UNIVERSITETET I AGDER

DAT215

Prosjekt Rapport
Rusta Vrak Bilutleige

Student:
Martin Engen

Veileder:
Arne Wiklund

Ved

Institutt for IKT
Fakultet for Teknologi og Realfag

24. november 2016

Innhold

Figurer

Kapittel 1

Innlending

1.1 Bakgrunn

Rusta Vrak bilutleige er et lite firma basert i Førde og Jøster i Sogn og Fjordane. Utleie firmaet blir driftet av verkstedmesteren Stein Olav Erikstad, og målet er å kunne tilby utleie av greie og fult funksjonelle biler til en rimelig pris. Firmaet ble startet i 2013, og har i dag en flåte på over 20 biler. Rusta Vrak fungerer i dag som et sideprosjekt, men det har gode muligheter for vekst i fremtiden.[1]

Dersom et bilutleige firma ønsker å være kompetitivt, er det viktig å kunne tilby en digital form for tilgjengelighet på lik linje som de mer etablerte bedriftene.

1.2 Problemdefinisjon

Rusta Vrak Bilutleie har i dag en nettside basert på tjenesten Blogspot [CITATION], som gjør at det er en enkel informasjonsside som oppgir informasjon om bilene, priser og kontaktinformasjon. Deretter må kunden ta kontakt enten vha. telefon eller epost for å foreta selve bestilling. Da firmaet blir driftet av én person som et sideprosjekt, kan det være vanskelig å alltid være tilgjengelig, og samtidig ha en oversikt over hvilke biler som er ledige til enhver tid. (ASD)

Dette kan bli forbedret ved å gjøre nettsiden mer interaktiv både for kunde og bedrift.

1.2.1 Oppgaven

Lag en nettside for bilutleie firmaet Rusta Vrak Bilutleige. Denne nettsiden skal kunne benyttes av kunder for å finne hvilke biler som passer best for kundens formål, få en oversikt over når de spesifikke bilene er ledige for utleie, og foreta en reservasjon av valgt bil.

Nettsiden skal også kunne benyttes av firmaet for å få en oversikt over bilene som er for utleie, samt kunder som leier eller har leid bil. Det må være mulig å legge til, fjerne og redigere bilene for utleie forløpende.

1.2.2 Krav

I samarbeid med firmaet utarbeidet vi noen krav over hva som er viktig å få med i nettsiden. Disse kan være både funksjonelle og ikke-funksjonelle.

Lett å bruke. Nettsiden skal være enkel å bruke, og skal helst være så intuitiv som mulig. Det betyr at den ikke skal for komplisert i bruk.

Selvstendig og Billig. Firmaet er lite, og derfor skal nettsiden kunne være så selvstendig som mulig. Dvs. arbeid med drift og opprettholdning skal være minimal. Utover dette burde hosting av nettsiden også være rimelig.

Håndere alle reservasjoner. Nettsiden skal både fungere som et reservasjons system for brukere, og som et verktøy for bedriften. Derfor skal brukere kunne benytte nettsiden for reservasjoner, og ansatt skal ha et administrasjons område hvor det er mulig å legge til nye bestillinger.

Fordel å leie i lange perioder. Det skal være en fordel å leie over lengre perioder. Systemet må ta hensyn til dette og det skal implementeres en funksjon for å kalkulere prisen.

1.3 Litteratur Studie

Det aller meste av studie mot denne oppgaven har foregått på internett. Dette inkluderer hovedsakelig bruk av dokumentasjonen til de forskjellige rammeverker og plattformer. Utover dette har det blitt benyttet noen bøker som har fungert som oppslagsverk gjennom hele prosessen.

Bøker

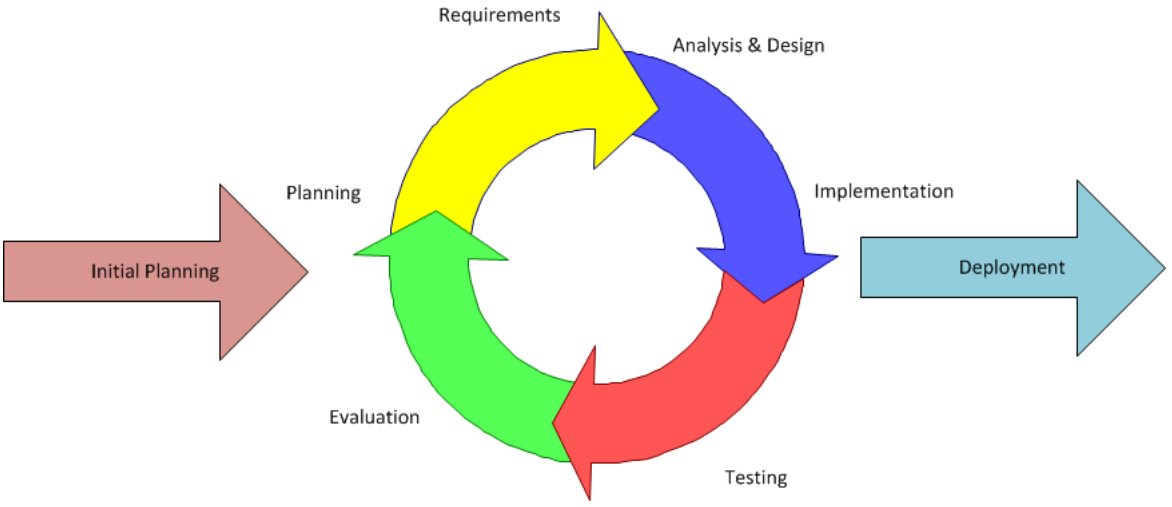
- Code Complete 2. Edition (Steve McConnel)
- Programming Google App Engine with Python (Dan Sanderson)
- HTML and CSS Design and Build Websites (Jon Duckett)

1.4 Problemløsning

1.4.1 Arbeidsmetode

Utviklingen av dette prosjektet har benyttet en iterativ og inkrementell arbeidsmetode. Denne metoden går ut på å gjennomføre utviklingen stegvis, man begynner utviklingen med et skjelett, som man vil videre legge til litt funksjonalitet, og deretter komme tilbake og legge til mer på toppen av dette igjen. Ved å benytte en slik prosess, blir det mulig å utvikle flere områder på nettsiden samtidig uten at det nødvendigvis krasjer mellom de forskjellige funksjonene.

En iterativ metode gjør det mulig å splitte et større prosjekt i mindre, mer håndterlige biter. Utvikling av kode, planlegging og design blir gjennomført i gjenntatte sykluser [2]. Dermed kan man følge figur 1.1 for prosessen gjennom hele utviklingen.



FIGUR 1.1: Oversikt over trinnene per syklus i iterativ utvikling.

1.4.2 Prosjekt Plan

Det første som ble gjort mot denne oppgaven var å lage en plan over alle funksjoner og arbeidsoppgaver som må med i det endelige produktet, og legge disse inn i kategorier basert på viktighet. Kategoriene går fra A til E, hvor A er viktigst og E er minst viktig.

Prioritet A

Lokalt utviklingsmiljø. Installere Python, oppsett av virtualenv, Installere og integrere JetBrains PyCharm, installere Django.

Lokal MySQL. Installasjon og opprette kobling mellom Django prosjekt og MySQL database.

Google Cloud Platform. Installering og oppsett av kobling mellom lokalt prosjekt og Google Cloud Platform. Dette inkluderer både hosting og database.

Prioritet B

Database tabeller. Planlegging og oppretting av database tabellene.

Forside. Enkel forside

Admin. Mulighet til å legge til / slette biler fra database.

Generere liste over biler fra database. Basert på bil kategori.

Bil Reservasjon. Mulighet å legge inn er reservasjon av bil som går mellom to datoer.

Prioritet C

Bilder. Mulighet for å kunne vise bilde av bilene. Både som thumbnail og alternativt galleri.

Kalender. Gjør det mulig for kunde å ha visuell oversikt når den spesifikke bilen er ledig for utleie.

Epost Bekreftelse. Send en email til kunde som bestiller. Denne skal inneholde informasjon om bestillingen.

Reservasjon Håndtering. Legg til funksjonalitet som stopper en reservasjon med å krasje med annen eksisterende reservasjon.

Søkefunskjon. Finne ledige biler på dato.

Prioritet D

Pris Funksjon. Det skal være en fordel å leie over lengre perioder. Derfor må en funksjon kunne dele ut riktig mengde rabatt over hvor mange dager som er i leieperioden. Videre skal denne kunne runde til nærmeste 5 for å slippe for mye småpenger.

PDF. Mulighet å laste ned PDF med bestillingsdetaljer.

Filtrering av bil liste. F.eks. kun biler som har automat.

Videre konfigurering av Admin. Søk bil på skiltnr., oversikt over alle reservasjoner og kunde informasjon.

Prioritet E

Google Calendar. Legg til informasjon i en Google Calendar for enkel oversikt over når biler hentes og leveres

1.4.3 Workflow

1.4.3.1 Kunde

Kunden skal kunne gå gjennom en enkel prosess for å få utført en bestiling på nettsiden. Man kan se på figur [FIG] fremgamgsmåten som skal benyttes. Man vil først møte en forside

1.4.3.2 Administrasjon

Kapittel 2

Oversikt over valgte Teknologier

Dette kapittelet omfatter en oversikt over informasjon fra de valgte teknlogoier, og begrunnelser bak dette.tst

2.1 Publisering

Ett av kravene til prosjektet var at nettsiden skal være så selvstendig som mulig. Det skal være minimalt med nedetid, serveren må kunne oppdateres uten manuelt inngrep og backup av database må foregå automatisk. Samtidig som å oppfylle dette burde selve hostingen være så billig som mulig. Derfor ble det noe moderne konseptet av Platform as a Service benyttet.

2.1.1 Platform as a Service

En Platform as a Service (PaaS) er et utviklings og distribusjons miljø basert i en skytjeneste [3]. En PaaS leverandør har som ansvar å levere og opprettholde en utviklings og distribusjons plattform for sine kunder. De har videre ansvaret på områdene rundt konfigurering, oppsett av servere og sikkerhet på server siden. Det er populært å referere til PaaS leverandører som IT-avdelingen til produktet, som gjør at utvikleren kan fokusere mer på selve utviklingen enn oppsett av infrastrukturen som ligger under [1, s. 10]

2.2 Programmeringsspråk og Rammeverk

Med dette prosjektet ville jeg videreutvikle min kompetanse innen bruken av rammeverket Django, basert på programmeringsspråket Python.

2.2.1 Python

Python er et skriptingspråk utviklet av nederlandske Guide van Rossum. Den tidligste versjonen kom frem i 1996 og det har stadig vært i utvikling siden. Python har et formål om å gjøre kode leselig og gjenbrukbar, og har et mindre fokus på ren hastighet. Utover dette har Python et fokus på å gjøre utviklingen, og debuggings fasene av prosjekter så raskt som mulig [1, s. 11].

2.2.2 Django

Django er et web-applikasjons rammeverk som kommer levert med en 'batterier inkuldert' filosofi. Dette betyr at vanlig funksjonalitet som ofte blir benyttet i en web sammenheng blir levert med i grunnpakken til Django. Dette inkluderer bruker-system, URL routing, template system, administrasjons system og database object-relational mapper (ORM) [**?**].

På figur 2.1 kan man se en oversikt over hvordan Django håndterer et HTML request.

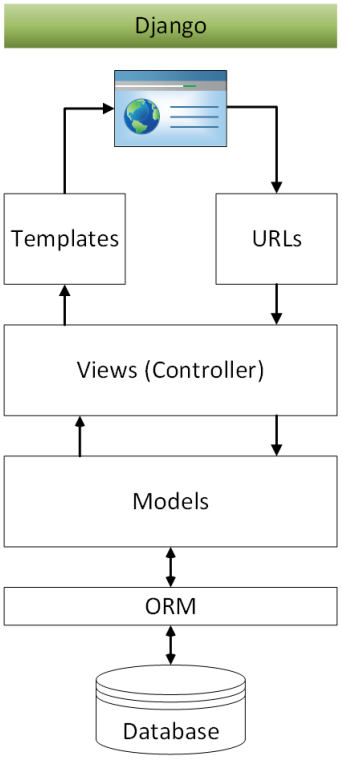
Først vil den tyde hvilke URL som kommer inn. Django inneholder et register over tilgjengelige URL som kan benyttes vha. regular expressions. Den vil sjekke dette registeret og finne hvilken funksjon (view) som tilhører den innkommende URL.

Den korresponderende funksjon i view vil nå håndtere data fra HTML requestet, dette innebærer å sjekke om det er et POST eller GET request, og hente ut eventuell data med henyn til dette.

Data vil bli hentet vha. Djangos Modeller, som er et objekt orientert metode å fremstille den tilkoblede databasen. Når det er klart hvilke data som skal hentes, vil Djangos ORM konvertere Djangos Database API til SQL Query. Den data som resulterer fra spørringen vil bli returnert tilbake til view.

Til slutt vil view samle data og generere den tilhørende template (html), som videre sendes tilbake til browseren.

FIGUR 2.1: Stegene Django kjører gjennom ved et HTML request.



2.3 Frontend

På klientsiden har det i hovedsak blitt benyttet 2 større

2.4 Verktøy

GitHub ble benyttet som versjons kontroll underveis i utviklingen. For hver nye feature som skulle introduseres, ble dette utviklet på en egen branch. Når utviklingen av en ny feature fungerte som planlagt, skulle den testes grundig før den ble pushet over på Master branch.

Google App Engine SDK. Google Cloud Platform kommer utstyrt med en SDK når man skal arbeide mot Google App Engine. Denne SDK inneholder et lokalt utviklingsmiljø som kjører på samme infrastruktur som det benyttet ved hostingen. Man får også publisering og administrasjons muligheter direkte vha. enkel GUI. Denne har hovedsakelig blitt brukt som et test område underveis i utviklingen, samt. Publisering av nyere versjoner etter hvert som nettsiden har blitt oppdatert.

JetBrains PyCharm har blitt brukt som IDE. Denne inneholder en bred rekke hjelpemidler for å gjøre utviklingsprosessen smidigere. Den støtter både utvikling av Django prosjekter og html/css/javascript. I tillegg har den et godt system for autocomplete av kode, og vil generere hint for forbedringer av bl.a. navn på funksjoner, indentering og sørger for at all kode benytter samme navn konvensjon.

LateX. Under rapportskrivingen har typesettingssystemet LateX blitt brukt får å generere dokumentet. Dette systemet hjelper med å skape en god utforming av hele dokumentet, som gir bedre muligheter for å fokusere på selve skrivingen. LateX gir et såkalt WYSIWYM (What You See Is What You Mean), som beskriver en form for skriving hvor det resulterende dokument bedre representerer den faktiske informasjonen man arbeider med.

Kapittel 3

Løsning

3.1 Krav

3.1.1 Ikke-funksjonelle krav

3.2 Designspesifikasjoner

3.3 Implementasjon

3.4 Admin

3.5 Database

Dette prosjektet benytter Django, og dermed vil dette rammeverket ta hånd om mye av arbeidet mot databasen. Django kommer utstyrt med en ORM (Object Relational Mapper), som håndterer overgangen fra Python klasseer til MySQL tabeller.

3.6 Testing og Validering

Det ble brukt en iterativ utviklingsmetode, og dette innebærer at testing skal foregå i hver syklus av utviklingsfasen. Dersom en ny funksjon blir introdusert, skal denne testes før man kan gå videre å jobbe på andre områder. Testingen har blitt gjennomført vha. Pythons Debugger, Pythons Logging bibliotek og Djangos debuggings modus.

3.6.1 Django Debugging

3.6.1.1 Django System Check

Kommandoen 'python manage.py check' kjører en bred sjekk av hele systemet [4]. Dette inkluderer blant annet:

Modellene. Går gjennom alle database modeller og sjekker om modeller, og felter er som de skal være, og reflekterer tabellene i databasen.

Admin. Ser på alle egendefinert Admin funksjoner og oversiktstabeller. Først og fremst sjekker at alle tabeller inneholder virkelig data, og at de grunnleggende admin funsjoner (Legge til, slette, redigere) fungerer.

Kompatibilitet. Denne vil se at koden benytter Djangos anbefalte funksjoner, og vil gi advarsel dersom en nyere versjon av Django har forandinger, og hva som bør endres for å gjøre koden kompitabel med nyere django versjoner.

Sikkerhet. Det blir utført en noe begrenset, men en såkalt 'low-hanging-fruit' sjekkliste. Dette er bl.a. en sjekk om at alle POST requests inneholder en Django spesifik csrf token (Cross-site request forgery).

Template. En gjennomgang av alle HTML som finnes til Templates folderene. Her blir det gjort en sjekk for å se om alle tags [1] blir benyttet på korrekt måte.

Debugging

Ved debugging ble editoren JetBrains PyCharm benyttet. Her kan man sette break points, og gå gjennom stegvis når koden treffer disse. Man vil da få en oversikt over hvilke data som befinner seg i variablene, og hvilke trinn som blir utført etter hver linje kode.

[1] Man kan utføre logikk i HTML i form av Tags. Disse kan f.eks. være variable, enkle IF setninger og løkker. Tags blir evaluert når den blir generert av serveren før den sendes til brukeren i form av ren HTML.[5]

Logging

Loggingen ble utført ved å bruke Python biblioteket 'Logging'. Her har man muligheten til å legge logge funksjoner rundt om i koden, og disse loggene kan differensieres i 5 nivåer [**?**]:

Critial Alvorlig feil, denne indikerer at programmet I seg selv kan være i en stand at den ikke kan fortsette uten inngrep.

Error Stor feil, programmet har ikke vært i stand til å kjøre en funksjon.

Warning Indikering på at noe uventet har forekommet, programmet skal fortsatte fungere.

Info Bekreftelse på at alt fungerer som det skal.

Debug Detaljert informasjon, skal benyttes når man skal diagnostiere et problem.

Disse loggene blir presentert i sanntid i det lokale utviklingsmiljøet, som gjør at man raskt kan identifisere hvor eventuelle feil forekommer. Når nettsiden blir koblet til gjennom Google App Engine, vil logger bli lagret til en egen modul på kontrollpanelet

Bibliografi

[1] Martin Engen. Bachelor rapport vår 2016. Technical report, Universitetet I Agder, 2016.

[2] URL https://www.inflectra.com/GraphicsViewer.aspx?url=Methodologies/Waterfall.xml&name=wordml://03000005.png. [Online; besøkt 14-November-2016].

[3] "what is paas?". URL https://azure.microsoft.com/nb-no/overview/what-is-paas/. [Online; besøkt 23-November-2016].

[] full stack python, django", . URL https://www.fullstackpython.com/django.html. [Online; besøkt 23-November-2016].

[4] system check framework". URL https://docs.djangoproject.com/en/1.10/ref/checks/. [Online; besøkt 22-November-2016].

[5] the django template language", . URL https://docs.djangoproject.com/en/1.9/topics/templates/#the-django-template-language. [Online; besøkt 22-November-2016].