Boligformidling

Sluttrapport

2014

Alexander Gård, s198585, 1. år IT

Arlen Syver Wasserman, s193956, 1. år IT

Petter Soland Wold Gjerstad, s181110, 3. år Dataingeniør



Innhold

[Om oppgaven 1](#_Toc387928105)

[Rammeverk for applikasjon 2](#_Toc387928106)

[Om programmet 2](#_Toc387928107)

[Ramme 2](#_Toc387928108)

[Programstruktur 3](#_Toc387928109)

[Produktdokumentasjon 3](#_Toc387928110)

[Avvik ifra oppdragsgivers ønsker 3](#_Toc387928111)

[Datastruktur 3](#_Toc387928112)

[Systemkrav 4](#_Toc387928113)

[Java-versjon 5](#_Toc387928114)

[Prosessdokumentasjon 5](#_Toc387928115)

[Arbeidsfordeling 5](#_Toc387928116)

[Alex: 6](#_Toc387928117)

[Arlen: 6](#_Toc387928118)

[Petter: 7](#_Toc387928119)

[Utviklingsverktøy 7](#_Toc387928120)

[Prioritering 7](#_Toc387928121)

[Testrapport 8](#_Toc387928122)

# Om oppgaven[[1]](#footnote-1)

*“Dere får nå i oppdrag å lage et datasystem for et firma som driver med boligformidling for utleiemarkedet. Firmaets kunder vil være utleiere og boligsøkende. Ved hjelp av applikasjonen, der utleieboliger og boligsøkende er registrert, skal kundebehandlere formidle kontakt mellom disse to aktørene i markedet.*

*Utleiere vil få sine boligtilbud presentert og får derved hjelp til å nå potensielle leietakere. Boligsøkende registrerer på sin side sine ønsker og krav til en eventuell bolig, samt personlige opplysninger som kan ha betydning for utleieren, og får derved hjelp til å finne en egnet bolig.*

*Ved å matche ledige boliger mot de boligsøkendes beskrivelser skal firmaet informere sine kunder om aktuelle leietakere og boliger. Hvordan denne informasjonen skal gis, har ikke oppdragsgiver noen klar formening om, men håper dere kan finne en passende løsning.*

*Firmaet skal registrere alle leieforhold som opprettes, slik at boligen og leietakeren da ikke lenger vil være registrert som henholdsvis ledig og boligsøkende.*

*For at firmaet skal kunne videreutvikle sin forretningsidé og strategi, ønsker de at programmet skal lagre all historikk nyttet til leiekontrakter som har blitt inngått i firmaets regi.”*

# Produktdokumentasjon

//En **produktdokumentasjon** som beskriver hvordan systemet (programmet) er bygget opp og hvordan ting virker sammen. Dette "dokumentet" er til for den som vil sette seg inn i systemet, f.eks. for å kunne vedlikeholde det, videreutvikle det, endre det, eller evaluere det ved sensur!

// Hvis dere ikke har rukket å innfri alle ønskene oppdragsgiveren har til programmet, må dere her skrive noe om hvilken funksjonalitet dere har prioritert og hvorfor, og hva som bør gjøres videre med programmet slik at andre programmerere kan fortsette der dere slapp.

// Har dere derimot lagt inn funksjonalitet utover oppdragsgiverens ønsker, bør dere også skrive noe om det.

## Om programmet

Applikasjonen vil fungere som en portal for både kunder og kundebehandlere hvor kunder som ønsker å leie ut sine leiligheter kan registrere disse, og kunder som ønsker å leie leiligheter kan klikke og søke etter disse registrerte leilighetene. Igjennom en kundebehandler kan man opprette leiekontrakter mellom utleier og boligsøker.

Programmet kan eksempelvis installeres på datamaskiner i firmaets lokaler hvor besøkende kunder kan klikke igjennom katalogen, legge ut en ny annonse, melde seg opp til varsler. En kundebehandler har også denne applikasjonen installert og kan opprette kontrakter ut fra forespørsler ifra kunder. Det er slik at man må igjennom en kundebehandler for å opprette leiekontrakter.

Da kravene i oppgaven “Boligformidling” henviser til at *“... ansatte i firmaet skal formidle kontakt mellom disse to aktørene i markedet*”, er systemet kun begrenset proporsjonalt i forhold til hvor mange kundebehandlere firmaet har.

## Rammeverk / Utgangspunkt

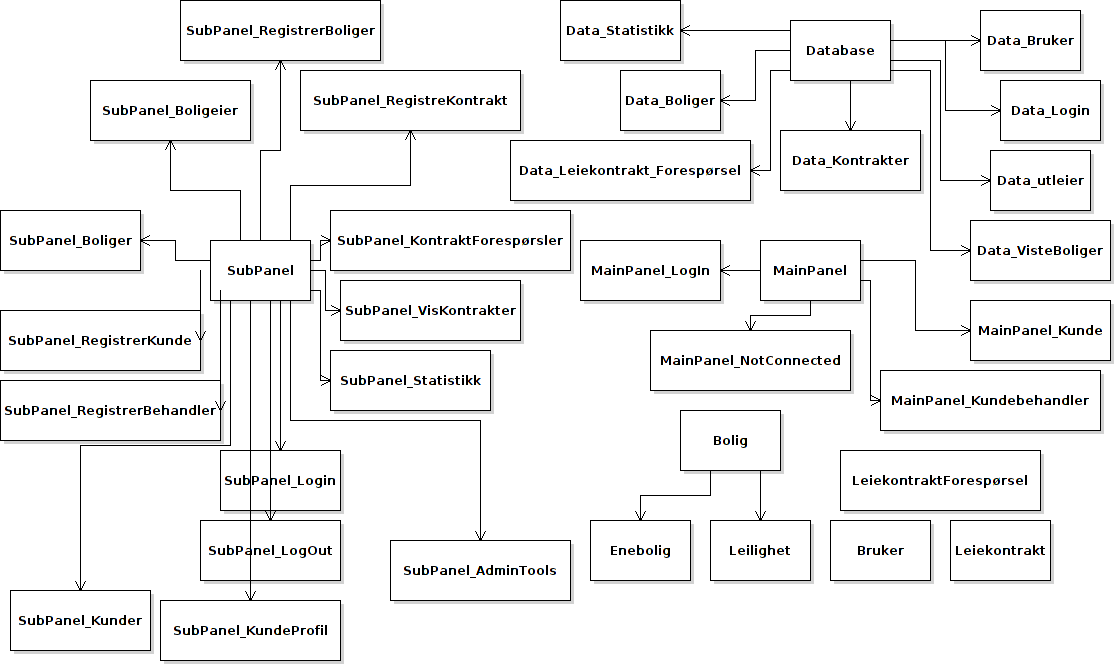
Siden applikasjonen skal utvikles for et firma vil vi legge vekt på at applikasjonen skal kunne benyttes av flere personer. Utviklingen vil også skje med tanke på at applikasjonen vil bli kjørt fra flere operativsystemer. Vi har gått ut ifra at det vil bli lagret data for X antall kunder, og applikasjonen har derfor blitt utviklet med tanke på skalerbarhet slik at hvis lagring av data skal bli utført lokalt på hver pc som kjører programmer vil dette medføre at hvert program har tilgang til ulik informasjon.

Vi har valgt å løse dette problemet ved å lagre dataen til applikasjonen i en SQL database. Selv om PC-en som applikasjonen kjører på vil være avhengig av internett tilgang, mener vi at det fortsatt er mer hensiktsmessig å løse problemet ved hjelp av en database.

Applikasjonen vil bli utviklet med tanke på skalerbarhet. Vi har også gått ut fra at applikasjonen vil bli kjørt fra flere operativsystemer, og applikasjonen vil bli utviklet med tanke på dette.

## Programstruktur

I bildene nedenfor vises oversikten over klassehierarkiet til applikasjonen. For en mer detaljer beskrivelse over klassene i applikasjonen, vennligst se den vedlagte Javadoc-filen.



For å gjøre program koden mest oversiktlig og justerbar har vi valgt å dele programmet i flere klasse typer. Database klassene og dens arvinger Data\_XXX inneholder SQL kommandoene som blir kjørt mot en MySQL server, og returnerer bearbeidet informasjon ifra databasen.

Programmet er også delt inn i JPaneler for å vise rett til informasjon, til rett tid.

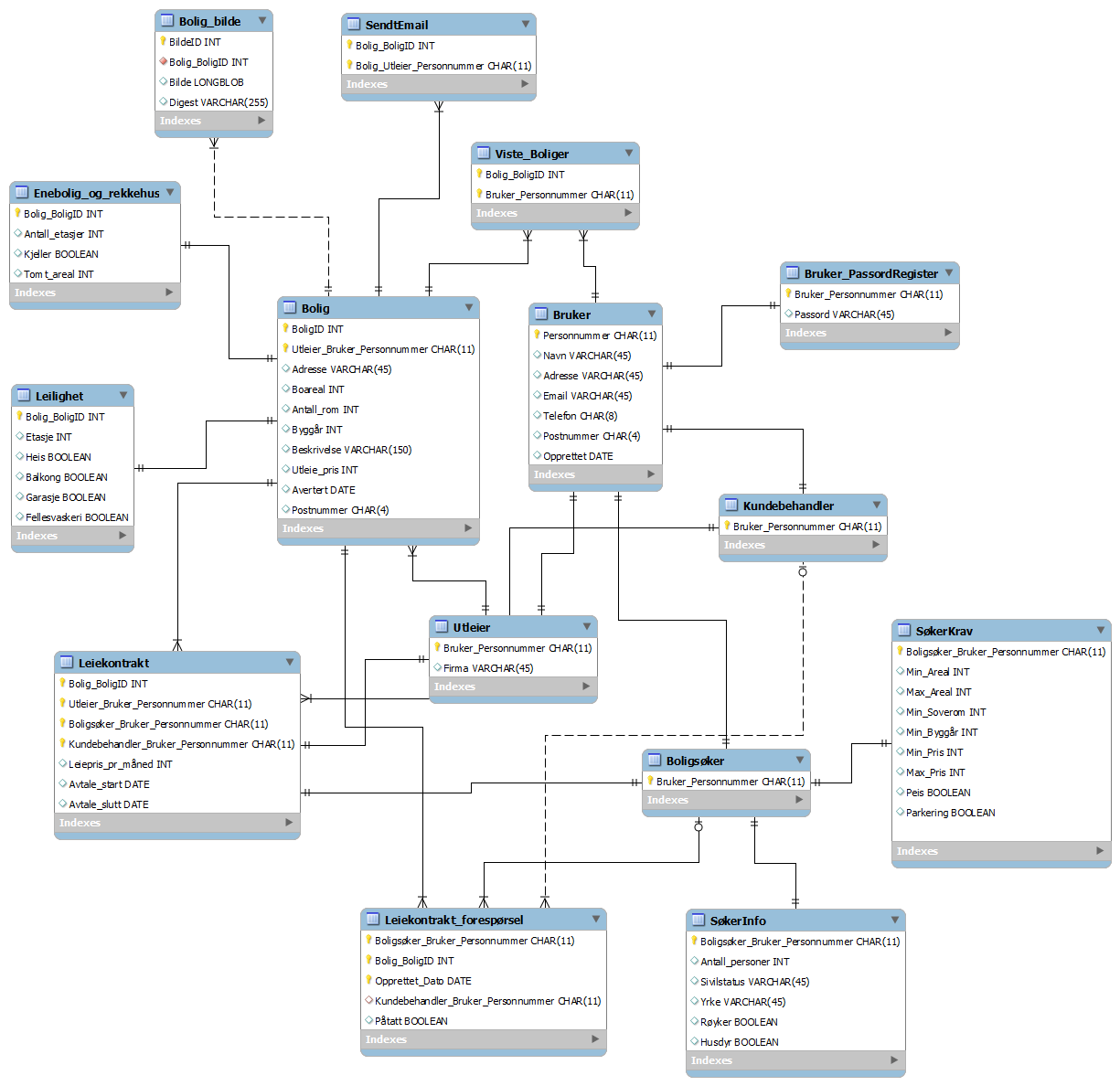
* «MainPanel», som exteder JPanel har som rolle å fylle hele vinduet. Ettersom «MainPanel» er abstrakt, vil arvingene ha som oppgave å hoste forsekjellige «SubPanel» i sin Tab Pane.
* «SubPanel», som extender JPanel, er paneler som ligger inni et «MainPanel» og er der hvor hoved funksjonalitetene til programmet ligger.
* Det finnes mange forskjellige «MainPanel» og mange forskjellige «SubPanel».

Her ser du en oversikt over hvilke «MainPaneler» som «hoster» hvilke «SubPaneler».

## Diagram_MP_SP_Relasjon.png

## Datastruktur

Selv om oppgaven var basert på at applikasjonen skulle utvikles med lokal lagring av data, ved hjelp av en liste struktur, så har vi valgt å bruke en annen fremgangsmåte. Ut ifra vårt inntrykk etter å ha lest gjennom oppgaven, virket det som applikasjonen skulle bli utviklet med tanke på flere brukere. En lokal datastruktur vil i dette tilfellet føre til problemer, ettersom dataen ikke vil være synkronisert mellom de forskjellige brukerne. For å løse dette problemet valgte vi å lagre dataene våre i en database. Dette løste problemet med synkronisering av data, samtidig som det skaper en klar datastruktur som er lett å jobbe med og har mulighet for skalering. Som databasehåndteringssystem har vi valgt å bruke MySQL, ettersom gruppen har mest erfaring med dette systemet.



Figuren ovenfor viser et EER (Enhanced Entity Relationship)-diagram over databasen hvor dataen vil bli lagret. Tabellen SendEmail er ikke tatt i bruk av applikasjonen, men hvis det skulle tenkes at applikasjonen skal utvikles videre, vil denne tabellen være nødvendig.

## Stuff

For å gjøre program koden mest oversiktlig og justerbar har vi valgt å dele programmet i flere klasse typer. Database klassene og dens arvinger Data\_XXX inneholder SQL kommandoene som blir kjørt mot en MySQL server, og returnerer bearbeidet informasjon ifra databasen.

Allerede i kravspesifikasjonen valgte vi å dele SQL databehandlingen mest mulig ifra GUI klassene. Av flere grunner:

1. Videreutvikling: En framtidig programmerer trenger ikke direkte å forholde seg til databasen og dens struktur, men heller bruker metoder og objekter til å manipulere databasen.
2. Samkjøring mellom utviklerne: For å minske dobbelt koding av SQL kommandoer er det bedre å samle like SQL kommandoer slik at et gruppemedlem ikke trenger å sette seg inn i alle database kommandoene, men heller fokusere på sine oppgaver.
3. Oversikt: Ved å se igjennom Data\_XXX klassene får man en god oversikt over hvordan databasen fungerer, og hvordan man selv skal kunne lage sine egne SQL kommandoer.

## Avvik ifra oppdragsgivers ønsker // ting som ble lagt til, og ting som ble tatt vekk

Datastruktur //se datastruktur ovenfor

Lagt til:

* Oppgradert søkefunksjon
* Kundebehandler
* Felt for bruker(Opprettet dato)

Tatt vekk:

* Boligsøker info, arbeidsforhold
* Krav til bolig, beliggenhet
* Utskrift av alle registrerte boligsøkere(med krav)

## Prioritering

Applikasjonen ble større enn det vi først hadde gått ut fra, så det ble nødvendig å dele opp i arbeidsoppgavene i to grupper, «prioritert» og «ikke-prioritert». «Prioritert» gruppen inneholder kjernen i applikasjonen, og alle funksjonaliteter som er absolutt nødvendig for at programmet skal kunne tas i bruk.

Prioriterte funksjoner:

* Registrering av bruker
* Bruker kan registrere bolig
* Bruker kan søke etter bolig med kriterier
* Bruker opprette kontrakt forespørsel
* Kundebehandler kan opprette kontrakt
* Kundebehandler kan ta sikkerhetskopi av database
* Kundebehandler kan se grunnleggende statistikk

I «ikke-prioritert» gruppen ligger funksjonaliteter som applikasjonen ikke er avhengig av, men bør bli implementert hvis man har mulighet for det.

Ikke-prioriterte funksjoner:

* Utvidet statistikk med grafer
* Sende mail til søkere hvis nye tilbud dukker opp
* Threading av potensielt lange operasjoner
* Bilderegister av hus og bruker
* Web registrering
* Kryptering/Hashing av passord
* Dynamisk bildelagring, dvs. at bilder hentet fra databasen blir lagret slik at de ikke trenger å bli lastet ned hver gang
* Printe diverse data til papir, for eksempel en kontrakt
* Sende passord på mail hvis brukeren har glemt passordet sitt

Vi fikk implementert alle de prioriterte- og flere av de ikke-prioriterte funksjonalitetene, men applikasjonen mangler fortsatt et par funksjonaliteter før den kan bli betegnet som et fullstendig produkt.

Nedenfor ligger en liste med funksjonaliteter som bør bli lagt til applikasjonen, hvis applikasjonen skulle blitt videre utviklet.

* Bedre GUI skalering
* Sende email hvis kunde har glemt passord
* Sende boliginformasjon til kunder på email
* Visning av kontrakter knyttet til en bolig, til utleiere.
* «Threading» av lengere operasjoner

## Systemkrav

|  |  |  |
| --- | --- | --- |
| **Type** | **Krav** | **Anbefaling** |
| Lagringsplass | 2 MB tilgjengelig på harddisk. | Kun for kundebehandler:  Lagringsplass for sikkerhetskopi vil variere etter antall kunder/boliger som er registrert. |
| RAM | 256 MB. | 256 MB eller mer |
| Prosessor | Pentium 2 266 MHz prosessor. | Pentium 2 266 MHz prosessor eller raskere |
| Skjermoppløsning | 1366 x 768 piksler. | 1366 x 768 piksler eller høyere |
| Operativsystem | Må oppfylle kravene for installasjon av Java 7, se link for mer informasjon.  <https://www.java.com/en/download/help/sysreq.xml> | Ingen anbefalinger |
| Internett | Dial-Up | Bredbånd/Fiberoptisk kabel |

## Java-versjon

Vi har valgt å utvikle applikasjonen vår i Java 7. Sammenlignet med Java 6 og eldre versjoner, har Java 7 generelt bedre stabilitet, hastighet og sikkerhet som alle er viktige egenskaper for applikasjonen vår. En mulighet kunne også vært og bruke Java 8, men ettersom denne versjonen fortsatt er i utviklingsfasen har vi valgt å se bort fra dette alternativet. Ettersom det vil bli brukt funksjonalitet som er spesifikt for Java 7 og høyere vil ikke applikasjonen la seg kjøre på en maskin som kun har Java 6 installert.

# Prosessdokumentasjon

## Arbeidsfordeling

Det viste seg at noen av oppgavene i arbeidsfordelingen som ble oppgitt i kravspesifikasjonen tok mye lengere tid å utføre enn andre. I tillegg til at noen av gruppemedlemmene startet litt sent med å utføre arbeidsoppgavene som ble gitt.

På grunn av dette ble andre gruppemedlemmer nødt til å påta seg flere oppgaver. Dette var nødvendig for at prosjektet skulle komme seg i mål, men resultatet ble at arbeidsfordelingen ble skjevt fordelt. Til ettertanke ville det vært mer effektivt å jobbe jevnt gjennom hele prosjektet med arbeidsoppgavene, slik at hvert gruppemedlem kan konsentrere seg om sine oppgaver.

Nedenfor ligger oversikten over hva som ble gjort gjennom prosjektoppgaven.

### Alexander:

Databasestruktur

Data\_Bruker

Data\_Kontrakter

Data\_Statistikk

Database

GraphPanel

Leiekontrakt

LeiekontraktForespørsel

Main

MainPanel\_Kunde

MainPanel\_Kundebehandler

MainPanel\_Login

MainPanel\_NotConnected

SubPanel\_AdminTools

SubPanel\_KontraktForespørsler

SubPanel\_LogIn

SubPanel\_LogOut

SubPanel\_RegistrerBehandler

SubPanel\_RegistrerKontrakt

SubPanel\_RegistrerKunde

SubPanel\_VisKontrakter

SubPanel\_Statistikk

Table

Vindu\_Main

### Arlen:

Programskjelletstruktur

Bolig

Enebolig

Leilighet

Bruker

Data\_Boliger

MainPanel

SubPanel

SubPanel\_Boligeier

SubPanel\_Boliger

SubPanel\_RegistrerBoliger

### Petter:

Databasestruktur(Mindre endringer på tabeller)

SubPanel\_KundeProfil

SubPanel\_Kunder

Data\_Bruker(lagt til tre metoder)

Data\_boligsøker

Data\_Leiekontrakt\_Forespørsel

Data\_Viste\_Boliger

Data\_Utleier

# Testrapport

For å kvalitetssikre applikasjonen har vi systematisk sjekket hvert panel i programmet for eventuelle feil. Testene ble utført med hensyn på bruker input og database koblingen. Ettersom begge disse elementene gir stort rom for feil, var det viktig å teste begge nøye.

For å forhindre feil i databasekoblingen har Database.java metoder som gjør det lett for å fange opp eventuelle feil. Klassene som «extender» Database.java må fange opp feilene som metodene forårsaker. Dette sørger for at problemene blir oppdaget med en gang, og ved bruk av «System.out.println(«feil i metodenavn: » + exception)» vil kilden til problemet også raskt bli funnet.

For å validere brukerinput har vi laget metoder som sjekker om feltene i vinduet inneholder gyldig informasjon. Hvis ikke vil brukeren bli vist en feilmelding, slik at han/hun kan rette opp feilen.

På grunn av størrelsen til applikasjonen har vi dessverre ikke klart å rette opp i alle «bugsene». Hvis applikasjonen eventuelt skulle utvikles videre, anbefales det at «bugsene» i listen nedenfor rettes opp først.

Liste over «bugs»:

* GUI skalering,
* Gjenoppretting av en database med eksisterende data, kan forårsake problemer.

## Ytelse

Vi har testet programmet på flere maskiner og vi kom fram til at programmet brukte ca. 50 - 80 MB RAM. Ellers fungerer alle funksjoner uten noen lange blokkerende operasjoner, forutsatt at man har en stabil, lav-latens forbindelse til serveren på HIOA, hvor databasen ligger.

1. Sitatet er hentet ut fra oppgaveteksten [↑](#footnote-ref-1)