FORORD

Tusen takk til Dyresenteret Ålesund for tilliten og ideen til oppgaven. Samt godt samarbeid og gode tilbakemeldinger.

En takk til Kjell Inge Tomren for veiledning og støtte.

En takk til Kenneth Vassbakk for hjelp og forståelse med JavaScript og jQuery.

INNHOLD

NTNU I ÅLESUND
BACHELOROPPGAVE

SAMMENDRAG

Verden vi lever i blir stadig mer digitalisert, og folk forventer enkle og smarte løsninger. Dyresenteret Ålesund har en suksess med bonuskort på enkelte varer i butikken, men får stadig tilbakemeldinger på at de fysiske papirkortene er vanskelig å huske å ha med seg, og burde bli digitalisert.

I samarbeid med Dyresenteret har jeg laget et webbasert system, som gjør det enkelt å registrere kunder, legge til bonuskort til kunden, og legge til kjøp på bonuskortene, slik at kunden enkelt kan spare opp til å få en gratis vare.

Ved bruk av et oversiktlig og enkelt brukergrensesnitt kan tiden brukt på å registrere et kjøp på bonuskort bli en brøkdel av det den var før.

Bonuskortsystemet er utviklet i HTML, CSS, PHP, MySQL og JavaScript/jQuery

TERMINOLOGI

Begreper

ER	(Entity Relationship) er et visuelt språk for å beskrive datamodeller ved hjelp av ER-diagrammer.
Klient	En nettleser eller en mobiltelefon er eksempel på klient
Server	En datamaskin som er på nett, oppfører seg som en tjener med en programvare.
Virtuell Server	Brukes som en lokal server på privat datamaskin før opplasting til nettbasert server
Primærnøkkel	En utvalgt kolonne i en databasetabell som brukes til identifisering av forskjellige rader i tabellen
Perl	Perl er et blokkstrukturert språk som JavaScript.
CGI	Common Gateway Interface. Standardprotokollen for å kjøre programmer på en webserver.
IDE	Et integrert utviklingsmiljø som kan brukes til å utvikle andre programvarer og verktøy
Backlog	En samling av oppgavene som gjenstår
Raspberry Pi	En datamaskin bygget på et enkelt kretskort
String	En sekvens av tegn (bokstaver/tall)

Forkortelser

HTML	Hyper Text Markup Language
CSS	Cascating Style Sheet
PHP	Personal Home Page (opprinnelig) men star nå for Hypertext PreProcessor
IKT	Informasjons- og kommunikasjonsteknologi
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
SSL	Secure Sockets Layer
TLS	Transport Layer Security
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
SQL	Structured Query Language
MySQL	My Structured Query Language
MAMP	macOS, Apache, MySQL and PHP
FTP	File Transfer Protocol
GUI	Graphical User Interface
URL	Uniform Resource Locator
VM	Virtual Machine
RPI	Raspberry Pi

INTRODUKSJON

1.1 Bakgrunn

Dyresenteret Ålesund er Sunnmøres største dyrebutikk, lokalisert i Breivika i Ålesund. Butikken tilbyr bonuskort til sine kunder, på forskjellig type varer. For eksempel hver 5. pose med godbiter gratis, der kunden får kortet stemplet og signert ved hvert kjøp.

Bonuskortene er en stor hit blant kundene, men kortene har en tendens til å forsvinne eller bli gjenglemt. Dagens lommebøker er ikke store nok til å huse x-antall kort og bonuskort fra forskjellige forretninger.

Dyresenteret har også en annen type bonuskort levert fra en fôrleverandør, som på grunn av størrelsen på kortene, blir oppbevart i et kartotek bak disken, for de kundene som ikke ønsker å ha det med seg. Det er mange kort og det kan til tider ta flere minutter å finne kundens kort, selv om de er kategorisert etter alfabetet.

Derfor ønsket jeg å utvikle en enkel webapplikasjon som kan erstatte de fysiske bonuskortene, og lette frustrasjonen både kunder og de ansatte føler på, ved gjenglemte og bortkomne kort.

Som ansatt på Dyresenteret ble dette en personlig og givende oppgave for meg.

Ved å jobbe med dette prosjektet håper jeg å kunne få en bedre forståelse av alle aspektene med å utvikle et system for en kunde, utvikle meg som programmerer og å bedre forstå hvilke tiltak man må ta for å lage en sikker løsning for alle parter.

1.2 Prosjektbeskrivelse

Oppgaven min ble å utvikle en applikasjon som kan gjøre det lettere for Dyresenteret å håndtere bruken av både egendefinerte bonuskort, men også fôrleverandørenes bonuskort. Systemet skal være enkelt å forstå, å bruke, og stå til dagens krav til informasjonssikkerhet.

Jeg vil sette fokus på brukervennlighet og et dynamisk design, som minimerer risikoen for feil og unødvendig tidsbruk.

1.2.1 Nøkkelegenskaper

Systemet skal kunne enkelt registrere en ny kunde, med de samme parameterne som de allerede eksisterende bonuskortene krever; navn, adresse, telefonnummer og epost.

Når en kunde kjøper noe som kan registreres på bonuskort vil den ansatte kunne søke opp vedkommende, velge et eksisterende bonuskort, eller lage et nytt, for så å registrere varen med verdi og dato.

Når en kunde har kjøpt nok produkter til å få et gratis produkt, blir bonuskortet arkivert. Dette for å kunne innhente informasjon på de varene som butikken kan få kreditert av leverandør, og også for å ha noe å vise til om en kunde er usikker på når/om h*n har mottatt et gratis produkt.

1.2.2 Sikkerhet

Systemet som utvikles skal operere på en webside på det åpne internettet. Informasjonen vi lagrer om kundene må beskyttes i henhold til lover og regler, i tillegg til at databasen og dens innhold må kunne sikres for sabotasje.

1.2.3 Studiefelt

- Databasedesign og webdesign
- Uvikling av webside med responsivt design
- Programmering med bruk av PHP og kontakt med databasen ved hjelp av SQL
- Sikre systemet mot vanlige angrep mot web-applikasjoner.

1.2.4 Restriksjoner

Språket på websiden vil være på norsk, da alle nåværende og kommende ansatte ved Dyresenteret må snakke og forstå norsk. Websiden er laget spesifikt for Dyresenterets formål, men kan med enkle endringer gjøres om til engelsk, hvis systemet skal utvikles og brukes videre på andre arenaer.

1.3 Målgruppe

Målgruppen for systemet er først og fremst de ansatte ved Dyresenteret, men også kundene som bruker bonuskort, da systemet vil forenkle handleopplevelsen for dem, selv om de selv ikke er i fysisk kontakt med systemet.

1.4 Akademisk bakgrunn og rolle

Bakgrunnen min er et 3-årig studieforløp som Dataingeniør hos NTNU Ålesund, med gjennomsnittlige datakunnskaper, og lite kunnskap om programmering fra før.

Som eneste person på gruppen har jeg ikke noe spesifikk rolle, jeg må fungere både som gruppeleder, sekretær, systemutvikler og designer.

Veilederen min for dette prosjektet er Kjell Inge Tomren.

Oppdragsgiver/Kunde for prosjektet er Dyresenteret Ålesund, representert av Ine Arlene Torvik Totland.

1.5 Rapportens oppbygging

Rapporten er delt inn i syv hoveddeler

- Introduksjon: Inneholder prosjektbeskrivelse, bakgrunnen og målgruppen til prosjektet.
- Teoretisk grunnlag: Introduserer og definerer utrykk som er essensielt for rapporten.
- Materialer og metode: En bekrivelse av arkitekturen, prosessene, og test-metodene til prosjektet.
- Resultater: Inneholder en beskrivelse av sluttproduktet
- Drøfting: Inneholder drøfting av valgt løsning og andre alternativer
- Konklusjon: Problemer som er støtt på underveis, konklusjon av det ferdige produktet og videre utvikling.
- Referanser: Inneholder en liste med referanser og kilder

2 TEORETISK GRUNNLAG

Dette kapittelet tar for seg den teoretiske bakgrunnen som danner grunnlaget for valg tatt underveis for å få det ferdige sluttproduktet.

2.1 Lover og regler

2.1.1 Personopplysningsloven

En personopplysning er alle opplysninger og vurderinger som kan knyttes til deg som enkeltperson. En typisk personopplysning kan være navn, adresse og fødselsnummer. Noen personopplysninger er mer personlige enn andre.[1]

Alle som behandler personopplysninger må opptre i samsvar med disse prinsippene:

Lovlig, rettferdig og gjennomsiktig

Det må finnes et rettslig grunnlag for en planlagt behandling av personopplysninger. Gjennomsiktig betyr at bruken av personopplysninger skal være oversiktlig og forutsigbar for den opplysningen gjelder. Dette setter enkeltpersonen i stand til å bruke sine rettigheter og ivareta sine interesser.

Formålsbegrensning

Ethvert formål med behandling av personopplysninger skal identifiseres og beskrives presist slik at alle berørte har samme forståelse av hva personopplysningene skal brukes til.

Dataminimering

Dataminimering innebærer å begrense mengden innsamlede personopplysninger til det som er nødvendig for å realisere formålet med innsamlingen. Dersom personopplysninger ikke er nødvendige for å oppnå formålet, skal man heller ikke samle dem inn.

Riktighet

Personopplysninger som behandles inn skal være korrekte, og skal om nødvendig oppdateres. Det betyr at den behandlingsansvarlige må sørge for å straks slette eller rette personopplysninger som ikke er rett ut i fra de formålene de er samlet inn for.

Lagringsbegrensning

Personopplysninger skal slettes eller anonymiseres når de ikke lenger er nødvendige for formålet de ble innhentet for.

Integritet, konfidensialitet og tilgjengelighet

Personopplysninger skal behandles slik at opplysningenes integritet, konfidensialitet og tilgjengelighet beskyttes. Den behandlingsansvarlige må sørge for å iverksette tiltak mot utilsiktet og ulovlig ødeleggelse, tap og endringer av personopplysninger.

Ansvarlighet

Virksomheten må kunne dokumentere at den har gjennomført tiltak for å etterleve personvernforordningen. Virksomheten må opptre proaktivt og etablere alle nødvendige organisatoriske og tekniske tiltak for å sikre at regelverket etterleves til enhver tid. [3]

2.1.2 GDPR (General Data Protection Regulation)

I 2018 fikk Norge en ny personopplysningslov. Loven består av internasjonale regler og EUs personvernforordning (GDPR). Dette er et sett regler som gjelder for alle EU/EØS-land. Et sammendrag av GDPR tilsier at bedrifter ikke kan prosessere personopplysninger uten at de har fått samtykke av personen informasjonen gjelder. [5]

En privatpersons rettigheter om bruk av ens personopplysninger:

- Hva formålet med innsamlingen av opplysningene er
- I hvilken anledning opplysningene kommer til å benyttes
- Hvor lenge opplysningene kommer til å bli lagret
- Hvem vil opplysningene bli delt med
- Grunnleggende personvernsrettigheter
- Om opplysningene overføres til andre land i Europa
- Muligheter til å trekke tilbake samtykke
- Rett til å sende inn klage
- Tjenestens kontaktinformasjon [4]

2.1.3 Samtykke

En virksomhet kan behandle personopplysninger dersom den gar innhentet gyldig samtykke fra personen eller personene det gjelder. For at et samtykke skal være gyldig, må det være:

- Frivillig
- Spesifikt
- Informert
- Utvetydig
- Gitt gjennom en aktiv handling
- Dokumenterbart
- Mulig å trekke tilbake like lett som det ble gitt

Når det gjelder selve samtykket, er det ingen formkrav. Samtykket kan gis skriftlig, muntlig, gjennom bevegelser eller på andre måter. Det eneste kravet er at samtykke må kunne dokumenteres. [14]

2.1.4 Universell utforming av IKT

Universell utforming handler om å utforme omgivelsene slik at vi tar hensyn til variasjon i funksjonsevne hos innbyggerne, inkludert personer med nedsatt funksjonsevne.

Likestillings- og diskrimineringsloven (lovdata.no) stiller krav om universell utforming til offentlige og private bedrifter.

IKT-løsninger i arbeidslivet er ikke omfattet av kravene. Et system du bare har tilgang til som ansatt i bedriften, er bedriftsinternt. [6]

2.2 Informasjonssikkerhet

En oppsummering av hvilke sikkerhetstiltak som er tatt i bruk, og hvilke potensielle trusler et websystem av denne typen kan stå imot.

2.2.1 Hashing av passord

Hash funksjoner er matematiske algoritmer brukt til å bekrefte identiteten til en spesiell melding og bekrefte at innholdet ikke har blitt endret. Hash algoritmer blir brukt til å lage hash verdier, ved å konvertere meldinger med variabel lengde til en verdi med fast lengde. Dette er et fingeravtrykk av den originale meldingen som blir sammenlignet med mottakerens lokalt kalkulerte hash av samme melding. Hvis begge verdier er identiske etter overføring, har meldingen blitt mottatt uten modifikasjoner. Hash funksjoner er en en-veis operasjon der den samme meldingen alltid vil gi samme hash verdi, men hash verdien alene kan ikke brukes for å avsløre innholdet i meldingen.

En nylig oppdaget angrepsmetode kalt *rainbow cracking* har laget usikkerhet om styrken til bruk av hashede passord. Hvis angriperne får tilgang til en fil med hashede passord kan de til slutt klare å avsløre brukernes passord. Passord som er laget fra vanlige ord eller dårlig sammensatt kan lett bli avslørt.

For å forsvare seg mot et slikt angrep må man beskytte filen med hashede passord, og ha en begrenset mengde forsøk på å forsøke å logge seg inn. Man kan også bruke en metode kalt passord hash salting. Ved salting legger man til litt tilfeldig data der passordet blir hashet. [10]

Når man bruker password_hash i PHP vil det bli generert et tilfeldig salt. [9]

2.2.2 HTTPS

HTTPS er en sikrere utgave av HTTP som er kommunikasjonsprotokollen som brukes for å utveksle informasjon på Internett. Kommunikasjon over HTTP er ukryptert og dermed åpen for at uvedkommende kan endre data eller stjele personlig informasjon som sendes.

HTTPS kommuniserer også over HTTP, men informasjonen som utveksles krypteres med «Transport Layer Security», før det sendes over den ikke-sikre HTTP-protokollen. Dette beskytter nettstedet og sluttbruker mot avlytting og angrep.

SSL er et lag som eksisterer mellom den råe TCP/IP protokollen og applikasjonslaget. SSL legger til flere funksjoner til nettverksstrømmen:

- Godkjenning av server, ved bruk av digitale signaturer
- Godkjenning av klienten, ved bruk av digitale signaturer
- Datakonfidensialitet ved bruk av kryptering
- Dataintegritet ved hjelp av meldingsautentiseringskoder

SLL beskytter mot «man-in-the-middle» angrep ved å bruke digitale sertifikat for å la webbrukeren lære det validerte navnet på nettstedet. [13]

Når man får kontakt med en server som bruker HTTPS ser man et ikon av en hengelås ved siden av adressen i URL-baren, som viser at dette er en sikker forbindelse.

Figur 2.1: URL til nettsiden som viser at HTTPS blir brukt

2.2.3 Kjente angrep og trusler

Man-in-the-middle
En rekke angrep der en person fanger en kommunikasjonsstrøm og forsøker å overbevise begge legitime parter om at han er den andre kommunikasjonspartneren. Noen man-in-the-middle angrep involverer krypteringsfunksjoner. [10]

SQL-injection
I et web-grensesnitt er det ofte et tekst- eller søkefelt som brukeren har tilgang til.
SQL-injection er et angrep som utnytter metategn i SQL (for eksempel ‘ # --) ved at uhåndterte metategn blir en del av SQL spørringen.
SQL-injection kan skje når utviklere unngår å korrekt validere bruker input før den blir brukt til å sende en spørring til en database.

String name = request.getParameter(“username”);
String pwd = request.getParameter(“password”);

query = “SELECT * FROM users WHERE username=’” + name + “’ AND password=’” + pwd + “’”;

I dette eksempelet vil programmet hente brukernavn og passord fra input parameterne, bygge en SQL-spørring og etterhvert sende spørringen til databasen. Så hvis brukeren skriver inn «Jens» som brukernavn og «hei123» som passord vil spørringen se slik ut:

SELECT * FROM users WHERE username=’Jens’ AND password=’hei123’;

Denne spørringen vil kjøre uten feil.
En måte å utføre et SQL-injection angrep på er å benytte --, som er kommentartegn i mange dialekter av SQL. Hvis man skriver Jens’ --, som brukernavn og lar passordfeltet være tomt, vil spørringen se slik ut:

SELECT * FROM users WHERE username=’Jens’—‘AND password=’’;

Hvis Jens eksisterer som et brukernavn i databasen vil man kunne logge inn med dette brukernavnet uten å bruke passordet. Her har man altså brukt SQL-metategnene for å avslutte en streng og kommentering til å fjerne passordsjekken.

En annen måte å utføre et SQL-injection angrep på er å benytte noe som alltid evaluerer ti «true», ved å for eksempel skrive inn ‘ OR ‘1’=’1’ som både bruker navn og passord.

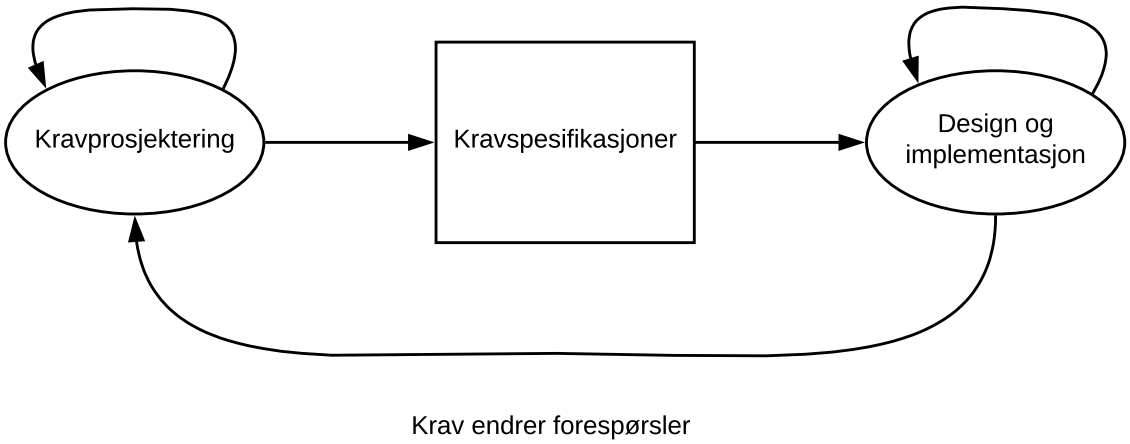
Hvis man har litt kjennskap til strukturen på databasen kan man slette innhold i databasen ved å for eksempel benytte ‘; DELETE FROM users—som brukernavn. [10]

2.3 Standarder og metoder

Denne delen vil beskrive og dokumentere standarder som er brukt i dette prosjektet

2.3.1 Agile metoder

På 1980- og tidlig 1990-tallet var det utbredt å bruke nøye planlegging før utvikling av et system. Dette synet kom fra programutviklere som var ansvarlig for å utvikle store systemer for eksempelvis luftfart og regjering. Slik plan-drevet utvikling involverer en stor mengde planlegging, designing og dokumentering av systemer.



Figur 2.2: Plan-drevet utvikling

Men når en slik utviklingsmetode brukes på små og mellomstore systemer i utvikling, blir det brukt mer tid på hvordan systemet skulle ha blitt laget, enn å faktisk lage og teste det.

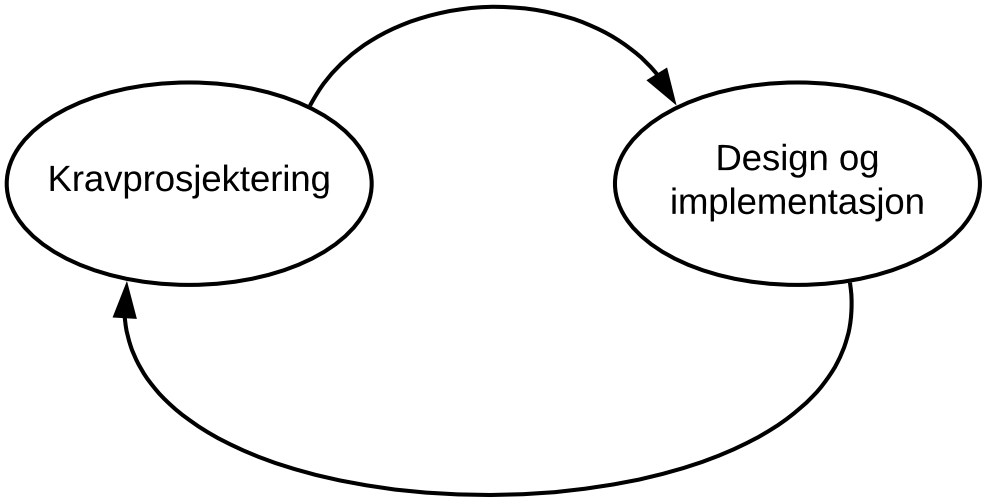
Dermed ble en ny form for utviklingsmetode utvikler på sent 1990-tallet:

Agile eller smidige metoder til utvikling brukes for systemutvikling der man regner med at spesifikasjonene og systemkravene kan endres flere ganger under utviklingen. Det blir brukt for å raskt kunne lage systemer som fungerer til brukerne, som igjen kan foreslå nye systemkrav og endringer som kan bli brukt i nye versjoner av systemet.

Prinsippene til smidig utvikling:

- Involvering av kunde: Kunder bør være nært involvert i hele utviklingsprosessen Deres rolle er å gi og prioritere nye systemkrav og å evaluere systemets iterasjon.
- Omfavn forandring: Forvent at systemkravene endres, så design systemet til å ta imot forandringene.

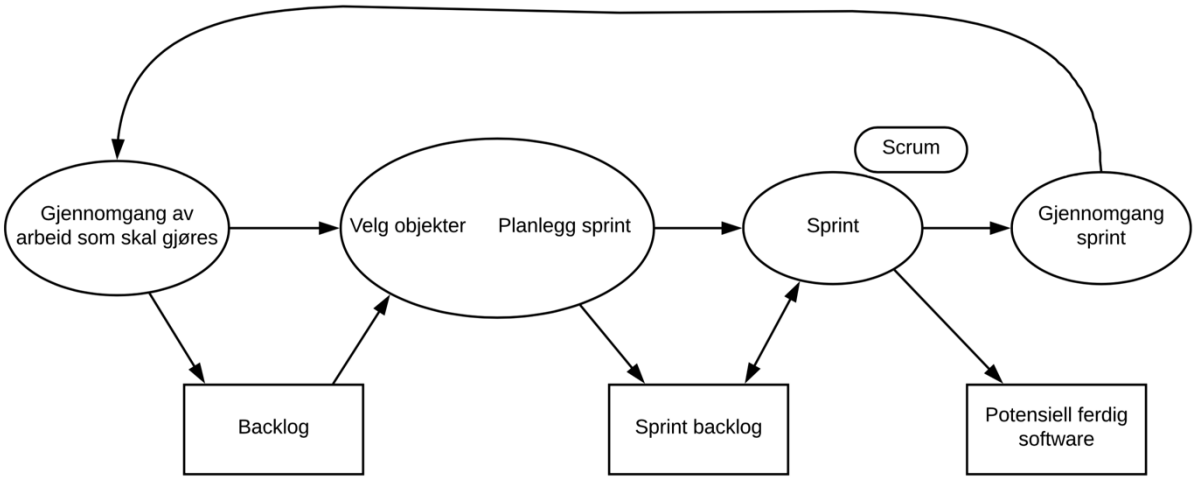
- Trinnvis levering: Systemet er utviklet i deler, der kunden spesifiserer systemkravene inkludert i hver del.
- Opprettholde enkelhet: Fokuser på enkelhet både i programvaren som utvikles og i utviklingsprosessen.
- Folk, ikke prosess: Utviklingsgruppens ferdigheter skal anerkjennes og utnyttes. [16]



Figur 2.3: Smidig utvikling

2.3.2 Scrum

Scrum er en mye brukt utviklingsmetode av den agile sorten som ble utviklet for å gi et rammeverk for å organisere smidige prosjekter og for å gi en ekstern synlighet over hva som foregår innen prosjektet.



Figur 2.4: Scrum

Startpunktet for Scrum sprint-syklusen er produktets backlog – listen over ting som funksjoner, krav og forbedringer som må jobbes med. Listen kan lages fra

kravspesifikasjonen, brukerhistorier eller annen beskrivelse av programmet som skal utvikles.
Hver sprint-syklus er en fast lengde, som oftest en tid mellom 2 og 4 uker. På begrunnelsen av hver syklus blir objektene i backlogen satt i prioritert rekkefølge. Sprinter blir aldri forlenget for å gjøre opp for uferdig arbeid. Det som ikke har blitt gjort i nåværende sprint blir ført tilbake til backlog.
Gjennom sprinten holder laget daglige møter (Scrums) for å få et overblikk over prosessen, ta opp problemer som kan ha oppstått, og planlegge dagens oppgaver. Dermed vet alle på laget hva som foregår og kan enkelt planlegge på nytt om uventede problemer oppstår.
På slutten av hver sprint er det et gjennomgangsmøte for å reflektere over hva som kunne blitt gjort annerledes og tilføre nye element til backlogen. [16]

2.3.3 Arkitektur og programmeringsmønster

Som en struktur- og arkitekturplan for implementeringen av brukergrensesnittet tok jeg utgangspunkt i Model-view-controller (MVC).
MVC er en populær måte å organisere koden på. Ideen bak er at hver seksjon av koden har forskjellige formål.

Model – kan være data i systemet, i mitt tilfelle tabeller i databasen
View – funksjonene som direkte har kontakt med brukeren, som gjør at systemet ser pent og funksjonelt ut
Controller – tar i mot brukerens input og bestemmer hva som skal gjøres med den. Hjernen bak applikasjonen sådan. [17]

2.3.4 Cohesion

Cohesion er definert som hvordan alle elementer i en modul, klasse eller komponent fungerer sammen som en funksjonell enhet. Høy cohesion er bra, og lav er dårlig. Den ideelle situasjonen er hvor en modul, klasse eller komponent bare styrer én funksjon eller et svært nært beslektet sett med funksjoner. Dette gjør at systemet er mer robust, lettere å forstå, å oppdatere og bruke om igjen i andre prosjekt.

2.3.5 Coupling

Coupling (kobling) defineres som graden av gjensidig avhengighet mellom to eller flere klasser, moduler eller komponenter. Tett kobling er dårlig, og løs kobling er god. [7]

2.4 Programmering

2.4.5 HTML og CSS

HTML er den standardiserte markup språket for å lage websider. HTML-elementer er byggeklossene til en webside og er representert med tagger, som markerer innhold som for eksempel paragrafer og tabeller. CSS beskriver hvordan HTML-elementene skal vises på skjermen. Det sparer mye arbeid da det kan styre oppsettet til flere websider samtidig. [18] [19]

2.4.6 PHP

PHP er et mye brukt skriptspråk med åpen kildekode som er gratis å laste ned og bruke.

Det brukes blant annet til å kjøre Facebook og brukes i kjernen til det største bloggsystemet, WordPress.

En PHP-fil kan inneholde tekst, HTML, CSS, JavaScript og PHP-kode.

Alle PHP kommandoer kjøres på serveren, og resultatet blir returnert til nettleseren i ren HTML.

PHP tolker kun kode mellom <?php og ?> [20]

Noen av nøkkelegenskapene til PHP er:

- Det er lett å bruke og veldig raskt. Selv om PHP-skript er tolket på kjøretid, så foregår tolkingen i webserveren. Som et resultat av dette kjører PHP sider (opptil 10 ganger) raskere enn tilsvarende Perl/CGI nettsider.
- I motsetning til CGI skript trenger ikke PHP sider å gjøres «kjørbare» eller å bli plassert i spesielle kataloger for å kjøres: det eneste man trenger å gjøre er å gi en HTML-fil en .php -ending og PHP vil automatisk kjøres.
- PHP-tolken viser feilmeldinger direkte på nettsiden, ikke i en loggfil.
- PHP kan cache tilkoblinger til MySQL databasesystemet. Som et resultat kan PHP-sider som er matet fra informasjon i en database, vises dramatisk raskere enn databasedrevne sider ved hjelp av andre systemer.
- PHP er ekstremt kraftig. Skript skrevet i PHP kan åpne filer, åpne nettverkstilkoblinger og utføre andre programmer. [13]

2.4.7 JavaScript

JavaScript er et kraftig og fleksibelt utviklingsspråk for nettsider. Sammen med HTML og CSS er det en av grunnsteinene i moderne web-utvikling, og brukes for å lage interaktive effekter på nettleserne. Javascript kan oppdatere og forandre både HTML og CSS, samt kalkulere, manipulere og validere data. [22]

2.4.8 SQL

SQL er et spørrespråk for databaser som benyttes til å formulere og kjøre operasjoner mot en relasjonsdatabase. Resultatet for en spørring er alltid én tabell. [11]

2.5 Databasedesign

Hensikten med begrepsmessig databasedesign/datamodellering er å finne frem til hva slags data som bør lagres i database.

En velstrukturert database:

- Sparer diskplass ved å eliminere redundans
- Opprettholder dataens nøyaktighet og integritet
- Gir tilgang til dataene på en nyttig måte

For å finne ut hvordan databasen skal se ut må man vite hva den skal brukes til. For å få tak i mest mulig informasjon kan man både intervjue personer i målgruppen og se gjennom eksisterende data, både digitale og fysiske filer.

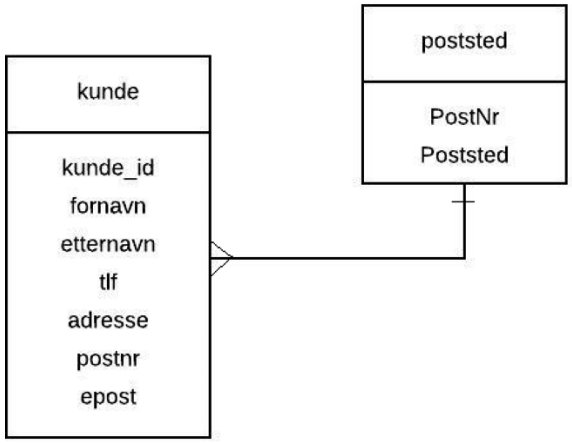
Et ER-diagram er en grafisk fremstilling av strukturen til en database. Diagrammene består av entiteter, attributter og forhold, som løselig svarer til tabeller, kolonner og fremmednøkler i en relasjonsdatabase.

Diagrammet egner seg som diskusjonsgrunnlag i møter mellom sluttbrukere og databasedesignere, for å avklare behov og begrepsbruk.

I et ER-diagram er entiteter tegnet som bokser, med navnet på entiteten øverst og attributtene listet under. Streker mellom boksene viser hvilke entiteter som er i relasjon med hverandre. På enden av strekene, nærmest hver boks, er et tegn som avgjør om relasjonen er en en-til-en-, en-til-mange- eller mange-til-mange relasjon.

Om mulig bør man unngå en-til-en, og mange-til-mange relasjoner.

Dette eksempelet viser en en-til-mange relasjon. Streken på linjen ved poststedboksen viser at en kunde kun kan ha et poststed, mens «kråkefoten» ved kundeboksen viser at et poststed kan ha flere kunder.



Figur 2.5: ER-diagram

Alle entiteter skal ha en identifikator. Noen typer entiteter har en naturlig identifikator, som registreringsnummeret på en bil, eller postnummeret til et poststed. Andre har ikke en attributt som kan brukes som identifikator, og dermed får de som oftest tildelt et løpenummer som kan håndteres med autonummerering. [11] [12]

2.6 GUI – Grafisk brukergrensesnitt

Alt brukeren ser og interagerer med kalles et grafisk brukergrensesnitt.

Funksjonene til en applikasjon kan fungere helt sømløst, men det hjelper ikke mye for brukeren hvis det ikke er tydelig hvordan den skal brukes.

Brukeren trenger ikke vite hvor mange linjer med kode som ligger bak en enkel knapp i nettleseren sin, eller hva som skjer når han fyller inn et skjema og trykker «Send». Det brukeren bryr seg om er om det er lett å forstå og lett å finne frem. Dette er ikke nødvendigvis så enkelt som man tror, og forskjellige rettningslinjer for grafisk brukergensesnitt har blitt utviklet siden 70-tallet. [15]

2.6.1 Retningslinjer

De to mest kjente listene med rettningslinjer for grafisk brukergrensesnitt:

Shneiderman (1987); Svhneiderman and Plaisant (2009)

- Streve etter konsistens
- Imøtekomme universell brukervennlighet
- Gi informativ tilbakemelding
- Forhindre feil
- Tillate enkel reversering av handlinger
- Få brukeren til å føle at de har kontroll
- Minimere kortsiktig minnebelastning

Nielsen and Molich (1990)

- Konsistens og standarder
- Synlighet av systemstatus
- Match mellom system og den ekte verden
- Brukerkontroll og frihet
- Forebygging av feil
- Anerkjennelse istedenfor tilbakekalling
- Fleksibilitet og effektivitet i bruk
- Estetisk og minimalistisk design
- Hjelpe brukerne gjenkjenne, diagnostisere og gjenopprette fra feil
- Gi online dokumentasjon og hjelp [15]

2.6.2 Gestalt-prinsippene

Gestalt-prinsippet om **Nærhet** handler om at den relative avstanden mellom objekter i et display påvirker vår oppfatning av om og hvordan objekter er organisert i undergrupper. Objekter som ikke er i nærheten av hverandre (i forhold til andre objekter), vises som i en gruppe, mens de som er lengre fra hverandre ikke gjør det.

En annen faktor som påvirker vår oppfatning av gruppering er i Gestalt-prinsippet om **Likhet**, der objekter som ser like ut ser ut som de danner en gruppe.

I tillegg til prinsippene som beskriver hvordan vi har en tendens til å gruppere objekter, så er det flere Gestalt-prinsipper som beskriver vårt visuelle systems tendens til å løse tvetydighet eller fylle inn manglende data.

Det første prinsippet, **Kontinuitet**, sier at vår visuelle oppfatning som regel oppfatter kontinuerlige former istedenfor frakoblede segmenter.

Ferdigstillelse, er prinsippet som sier at vårt visuelle system automatisk prøver å gjøre ferdig åpne figurer, slik at de oppfattes som hele objekt enn separate deler.

Gestalt-prinsippet om **Symmetri** sier at vi har en tendens til å analysere komplekse scener på en måte som reduserer kompleksiteten. Dataen vi ser på har som oftest mer enn én måte å bli oppfattet på, men synet vårt organiserer og tolker den automatisk for å forenkle og gi det symmetri.

Det neste Gestalt-prinsippet som beskriver hvordan vårt visuelle system strukturerer dataen det mottar er **Figur/Bakgrunn**. Dette prinsippet sier at hjernen vår separerer det visuelle feltet som en figur (forgrunnen) og bakken (bakgrunnen).

Felles skjebne handler om hvordan vi oppfatter objekter som beveger seg likt som objekter i samme gruppe.

Etter å ha designet en skjermbilde er det anbefalt å se på det med alle Gestalt-prinsippene i minne, for å se om designet tilsier at det er andre relasjoner mellom objektene enn de man hadde regnet med. [15]

2.6.3 Respons

Responsive systemer lar brukeren bli informert selv om det ikke kan utføre forespørselen øyeblikkelig. Det gir tilbakemelding på hva brukeren har spurt etter, og hva som skjer.

- Lar deg umiddelbart vite at forespørselen er mottatt.
- Gir noe informasjon om hvor lang tid operasjonene kan ta.
- Frigjør deg til å gjøre andre ting mens du venter.
- Administrerer køer på en intelligent måte.
- Utfører «husarbeid» og lavt prioriterte oppgaver i bakgrunnen
- Forutser dine mest brukte forespørsler. [15]

2.6.4 Varsler

For å gi tilbakemelding på feil i for eksempel forespørsler fra brukeren, er det flere velkjente metoder for å forsikre seg om at meldingen blir sett:

- Plasser meldingen der man ser. Når man trykker på en knapp eller en link, er det trolig at man ser direkte på den i et par øyeblikk etterpå. Designere kan bruke dette til å plassere feilmeldinger nært der man antar at brukeren ser.
- Bruk et symbol som varsler om feil, som for eksempel en varseltrekant eller et stopp-tegn.
- Reserver fargen rød til feilmeldinger. [15]

2.7 Prototype

En prototype er en tidlig versjon av et produkt eller et system, ofte brukt for å teste utviklingen før systemet ferdigstilles. Her er det ikke stort fokus på det ferdige brukergrensesnittet og designet, men heller på funksjonene og kravspesifikasjonen, slik at man kan bygge videre på det som fungerer og er ønsket fra oppdragsgiver, og heller forkaste ting som ikke behøver like mye energi. [25]

2.8 Testing

For å teste et system med levende brukere, er det en god ide å invitere en håndfull brukere fra ulike målgrupper og gi dem oppgaver de skal løse. Dette får du ved å brukerteste:

- Kunnskap (istedefor spekulasjon) om hvordan nettstedet fungerer. Hvis testpersonene går seg fast på samme sted hver gang, så gjør brukerene det også.
- Diskusjons- og konklusjonsgrunnlag for avgjørelser på design, slik at man gjør de rette valgene for videreutvikling og eventuelt endring av nettstedet.
- Enighet om hvilke behov nettstedet skal dekke – tilsvarer funksjonaliteten det brukerene ønsker? [21]

3 MATERIALER OG METODE

3.1 Prosjektstyring

Den største utfordringen med prosjektstyringen er å håndtere et helt prosjekt alene. Det er lettere å kjøre seg fast i et mindre problem, vanskeligere å vite hvem man skal gå til for hjelp, lettere å utsette ting og litt mindre sosialt, enn om prosjektet hadde vært et gruppeprosjekt. Allikevel fikk jeg til å styre tiden min greit selv, og fikk god hjelp av utviklingsmetoder.

3.1.1 Scrum som enkeltperson

Et Scrum team består vanligvis av en liten gruppe mennesker, men jeg tok med meg prinsippene og formet dem til å passe meg som enkeltperson.

Jeg laget en kravspesifikasjon sammen med oppdragsgiver, og ut ifra den ble de første deloppgavene i backlogen min til. Etter hvert som jeg støtte på problemer, fikk nye spesifikasjoner eller krav, eller jeg fant en måte å dele opp større oppgaver på, så ble backlogen fylt på.

Jeg bestemte meg tidlig for å ha iterasjoner på 2 uker, der jeg bestemte hvilke deloppgaver som skulle prioriteres, og i hvilken rekkefølge de skulle prioriteres, og så jobbet jeg ut i fra det. Hvis jeg satt fast på en oppgave over en tid, byttet jeg den ut med den neste på listen, og slik fikk jeg effektivisert utviklingen.

Jeg trengte tilbakemelding på systemet mens jeg jobbet, og brukte både datakyndige og ikke-datakyndige personer som testpersoner til å teste funksjoner og gi tilbakemelding på brukervennlighet og generell design.

3.1.2 Møter med oppdragsgiver

De første møtene med oppdragsgiver ble brukt til å diskutere hvilke kriterier som var helt nødvendige, og sette dem i prioritert rekkefølge, og lagde en enkel kravspesifikasjon.

Ved 2. møte kunne jeg vise frem protyper og nettverkskart.

Ved 3. møte kunne jeg vise første utkast av nettsiden, med de viktigste funksjonene på plass. Jeg tok imot tilbakemelding på endringer, flere funksjoner og design. Oppdragsgiver fikk prøve siden og jeg fikk svært positiv respons.

Ved 4. møte var siden klar til bruk, og etter en kort veiledning fikk de ansatte begynne å bruke siden. Jeg fikk tilbakemeldinger om noe var uklart eller burde endres på.

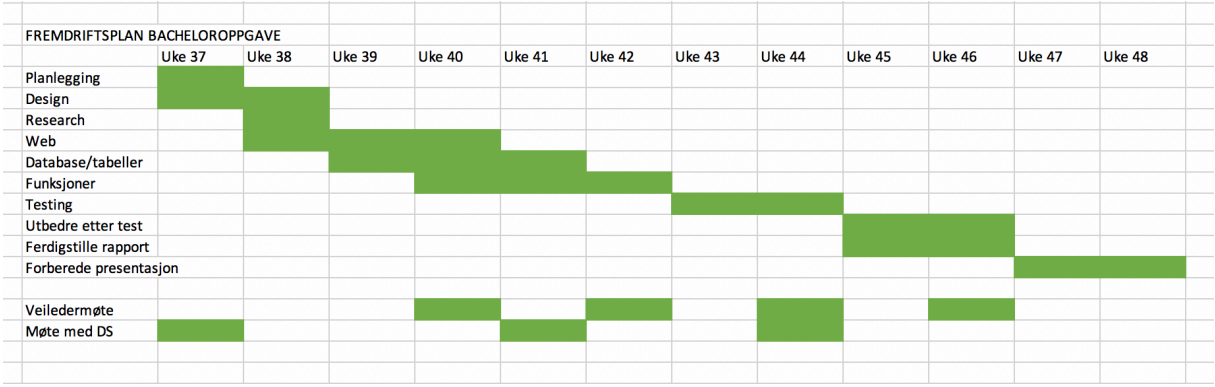
3.1.3 Møter med veileder

Jeg hadde møte med min veileder én gang annenhver uke, der jeg oppdaterte om hvordan det gikk, hvilke problemer jeg hadde støtt på, og hvordan jeg lå an i forhold til planen.

3.2 Planlegging

3.2.1 Fremdriftsplan

Tidlig i prosjektet satte jeg sammen en grov fremdriftsplan, for å måle fremdriften og holde kontroll på når de forskjellige delene burde være ferdige. Planen var til god hjelp for å sette opp tidsfrister.



Figur 3.1: Fremdriftsplanen

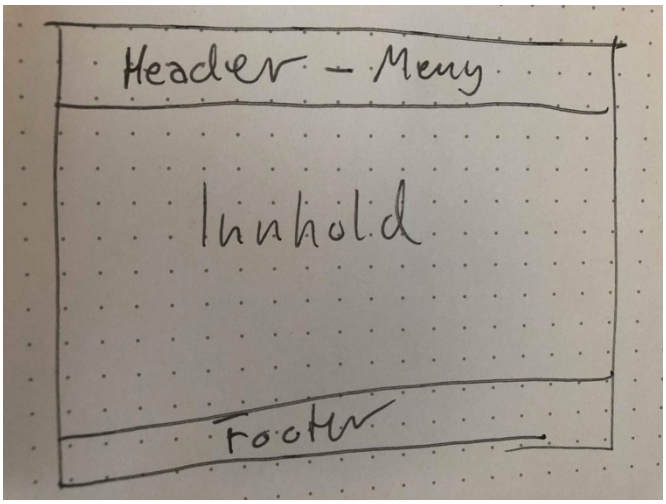
3.2.2 Oppgavetavle (Task Board)

For å organisere småoppgaver, brukte jeg programmet Asana. Her kunne jeg dele opp oppgavene etter tema (design, programmering, rapport) og sette tidsfrist. Jeg brukte dette som en backlog, der jeg satte kortere tidsfrister på de viktigste tingene, og lot de andre stå uten.

Figur 3.2: Oppgavetavle

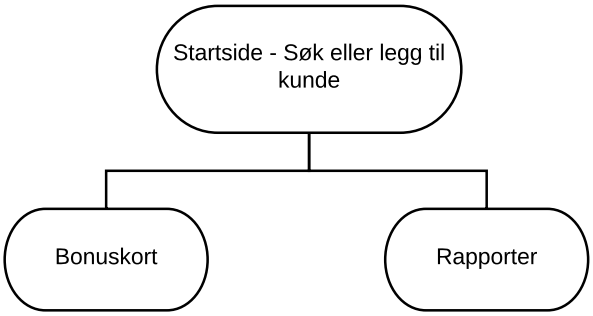
3.2.3 Webdesign og nettverkskart

Et nettverkskart eller en wireframe, er en visuell guide til hvordan skjelettet til nettstedet skal se ut. Den mangler som oftest stil, farge og grafikk siden hovedfokuset er hva et skjermbilde skal gjøre, ikke hvordan det ser ut. (Dan M. Brown 2011)



Figur 3.3: Wireframe

Deretter lagde jeg et nettverkskart for å klargjøre de forskjellige funksjonene nettsiden skal kunne tilby, og hvordan de henger sammen med hverandre



Figur 3.4: Nettverkskart

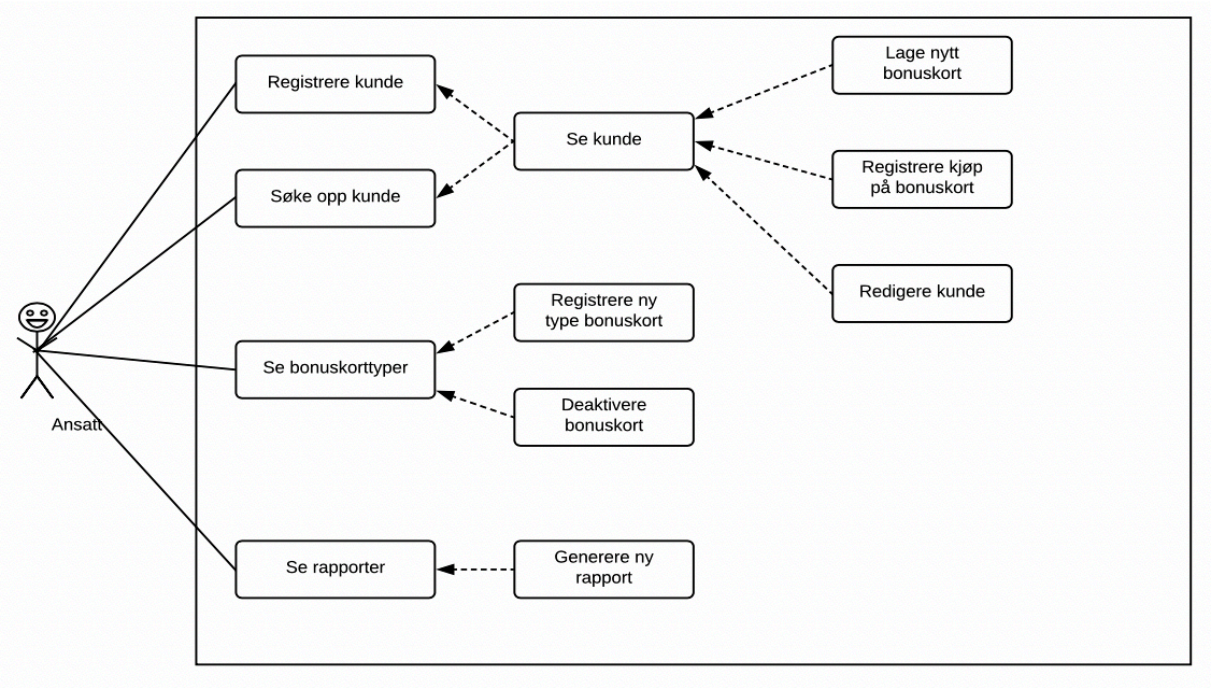
3.3 Bruker

Systemet håndterer kun én type bruker, en butikkmedarbeider hos Dyresenteret.

Eventuell administrering må skje gjennom phpMyAdmin

3.4 Bruksmønster

For å hjelpe til å organisere de forskjellige funksjonene til systemet, lagde jeg et såkalt Use Case diagram, eller et bruksmønster. Denne metoden identifiserer den individuelle kontakten mellom systemet og brukeren, og eventuelt med andre systemer.



3.5: Use Case

3.5 Programmeringsspråk

Her gjør jeg rede for hvilke språk som er brukt til utviklingen av prosjektet

3.5.1 HTML

Jeg har brukt HTML i prosjektet for å lage basen til nettsiden, utseende, menybar og knapper, alt det brukeren ser.

3.5.2 PHP

Alle funksjonene til nettsiden er skrevet i PHP. For eksempel; når en annen fil skal inkluderes i koden, for å håndtere input fra brukeren, for å håndtere SQL-spørringer mot databasen eller for å håndtere feil og feilmeldinger.

3.5.3 SQL

For å snakke til databasen, vise, endre, legge til, eller fjerne data fra databasen brukes SQL-spørringer. Disse spørringene er skrevet inn i PHP-koden

3.5.4 JavaScript

JavaScript med jQuery ble brukt for dynamiske oppgaver i funksjonene. Jeg hadde aldri jobbet med JavaScript fra før, og fikk derfor noe hjelp og eksempler til løsninger med en jeg kjente fra før som jobber med språket.

3.6 Utviklingsverktøy

Her er en oversikt over de forskjellige utviklingsverktøyene jeg brukte for å utvikle systemet.

3.6.1 Sublime Text

Når jeg utviklet koden til systemet brukte jeg Sublime Text som teksteditor. Dette fordi det var den jeg var vant med å bruke, og jeg syns den fungerer raskt og er lesevennlig. Jeg liker også godt kartet på høyre del, som kan brukes til å hoppe til gjenkjennelige punkter i koden, samt søkefunksjonen. Å bruke Sublime Text bød ikke på noen problemer.

3.6.2 NetBeans

Jeg vurderte også å bruke NetBeans som er en programvareutviklingsplatform skrevet i Java, og åpnet og redigerte deler av prosjektet i det, men jeg fant ikke noen store fordeler med å bruke det foran Sublime Text, så jeg gikk tilbake til å kun bruke Sublime.

3.6.3 MAMP

Jeg brukte MAMP for å utvikle og teste databasen og nettsiden etter hvert som den ble utviklet. MAMP er et akronym som står for de forskjellige originale komponentene av systemet; macOS (operativsystemet), Apache (web serveren), MySQL (database administreringssystemet) og PHP, Perl eller Python (programmeringsspråk for webutvikling)

Når jeg slo på serverene i MAMP kunne jeg enkelt åpne nettsiden lokalt, og få kontakt med databasen, for testing og utvikling.

3.6.4 phpMyAdmin

Ved å bruke MAMP fikk jeg tilgang til phpMyAdmin for å administrere databasen. Derfra kunne jeg lage eller laste opp databasen, med tabeller og utfylte kolonner.

3.6.5 Server og domene

Når alle funksjonene var kommet på plass var det klart for å få siden på nett for bruk i butikken. Min første idé var å bruke en Raspberry Pi som server, da dette var noe jeg hadde lest noe om, og jeg hadde allerede en RPI tilgjengelig. Men på grunn av dårlig tid tok jeg avgjørelsen om å kjøre hele websiden, med databasen og alt fra one.com, da det var lett tilgjengelig, lett å bruke og trygt.

Jeg logger enkelt inn på one.com for å redigere filer, eller administrere databasen fra phpMyAdmin.

3.7 Testing

Et nettbasert system trenger å testes fra ende til annen før det kan bli gitt ut til brukerne.

Jeg fant ikke noen gode løsninger på å teste siden automatisk, så jeg gjorde det stegvis og manuelt underveis, etter hvert som jeg la til flere funksjoner.

Når systemet hadde begynt å få på plass noen funksjoner, tok jeg det med til Dyresenteret slik at daglig leder kunne se første skikkelige utkast, og prøve seg på de funksjonene som var på plass.

3.8 Debugging

Når jeg støtte på problemer som jeg ikke fant ut av med det samme, brukte jeg debugging-funksjonen fra NetBeans, samt php_error.log fra MAMP.

3.9 Eksterne biblioteker

3.9.1 Bootstrap

Bootstrap er en av de mest populære front-end rammeverkene i verden Det erstatter store deler av det en CSS-fil gjør, og man får et universelt, responsivt og funksjonelt design på en enkel måte. Både skrift, knapper, tabeller, menyvalg og bildefremvisning er noen eksempler på hva Bootstrap enkelt kan forme.

Jeg har brukt Bootstrap for å forenkle designet på siden, og gjøre det enklere for eventuelt andre å jobbe videre med prosjektet.

Bootstrap-filene kan lastes ned og kjøres fra serveren eller lokalt, eller blir inkludert fra et CDN (Content Delivery Network). Jeg har valgt det siste.

Mange brukere har allerede lastet ned Bootstrap fra MaxCDN når de har besøkt en annen side. Som et resultat vil Bootstrap bli lastet fra en cache når de besøker siden, noe som fører til raskere innlasting.

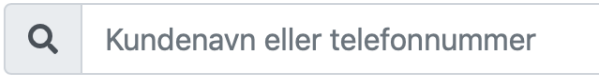
I tillegg til Bootstrap kjører jeg en liten egen CSS-fil.

3.9.2 JQuery, Poppers and Bootstrap JS

Mange av komponentene til Bootstrap krever JavaScript for å fungere, og må derfor bli inkludert. For å få med alt må både JQuery, Poppers og Bootstraps egen JavaScript bl inkludert fra et CDN.

3.9.3 Font Awesome

Font Awesome gjør det enkelt å legge til og bruke ikoner og små figurer for å gjøre nettsiden så enkel som mulig. Biblioteket legges enkelt til <head> på siden, og så er det bare å linke til de ikonene man vil bruke.



Figur 3.6: Et eksempel; et forstørrelsesglass som for de aller fleste forbindes med et søk. Ikonet er hentet med Font Awesome.

4 RESULTATER

Jeg har utviklet en webapplikasjon som vil kunne erstatte de fysiske bonuskortene til kundene som handler hos Dyresenteret Ålesund.

I dette kapittelet vil jeg gi en gjennomgang av resultatet av prosjektet, basert på den teoretiske delen av rapporten, samt forklare hvordan systemet vil fungere fra databasenivå til brukergrensesnittet.

4.1 Kravspesifikasjon

Kravspesifikasjonen ble utviklet sammen med oppdragsgiver. Den fullstendige kravspesifikasjonen ligger ved rapporten som vedlegg.

Tekniske spesifikasjoner
Nettsiden skal brukes via en nettleser på kassasystemet. Den skal kunne:

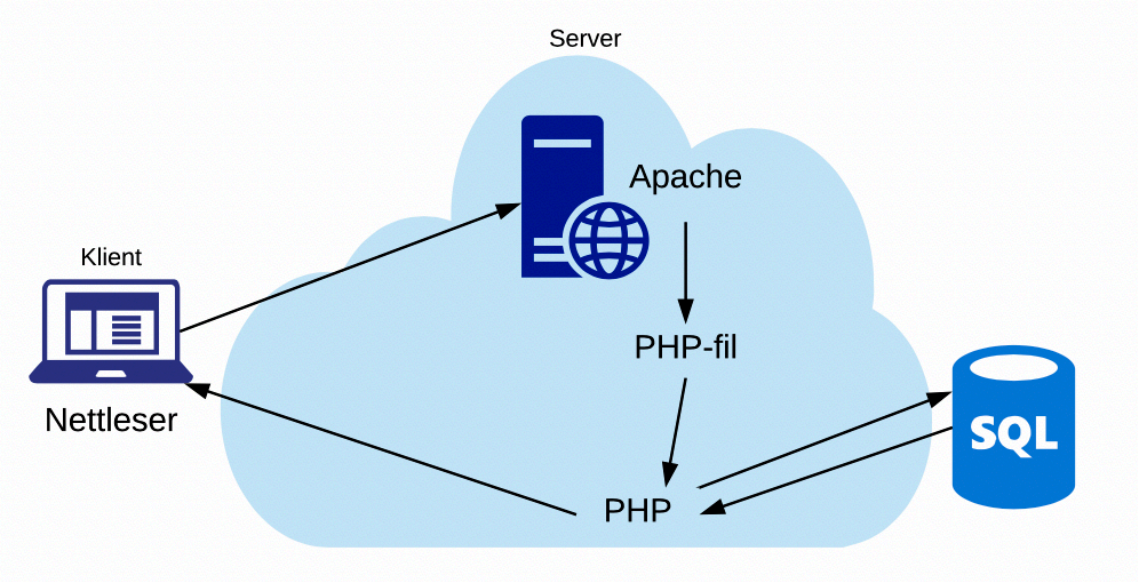
- Søke opp blant eksisterende kunder etter navn og/eller telefonnummer
- Registrere ny kunde
- Registrere et eller flere bonuskort på kunden
- Vise hvor mange «stempel»/kjøp kunden har på hvert bonuskort, med dato og pris
- Administrere forskjellige gyldige typer bonuskort (med forskjellig antall «stempel»)
- Arkivere «oppbrukte» bonuskort og ha mulighet til å lete dem opp
- Ha oversikt over hvilke varer som er gitt ut gratis som skal krediteres fra leverandør, med den informasjonen om kunden som leverandøren krever.

Ikke-funksjonelle krav

- Plattform: Hovedsakelig nettleser på kassasystemet. Eventuelt mulighet for å bruke mobil/nettbrett ved tekniske problemer, strømbrudd eller når butikken har stand på messe eller utstilling.
- Sikker løsning som hindrer sabotasje.
- Drift og vedlikehold skal utføres av utvikler, men bør være på et minimum.
- Krav om sikker lagring av personlige opplysninger, samt samtykkeerklæring ved registrering.

4.2 Struktur og oppsett

Figur x viser et veldig generelt overblikk over arkitekturen til prosjektet.

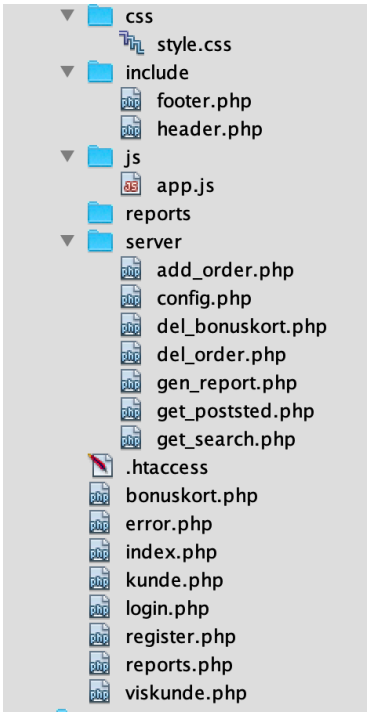


Figur 4.1: Arkitektur

En nettleser åpner nettstedet, som da får kontakt med serveren der filene til nettsiden er lagret. De rette filene blir kjørt i PHP, som da tar kontakt med databasen for å respondere med den.

4.2.1 Filstruktur

Filene til prosjektet ble sortert i mapper, for å gjøre det enklest mulig å finne fram underveis og i etterkant av prosjektet.

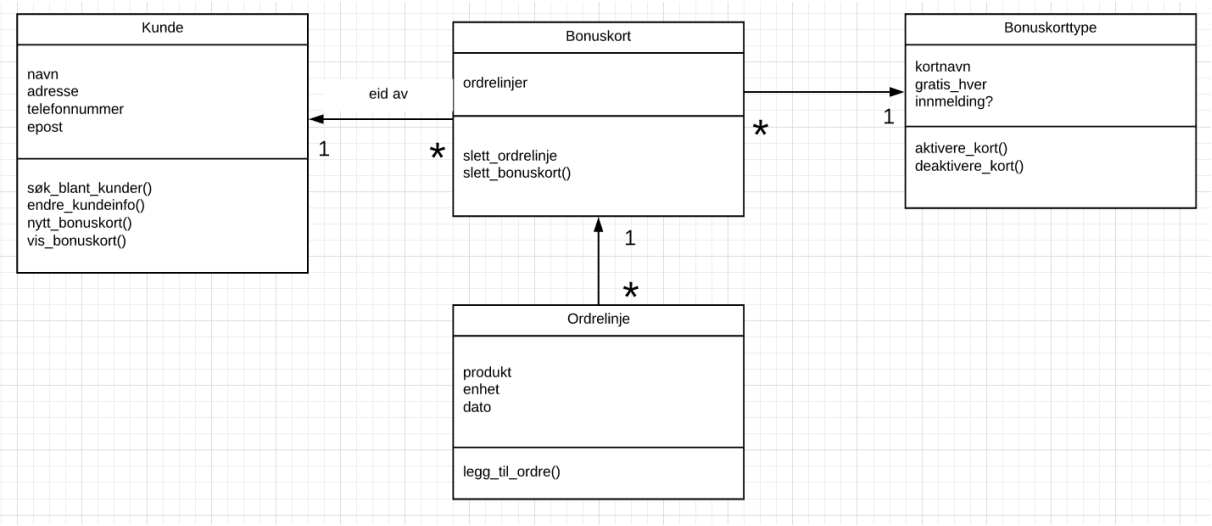


Figur 4.2: Organiseringen av filer

Koden er skrevet med tanke på low coupling og high cohesion. Dette gjør koden brukervennlig med tanke på endring, og også mer lesbar. En fil/modul har ansvar for en nettside, eller et begrenset antall funksjoner som er svært beslektet. Deler som gjentar seg i flere vindu lages som egne filer, eksempelvis header.php og footer.php. Dette gjør at det er enkelt å foreta større endringer.

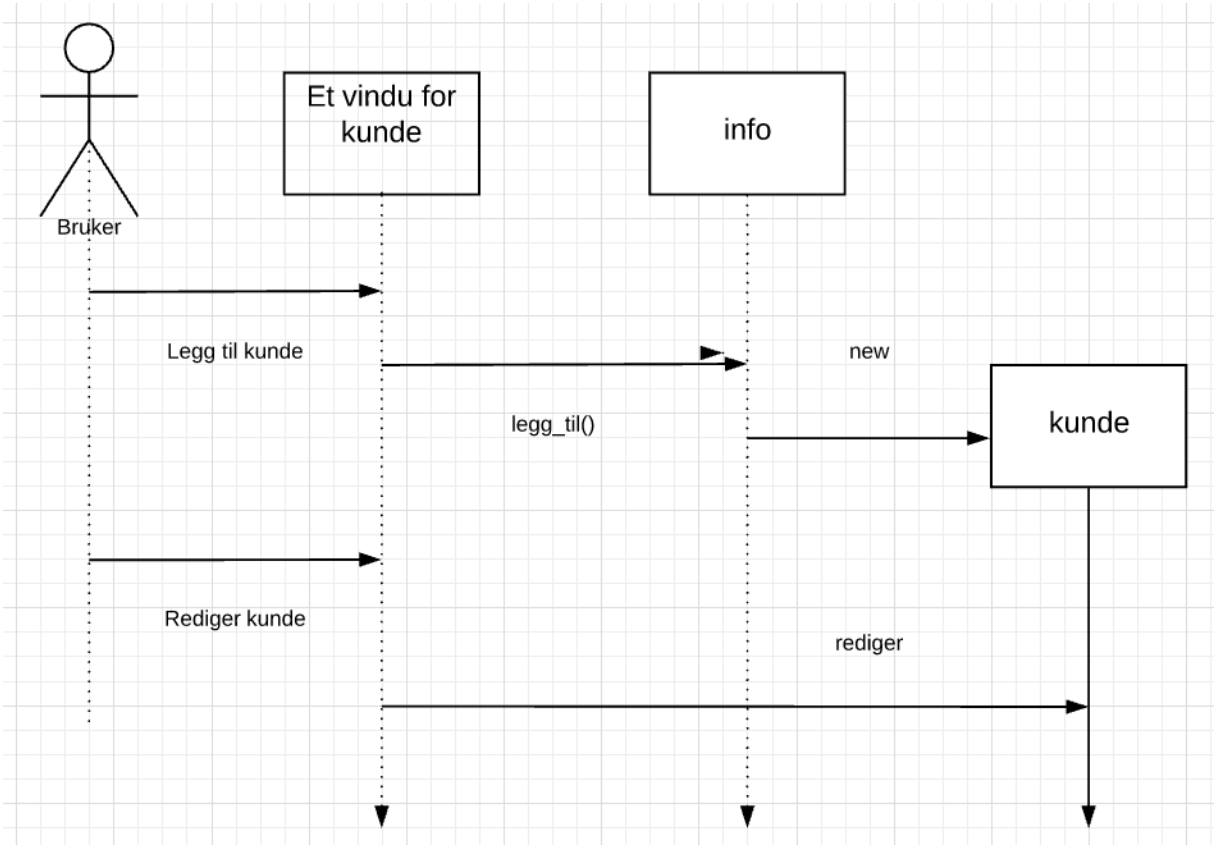
4.2.2 UML-Diagrammer

Klassediagram



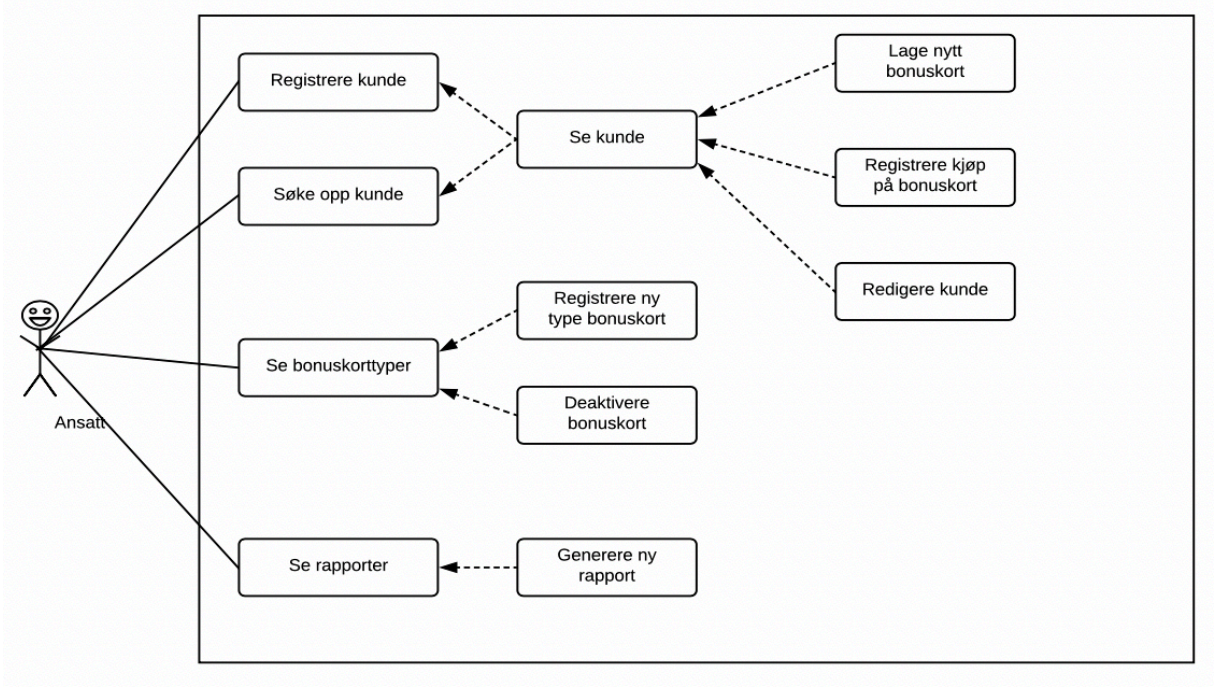
Figur 4.3: Et klassediagram som viser funksjoner og parametere

Sekvensdiagram



Figur 4.4: Sekvensdiagram som viser et eksempel på en funksjon

Use Case diagram



Figur 4.5: Use Case diagram

4.3 Programmeringsspråk og metode

Jeg valgte å lage en webapplikasjon fremfor for eksempel en installert applikasjon, da Dyresenteret har leid utstyret de bruker til kassesystemet, og jeg derfor ikke kunne installere noe der uten å klargjøre dette med utstyrsleverandør.

Løsningen ble en webapplikasjon, som dermed kan brukes fra hvor som helst om det skal være behov for det. Dyresenteret deltar noen ganger på messer og utstillinger, der de ikke har med seg elektronisk kassesystem, og dermed er det fremdeles mulig å bruke bonuskort om det skal være ønskelig.

Jeg valgte å skrive systemet i PHP, da jeg hadde litt erfaring med det fra et tidligere gruppeprosjekt. Og det er den mest allsidige til denne typen prosjekt.

Jeg brukte Bootstrap som rammeverk. Bootstrap var noe jeg hadde lest mye om, men ikke tatt i bruk selv, så det ble en utfordring å lære seg å bruke. Men det gjorde utseende på nettstedet veldig pent og rent uten å bruke for mye tid på det.

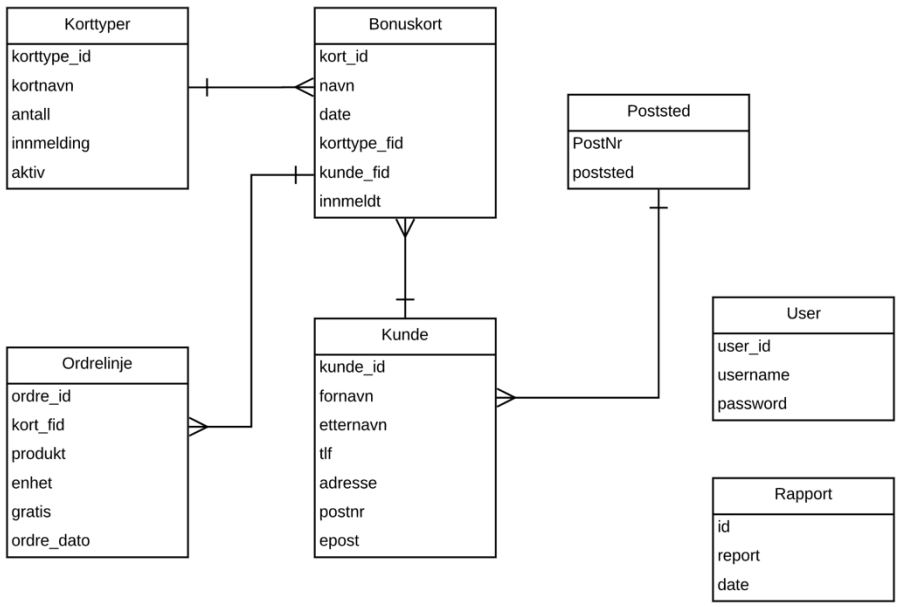
4.4 Databasestruktur

Strukturen på databasen ble kartlagt ganske tidlig i prosjektet, og det er stort sett attributter som er blitt endret underveis.

Jeg vurderte å samle inn mer data fra kunder, som for eksempel hvilke dyr de har, for å ha noe å lage statistikk av. Men ifølge personvernsloven er det ikke tillatt å samle inn mer informasjon enn vi kan dokumentere er nødvendig. Dermed holdt jeg meg til de feltene som de gamle bonuskortene allerede hadde, som blant annet fôrleverandørene krever for å kreditere produkter som er gitt ut gratis til kunder.

4.4.1 ER-diagram

Figuren under er et ER-diagram som viser den endelige databasestrukturen med eventuelle relasjoner.



Figur 4.6: ER-diagram av databasestrukturen

4.4.2 Entitets relasjoner

Som ER-diagrammet viser, er fem av tabellene koblet sammen, og hver relasjon er en en-til-mange relasjon.

- En kunde kan ha flere bonuskort. Et bonuskort er registrert på en kunde.
- Det kan bo mange kunder på samme poststed. En kunde er registrert på kun ett poststed.
- Et bonuskort er av en bestemt korttype. Mange bonuskort kan ligge under samme korttype.
- En bestemt ordrelinje finnes kun på ett bonuskort. Et bonuskort kan ha mange ordrelinjer.

4.4.3 Overblikk på alle attributter

Nedenfor følger figurer som viser alle entiteter med navn og datatype på alle attributtene. Man kan også se tegnet for primærnøkkel/identifikator, og for eventuelle fremmednøkler. Primærnøkkel er det sterke bildet av en nøkkel, mens fremmednøkkel er det svakere bildet.

#	Name	Type
1	**kort_id**	int(11)
2	**navn**	varchar(255)
3	**date**	timestamp
4	**korttype_fid**	int(11)
5	**kunde_fid**	int(11)
6	**innmeldt**	int(1)

Figur 4.7: Entitet Bonuskort med navn og datatype på attributtene. Primærnøkkel er kort_id og korttype_fid og kunde_fid er fremmednøkler

#	Name	Type
1	**korttype_id**	int(10)
2	**kortnavn**	varchar(255)
3	**antall**	int(10)
4	**innmelding**	int(1)
5	**aktiv**	int(1)

Figur 4.8: Entitet Korttype med navn og datatype på attributtene. Primærnøkkel er korttype_id.

#	Name	Type
1	kunde_id	int(11)
2	fornavn	varchar(50)
3	etternavn	varchar(50)
4	tlf	int(8)
5	adresse	varchar(255)
6	postnr	char(4)
7	epost	varchar(255)

Figur 4.9: Entitet Kunde med navn og datatype på attributtene. Primærnøkkel er kunde_id og postnr er fremmednøkkel.

#	Name	Type
1	ordre_id	int(11)
2	kort_fid	int(11)
3	produkt	varchar(255)
4	enhet	varchar(255)
5	gratis	int(1)
6	ordre_dato	timestamp

Figur 4.10: Entitet Ordrelinje med navn og datatype på attributtene. Primærnøkkel er ordre_id, og kort_fid er fremmednøkkel.

#	Name	Type
1	PostNr	char(4)
2	Poststed	varchar(50)

Figur 4.11: Entitet Poststed med navn og datatype på attributtene. PostNr er primærnøkkel.

#	Name	Type
1	id	int(11)
2	report	varchar(255)
3	date	timestamp

Figur 4.12: Entitet Rapport med navn og datatype på attributtene. Primærnøkkel er id.

#	Name	Type
1	user_id	int(11)
2	username	varchar(255)
3	password	varchar(255)

Figur 4.13: Entitet User med navn og datatype på attributtene. Primærnøkkel er user_id.

4.5 Sikkerhet

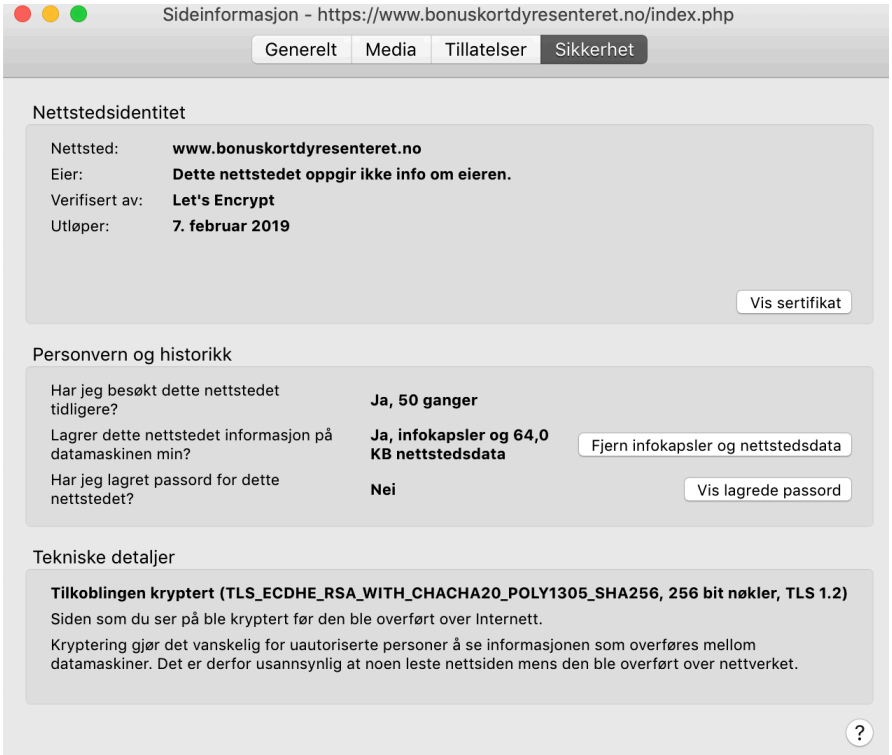
Denne delen beskriver hvordan jeg har tatt hånd om kjente sikkerhetsproblemer med webløsninger. Jeg trenger å være trygg på at personopplysninger sikres, og at databasen ikke kan saboteres av uvedkommende.
Det vil alltid være en viss sikkerhetsrisiko i et system. Men man kan gjøre mye for å begrense at uvedkommende kan bryte seg inn i systemet.

4.5.1 Hashing av passord

Ved registrering av bruker brukes *password_hash* før passordet lagres i databasen. Dermed blir ikke passordet sendt i klartekst, men blir lagret i databasen kryptert.

4.5.2 HTTPS

Domenet jeg benytter har et SSL-sertifikat. Dermed kan HTTPS benyttes. Ved hjelp av en .htaccess-fil omdirigeres HTTP til HTTPS når noen besøker nettstedet.
Ved å bruke sideinnformasjon til Mozilla Firefox, kan jeg verifisere at tilkoblingen er kryptert og hvilken kryptering som er brukt.



Figur 4.14: Sideinnformasjon fra Mozilla Firefox

4.5.3 Beskyttelse mot SQL-injection

For å beskytte siden mot SQL-injection bruker jeg «prepared statements», som vil si at jeg forbereder spørringen som en string, så den ikke kan manipuleres fra et input-felt.

```
$stmt = $mysqli->prepare("  SELECT *
                            FROM kunde
                            LEFT JOIN PostSted ON PostSted.PostNr = kunde.postnr
                            WHERE kunde.kunde_id = ?");

$stmt->bind_param("s", $_GET['kunde']);
$stmt->execute();

$kunde = $stmt->get_result()->fetch_assoc();
```

Figur 4.15: Eksempel på et «prepared statment» i koden.

4.5.4 Sikre personopplysninger

Innholdet som bli lagret om hver enkelt kunde faller under personopplysningsloven. Systemet samler ikke inn sensitive opplysninger, som for eksempel: opplysninger om rase, etnisk opprinnelse, politisk oppfatning, religion med mer.

Systemet skal bare håndtere og lagre de opplysningene om kunden som er nødvendig. Kunden kan når som helst be om å endre eller slette sine personopplysninger.

4.5.5 Personvernsloven – samtykke

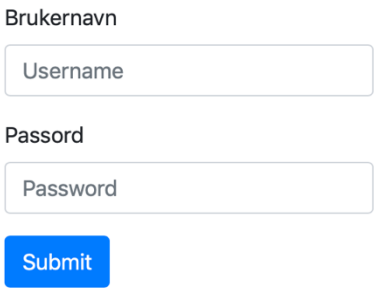
Siden informasjonen lagres av den ansatte, og ikke av kunden selv, kunne vi ikke be om samtykke elektronisk. Dette løste jeg med å skrive ut et skjema, der det kort ble fortalt hvilken informasjon som ble lagret og hvorfor, hvem som får se informasjonen, og at den når som helst kan slettes på forespørsel. Kunden skriver så navnet sitt og en signatur i en kolonne. Dette var en rask og enke måte å løse det på. [Vedlegg 3]

4.6 Brukergrensesnitt

Denne delen beskriver hvordan jeg har vurdert designet på systemet, spesielt med tanke på grafisk brukergrensesnitt (GUI). Dette er noe jeg har lagt mye vekt på at skal være funksjonelt, i tillegg til enkelt og raskt å bruke.

4.6.1 Innlogging

For å spare tid på innlogging for hver gang systemet skal brukes, så er det kun én bruker for alle de ansatte ved Dyresenteret. Derfor er det ikke mulighet for å registrere flere brukere etter at den første ble registrert.

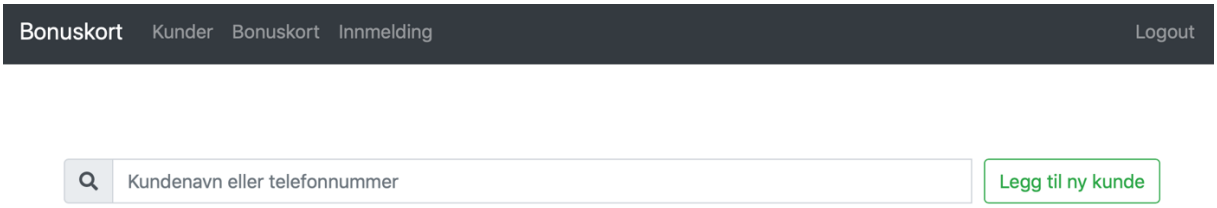


Figur 4.16: Innlogging

4.6.2 Startside / Kunder

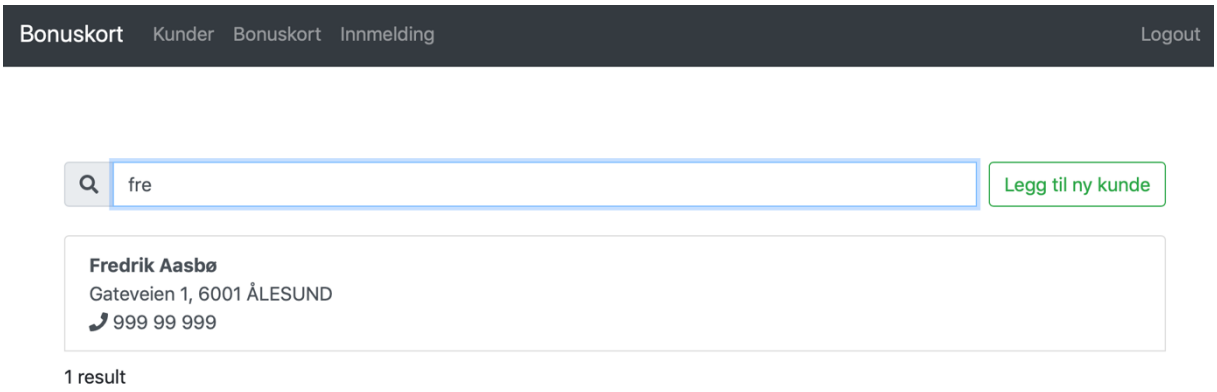
Når brukeren har logget inn kommer han direkte til siden der man kan søke opp kunder som allerede er registrert, eller registrere en ny kunde.

Øverst på siden ligger en oversiktlig navigasjonsmeny og mulighet for å logge ut. Denne menyen ligger alltid øverst på alle sidene.



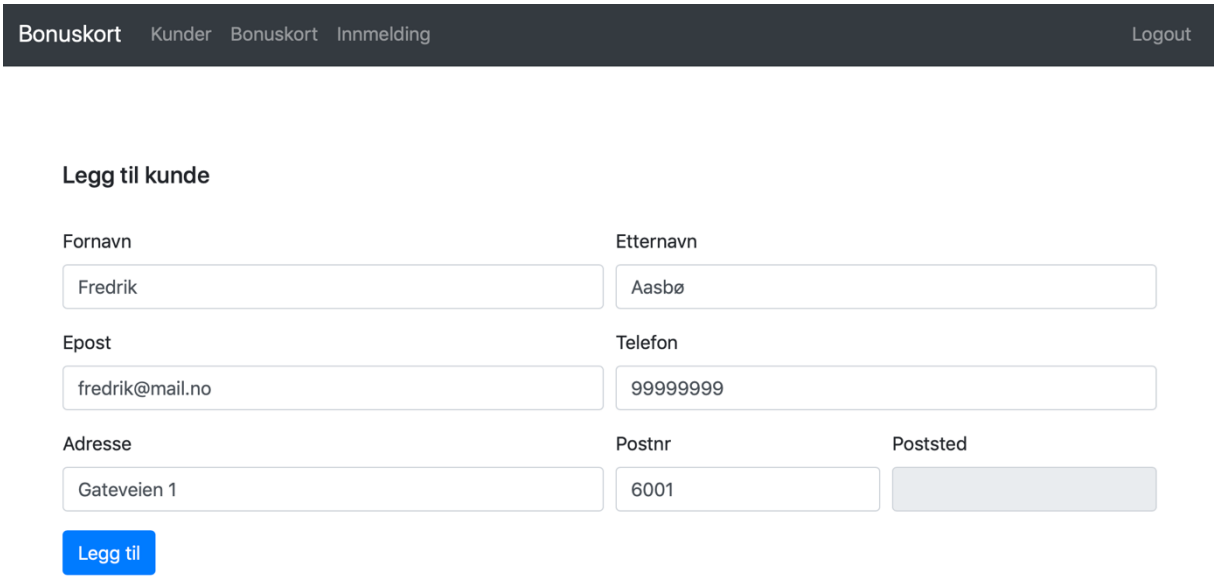
Figur 4.17: Startside

Figur 4.18: Et ikon av en roterende spinner viser at siden jobber med å finne kunder



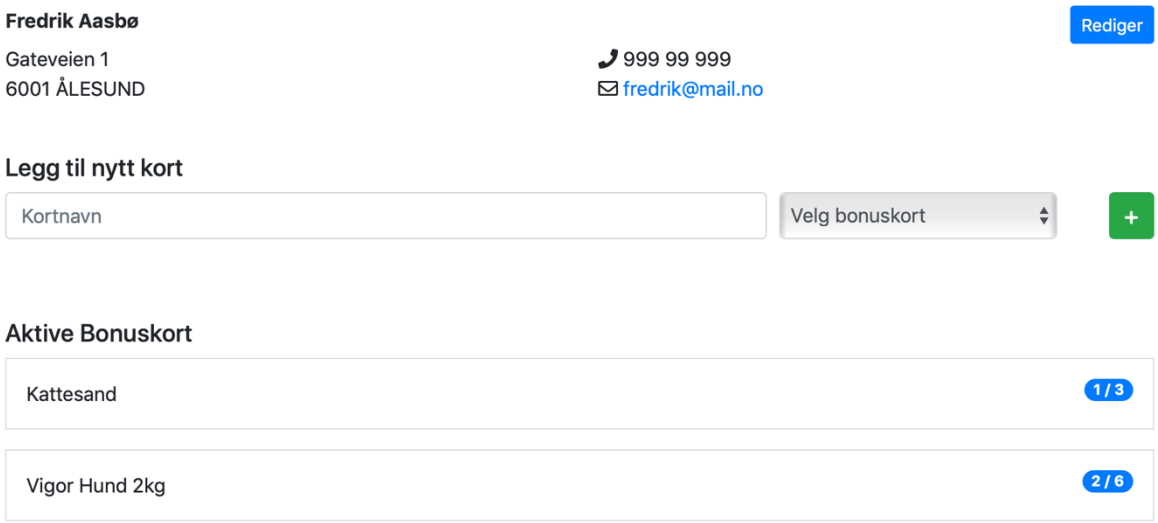
Figur 4.19: Søkeresultater

For å registrere en ny kunde må man fylle ut et skjema. Poststed fylles inn automatisk etter postnummer.



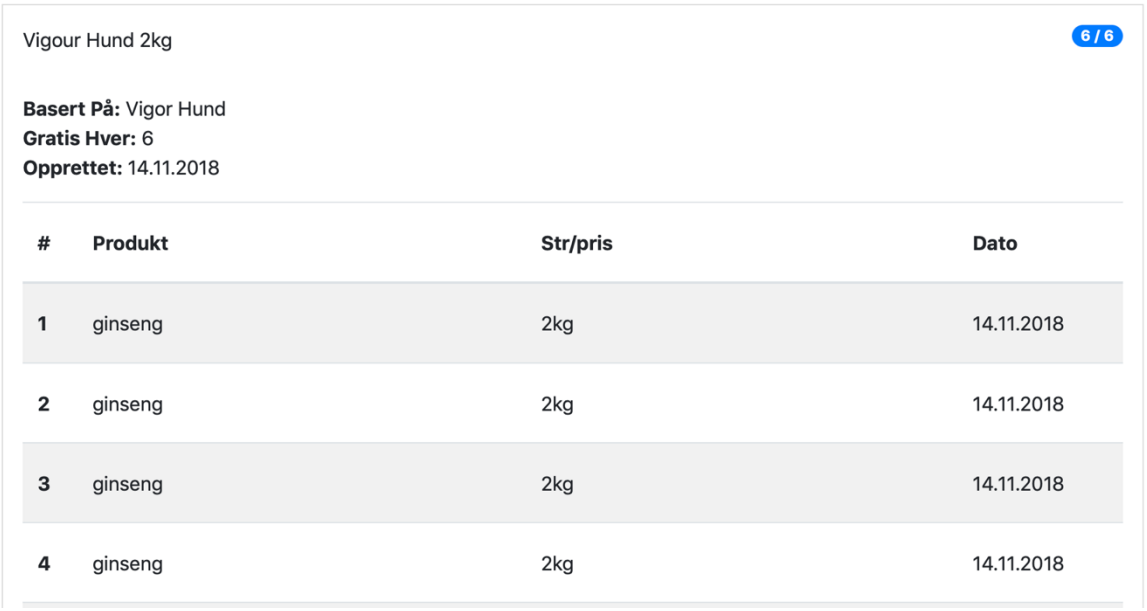
Figur 4.20: Skjema for å legge til kunde

Etter å ha søkt opp og trykket på valgt kunde, eller registrert en ny kunde, åpnes den aktuelle kundens side. Her vises kundens informasjon, med mulighet for redigering. Nedenfor kan man legge til et nytt bonuskort fra listen av aktive bonuskort. Deretter følger bonuskortene som kunden allerede har tatt i bruk.



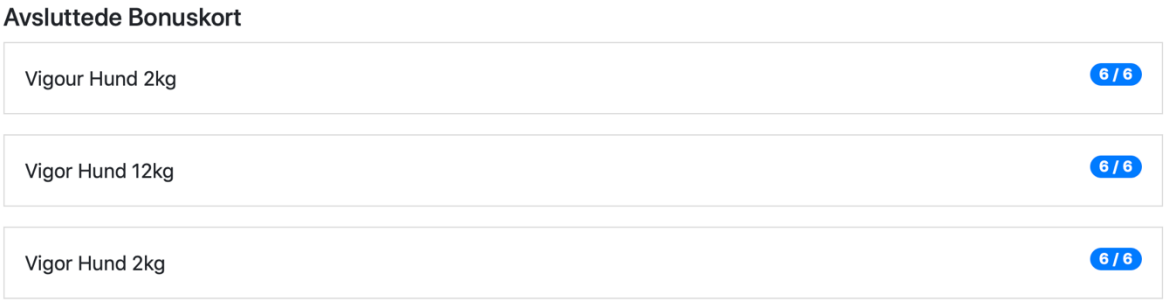
Figur 4.21: Kundeside

Når man trykker på et aktivt bonuskort vises de kjøpene som allerede er gjort, med enkel tekst om hvilken type (gjerne for de kundene som ikke alltid husker hva de skal ha), størrelse eller pris, og datoen for kjøpet.



Figur 4.22: Innhold på et bonuskort

Nederst på siden finner man bonuskortene som er fylt ut, og dermed brukt opp. Jeg valgte å spare på de ferdige bonuskortene for å kunne vite hvilken type vare kunden bruker, og for å ha noe å vise til om kunden er overasket over å ikke ha et aktivt bonuskort lenger.



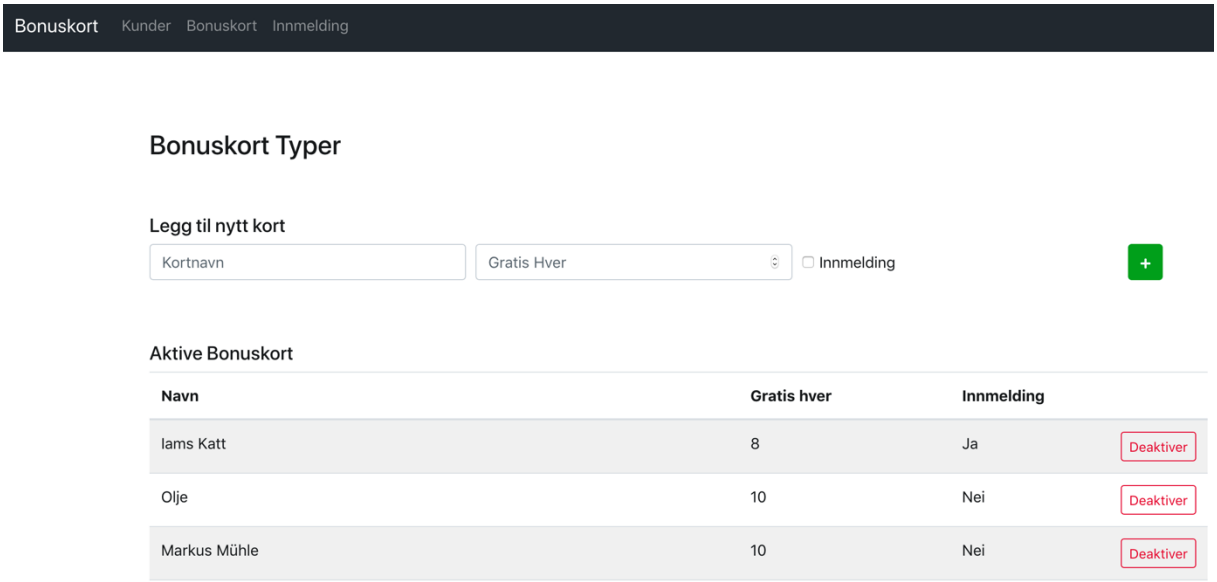
Figur 4.23: Ferdige/Avsluttede bonuskort

4.6.3 Bonuskort

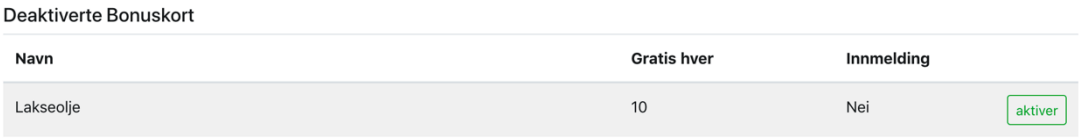
På menyvalget Bonuskort er en oversikt over de bonuskortene som gjelder. Her står informasjon om hva kortet gjelder, hvor mange enheter med en vare som trengs for å fylle opp et bonuskort, og også om bonuskortet skal meldes inn til fôrleverandør når det er fylt ut, for kreditering av vare utlevert gratis.

Et bonuskort kan også enkelt deaktiveres herfra, for eksempel hvis en leverandør slutter å tilby bonuskort.

Øverst ligger et enkelt skjema for å registrere en ny type bonuskort.



Figur 4.24: Registrere ny type bonuskort



Figur 4.25: Deaktiverte bonuskort

4.6.4 Innmelding

På det siste menyvalget, Innmelding, finner man rapporter.

I en rapport finnes informasjonen om kunden og varen som er gitt ut gratis ved slutten av et bonuskort, hvis det er av typen bonuskort som skal meldes inn til leverandør.

Den ansatte som har ansvar for å sende krav til leverandøren genererer en rapport ved å trykke på generer. Rapporten havner øverst i listen av rapporter.

Figur 4.26: Rapporter

4.6.5 Design

Jeg valgte å legge mye vekt på designet på siden, og gjøre siden lettlest og enkel. Jeg brukte punktene til Jeff Johnsen [2.6.2] når jeg tok valg angående utformingen av siden og blant annet farger på tekst, bakgrunn og tilbakemeldinger.

Ren hvit og åpen bakgrunn gjør det enkelt å navigere. Siden viser bare relevant informasjon til brukeren, slik at det er enkelt for brukeren å finne frem til ønsket funksjon.

Ved å bruke Bootstrap lagde jeg designet på knapper, tilbakemeldinger, tabeller, og lister blant annet. Sammen med JavaScript gjorde jeg siden dynamisk og interaktiv, for en positiv brukeropplevelse.

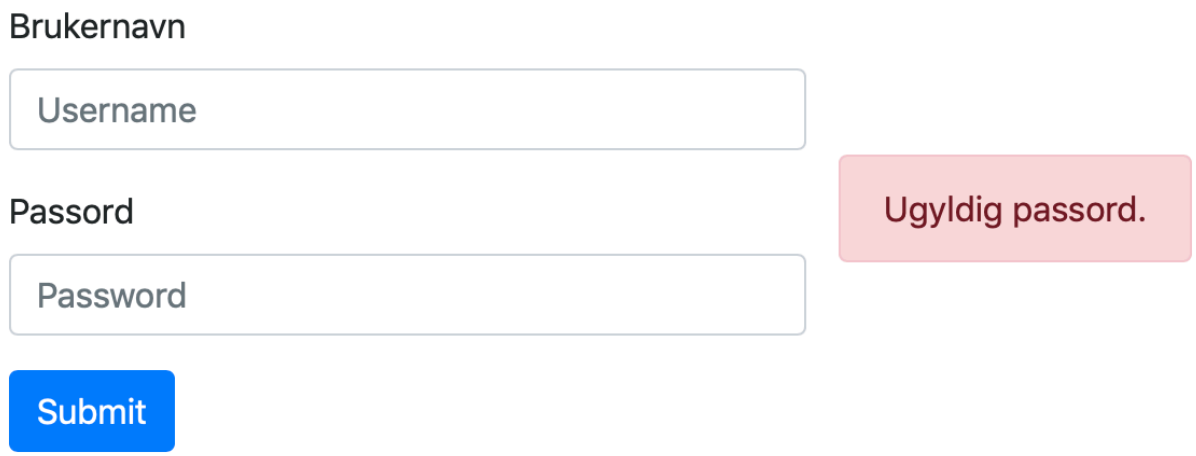
Blant annet har jeg brukt JavaScript til å hindre siden fra å oppdatere seg ved for eksempel feil eller manglende input i skjema.

```
// Bootstrap Form Vailidation.
(function() {
    'use strict';
    window.addEventListener('load', function() {
      // Fetch all the forms we want to apply custom Bootstrap validation styles to
      var forms = document.getElementsByClassName('needs-validation');
      // Loop over them and prevent submission
      var validation = Array.prototype.filter.call(forms, function(form) {
        form.addEventListener('submit', function(event) {
          if (form.checkValidity() === false) {
            event.preventDefault();
            event.stopPropagation();
          }
          form.classList.add('was-validated');
        }, false);
      });
    }, false);
  })();
```

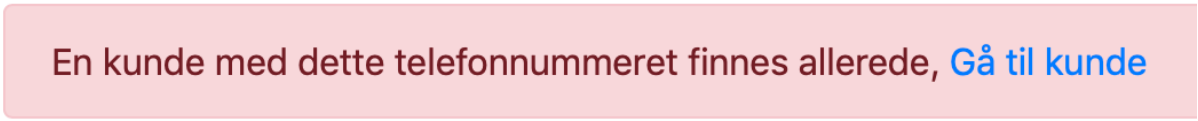
Figur 4.27: Eksempel JavaScript. Denne funksjonen stopper nettleseren fra å «gjøre det den skal» og oppdatere siden når man trykker «ok» på for eksempel en registrering og man ikke har fylt inn skjema korrekt, slik at man slipper å fylle inn alt på nytt og scrolle ned til rett plass.

4.6.6 Tilbakemelding til bruker og advarsler

Når det oppstår en feil, enten ved manglende eller feil type input fra bruker, eller en feil som gjør at brukeren ikke får informasjonen han vil ha, vises en feilmelding med rød bakgrunn for å fange brukerens oppmerksomhet.



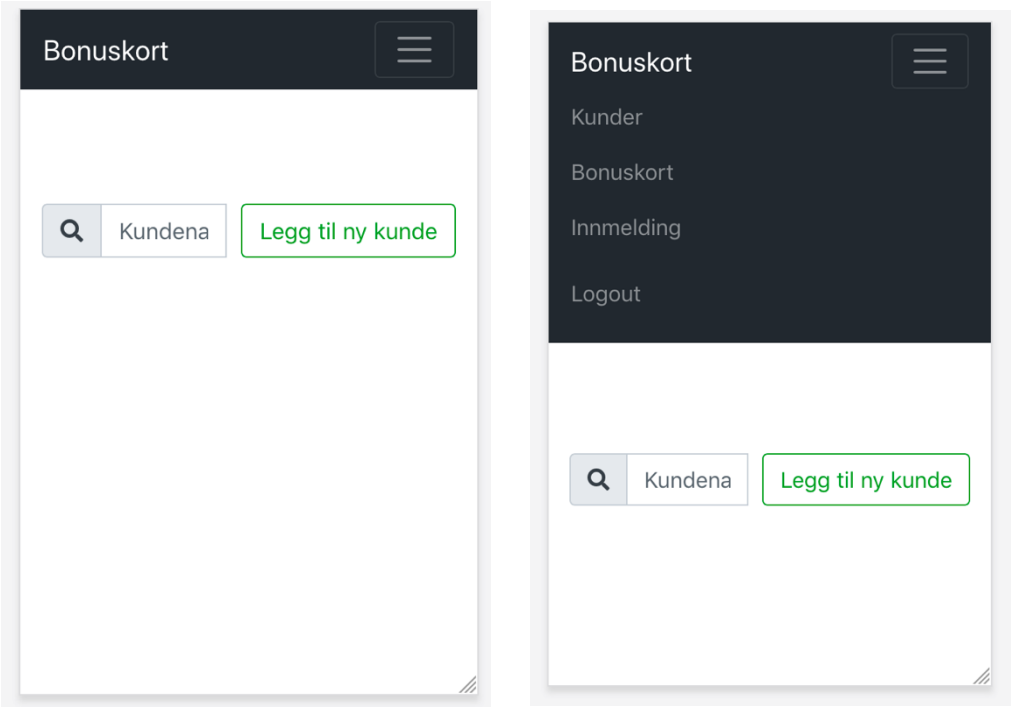
Figur 4.28: Feilmelding ved ugyldig passord



Figur 4.29: Feilmelding under registrering

4.6.7 Mobil og andre mindre skjermer

Når siden blir lastet inn på en mobilskjerm, et nettbrett eller en annen mindre skjerm, vil elementene i navigasjonsmenyen gjemmes, og erstattes av et ikon. Når man trykker på dette ikonet vil navigasjonsmenyen falle ned og vise elementene. Uten denne funksjonen hadde siden blitt vanskelig å lese å navigere på en liten skjerm, og navigasjonsmenyen hadde tatt opp store deler av skjermen.



Figur 4.30: Navigasjonsmenyen på mobil

4.7 Testing

Jeg brukte noen få personer jeg kjenner som testbrukere til siden. Men hovedsakelig var det meg selv som bruker, som testet siden gjennom utviklingen. Når siden hadde fått på plass de viktigste funksjonene, lot jeg oppdragsgiver prøve funksjonene og komme med tilbakemelding.

5 DRØFTING

I dette kapittelet følger en vurdering av oppnådde resultater, utførelsen av prosjektet og utfordringer som jeg støtte på.

5.1 Planlegging og prosjektstyring

Jeg begynte planleggingen av prosjektet i begynnelsen av september, og brukte noe tid på å lese teori om utviklingsmetoder, sjekke ut lignende løsninger, og snakke med oppdragsgiver om hva som ble forventet av systemet. Jeg bestemte meg tidlig for utviklingsmetode og begynte planleggingen av funksjoner. Etter å ha lest mye om Bootstrap og hva det kunne tilby, bestemte jeg meg for at det var noe jeg ville bruke istedenfor å kun bruke egen CSS som jeg var vant med.

Jeg planla først alle hovedfunksjonene, og deretter delte jeg alt opp i mindre deler. Dette hjalp meg å få oversikt over alle de mindre funksjonene som måtet til for å få løsningen jeg ønsket.

Jeg planla å bruke en Scrum-lignende metode, som enkeltperson. Jeg dokumenterte og planla hver iterasjon, som varte 2 uker hver. Det var vanskelig å planlegge hvor lang tid en enkelt oppgave skulle ta, samt i hvilken rekkefølge det kunne gjøres, så det ble noe avvik i forhold til planen.

Å få på plass alle funksjonene var det som tok mest tid, og påvirket de siste ukene av prosjektet før systemet var klar til å begynne å brukes. Jeg hadde ikke klart å dele opp funksjonene i nok mindre deler, noe som gjorde noen av funksjonene uoverkommelige og vanskelige. Etter å ha klart å dele opp funksjonene, og brukt noen sene nattetimer, falt mange av brikkene endelig på plass og jeg kunne se enden på prosjektet.

Etter å ha skaffet et domene måtte jeg endre på deler av koden, da det dukket opp mye småfeil som ikke ble oppdaget når prosjektet kjørte lokalt på maskinen.

Å holde tidskjema til prosjektet var en stor utfordring, da det har dukket opp flere ting i høst som tvang meg til å legge prosjektet til side i perioder. I tillegg til noe sykdom.

5.2 Testing

Først og fremst er funksjonaliteten testet fortløpende under utviklingen. Det gikk mye tid med til å utelukke feil. Systemet ble testet med både logiske og ulogiske inputs for å se hva slags tilbakemelding man fikk.

Siden jeg selv jobber på Dyresenteret, og har førstehånds erfaring med hva systemet trenger var det lite å sette fingeren på av oppdragsgiver når prosjektet ble vist frem for første gang. Oppdragsgiver mente at brukersnittet var pent å se på, enkelt å forstå, og hadde noen gode funksjoner for å oppnå ønsket bruk.

Systemet ble testet for alvor fra midten av november, da de ansatte ved Dyresenteret fikk tilgang til å begynne å registrere kunder og bonuskort. Fra systemet ble tatt i bruk har det eneste problemet vist seg at ikke alle postnummer var lagt til i databasen. Dermed fikk man ikke registrert en kunde riktig, da systemet nektet å godta et postnummer. Den ansatte løste det ved å legge inn et kjent postnummer, og skrive ned hvilken kunde det gjaldt og hvilket postnummer det egentlig skulle være, slik at jeg kunne sjekke dette opp mot databasen og legge til de postnumrene som manglet, og etterpå redigere kundens informasjon.

Noe annet som kom opp etter oppstart var angående registrering av epost. Selv om de originale bonuskortene inneholdt et felt for e-post, så har vi ikke nektet kunder å bruke

bonuskort om de ikke eier en epost, eller ikke vil oppgi eposten sin. Dette feltet sto originalt som «required», men ble endret til å være et valgfritt felt.

5.3 Server

Noe jeg ikke hadde fått nok tid på til å planlegge, var hvordan jeg skulle få siden på nett. Jeg hadde noen alternativer:

- å bruke en stasjonær pc plassert på Dyresenteret som server
- å bruke en Raspberry Pi som server, plassert hjemme hos meg selv
- eller kjøpe et domene med tilhørende webhotell

De to første alternativene virket mest spennende og som en del av prosjektet, men hadde potensielt mange feilkilder.

Etter som tiden forsvant fortere og fortere, bestemte jeg meg for enkleste løsning, et domene hos one.com, med tilhørende webhotell og database inkludert. I tillegg er filene sikret.

Jeg kan enkelt administrere siden selv, via one.com, både redigere noe i databasen eller filene.

Dette ble en fin løsning for Dyresenteret også, da det gjør det enkelt å administrere av andre personer.

5.4 Statistikk og flere registreringsfelt

I tidlig fase av prosjektet hadde jeg og oppdragsgiver tanker om å kunne registrere flere punkter om hver enkelt kunde, som for eksempel hvilke typer dyr de eier. Dette for å kunne lage statistikker til senere bruk. Men med tanke på både personopplysningsloven og tidsbruk ved registrering av kunde, så ble denne ideen forkastet.

En annen idé om statistikk var en enkel statistikk på postnumrene til de registrerte kundene. Da det kan være interessant å se fra hvilke områder butikken har flest kunder, og om det er mange som er villig til å reise lengre avstander for å handle hos Dyresenteret.

5.5 Sikkerhet og personvern

Jeg brukte tid på å finne ut hvilke sikkerhetstiltak som var viktige, og hva jeg kunne gjøre for at systemet skulle være så sikkert som mulig. Et mulig skadeomfang hvis sikkerheten ikke er på plass er lekkasje av personopplysninger. Selv om systemet ikke lagrer noen sensitive opplysninger, eller opplysninger som kan bli brukt til for eksempel ID-tyveri, så bør man være sikker på at man har gjort det man kan for å beskytte informasjonen.

En annen sikkerhetsrisiko kunne være at uvedkommende kommer seg inn i systemet. Per nå er det ingen automatisk utloggingsfunksjon på siden, noe som gjør at man står som innlogget hele arbeidsdagen. Sjansen for at uvedkommende tar seg inn i systemet med å bruke Dyresenterets maskin er derimot meget lav.

5.6 Design

Jeg har designet utseende og funksjonene på siden til å være dynamisk og enkelt å forstå seg på. Systemet virker hurtig og oversiktlig. Feilmeldinger vises med rød farge, og gir lite rom for feiltolkning. Fargene på siden er enkle; hvit bakgrunn, med lyse bakgrunnsfarger på felt, og knapper i klar og sterk farge.

Jeg har prøvd å følge prinsipper og logikk forklart av Jeff Johnson i boken «Designing with the mind in mind», [2.6.2] for å øke sannsynligheten for at siden blir forstått på rett måte. For eksempel har jeg plassert objekter som hører sammen i nærheten av hverandre, slik at de ser grupperte ut for brukeren. Knappene er i sterke farger og satt helt til siden eller under det den gjelder, for å skille dem fra resten av siden.

5.7 Etterspørsel og lignende systemer

Det finnes mange systemer som tar for seg kunderegistrering, men jeg har ikke funnet noen som hadde levert en løsning der man kan lage og registrere bonuskort på registrerte kunder.

Før prosjektets start hadde oppdragsgiver stilt et spørsmål i en lukket faghandelgruppe, om noen hadde en enkel løsning på å slippe unna de fysiske bonuskortene. Den eneste løsningen som ble presentert var et Excel-dokument, der man skrev kundens navn og kjøp i rekkene ved siden av. Kundens navn kunne da søkes opp i dokumentet. En billig og enkel løsning, men med en stor sjanse for feil, sletting og mangel på dokumentasjon. Ikke minst vil det gjerne bryte personopplysningsloven på flere punkter.

5.8 Bruk av kilder

Jeg har prøvd å finne det meste av teori i bøker eller artikler. Noen definisjoner var best forklart på internett, så en del teori ble basert fra artikler og nettsted som omhandlet temaet.

Jeg har brukt mange av bøkene jeg har blitt kjent med gjennom studiet, da jeg vet at de er relevante i forhold til hva jeg har lært, og hva studiet forventer at jeg skal kunne.

Under utviklingen av prosjektet har jeg brukt mye ideer fra offisielle sider, som for eksempel https://getbootstrap.com for å lære og ta i bruk Bootstrap. Jeg har også hentet ideer fra tutorials og eksempler, fra å google et konkret problem, sett på blogger, forum og eksempelkode. Jeg fikk også hjelp til å forstå og til bruke jQuery på noen funksjoner, for å gjøre siden mer dynamisk.

6 KONKLUSJON

Jeg konkluderer med at prosjektet er gjennomført i henhold til Kravspesifikasjonen

[Vedlegg 2]

Systemet inneholder alle funksjonene det var krav om å få med. Det er likevel noen få funksjoner og detaljer jeg ønsker å implementere etter levering av rapport, som for eksempel en enkel statistikk over postnummer (er det bare lokale kunder som handler hos Dyresenteret, hvor langt er kundene villig til å kjøre for å handle der?) og et mer personlig design, med bilder og figurer.

Jeg konkluderer med at systemet kan brukes i lang tid fremover, der jeg vil bistå om det oppstår problemer som jeg kan rette opp i. I løpet av 11 dager er det registrert over 170 kunder, der hver kunde har ett eller flere aktive bonuskort.

Gjennom møter med oppdragsgiver, og diskusjoner om hvordan systemet måtte fungere, fikk jeg god oversikt over hvilke krav som gjaldt, og det var aldri noen stor fare for sprik mellom hva jeg trodde oppdragsgiver ønsket, og hva de egentlig ønsket seg av funksjoner. Systemet ble presentert i delvis ferdig form så tidlig som mulig, og responsen fra oppdragsgiver har utelukkende vært positiv og engasjert. I innspurt av prosjektet har jeg fått gode tilbakemeldinger fra ansatte ved Dyresenteret, og de er svært fornøyd med en løsning som fungerer. [Vedlegg 4]

Jeg konkluderer også med at prosjektet er gjennomført på en ansvarlig måte i forhold til sikkerhet og behandling av personopplysninger.

Gjennom arbeid med dette prosjektet har jeg utviklet en bedre forståelse for hvordan arbeid med denne typen prosjekt vil kunne fungere. Alt fra planleggingsfase, møter med oppdragsgiver, og ikke minst de mange problemene man kan støte på under utvikling av et system. Jeg har dessverre ikke fått jobbet i et gruppeprosjekt med andre studenter, og føler jeg hadde kunne jobbet med bedre flyt om jeg hadde hatt noen andre å støtte meg på og dele problemer som dukket opp underveis med.

Kunne jeg begynt på nytt hadde jeg lagt større vekt på planlegging og dokumentering fra start av, og hatt en større feilmargin i forhold til tidsbruken på utviklingen av funksjoner.

Dette tar jeg med meg videre:

- Dele oppgaven opp i mindre deler, for å ha bedre oversikt og øke sjansen for å lykkes.
- Bruke god tid på de første møtene med oppdragsgiver, slik at jeg har en stor forståelse og oversikt over hva kunden ønsker seg fra første stund.
- Ha en klar idé og tanke på hvordan oppgaven skal løses før jeg setter i gang.

7 REFERANSER

[1] Datatilsynet – Hva er en personopplysning https://www.datatilsynet.no/rettigheter-og-plikter/personopplysninger, publisert: 13.06.2018, lest 20.10.2018.

[2] Datatilsynet – Om personopplysningsloven med forordning og når den gjelder https://www.datatilsynet.no/regelverk-og-verktoy/lover-og-regler/personvernregelverket/om-personopplysningsloven-og-nar-den-gjelder, publisert: 29.05.2018, lest 20.10.2018.

[3] Datatilsynet – Personvernprinsippene https://www.datatilsynet.no/rettigheter-og-plikter/personvernprinsippene/, publisert 13.06.2018, lest 20.10.2018

[4] Lovdata – Personopplysningsloven https://lovdata.no/dokument/NL/lov/2018-06-15-38/KAPITTEL_gdpr#KAPITTEL_gdpr, publisert 27.04.2016, lest 20.10.2018

[5] Kai Jæger Kristoffersen NRK – Dette er den nye personvernsloven https://www.nrk.no/nordland/storste-endringen-pa-over-20-ar-_-dette-er-den-nye-personvernloven-1.14039349 publisert 10.05.2018, lest 04.11.2018

[6] Difi – Kva seier foreskrifta? https://uu.difi.no/krav-og-regelverk/kva-seier-forskrifta#nettloysingar, publisert 16.03.2016, lest 23.10.2018

[7] Jerome Carter, EHR Science, Coupling and Cohesion: A view of software design from the inside out, publisert 12.11.2012, lest 25.10.2018

[8] Alex Andrews, Scrum of one, https://www.raywenderlich.com/585-scrum-of-one-how-to-bring-scrum-into-your-one-person-operation, publisert 02.06.2017, lest 01.11.2018

[9] PHP – Safe Password Hashing, http://php.net/manual/en/faq.passwords.php, online, lest 01.11.2018

[10] Michael E. Whitman and Herbert J. Mattord, Principles of Information Security, Fifth Edition, ISBN-13: 978-1-285-44836-7, Cengage Learning, 2016.

[11] Bjørn Kristoffersen, Databasesystemer, 4. utgave, ISBN 978-82-15-02708-1, Universitetsforlaget, 2016.

[12] Databasedesign, https://www.lucidchart.com/pages/database-diagram/database-design, online, lest 05.11.2018.

[13] Simson Garfinkel with Gene Spafford, Web Security, Privacy & Commerce, Second Edition, ISBN: 0-596-00045-6, O'Reilly Media, 2002.

[14] Datatilsynet – Samtykke https://www.datatilsynet.no/regelverk-og-verktoy/veiledere/veileder-om-behandlingsgrunnlag/samtykke/, online, lest 16.11.2018

[15] Jeff Johnson, Designing with the Mind in Mind, Second Edition, ISBN: 978-0-12-407914-4, Elsevier - Morgan Kaufmann, 2014

[16] Ian Sommerville, Software Engineering, Tenth Edition, ISBN 10: 1-292-09613-6, Pearson Education Limited, 2016.

[17] Codecademy – MVC

https://www.codecademy.com/articles/mvc , online, lest 15.10.2018

[18] CSS - https://www.w3schools.com/css/css_intro.asp , online, lest 16.11.2018

[19] HTML - https://www.w3schools.com/html/html_intro.asp, online, lest 16.11.2018

[20] PHP - https://www.w3schools.com/php/default.asp, online, lest 16.11.2018

[21] Netlife Research – Brukertesting
http://iallenkelhet.no/2010/03/22/gjør-det-selv-5-trinn-til-brukertestet-nettsted/, publisert 22.03.2010, lest 16.11.2018

[22] JavaScript - https://www.w3schools.com/whatis/whatis_js.asp, online, lest 16.11.2018

[23] Dan M. Brown, Communicating Design: Developing Web Site Documentation for Design and Planning, Second Edition, ISBN: 978-0321712462, New Riders Press, 2011.

[24] UML - https://www.ntnu.no/wiki/display/tdt4140/En+rask+introduksjon+til+UML, publisert 10.02.15, lest 20.11.2018

[25] Prototype - https://no.wikipedia.org/wiki/Prototyp, online, lest 16.11.2018[27]

[26] Chris Bates – Web Programming: Building Internet Applications, Third Edition, ISBN: 978-0-470-01775-3, John Wiley & Sons, Ltd, 2006.

[27] https://www.one.com

[28] https://www.mamp.info/en/

[29] https://www.w3schools.com

[30] http://www.yogihosting.com/what-is-bootstrap/

[31] https://getbootstrap.com

[32] https://stackoverflow.com

[33] https://www.youtube.com