Forprosjektrapport

Monitorering av produksjonsløyper ved Nasjonalbiblioteket

- Project BAKE

Utarbeidet av:

Einar Wågan

Kristian Akerhei

Studium:

Informasjonssystemer

Innlevert:

26.05.2015

Forord

Vi er to studenter som studerer Informasjonssystemer ved Høgskolen i Nesna. Vi er inne i 6.semester og skal nå skrive vår bacheloroppgave. Einar er født i 1991 og tidligere har han vært lærling i IKT-Servicefag ved Brønnøysundregistrene og har også fagbrev derfra. Einar jobber til daglig 40% på DBU-Drift ved Nasjonalbiblioteket. Kristian er født i 1986 og har et årskurs i Business og IT fra UiN, og har tidligere jobbet på driftsavdelingen i Finnmark fylkeskommune. Kristian jobber også 40% på DBU-Drift ved Nasjonalbiblioteket. Begge har hatt interesser innen data og IT siden tidlig ungdom, og yrkesvalg innen dette fagfeltet har alltid vært ønskelig.

Sammendrag

I denne rapporten kan man lese mer om hvordan vi skal utføre bachelorprosjektet vårt i samarbeid med Nasjonalbiblioteket (NB). De ønsket en web-applikasjon for overvåkning av produksjonsløypene, og vår oppgave blir å designe og utvikle dette. De har en lignende løsning fra før, men den er både treg, består av gammel teknologi og er tidkrevende å vedlikeholde. Vi har funnet ut hvilke teknologier vi skal bruke og hva som må være på plass før vi kan begi oss ut på oppgaven. Utviklingsmetodikken vi har valgt å bruke er en blanding av Scrum og XP fordi det hjelper oss til å få struktur og produktivitet i utviklingen av applikasjonen.

I tillegg har vi oppdaget at nå-situasjonens loggformat ikke er helt kompatibel med en slik type løsning. For prosjektet har dette ingen betydning da vi kan lage systemet, selv om funksjonalitet vil mangle inntil jobben med å endre logstandard er utført. Prosjektet i sin helhet er i aller høyeste grad utførbart.

Innhold

1 Innledning

1.1 Prosjektets forhistorie - Opprinnelse

Siden vi begge jobber på Nasjonalbiblioteket var det naturlig å spørre om de hadde noen prosjekter vi kunne ta for oss, og bruke til vår bacheloroppgave. Vi avtalte et møte med lederen og endringsagentene på DBU-Drift på Nasjonalbiblioteket. Thomas Langvann (Leder DBU-Drift) hadde et ønske om å kunne se i nåtid hvor mange objekter som ligger i produksjonsløypene til enhver tid. Dette for å bedre kunne identifisere flaskehalser og å se eventuelle feil og opphoping av objekter i produksjonsløypene. Bakgrunnen for dette var at når det blir satt et mål om å øke produksjonen med for eksempel 50% er det ingen system som overvåker dette. De eneste tallene man har, er hvor mange objekter som går inn i produksjonsløypene og ikke hvor mange som kommer ut.

1.2 Hvorfor vi valgte denne oppgaven

Vi valgte denne oppgaven fordi den har aspekter fra både utvikling og drift, og fordi vi får muligheten til å utvikle et system som gir god nytteverdi for Nasjonalbiblioteket. Vi syntes også denne oppgaven er spennende da den vil gi oss mulighet til å bruke mye av kunnskapen vi har tilegnet oss gjennom årene fra både skole og jobb. Oppgaven vil også gi oss erfaring i hvordan slike prosjekter gjennomføres i en større virksomhet, noe vi kanskje ikke hadde fått dersom vi hadde utviklet et system for oss selv. Det gir oss også mulighet til å benytte ny teknologi og infrastruktur, og få erfaring med dette.

1.3 Samarbeidspartnere

Nasjonalbiblioteket har mål om å være nasjonens hukommelse og bevare alle norske publikasjoner i det offentlige rom. Dette inneholder de fleste medier som for eksempel bøker, aviser, radio og tv. Det er et mål for Nasjonalbiblioteket å digitalisere dette for å kunne lettere tilby det til offentligheten.

Denne oppgaven skriver vi for DBU drift ved Nasjonalbiblioteket. DBU drift forvalter tjenester og infrastruktur ved Nasjonalbiblioteket. Konkrete interessenter er Thomas Langvann (Seksjonsleder DBU drift), Sverre Bang(Endringsagent DBU drift) og Marianne Drotninghaug(Endringsagent DBU drift) og ellers de som jobber med produksjonsløypene daglig.

1.4 Hensikt med rapporten

Hensikten med et forprosjekt er å planlegge, evaluere og beskrive hovedoppgaven. Det er her man finner ut om prosjektet er gjennomførbart eller ikke.

1.5 Oppgavebeskrivelsen fra Nasjonalbiblioteket

Nasjonalbiblioteket (NB) har etablert produksjonsløyper som sikrer at materiale publisert i det offentlige rom blir bevart for ettertiden i digital form. Produksjonsløypene håndterer både innhold som er født digitalt og innhold som foreligger i analog form der digitalisering av innholdet inngår som en del av produksjonsprosessen. Produksjonsløyper er etablert for innholdstyper som bøker, aviser, tidsskrifter, foto, musikk, radio og tv.

Produksjonsløypene foreligger i ulike varianter,- de krever betydelig manuell innsats og har varierende grad av overvåking, feilhåndtering, varsling, logging og indikatorer for kvalitet, ytelse og mengde.

NB ønsker at prosjektet designer og bygger et system som gir mulighet for å fremskaffe produksjonstall, avvik, flaskehalser og andre indikatorer som kan gjøre oss bedre i stand til å identifisere og beslutte tiltak for effektivisering og automatisering av produksjonsløypene. I prosjektet skal logstash (http://logstash.net/) anvendes som rammeverk for datafangst og det skal bygges en enkel web-tjeneste for presentasjon av utvalgte indikatorer.

1.6 Mål for prosjektet

Målet for prosjektet vil være å lage et system som kan gjøre jobben til kontrollørene (Se kapittel 4.1) av produksjonløypene ved Nasjonalbiblioteket lettere, og på bakgrunn av userstories fra kontrollørene vil målene våre være knyttet opp mot disse.

Utover dette vil første mål være å gjennomføre slik at systemet oppfyller de kritiske userstoriene (Se kapittel 8.1).

2 Relevant arbeid – Vårt system sammenlignet med andre løsninger

Siden slike tjenester ofte er skreddersydde for hver enkelt bedrift, blir det vanskelig å finne noe som kan direkte sammenlignes med vår løsning.

Vi har sett på forskjellige SCADA(Supervisory Control And Data Acquisition) systemer som har tilsvarende tjenester, men disse løsningene er som regel lite brukervennlige og krever at brukerne må installere et program. Disse løsningene er vanlige i større produksjonsbedrifter og brukes blant annet i olje og gass industrien for å overvåke og kontrollere produksjonen. Da vi bare er ute etter monitorering, blir disse løsningene for avanserte og tungvinte. De ville også vært dyre i både innkjøp og support, og i tillegg vanskelige å konfigurere i forhold til Nasjonalbibliotekets (NB) eksisterende systemer. NB er ikke en tradisjonell produksjonsbedrift, med PLS og sensorer til å registrere avvik og lignende, men en digital fabrikk. Produksjonsløypene foregår i og mellom forskjellige servere, og det som prosesseres er digitalt materiell.

På Nasjonalbiblioteket finnes det et lignenede system som monitorerer produksjonsløypene og lager grafer av objekter i produksjonsløypene. Den består av en treg web-applikasjon, og det er ikke uvanlig når man laster siden at den bruker opp til 5 minuter før status på produksjonsløypene er visuelt framstilt. Problemet med denne løsningen er at den er laget med gammel tekonologi av en enkelt person på NB og få vet om den. Dette har ført til at systemet ikke blir driftet og oppdatert ved endringer i produksjonen på en tilstrekkelig måte.

Et annet problem er at hvis denne løsningen brukes av flere, krever den mye ressurser fra produksjonsdatabasen og kan i værste fall føre til avvik i produksjonen. Systemet er også veldig vanskelig å forstå for folk som ikke har mye kunnskap om produksjonsløypene.

Vår løsning vil være lettere å bruke, da vi skal gjøre systemet lett forståelig for folk som ikke har kunnskap om produksjonløypene. Tjenesten vil bestå av en enkel web-applikasjon, og dermed trenger ikke brukere ekstra programvare for å få tilgang til denne. Vi kommer til å bruke Spring (Se kapittel 10.1) til utviklingen av web-applikasjonen. Spring er også rammeverket utviklingsavdelingen på NB bruker, så de har mye kompetanse på dette og kan oppdatere applikasjonen ved behov.

Vi blir å pre-fetche eller cache data slik at web-applikasjonen ikke trenger å bruke produksjonsdatabasen for hver person som bruker tjenesten, og derfor vil den være mye raskere enn den eksisterende løsningen. Koden vil bli skrevet så enkel og standarisert som mulig slik at det blir lett å legge inn nye produksjonløyper, som vil gjøre drift og vedlikehold av applikasjonen mye lettere. Vi kommer også til å koble tjenesten opp mot logstash (Se kapittel 10.1) som gjør det mulig å gå inn på hvert enkelt objekt og få opp alle logg-innslagene om objektet. På denne måten kan denne tjenesten også brukes til feilsøking, istedefor å manuelt gå inn og søke i loggene.

3 Kravinnsamling

3.1 Brukergruppe

De som blir brukere av applikasjonen er de som daglig jobber med produksjonsløypene. Det innebærer de som jobber i selve produksjonen og scanner bøkene, avisene eller hva det måtte være. Disse kan bruke applikasjonen for å få en oversikt over hvor mye de har fått gjort i løpet av en dag, uke, mnd, etc. En annen brukergruppe er kontrollørene som jobber med objektene i produksjonsløypene. Deres oppgave er å drive med feilretting og sørge for at objektene kommer seg igjennom løypene. De vil ha et stort utbytte av et godt system som logger, viser feilmeldinger og hvor objektene befinner seg i produksjonsløypen. Det kan bidra til bedre oversikt, og raskere komme til konklusjon hvor, når og hva som har gått galt.

Andre som vil ha en innblikk i produksjonen kan dra nytte av et slikt system. Det kan for eksempel være ledere vil se om målene om å digitalisere x-antall aviser er nådd, eller rett og slett bare se trykket i produksjonen for et gitt tidsrom.

3.2 Funksjonelle krav

User stories

User stories er det vanlig at de som kommer til å bruke systemet skriver. De er veldig lett formulert slik at alle skal kunne komme med en historie. Oppbygningen av en user story er: Som <Brukerrolle> vil jeg <mål/krav> slik at <nytte>.

I forbindelse med dette prosjektet hadde produkteier og et utvalg av brukere møte der de fikk diskutert seg i mellom hva systemet skal gjøre og kom fram til 4 userstories.

Dette gjør det lettere for oss å holde rett fokus på hva brukerne vil ha og ikke hva vi vil ha. Produkteier skal prioritere user stories, men i skrivende stund er ikke dette gjennomført. Vi har derfor prioritert disse på bakgrunn på hva vi syntes er viktigst for systemet på en skala fra 1 til 10, der 10 er høyest prioritet.

User story 1

Som kontrollør vil jeg ha informasjon om antall objekter fordelt på stegene i produksjonsløypa i sanntid, slik at jeg til enhver tid kan få et øyeblikksbilde av produksjonen.

Prioritet: 10

User story 2

Som kontrollør vil jeg ha informasjon om hvordan objektene i produksjonsløypa er fordelt på status (ok, failed etc.) i sanntid, slik at jeg kan få et øyeblikksbilde av status for enkelt objekter i produksjon.

Prioritet: 7

User story 3

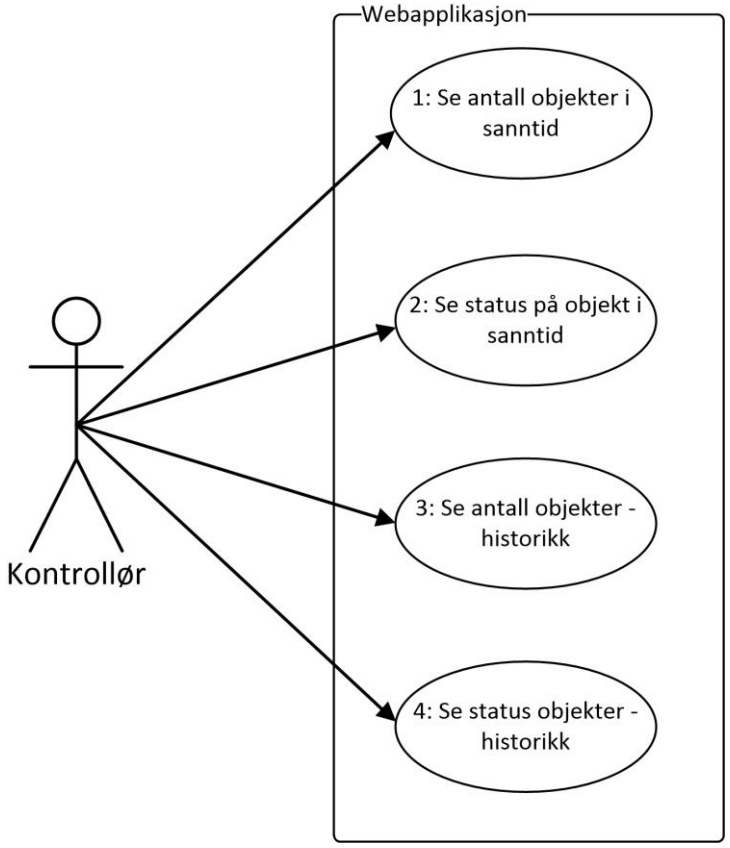
Som kontrollør vil jeg ha tilgang til trendkurver for antall objekter fordelt på stegene i produksjonsløypa slik at jeg kan se utviklingen over tid.

Prioritet: 4

User story 4

Som kontrollør vil jeg ha tilgang til trendkurver for hvordan objektene i produksjonsløypa har vært fordelt på status (ok, failed etc.) slik at jeg kan se utviklingen over tid.

Prioritet: 5



Figur 1: Use-case diagram basert på user stories.

3.3 Kritiske userstories

User story 1

User story 1 er den mest kritiske på grunn av at resten av prosjektet vil bygge på funksjonaliteten denne medbringer og vil være den funksjonaliteten som vil bli brukt mest.

User story 2

User story 2 er også viktig for systemet. Denne funksjonaliteten gir kontrolløren mulighet til å se avvik og feil i produksjonsløypa. User story 3 og 4 bygger videre på denne funksjonaliteten.

3.4 Ikke-funksjonelle krav

Brukeropplevelse

Systemet skal ha et godt design, og skal oppleves som et lett og brukervennlig system. Dette får vi til ved å spørre brukerne om det er noe som kan gjøres for at webapplikasjonen skal være lettere å bruke.

Ytelse

Systemet skal oppleves som raskt og responsivt, det vil si at det ikke skal gå flere minutter å laste siden. Det skal heller ikke bruke unødvendig mye ressurser fra produksjonsmiljøet, noe vi vil få til gjennom en cache/prefetching løsning. Det skal ikke ta mer en 30 sekund å laste webapplikasjonen.

Drift- og vedlikehold

Systemet skal være lett for Nasjonalbiblioteket (NB) og vedlikeholde. Dette vil vi få til ved å bruke teknologier som er kjent og i bruk hos NB. Det skal ikke brukes teknologier som drift/utviklingsavdelingen hos NB ikke har kjennskap til. Koden skal skrives slik at den kan gjenbrukes til andre prosjekter og applikasjoner hvis man ser behov for det.

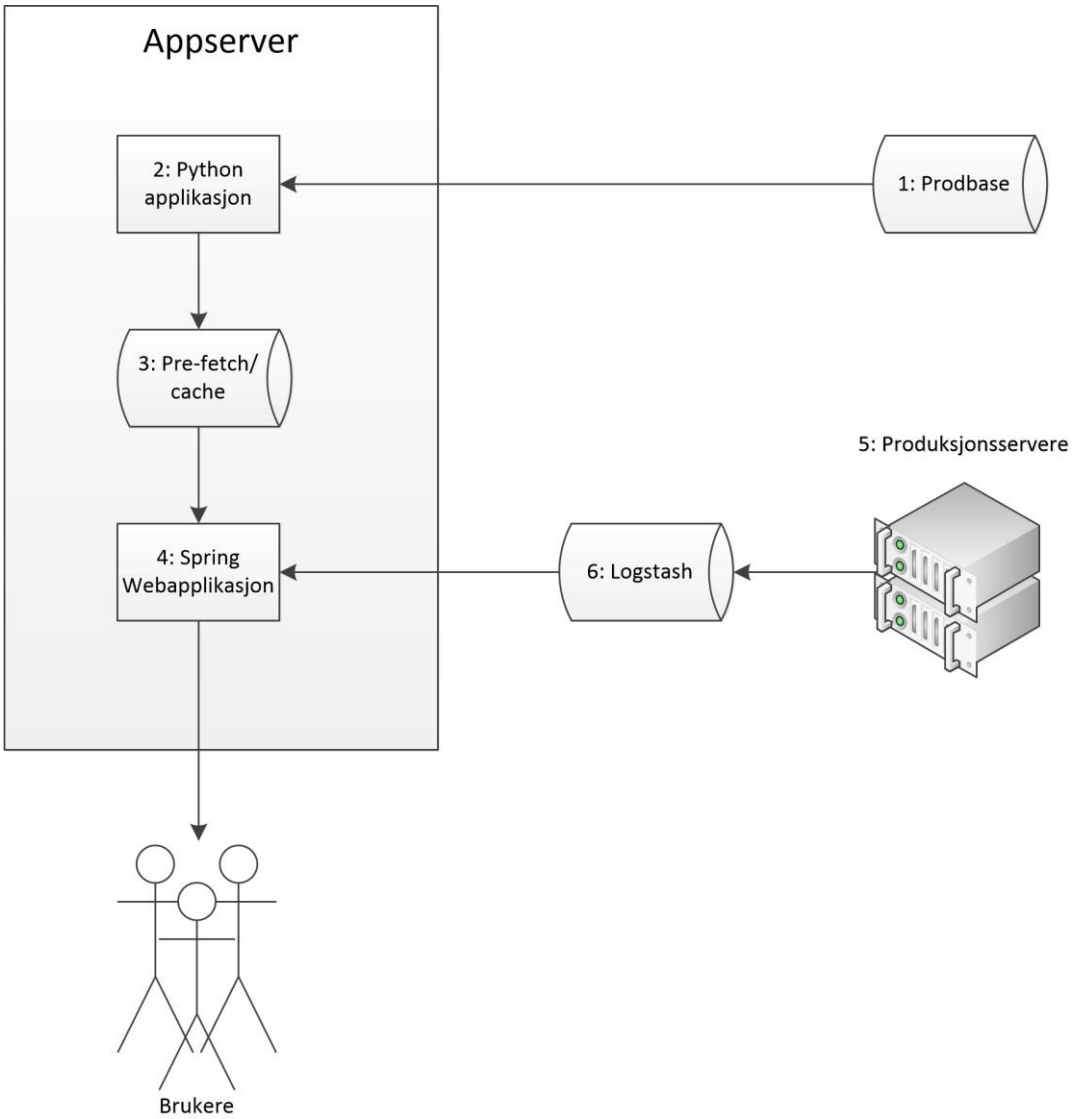
Konsistens

I user stories blir begrepet sanntid brukt ofte, med sanntid i disse tilfellene menes det forholdsvis ny data. Dette systemet har ikke behov for data fra sanntid, vi vil derfor senke på kravet om sanntidsdata og heller fokusere på ytelse. Det er her pre-fetch/cache løsnigen kommer inn, ved at vi sanker inn data med for eksempel 10 minutters mellomrom, vil vi kunne øke ytelse betraktelig. Konsistensen mellom data fra vårt system og data fra produksjons databasen vil lide litt, men for dette systemet har ikke dette stor betydning. Vi velger derfor å sette som krav at data som brukes ikke skal være eldre en 20 minutter, og det vil vises i applikasjonen når det sist ble hentet data fra produksjons databasene.

4 Arkitektur og prototype

4.1 Arkitektur

Vi har allerede sett for oss en overordnet arkitektur. Informasjonen om objektene ligger på to plasser, det meste ligger på produksjonsbasene, mens noe blir skrevet til logger på produksjonsserverene. Derfor blir vi nødt til å ha to måter å hente data på, en via produksjonsdatabasen og en gjennom Logstash og produksjonsserverne. Dette kan selvsagt endres iløpet av prosjektet, men det er slik vi ser for oss at det vil fungere.



Figur 2: Overordnet arkitektur av systemet.

1: I produksjonsdatabasene på Nasjonalbiblioteket ligger det generell informasjon om objektene, det vil si hvilket steg de befinner seg i og hvilken status de har, samt dato og klokkeslett for de jobbene som blir kjørt.

2: Python applikasjonen er noe vi må lage. Den skal hent data fra produksjonsbasen, og sende det til pre-fetch/ cache.

3: Pre-fetch/cache for dette systemet eksisterer ikke per dags dato. Dette er noe vi må sette opp, og vil bestå av en NoSQL database.

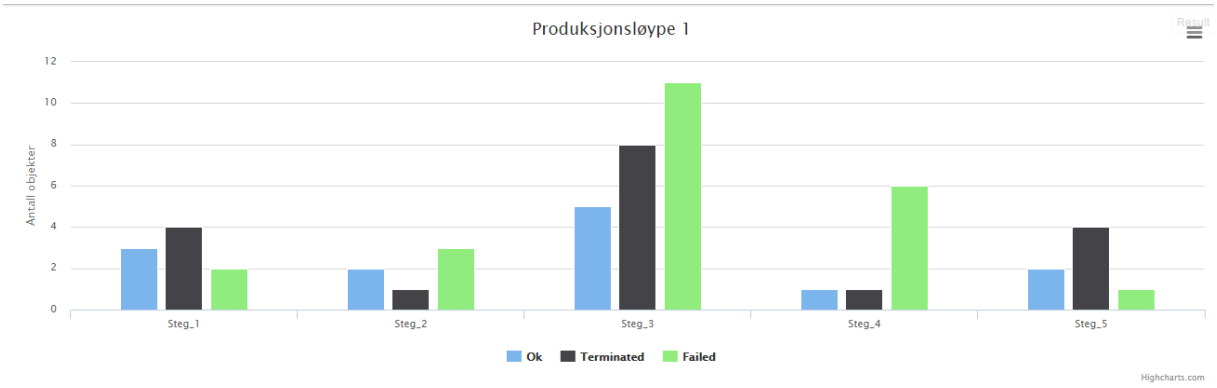
4: Web-applikasjonen er også noe vi må lage. Denne vil hente data fra NoSQL basen og loggene fra logstash, deretter presentere dette i applikasjonen.

5: På produksjonsserverne kjøres det en del jobber som skriver logger. Disse loggene vil bli sendt til logstash, for videre håndtering.

6: Logstash er implementert på Nasjonalbiblioteket, denne tar i mot logger og lagrer de i en database, som er rask og lett å søke i.

4.2 Prototype

Dette er en prototype på hvordan webapplikasjonen kan se ut. En lett forståelig graf der stegene i produksjonsløypene og antall objekter er visualisert. Man kan klikke på hver enkel søyle for å få informasjon om hvilke objekter som ligger der.



Figur 3: Prototype av hvordan webapplikasjonen kan se ut.

5 Fremgangsmåte

5.1 Forprosjektrapport

Det kan være verdt å nevne at større og grundige forprosjektetrapporter som dette er en del av det første steget i RUP (Inception fasen). Vi henter flere elementer fra denne fasen, slik at prosjektet skal være godt gjennomtenkt og planlagt før vi begynner arbeidet med utviklingen. Det er også flere elementer som strider i mot utviklinsmetodikken vi har valgt å bruke. Disse har vi enten tilpasset eller skrapet.

Inception[Jacobson, Booch, Rumbaugh. 2007, kapittel 13]:

- Kravinnsamling med funksjonelle (fra brukerne) og ikke fuksjonelle krav, samt definere scope på prosjektet og finne de kristiske user storyene.
- Diskutere risiko og kostnader.
- Utarbeide en designskisse og sikre brukskvalitet.
- Planlegge og sette deadlines på prosjektet

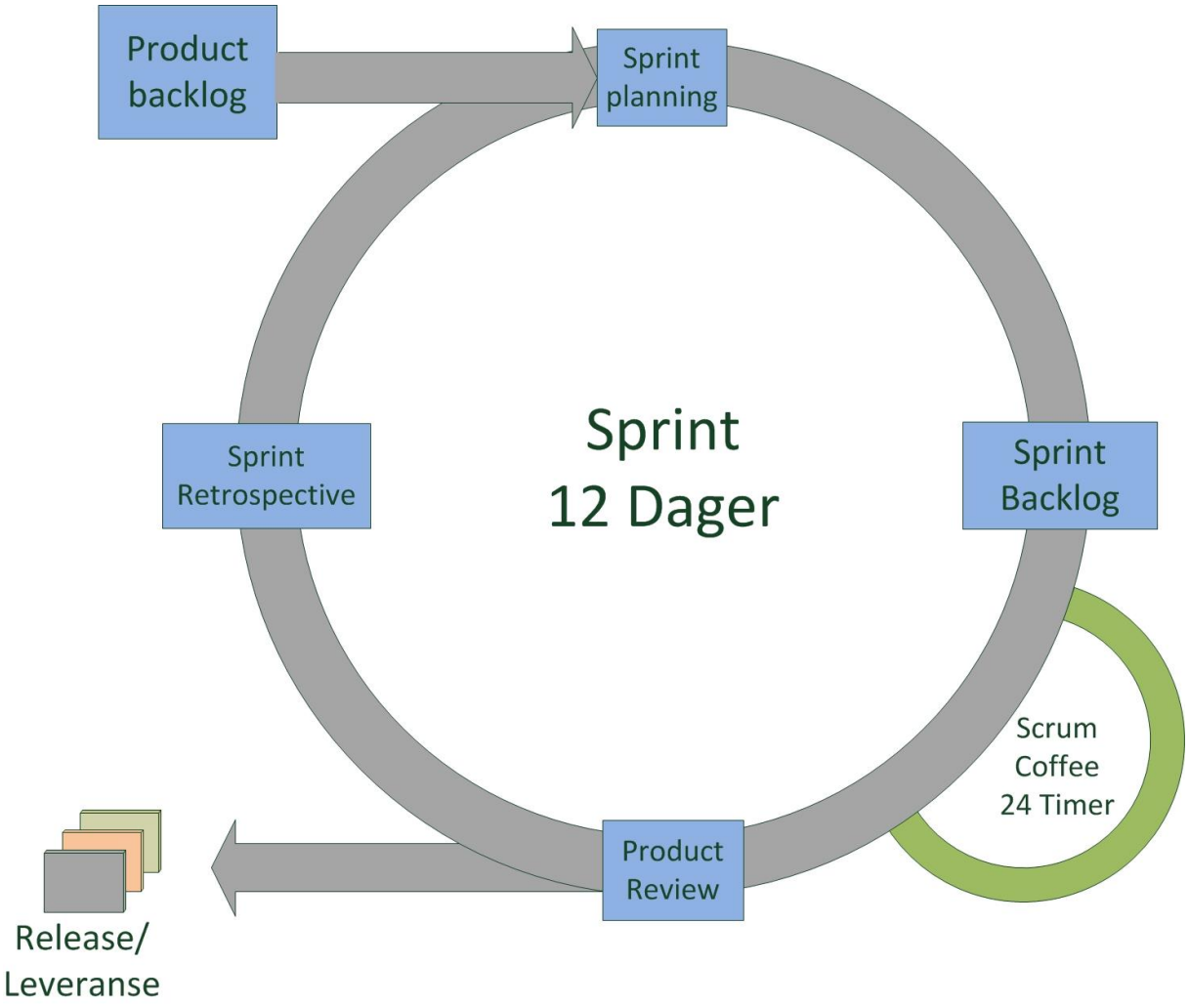
5.2 Utviklingsmetode

Vi har valgt å bruke en blanding av Scrum og XP (Extreme Programming) [Kniberg, 2007] for dette prosjektet, og kun ta med de elementene som er mest hensiktsmessig for oss. Vi har sett på forskjellige utviklingsmetoder og funnet ut at for et så lite team som to personer er dette et godt alternativ. Dette er på grunn av at vi tar bort mye av det vi ikke trenger, og benytter oss av det vi føler kan hjelpe oss og holde struktur på prosjektet. Begge er smidige utviklingsmetoder som hjelper oss å være fleksibel, slik at kravspeccen kan endres underveis. Det kan for eksempel være at nye tasks blir lagt til under sprint-planleggingen, fordi en bruker ser behov for utvidet funksjonalitet.

Scrum har gode teknikker for organisering og planlegging noe som vil hjelpe oss å være produktiv, men siden scrum ikke sier noe om programmeringen i praksis, har XP mange gode elementer om dette. Vi vil bruke noen av disse i sprintene når vi programmerer, og det hjelpe oss til å få kvalitet i koden. Vi kommer også til å bruke Jira(Se kapittel 10.2) under hele prosjektet, slik at vi kan visualisere arbeidsflyten på en oversiktelig måte i form av swimlanes med backlog, todo, in progress og done.

Vi fikk være med på sprint-planlegging og review hos utviklingsavdelingen på Nasjonalbiblioteket, for å se hvordan de gjorde det der og for å få en bedre oversikt av hvordan det skjer i praksis. Det var både spennende og lærerikt.

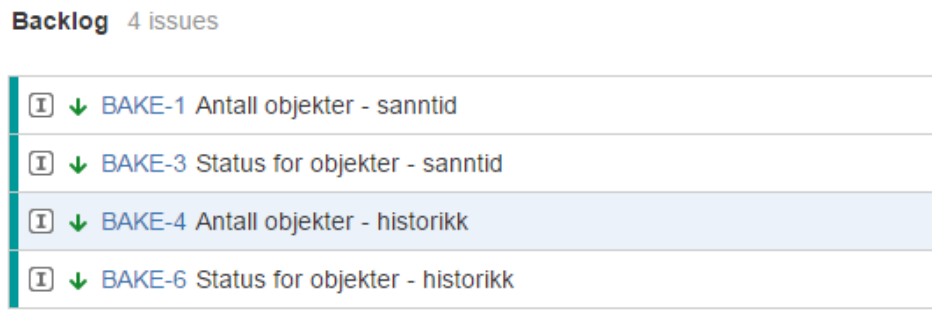
5.3 Scrum:



Figur 4: En skisse over arbeidsflyten i Scrum.

Product backlog

For å holde oversikt over det som må være ferdigstilt for et endelig produkt. Dette kan være krav, funksjoner eller lignende, i en prioritert rekkefølge, og kalles backlog stories. Disse vil inneholde noen felt som beskriver storiene, det kan være for eksempel navn(en kort beskrivelse), estimert tid, og prioritet, eksemplet under viser hvordan det kan se ut:



Figur 5: Skjermbilde tatt av backlogen i Jira.

Sprint planning

Kanskje det viktigste møtet i scrum, og før vi kan begynne med dette må product backlog'en eksistere. I dette møtet skal produkteier og teamet være representert. Backlog stories må ikke være perfekt og veldefinert, men prioriteringen burde være på plass. Hensikten med Sprint-planning er at teamet skal få nok informasjon slik at de kan jobbe i fred og ro i et par uker. Man skal også få fram målet med sprinten, avtale demo (review), tid og sted for daily scrum. I sprint planning vil man også velge hvilke product-backlog stories man skal ta med i sprint-backlogen. Disse blir analysert og "hugget" opp i mindre biter slik at man kan lage mer konkrete oppgaver. Disse oppgavene må prioriteres og estimeres og kalles task's. Estimeringen gjør man ofte ved hjelp av planning-poker for å sikre gode estimat.

Sprint backlog

Er noe som må lages etter sprint-planning og før daily-scrum møtet. Det er i sprint backlog man legger til tasks, og det er fra denne man henter oppgaver når man jobber i en sprint. Vi skal bruke Jira for å visualisere arbeidet, backlogen og samtidig få tilgang på burndown-chart. Burndown-chart er en visuell representasjon av arbeid og tid som gjenstår. Dette vil gi oss en indikasjon på om vi blir ferdig med alle tasks før sprinten er ferdig.

Scrum coffee

Vår egen versjon av daily scrum, for å tilpasse et mindre team har vi kuttet ned tiden. Normalt sett vil denne ta 5-10 minutter, og foregår på starten av dagen. Her kan vi oppdatere arbeidstavla og komme med nye estimat, hvis for eksempel uforutsette hendelser oppstår.

Sprint

I en sprint er der hvor utviklingen foregår. Vi velger at sprintene skal foregå over 12 dager, rett og slett pga tiden. På denne måten slipper vi også å skifte fokus så ofte, med tanke på Sprint-Review og Sprint-planning, og får konsentrert oss om arbeidet. Samtidig legger vi ikke beslag på så mye av de involvertes tid og ressurser. Blir vi ikke ferdig med en task i løpet av en sprint, vil denne bli tatt med i den neste sprinten.

Sprint Review

Vil foregå hver andre uke, etter hver sprint. Her skal vi se på hva som har blitt ferdig, og eventuelt hva som ikke ble ferdig. Vi vil også gi en kort presentasjon av ferdig vare til produkteier og brukere, slik at vi kan få tilbakemelding på systemet underveis. Her er det også satt av 30 minutter.

Sprint Retrospective

Hver andre uke etter hver sprint vil vi ha et lite møte. Her vil det kun være medlemmene i teamet, og vi vil reflektere over den foregående sprinten, og finne ut hva som gikk bra, og hva som kan gjøres bedre. Har satt av 15 minutter til dette.

Sprint Review, Retrospective og Planning vil som regel foregå på samme dag.

5.4 Extreme Programming

Pair programming

Jobbe i par, to utviklere foran en maskin. Dette er for å sikre kvalitet i koden, da den som ikke sitter foran tastaturet bidrar med å gå over koden mens den blir skrevet. Begge får da en god forståelse for koden og man får i tillegg en større mulighet til å komme med forbedringer og avdekke feil i koden med en gang. "Navigator"; (den som ikke sitter foran tastaturet) burde ha en datamaskin, slik at han kan lete opp dokumentasjon og andre ting når "driver"(den som skriver kode) sitter fast på noe, slik at den slipper å endre fokus. Vi vil prøve å benytte oss av dette der det er mulig.

TDD(Test-Driven Development)

TDD er metode der man bruker tester for å drive utviklingen fremover. Før man lager en funksjon skal testen skrives, og hver funksjon må bestå testen før den kan tas i bruk. Siden vi bruker Spring framework kan vi bruke Junit, som er et test-rammeverk. Dette vil hjelpe oss med å lage gode tester.

Simple design

XP oppfordrer til lette løsninger, og at ekstra funksjonalitet kan bli lagt til senere. Dette sørger for at vi ikke bruker tid og ressurser på noe som ikke nødvendig der og da, og kan implementere dette på et senere tidspunk ved behov.

Refactoring

Gå over kode gjevnlig for å se om for eksempel en funksjon kan skrives lettere og/eller penere. Gjerne en stund etter at koden er skrevet, slik at man kan få "nye" øyne på det.

Collective owership

For å unngå "min" og "din" kode. Dette fungerer bra sammen med Pair programming, da vi begge blir å få en god forståelse for koden og kan endre den ved behov.

Coding standards

Siden Nasjonalbiblioteket ikke har en egen kodestandard, vi velger å bruke IntellJ IDEA (se kapittel 9.2) sin egne kodestandard. Fordelen med dette er at vi kan bruke funksjoner i Intellij for å formatere koden automatisk, uten å måtte konfigurere dette. Grunnen til at man har en slik kodestandard er at koden skal være lett leselig for alle, slik at ikke bare de som skrev den skjønner koden.

6 Fremdriftsplan

Her har vi laget et Gantt-diagram. Det gir en god visuell oversikt over tidsplanen, med milestones og deadlines. Her har vi valgt og sette opp 5 sprinter på grunn av tidsrammen på prosjektet. Det er også satt av litt slakk mellom hver sprint, slik at vi har litt å gå på, om vi havner bakpå, vi vil få indikatorer på dette i burndown-chart'en. Vi har valgt å ikke sette opp milestones for hver sprint, rett og slett fordi vi ikke vet hva som blir å komme med i sprintene. Det er noe som blir avgjort i sprint-planleggingen som skjer før hver sprint.

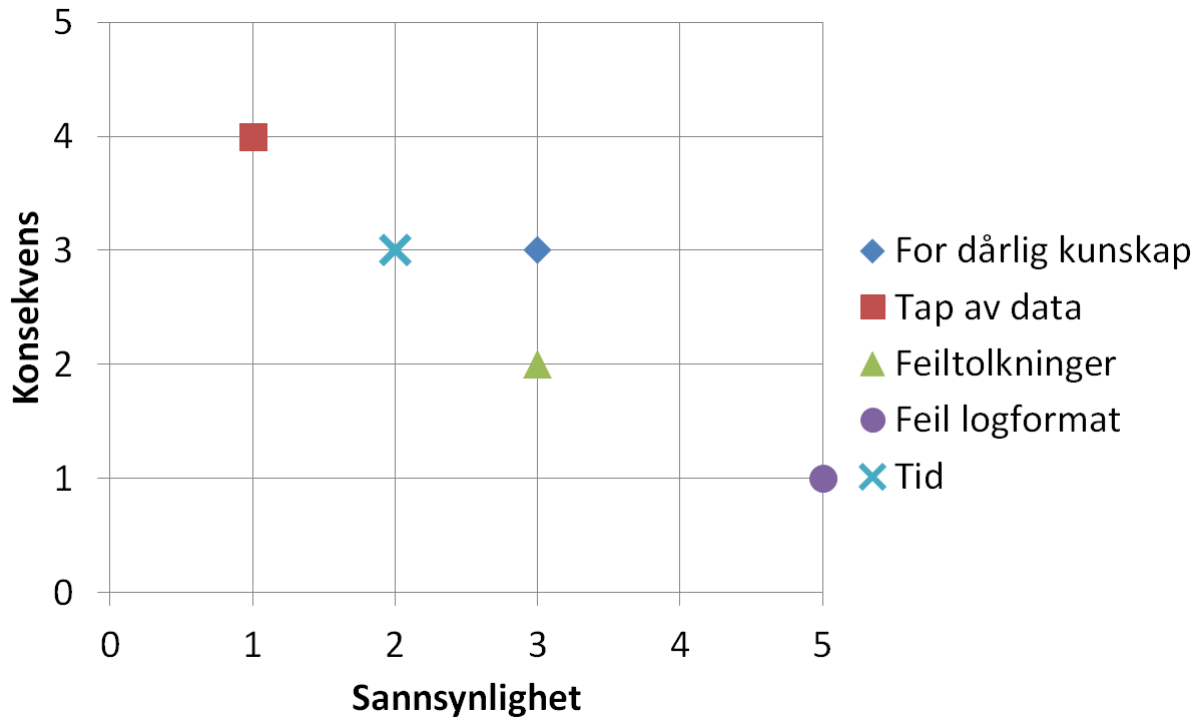
Hvis vi mot formodning ikke skulle få implementert alle user stories, vil vi uansett ha fått stor lærdom i prosjektorganisering og estimering, og kommer til å fortsette med prosjektet fram til all funksjonalitet er på plass. Skulle vi derimot bli for tidlig ferdig, arrangerer vi nye møter med produkteier, og kommer fram til ekstra funksjonalitet vi kan legge til.

ID	Aktivitetsnavn	Start	Slutt	Varighet
1	Oppgavebeskrivelse	05.01.2015	14.01.2015	10d
2	Forprosjekt rapport	15.01.2015	28.02.2015	45d
3	Sprint 1	02.03.2015	13.03.2015	12d
4	Sprint 2	16.03.2015	27.03.2015	12d
5	Sprint 3	30.03.2015	10.04.2015	12d
6	Sprint 4	13.04.2015	24.04.2015	12d
7	Sprint 5	27.04.2015	08.05.2015	12d
8	Sluttrapport	11.05.2015	01.06.2015	22d

Figur 6: Gant diagram.

7 Risikovurdering

For å redusere muligheten for at noe galt skal skje under prosjektet lager man en risikoanalyse. Med denne skal man prøve å kartlegge de viktigste risikoelementene, og konsekvensene av disse. Risikoene er i korte trekk beskrevet under, og viser sannsynlighet og konsekvens på en skala fra 1 til 5, der 1 er lavest og 5 er høyest.



Figur 7: Risikodiagram

For dårlig kunnskap

En risiko for denne type oppgave er at vi sitter med for lite kunnskap på enkelte områder, dette kan føre til at vi støter på problemer under programmeringen av systemet. Det er for eksempel lenge siden vi har brukt Spring framework så det kan hende at vi blir å sette oss fast en del med dette. Hvis dette skjer må vi være flinke å bruke de ressursene vi har tilgjengelig, som for eksempel veiledere på skolen, utviklere på Nasjonalbiblioteket eller bøker og dokumentasjon.

Tap av data

Tap av kode under utvikling er også en reell risiko. Det kan oppstå om man glemmer å lagre eller får maskinvaresvikt. Dette kan føre til tap av for eksempel kode eller dokumenter. Tap av vår egen kode eller dokumenter er ikke ønskelig, da konsekvensen av dette kan bli fatal for prosjektet. Derfor har vi gode forhåndsregler slik at det ikke skal skje. Vi bruker Google Drive til lagring av dokumenter, og git (se kapittel 9.2) til versjonskontroll og lagring. På denne måten har vi et eksemplar lokalt og en i det interne nettet til Nasjonalbiblioteket. Det er ikke fare for tap av Nasjonalbibliotekets data i produksjonsløypene.

Feiltolkninger

For at ikke feiltolkninger og misforståelser skal oppstå vil vi være påpasselig å spørre om det er noe som er uklart. Vi kommer til å diskutere oss i mellom om vi har samme oppfatning av en oppgave eller hvordan systemet skal fungere. Om en feiltolkning inntreffer vil vi raskt få avdekket dette, siden vi bruker Scrum som utviklingsmetodikk har vi ofte møter med produkteier der vi får avdekt eventuelle misforståelser.

Feil logformat

For at dette systemet skal fungere optimalt må loggene fra enkelte steg i produksjonsløypene være på et spesielt format. Å konvertere disse loggene er en stor oppgave som er satt i gang hos Nasjonalbiblioteket. På grunn av størrelsen til oppgaven er det svært usannsynlig at dette blir gjennomført i tide. For prosjektet har ikke dette stor betydning siden vi kan lage disse loggene (mocking) for å teste at systemet fungerer. Så når loggene kommer i rett format er systemet klart for dette.

Tid

Om det skulle bli for liten tid til å implementere alle user stories, blir i verste fall målet med prosjektet ikke nådd. Vi vil forhåpentligvis ha fått implementert noen user stories, og dermed kan man likevel dra nytte av systemet. Noen forhåndsregler har vi tatt, blant annet har vi litt slakk mellom sprintene, og er forberedt på å jobbe både på ettermiddager og helger - slik at vi skal nå målene.

8 Kostnadsvurdering

Å regne ut kostnader for et slikt prosjekt regner vi ut arbeidstimene vi legger i prosjektet, og hva summen hadde blitt om vi hadde fått lønn for arbeidet. Det er ingen lisenskostnader til programvaren vi har valgt å bruke.

Vi har begge to en timevikarstilling ved Nasjonalbiblioteket, vi har brutto 339 000 i året i 100% stilling. Vi tar utgangspunktet i dette.

Vi regner 100 dager og ganger dette med 7,5 timer, som er en vanlig arbeidsdag per person. Etter tidligere erfaringer med å utvikle applikasjoner i skolesammenheng regner vi med at vi blir å gå noen ettermiddager også, men vi velger å se bort i fra dette.

- 100 dager arbeid
- 7,5 timers arbeidsdager
- 2 personer
- 100 x 7,5 x 2 = 1500
- 1500 x 188 = 282500

Det ville ha kostet Nasjonalbiblioteket 282 500,- brutto om vi skulle gjort dette i arbeidstiden. Andre kostnader vi ikke velger å ta med i regnestykket er arbeidstiden til de involverte på NB.

9 Teknologier og verktøy:

Teknologiene vi skal bruke er i hovedsak open-source, dette gjør vi for å redusere kostnader men også fordi open-source er mye benyttet på Nasjonalbiblioteket (NB). Noen av disse teknologiene er vi nødt til å bruke da NB allerede har implementert disse. Verktøyene vi kommer til å bruke er verktøy vi får gjennom skolen eller Nasjonalbiblioteket.

9.1 Teknologier

Elasticsearch

Elasticsearch er en fleksibel og kraftig open source søk og analysemotor. Den har ytelsen som trengs for en slik oppgave og kan skaleres i bredden.

Logstash

Logstash er et verktøy for å håndtere hendelser og logger, og brukes til å motta og foredle logger og lagre dem til senere bruk. Vi finner ikke et annet produkt som kan konkurrere med dette i fleksibilitet og ytelse.

Spring Framework

Spring er et rammeverk for koding med Java, og brukes ofte i store bedrifter. Vi velger å bruke Spring fordi at Nasjonalbiblioteket bruker Spring på flere av deres egenutviklede applikasjoner, dette vil gjøre drift av denne applikasjonen mye lettere.

MariaDB

MariaDB er en SQL database som er basert på MySQL. Produksjonsbasene på Najonalbilioteket kjører MariaDB, og det er der vi må hente data om hvert enkelt objekt i produksjonsløypene.

Rsyslog

Rsyslog er et logging program for Linux, og blir brukt som standard i de fleste nye distribusjoner av Linux. Det kan også brukes for å sende logger mellom maskiner. Vi velger å bruke dette for å sende logger til logstash av flere grunner, den største grunnen er at man kan legge ved informasjon på hver enkelt loglinje om hvor loggen kommer fra.

Centos 7

Centos 7 er en Linux distribusjon som er basert på Redhat og Fedora, og er på mange måter en gratis versjon av Redhat. Vi velger å bruke et CentOS 7 miljø fordi det brukes på alle maskiner ved Nasjonalbiblioteket, dette vil gjøre jobben for drift lettere.

Python 2.7/3.3

Python er et open source kodespråk som er blitt veldig populært de siste årene. Vi velger å benytte Python for å kjøre spørringer mot produksjonsdatabasen og bruke resultatet vi mottar som cache lokalt på serveren, eller pre-fetching til en database. Grunnen til at vi velger Python er fordi det er standard i Linux distribusjoner.

Microservices

Microservices er en tankegang der man tenker at man istedetfor å lage en stor tjeneste som gir kunden det han vil ha, lager mange små tjenester som kommuniserer med hverandre og hver enkelt tjeneste er veldig god på det den gjør. Dette gjør at man gjenbruke deler av koden til andre prosjekter uten å skrive om noe som helst.

9.2 Verktøy

Git

Git bruker vi for versjonshåntering av koden, vi kommer til å bruke Nasjonalbibliotekets egen git installasjon.

Jira

Vi kommer til å bruke Jiras swimlane tavle, dette gjør det veldig enkelt å planlegge sprinter og gir oss veldig god oversikt om hvordan det går i prosjektet med burndowncharts og andre grafer.

IntelliJ IDEA

Som IDE skal vi bruke IntelliJ fra JetBrains for å skrive kode, vi bruker denne fordi det er en veldig god editor med mange bra funksjoner, og vi er godt vant med denne fra før.

PyCharm

Til Python kommer vi til å benytte IDE'en PyCharm fra JetBrains.

10 Litteraturliste

Beck, K. (1999). *Extreme Programming Explained: Embrace Change.* [USA], Addison-Wesley

Jacobson, I. Booch, G. Rumbaugh, J. (2007). *The Unified Software Development Process*. [USA], Addison-Wesley.

Kniberg, H. (2007). *Scrum and XP from the trenches : how we do scrum*. [USA], c4media.

Schwaber, K. Beedle, M. (2001). *Agile Software Development with Scrum.* [USA],Prentice Hall.

Sommerville, I. (2011). *Software Engineering*. [USA], Pearson.

CentOS (udat). *CentOS Project.* http://www.centos.org/

Elasticsearch (udat). *Open Source Distributed Real Time Search & Analytics*. http://www.elasticsearch.org/

Git (udat). *Version Control on your Server.* https://about.gitlab.com/

IntelliJ IDEA (udat). *The Most Intelligent Java IDE.* https://www.jetbrains.com/idea/

Jira (udat). *Issue & Project Tracking Software.* https://www.atlassian.com/software/jira

Logstash (udat). *Open Source Log Management.* http://logstash.net/

MariaDB (udat). *MariaDB.* https://mariadb.org/

Rsyslog (udat). *Rsyslog.* http://www.rsyslog.com/

Pycharm (udat). *Python IDE & Django IDE for Web developers.* https://www.jetbrains.com/pycharm/

Python (udat). *Python.* https://www.python.org/

Spring Framework (udat). *Spring Framework.* http://projects.spring.io/spring-framework/