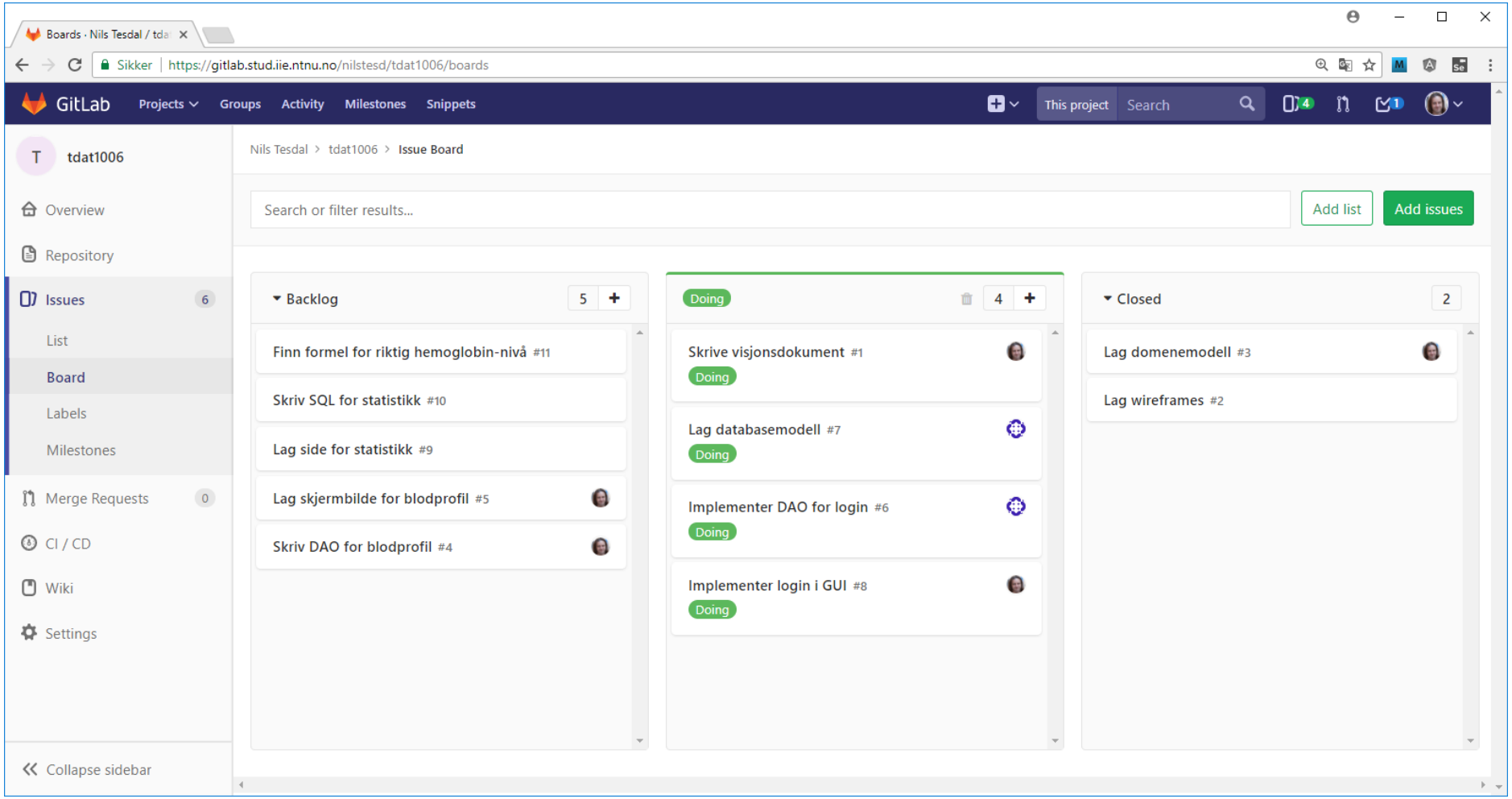



NTNU
Kunnskap for en bedre verden

# Backlog

- En backlog er en liste med oppgaver som må utføres; en todo-liste.
- En ***product backlog*** er en slik liste for ett prosjekt.
- I *Scrum* bruker vi et ***Scrum-board***, i Kanban et ***Kanban-board***.
- Vi skal bruke et generisk ***issue-board*** på *GitLab* med tre kolonner; ***todo***, ***doing*** og ***done***
- Vi deler opp et Use-Case opp i flere håndterbare oppgaver
- Vi kan også lage oppgaver relatert til dokumentasjon, kunnskapsheving eller andre ting som vil ta tid i løpet av prosjektet.
- Lag oppgaver med en enkel tittel som forteller hva som skal gjøres.
- Legg til bilde på brukeren din slik at dere kan vise tydelig hvem som har fått tildelt en oppgave
- Dra oppgavene mellom de forskjellige kolonnene etter hvert som dere starter eller ferdigstiller oppgaver.

# Issue Board






NTNU
Kunnskap for en bedre verden

# JavaDoc

```java
/**
 * Repository for the MyEntity-entity
 *
 * @author nilstes
 */
public class MyEntityRepo {

    /**
     * Get object with given id
     *
     * @param id   the entity id
     * @return     an instance of MyEntity
     */
    public MyEntity getMyEntity(String id) {
```

Genererer HTML basert på taggene. IntelliJ: Tools->Generate Javadoc