HOVEDPROSJEKT: KIKKI'S PIZZA

TITTEL

Grandis

FORFATTER(E): FREDRIK LYSAKERUD
MORTEN KRISTOFFERSEN
OLE JØRGEN REIERSTAD

Dato: 23.05.01

SAMMENDRAG AV HOVEDPROSJEKT

Tittel:	Grandis	Nr. :
		Dato : 23.05.01
Deltaker(e):	Fredrik Lysakerud	
	Morten Kristoffersen	
	Ole Jørgen Reierstad	
Veileder(e):	Svein Gautestad	
Oppdragsgiver:	Kikki's Pizza Gjøvik	
Kontaktperson:	Sveinung Mikalsen	
Stikkord (4 stk)	ISDN, Capi, Database, C++ Builder	

Antall sider: 48	Antall bilag:3	Tilgjengelighet (åpen/konfidensiell): Åpen

Kort beskrivelse av hovedprosjektet:

Hovedprosjektet gikk ut på å lage et helt nytt betillingssystem for Kikki's pizza da det de har brukt frem til nå er blitt gammeldags og innefektivt. Dette ble gjordt for hånd, med penn og papir.
De ønsket seg et brukervennelig, effektivt og stabilt system på en datamaskjin.
De var også inne på tanken om å bruke touch-screen for å forenkle bruken, noe som er blitt tatt høyde for under utviklingen.
Det er, etter ønske fra Kikki's, lagt vekt på brukervenlighet og effektivitet.

Forord

Denne rapporten er skrevet på grunnlag av hovedprosjektet vårt ”Grandis”. Hovedprosjektet er avslutning på den treårige dataingeniørutdanninga vi går på. Hovedprosjektet skal gi oss en realistisk og faglig relevant problemstilling, og legges opp slik at kunnskap og ferdigheter fra flere fagområder benyttes. Gjennom hovedprosjektet får vi erfaring i gruppearbeid i prosjekt som vi vil få bruk for senere i arbeidslivet. Hovedprosjektet er obligatorisk og teller 4 av 60 vekttall ved ingeniørstudiet.

Oppdragsgiver for prosjektet er Kikki’s Pizza. Kikki’s leverer pizza i Gjøvik området. Oppgaven vår går ut på å utvikle en applikasjon som benyttes til å ta imot bestillinger.

Vi vil gjerne takke:
Daglig leder av Kikki’s Pizza, Sveinung Mikalsen
Veileder, Svein Gautestad
Tor André Johansen, C++ nerd

Gjøvik 23.05.2001

Fredrik Lysakerud	Ole Jørgen Reirstad	Morten Kristoffersen

1 Innledning

1.1 Problemområde

1.1.1 Definering av oppgaven

Situasjonen oss Kikki's nå er at bestilinger kommer inn over telefon. Bestillingen må da skrives ned på lapper. Dette kan fort bli rotete og uoversiktlig. Om morgenen og tidlig på dagen, da det er liten trafikk, er det kun kokken som er der. Bestillingene gjøres da manuelt av kokken og tar mye tid da han har alt ansvaret rundt en bestilling. Da kan det fort bli hektisk å ta imot bestillinger og lage maten samtidig. Når pizzaen så skal kjøres ut, må budet selv finne frem bestillingene og organisere kjøreruten. Når det er mye bestillinger blir dette en vanskelig oppgave. Det blir fort veldig uoversiktlig. Det blir ofte en del unødvendige lange turer på pizzabudet, på grunn av dårlige planlagte kjøreruter. Kikki's Pizza har også et problem med svinn av råvarer. Forskjellige kokker bruker ulik mengde med ingredienser når de lager pizza. Dette fører til dårlig oversikt over lagerbeholdningen av råvarer

Målet for prosjektet er å utvikle en Windows applikasjon for bestilling og administrasjon etter ønsker fra Kikki's pizza

1.1.2 Henvisning til tidligere arbeid

Steve O'Conor har laget et pizza program som kan lastes ned på:
http://members.iweb.net.au/~steveoc/gtk_pizza/

1.1.3 Målgruppen

Brukerne av det ferdige systemet er ansatte hos Kikkis. Brukerne kan ha alt fra ingen til bra erfaring med bruk av datamaskiner og Windows applikasjoner.

Rapporten er skrevet for veilederen og sensoren av hovedprosjektet. I tillegg til Kikkis som her får kravspesifikasjonen og design/analyse av systemet. Kikkis kan gi denne rapporten til utviklere av eventuelle utvidelser og endringer.

1.1.4 Studentenes faglige bakgrunn

Alle deltakerne i prosjektgruppa går på det tredje og siste året på dataingeniørstudiet.

I faget "Grafiske brukergrensesnitt" som vi hadde høsten 2000, fikk vi kunnskap om å lage windows- og database-applikasjoner i Borland utviklingsverktøy.

Vi hadde ingen kunnskap om CAPI, (Common ISDN API) som er protokollen som brukes for å hente og sende informasjonen ved hjelp av et ISDN-kort.

Studentene har kunnskap innen:
C++ - Objektorientert Programmeringsspråk, plattformavhengig
Databaser - Databasedesign og implementering
Systemutvikling - Strukturert og objektorientert systemutvikling, *UML*, og *Rational Rose*

1.1.5 Prosjektgruppas valgte arbeidsform

I begynnelsen av prosjektet, under analyse og designfasen jobbet hele gruppa sammen. På denne måten fikk vi fram alles synspunkter og ideer. Da vi gikk over til koding hadde vi felles møter innad i gruppa, og fordelte oppgaver til hver enkelt deltaker.

Vi opprettet et "arbeidskontor" hos Ole Jørgen Reierstad der vi hadde statusmøtene og jobbet med fellesarbeid.

1.1.6 Terminologibruk

Det er brukt terminologi som letter forklaring av utviklingsproblemer og løsninger.
Ord og uttrykk som særskilt brukes i denne sammenhengen står i ordlisten. Det har vært forsøkt å holde teknologi-faktoren på et lavt nivå i språkbruken, slik at rapporten blir lettlest for utenforstående.

CAPI – (COMMON-ISDN-API) Er en standard for applikasjonsprogrammering mot ISDN-kort. Ved å benytte seg av denne standarden kan man kalle ferdig definerte funksjoner

ISDN - (Integrated Services Digital Network)

SQL – (Structured Query Language) Er et standardspråk for spørringer mot databaser.

Rational Rose – Verktøy for objektorientert analyse og design.

UML - (Unified Modeling Language) Standard for objektorientert analyse og design.

BDE – (Borland Database Engine) Applikasjon fra Borland som er et bindeledd mellom databasen og applikasjonen.

INTERBASE - Databasesystem.

Pascal – Programmeringsspråk.

C++ - Programmeringsspråk.

Delphi – Pascal basert utviklingsverktøy fra Borland.

Borland C++ Builder – C++ basert utviklingsverktøy fra Borland.

GUI – (Graphical User Interface) Grafisk Brukergrensesnitt.

WAP - Wireless Application Protocoll, surfe med mobil på Internett

SMS – Tekstmeldinger fra mobiltelefon

AnyWeb – Telefon fra Samsung med internet tilgang over ISDN

1.2 Organisering av dokumentet

Litteraturlisten gir en oversikt over litteratur som er brukt i prosjektet, mens ordliste forklarer ord som er av teknisk art eller som er uklare.

1. **Innledning**, gir en kort beskrivelse av oppgaven, arbeidsformer og studentenes bakgrunn. Inneholder også en ordliste.

2. **Kravspesifikasjon**, forklarer dokumentprosessen frem til den endelige kravspesifikasjonen. Selve kravspesifikasjonen er innlemmet i dette punktet.

3. **Design,** begrunner metodevalg og teknikkbruk for designfasen. Designdokumentet ligger som vedlegg til rapporten.

4. **Implementering,** utreder om verktøyvalg og implementeringsteknikker da vi kodet.

5. **Testing og kvalitetssikring,** viser hva vi gjorde for å sikre et godt produkt. Det demonstreres også hvilke fremgangsmåter vi bruke for å teste programkoden.

6. **Diskusjon av resultatet** viser hva systemet gjør og hvilke krav som ønskes. Viser lit om hvilke avvik det ble fra ønskelige krav.

7. **Konklusjon** beskriver hva gruppa kom fram til og enkeltpersoners meninger.

8. **Litteraturliste**

9. **Innholdsfortegnelse over vedlegg**

2 Kravspesifikasjon

2.1 Overordnet kravspekk

2.1.1 Bakgrunn

Kikkis Pizza har et ønske om å få et automatisert bestillingssystem. Dagens situasjon er at alle bestillinger tas imot manuelt. Bestillingen må skrives ned på lapper. Dette kan fort bli rotete og uoversiktlig. Om morgenen og tidlig på dagen, da det er liten trafikk, er det kun kokken som er der. Bestillingene gjøres da manuelt av kokken og tar mye tid, da han har alt ansvaret rundt en bestilling. Da kan det fort bli hektisk å ta imot bestillinger og lage maten samtidig. Når pizzaen så skal kjøres ut, må budet selv finne frem bestillingene og organisere kjøreruten. Når det er mye bestillinger blir dette en vanskelig oppgave. Det blir fort veldig uoversiktlig. Det blir ofte en del unødvendige lange turer på Pizzabudet, på grunn av dårlige planlagte kjøreruter. Kikki's Pizza har også et problem med svinn av råvarer. Forskjellige kokker bruker ulik mengde med ingredienser når de lager pizza. Dette fører til dårlig oversikt over lagerbeholdningen av råvarer

2.1.2 Kort om krav til systemet

Applikasjonen skal koble bestemte telefonnummer mot bestemte kunder via en database slik at arbeidet til kokken lettes. Når det ringes inn en bestilling kommer telefonnummeret automatisk opp på skjermen med navn og adresse til den som eier telefonen.
Kokken slipper å skrive ned bestillingen for hånd slik som situasjonen er nå. Kokken styrer hele bestillingsdelen fra en terminal, gjerne med hjelp av en touch-screen.
Hvis det skulle være nødvendig å gjøre om noen data til en kunde, for eksempel en annen adresse en det som står oppført, er det bare å skrive dette inn i de tilsvarende feltene.

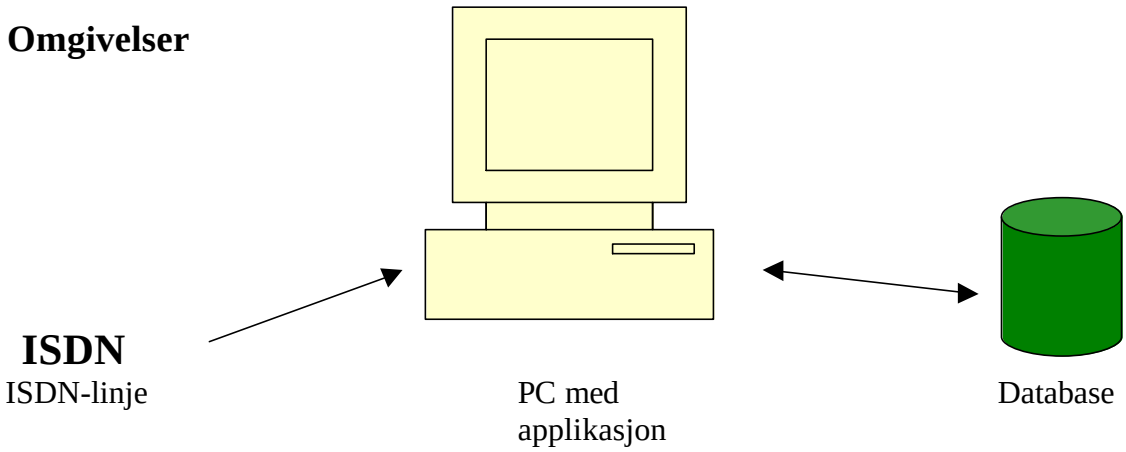
Adressene skal organiseres i geografiske soner for å lette arbeidet til pizzabudene. Sonene blir bestemt av kartreferanser over Gjøvik området. Kommer det inn bestillinger som skal til en sone som er på veien til en annen sone skal denne bestillingen bli tatt med på denne ruten. Når en bestilling kommer inn blir den lagt til en sone slik at sjåføren kan se etter om det er en bestilling som snart er klar. Han kan da ta en vurdering om han vil vente til den bestillingen er ferdig eller kjøre ut den han har og overlate den til neste kjøring. På denne måten vil det bli letter å legge opp mer effektive kjøreruter.

Systemet har muligheter for å skrive ut forskjellige rapporter. Det er ett behov for å sammenlikne det faktiske bruk av råvarer med det som er budsjettert.

CAPI protokollen brukes til kommunikasjon mellom datamaskinen med programmet på og ISDN-linja. Det skal opprettes en database med register over alle kundene. GUI skal lages slik at en hvilken som helst person skal kunne skjønne og bruke programmet.
Systemet skal gi mulighet for å kunne kontrollere forbruket av varebeholdningen, slik at det blir en bedre kontroll over hva som er igjen av råvarer. Det skal føres et regnskap over antall solgte Pizzaer og andre varer.

2.2 Bruker beskrivelse

2.2.1 Omgivelser



Figur 2.2.1

ISDN-linje:
Kikkis ISDN linje er koblet til ISDN-kortet til PC'en applikasjonen kjører på. Telefonnummeret til kunden kan da leses inn.

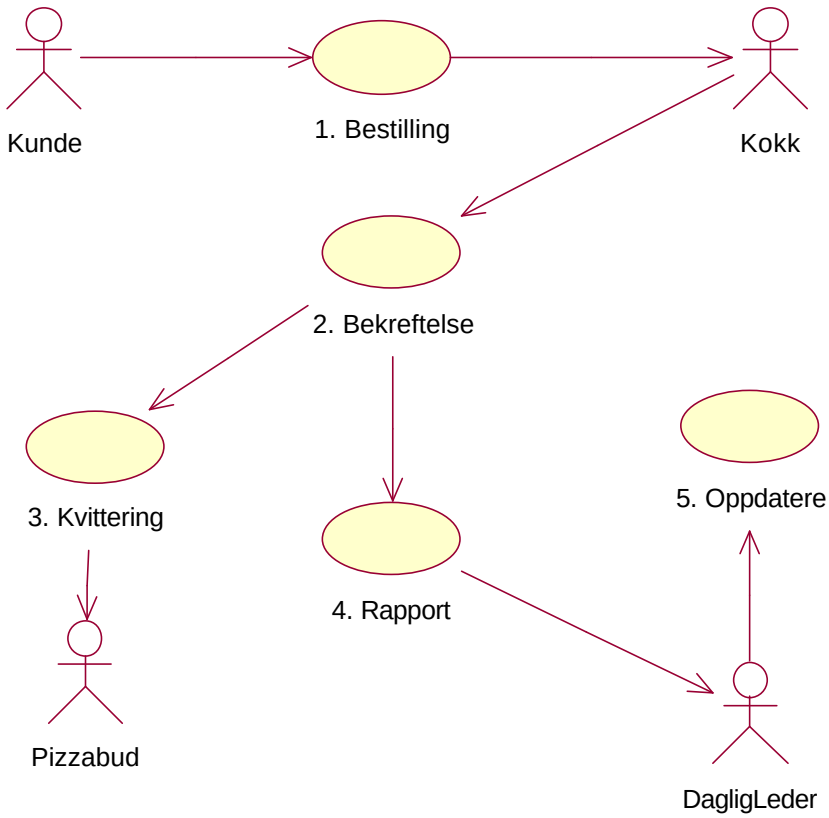
PC med applikasjon:
PC'en der programmet kjøres. Brukerne på Kikkis legger inn bestillingene her og kan ta ut rapporter og lignende.

Database:
Her ligger kunderegisteret og alle registrerte bestillinger. Det kan tas backup på forespørsel og ved bestemte tidspunkter. Databasen skal kjøres på Interbase 6 server. Dette databasesystemet er en SQL-database som er "open-source".

2.2.2 Systemets brukere

Brukeren av applikasjonen er de ansatte ved Kikkis. De kan ha alt fra ingen til god kunnskap om bruk av Windows programmer. Applikasjonens GUI må derfor være selvforklarende.

2.2.3 Funksjoner



Figur 2.2.3

2.2.3.1 Kunde

Kunden er den som ringer inn og bestiller pizza og drikke over telefon. Kunden kan også møte opp hos Kikki's lokaler og hente pizzaen direkte.

2.2.3.2 Bestilling

Use caset startes ved at kunden ringer inn en bestilling. Telefonnummeret blir sammenliknet mot systemets database. Her ligger data om kundens navn, adresse og telefonnummer.

2.2.3.3 Kokken

Kokken er den personen som skal ta imot bestillingen.

2.2.3.4 Bekreftelse

Use caset startes ved at kokken bekrefter at adressen til kunden stemmer, hvis ikke skal kunden legges inn i databasen. Bestillingen til kunden skal registreres i systemet og når pizzaen er ferdig laget skal kokken kvittere den ut.

2.2.3.5 Kvittering

Det blir skrevet ut en kvittering til kunden. Inneholder kunde info, priser og hva som er bestilt.

2.2.3.6 Pizzabud

Pizza budet får kvitteringen slik at han kan se hvor bestillingen skal hen. Kvitteringen viser hvilken sone og adresse den skal til, samt informasjon om bestillingen.

2.2.3.7 Rapport

Systemet skal lage rapporter til daglig leder. Rapportene skal gi oversikt over hvilke varer og pizzaer som er blitt solgt.

2.2.3.8 Daglig leder

Daglig leder står for oppdateringer av system og mottar ulike rapporter.

2.2.3.9 Oppdatere

Daglig leder skal kunne legge inn nye pizza typer og varer i systemet, endre priser, påfyll av råvarer og lignende.

2.2.4 Operasjon

Systemet skal være oppe under hele åpningstiden for Kikki's Pizza.

Feil: Telefonlinjen kan være nede
Systemkrasj
Hardware problemer

2.2.5 Aspekter omkring livssyklus

Vedlikehold og oppdatering av systemet gjøres hos Kikki's.

Å legge inn nye pizzaer og varer gjøres av ansatte hos Kikki's. Det samme med nye kunder i databasen.

Systemet kan flyttes til andre bedrifter som selger de samme varene som Kikki's.

2.2.6 Ytelse

Minimum Pentium I og 64MB RAM. Vi anbefaler Pentium II og 128MB RAM
Harddisk bør være stabil å ha 7200 rpm.

2.2.7 Begrensninger

Økonomiske begrensninger
For at investeringen skal bli minst mulig for Kikki's, skal det benyttes lisens frie programmer som skal inngå i det ferdige resultatet og som ikke skal utvikles av prosjektgruppa.

2.2.8 Antagelser

For å kunne utvikle, spesielt telefon delen, forutsetter det at vi har tilgang til alle komponenten til testing under utviklingen.

2.3 Detaljert Kravspesifikasjon

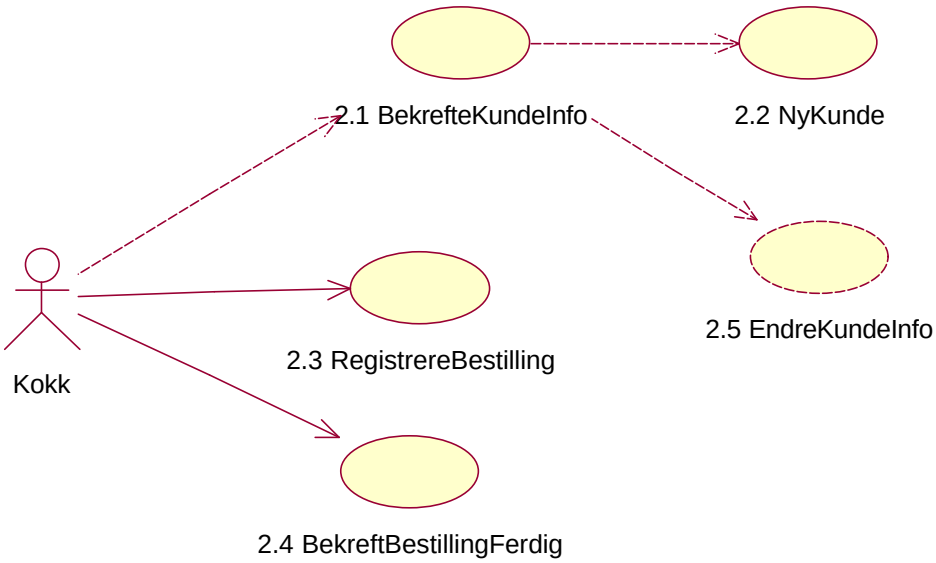
2.3.1 Funksjonell struktur og tverrelasjoner

2.3.1.1 Bestilling

Figur 2.3.1.1

1.1 RingInnBestilling
Kunden ringen inn, og det blir registret et telefonnummer. Telefonnummeret blir lest inn ved hjelp av et ISDN kort og CAPI protokollen. Telefonnummeret blir satt opp på spørring mot databasen. Det vil komme opp på skjermen om kunden ligger i databasen eller ikke.

2.3.1.2 Bekreftelse



Figur 2.3.1.2

2.1 BekrefteKundeInfo
Hvis kunden finnes i databasen vil adresse og navn komme opp på skjermen. Kokken skal da bekrefte om navn og adresse er riktig.

2.2 NyKunde
Hvis kunden ikke ligger lagret i databasen skal kokken skrive inn informasjon på det telefonnummeret som ringer inn.

2.3 RegistrereBestilling
Kokken skal fylle inn de bestillingen som blir gjort av kunden.

2.4 BekreftBestillingFerdig
Når en pizza er ferdig laget, skal den tas bort fra ventelisten og bli gjort klar til utkjøring.

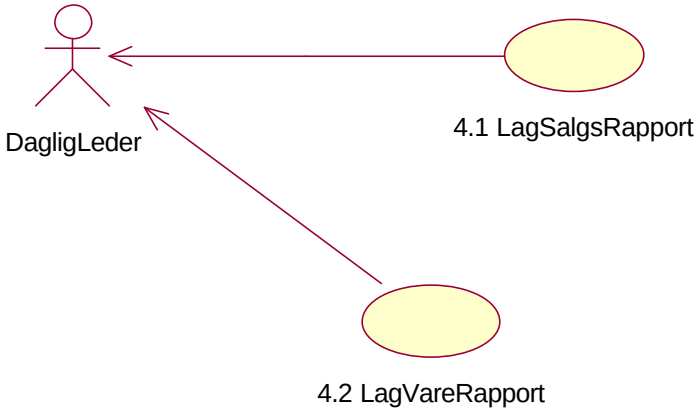
2.5 EndreKundeInfo
Navn og adresse kan endres hvis bestillingen ikke skal til den opprinnelige adressen som er registrert. Denne forandringen blir ikke registrert i databasen.

2.3.1.3 Kvittering

Figur 2.3.1.3

3.1 Kvittering
Det blir skrevet ut en kvittering med navn, adresse, telefonnummer, sone og hvilke bestillinger som er gjort.

2.3.1.4 Rapport

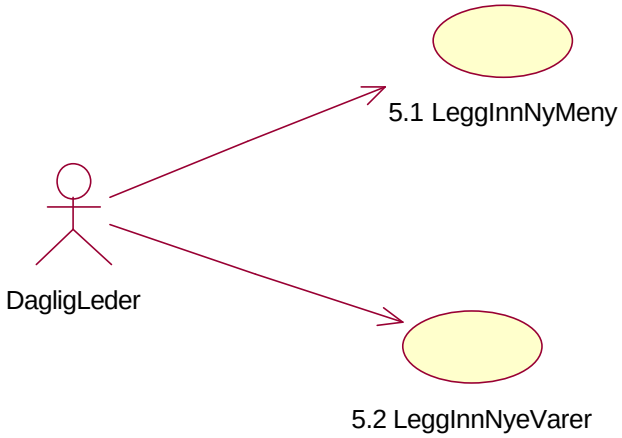


Figur 2.3.1.4

4.1 LagSalgsRapport
Daglig leder skal ha muligheten til å få skrevet ut en rapport på hvor mange bestillinger som er blitt gjort.

4.2 LagVareRapport
Daglig leder vil ha en oversikt over hvor mage pizzaer/varer og hvilke pizzaer/varer det er blitt solgt fra en bestemt dato. Kan også velge å skrive ut hvilke ingredienser som er blitt brukt.

2.3.1.5 Oppdatere



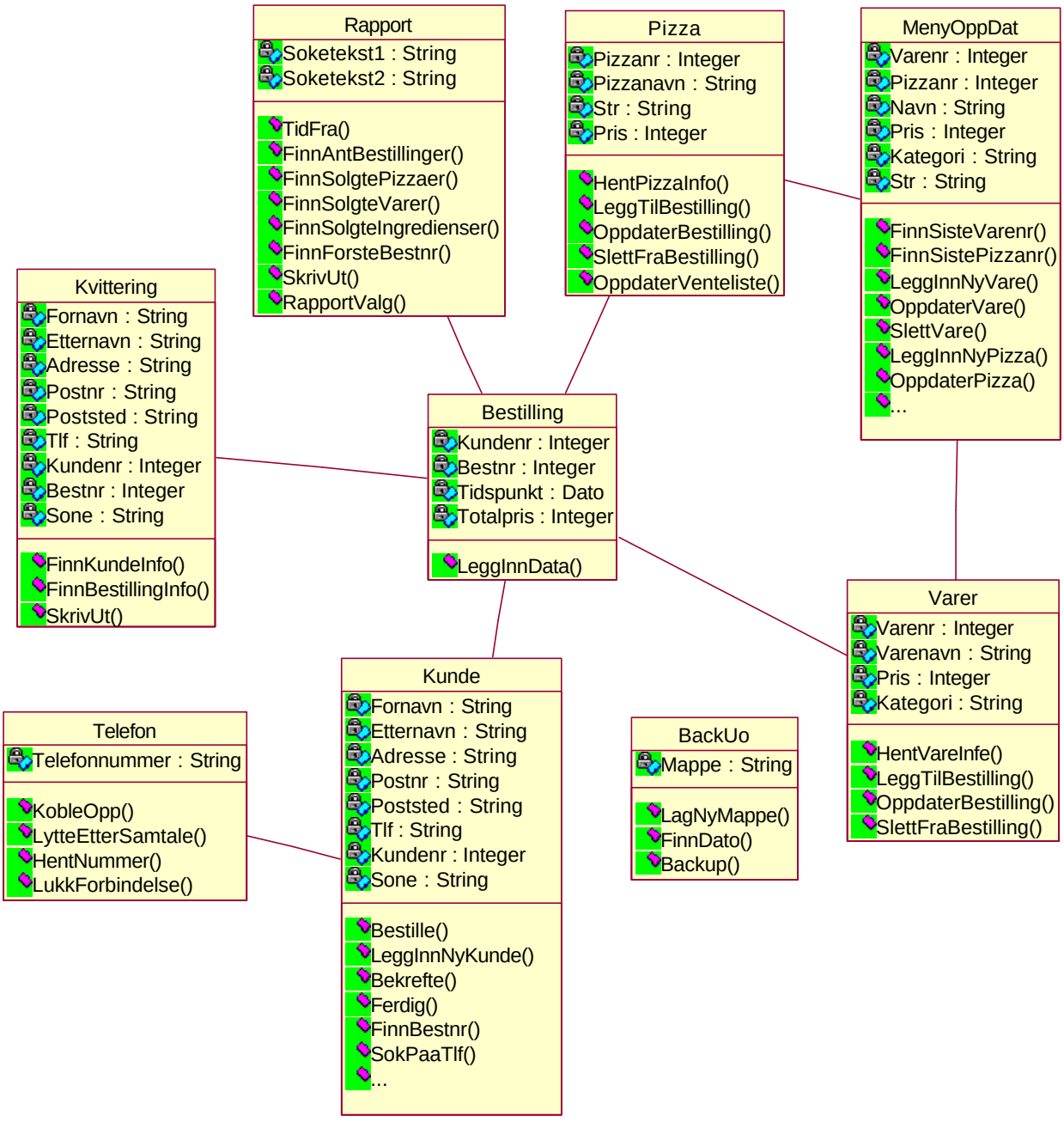
Figur 2.3.1.5

5.1 LeggInnNyMeny
Det skal være mulighet for å gjøre forandringer på menyen. Nye pizza typer og nye priser. Kan koble hvilke ingredienser som er på pizzaene.

5.2 LeggInnNyVarer
Legger inn nye varer som drikke, sauser, sigaretter og ingredienser. Kan oppdatere prisene på de som allerede er der og det er mulighet for å slette varer.

2.3.2 Data spesifikasjoner

2.3.2.1 Data rammeverk



Figur 2.3.2.1 Data rammeverk

Programmet består av følgende klasser:

Kunde inneholder funksjoner for å legge inn nye kunder, finne nytt bestillingsnummer og eventuelt nytt kundenummer. Den søker opp telefonnummer i databasen.
Den starter en bestilling som er knyttet opp mot en knapp.

Bestilling legger til ny data inn i bestillingene. Dette er stort sett SQL spørringer mot databasen.

Pizza henter informasjon fra databasen på de enkelte pizzaene. Passer på at vinduer som lister opp pizzaene alltid er oppdaterte. Legger pizza til en bestilling.

Vare henter informasjon fra databasen på de enkelte varene. Passer på at vinduer som lister opp varene alltid er oppdaterte. Legger vare til en bestilling.

Rapport søker i databasen på informasjon som blir valgt fra brukeren. Skriver ut denne informasjonen.

Kvittering henter ut informasjon om kunden og dens bestilling.

MenyOppDat oppdaterer tabellene i databasen over pizza og vare. Sletter, legger til ny og endrer data.

BackUp tar kopier av databasen med jevne mellomrom tilfelle uforutsigbare hendelser.

Telefon lytter på ISDN kortet om det kommer en oppringning.

2.3.2.2 Tverr-funksjonelle data definisjoner

Kommunikasjon modulene imellom vil foregå ved parameteroverføring. Objekter som er attributter i andre objekter kalles her parametere når de overføres. Annen kommunikasjon mellom modulene er beskjeder eller funksjonskall fra en klasse om å starte en funksjon i en annen klasse.

2.3.3 Overordnede operasjonelle systemkrav

2.3.3.1 Normal Operasjon

2.3.3.1.1 Modus og kontroll

Systemet skal være oppe under Kikki's åpningstid.
Ved oppstart skal applikasjonen opprette en forbindelse mot ISDN kortet.
Når en samtale kommer sender ISDN kortet informasjon til applikasjonen.
Når systemet tas ned stenges denne forbindelsen.

2.3.3.1.2 Ytelse

Responstid: Skal være minimal, da telefonnummeret skal komme opp på skjermen med navn og adresse. Max 1 sekund.

Kapasitet: Databasen over kunder skal kunne håndtere et register på minimum 10000 forskjellige kunder. Det skal kunne håndtere minimum 50 bestillinger samtidig.

Antall brukere: Enbruker system.

Brukervennlighet: Systemet skal ha stor brukervennlighet. Brukeren har liten eller ingen kjennskap til bruken av dataverktøy. Brukergrensesnittet vil enten være Windows basert eller touch-screen.

Robusthet: Det skal ikke kunne gjøres feil fra brukerens side slik at systemfeil skjer. Det er stor sannsynlighet for at brukeren kan gjøre feil.

2.3.3.1.3 Sikkerhet

Systemet skal kun brukes av Kikki's ansatte. Ikke alle brukere skal ha samme tilgang til systemet. Det er kun daglig leder som skal kunne skrive ut rapporter og legge inn nye pizzatyper. Dette skal gjøres ved hjelp av brukernavn og passord.

2.3.3.1.4 Oppstart og nedtagning

Det skal ikke erstatte noen eldre systemer.
Installering skjer ved hjelp av installerings programmet Setup Generator.

2.3.3.1.5 Innebygde tester

Når systemet startes opp skal det testes om ISDN kortet virker som det skal. Brukeren skal få beskjed om dette.

2.3.3.2 Operasjon i feilsituasjon

2.3.3.2.1 Feilrapportering

Dersom programmet avsluttes feil, vil brukeren få beskjed om dette ved neste oppstart.

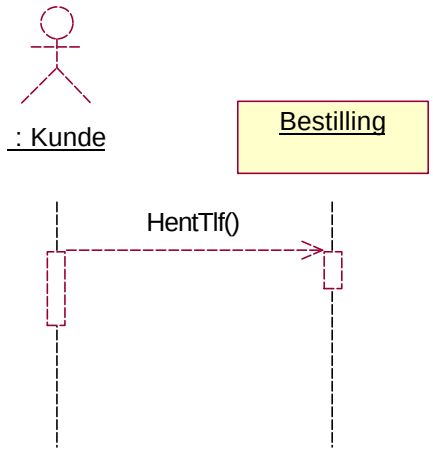
2.3.3.2.2 Gjenervervelse etter feil

Gjenervervelse etter feil skjer manuelt av personellet ved Kikki's pizza.

2.3.3.2.3 Sikkerhet

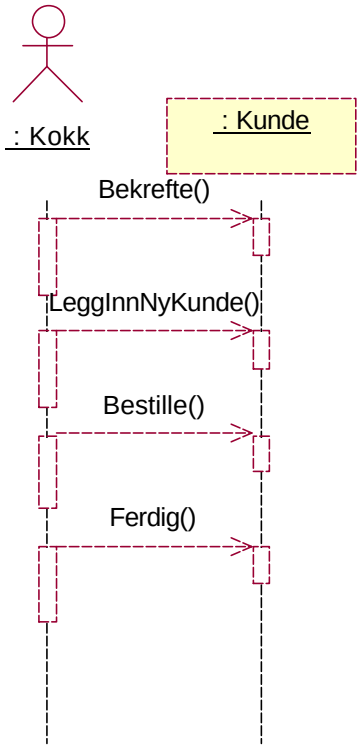
Det blir tatt backup av databasen hvert 5 min gjennom hele oppetiden.

2.3.4 Funksjonelle krav



Figur 2.3.4.A

Navn	**HentTlf()**
Ansvar	- Skal finne telefonnummeret
Type	System
Kryssreferanser	Fra Use Caset Bestilling
Unntak	Hvis det er anonymt telefonnummer
Prebetingelser	En kunde må ha ringt inn
Postbetingelser	Programmet er klar til å ta imot en bestilling



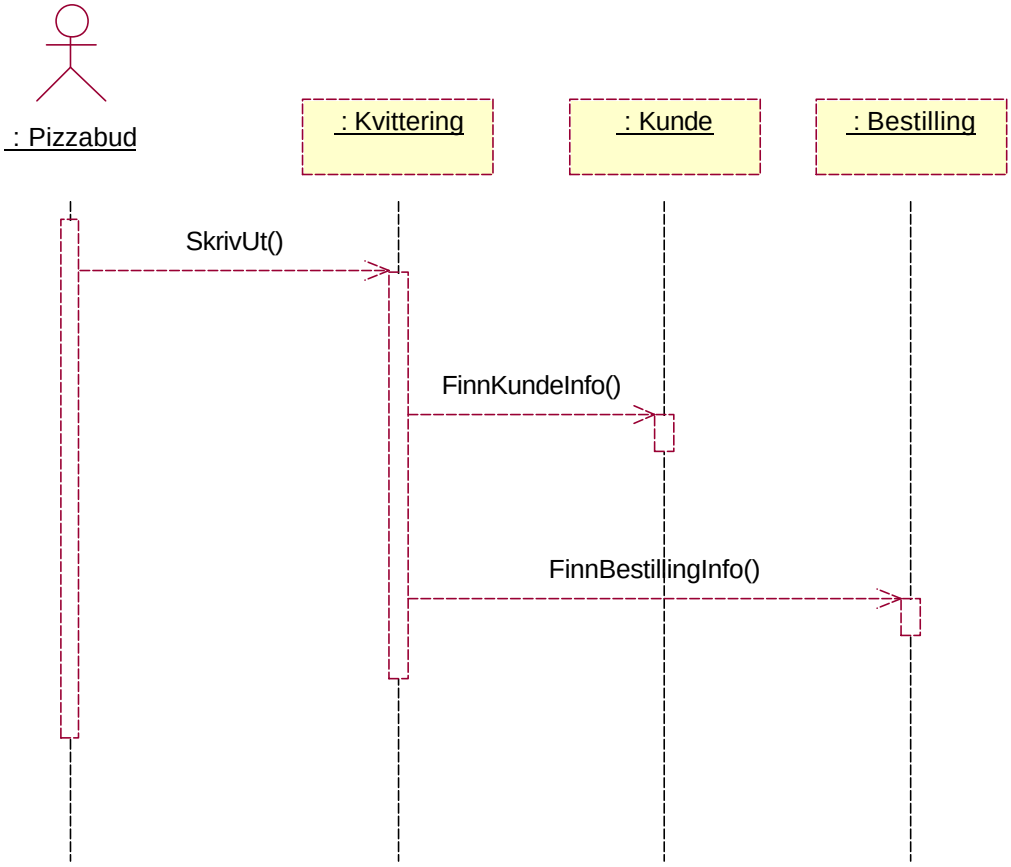
Figur 2.3.4.B

Navn	**Bekrefte()**
Ansvar	- Skal bekrefte at nødvendig informasjon om kunden blir registrert
Type	System
Kryssreferanser	Fra Use Caset Bekreftelse
Unntak	Hvis det ikke skal registreres en bestilling
Prebetingelser	Det må ha blitt gjort en bestilling
Postbetingelser	Klar til å skrive ut kvittering

Navn	**LeggInnNyKunde()**
Ansvar	- Skal ta imot informasjon om kunden som skal legges inn - Det skal tildeles et kunde nummer
Type	System
Kryssreferanser	Fra Use Caset Bekreftelse
Unntak	Hvis kundens telefonnummer allerede eksisterer
Prebetingelser	Kunde nummer må være tildelt Telefonnummeret må ikke finnes fra før
Postbetingelser	Kunden kan bestille

Navn	**Bestille()**
Ansvar	- Skal tildele et unikt bestillingsnummer.
Type	System
Kryssreferanser	Fra Use Caset Bekreftelse
Unntak	En kunde må ha ringt inn
Prebetingelser	Kunden må være registrert Må ha fått tildelt et bestillingsnummer
Postbetingelser	

Navn	**Ferdig()**
Ansvar	- Skal ta bort en pizza fra ventelisten
Type	System
Kryssreferanser	Fra Use Caset Bekreftelse
Unntak	
Prebetingelser	En pizza må være valgt
Postbetingelser	Ventelisten blir oppdatert

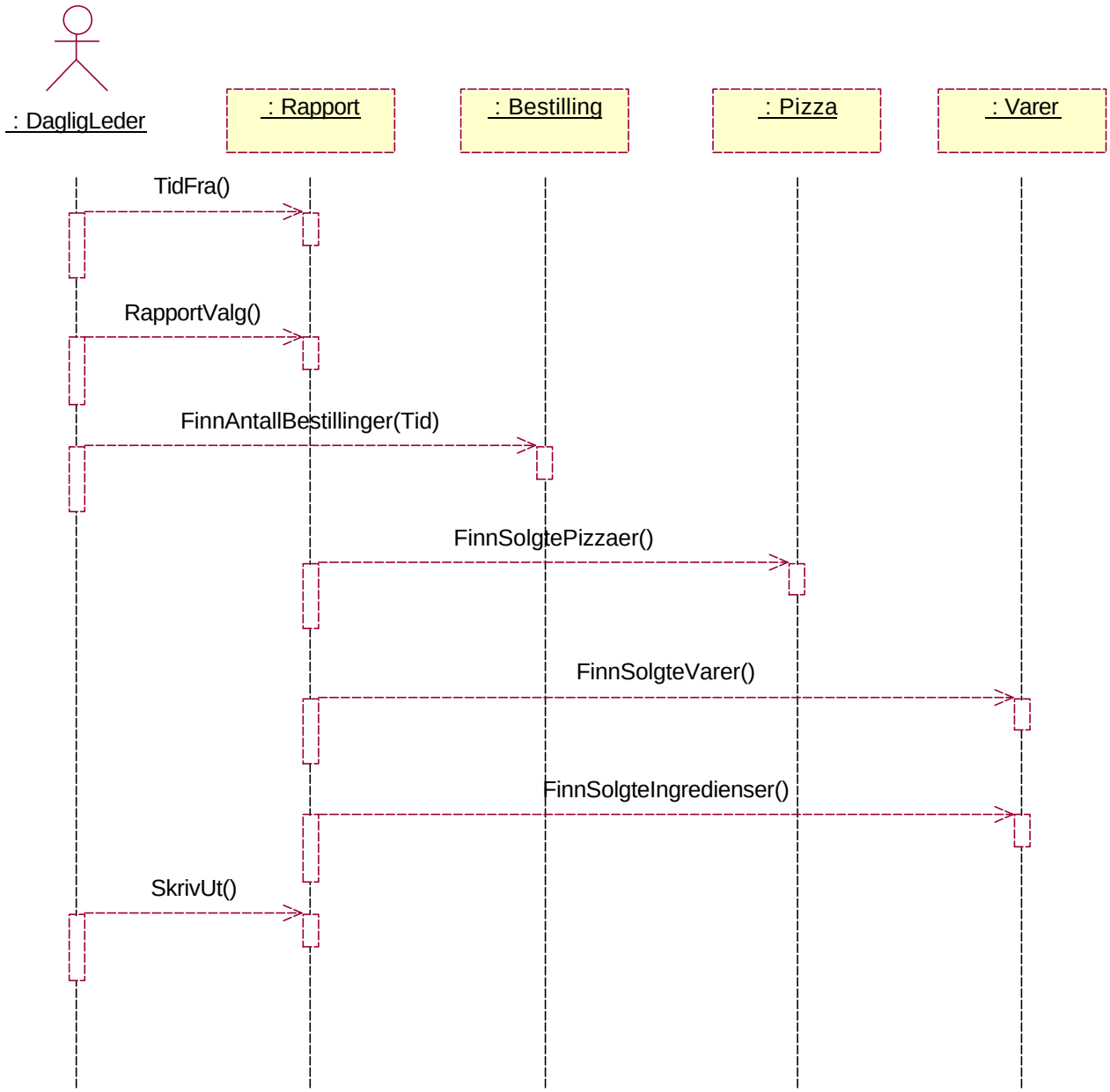


Figur 2.3.4.C

Navn	**FinnKundeInfo()**
Ansvar	- Skal finne all kunde informasjon som er nødvendig for kvitteringen
Type	System
Kryssreferanser	Fra Use Caset Kvittering
Unntak	
Prebetingelser	En bestilling må være gjort
Postbetingelser	

Navn	**FinnBestillingInfo()**
Ansvar	- Skal finne hvilke varer og pizzaer som er bestilt - Finner priser
Type	System
Kryssreferanser	Fra Use Caset Kvittering
Unntak	
Prebetingelser	En bestilling må være gjort
Postbetingelser	

Navn	**SkrivUt()**
Ansvar	- Skal skrive ut en kvittering til kunden
Type	System
Kryssreferanser	Fra Use Caset Kvittering
Unntak	
Prebetingelser	En bestilling må være gjort
Postbetingelser	


: DagligLeder
: Rapport
: Bestilling
: Pizza
: Varer
TidFra()
RapportValg()
FinnAntallBestillinger(Tid)
FinnSolgtePizzaer()
FinnSolgteVarer()
FinnSolgteIngredienser()
SkrivUt()

Figur 2.3.4.D

Navn	**TidFra()**
Ansvar	- Skal finne et tidspunkt/Dato
Type	System
Kryssreferanser	Fra Use Caset Rapport
Unntak	Nåtid ikke registrert
Prebetingelser	
Postbetingelser	

Navn	**RapportValg()**
Ansvar	- Skal finne ut hvilken rapport type som er valgt
Type	System
Kryssreferanser	Fra Use Caset Rapport
Unntak	En rapport type må være valgt
Prebetingelser	En Tid/Dato må være valgt
Postbetingelser	

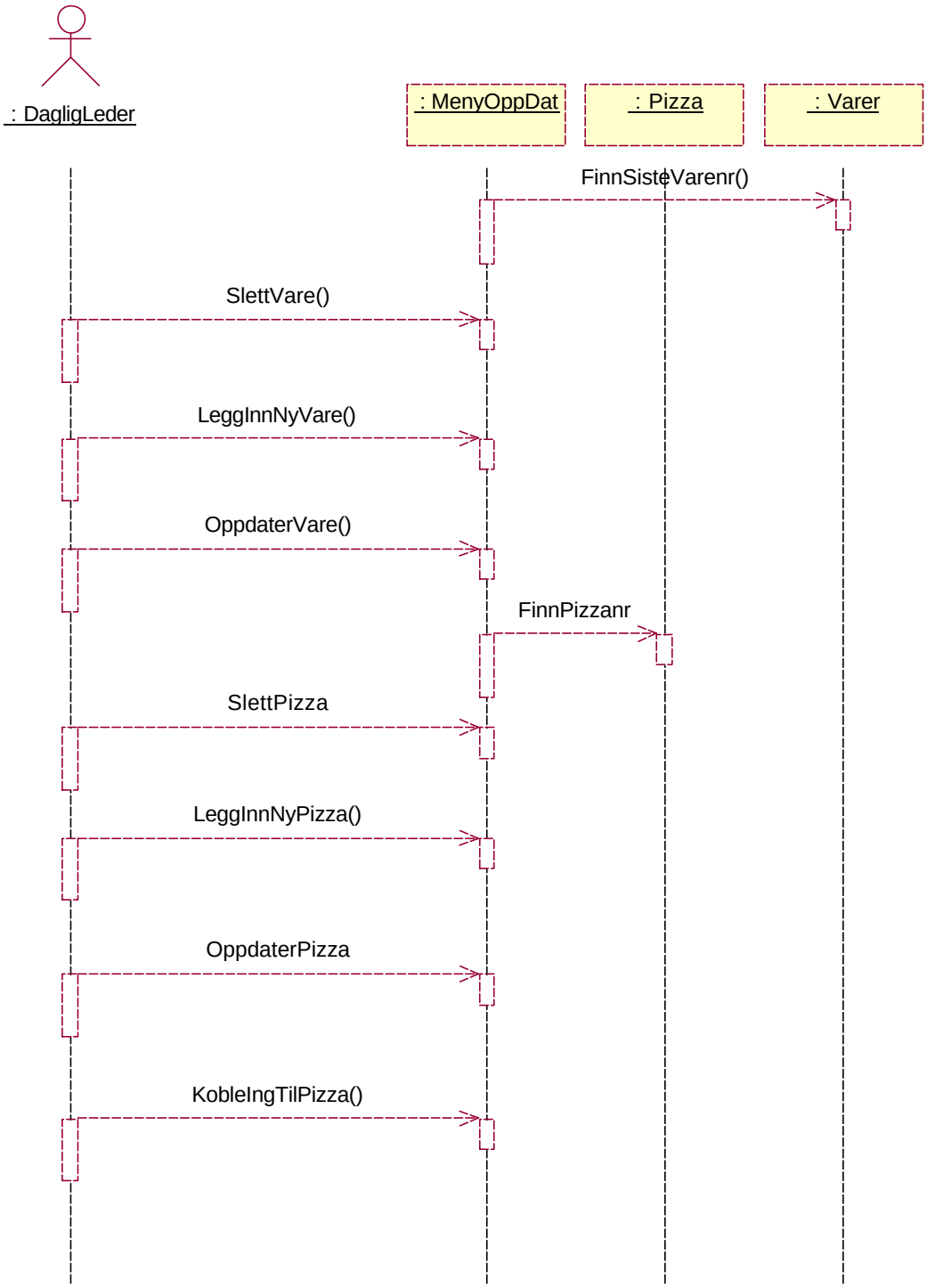
Navn	**FinnAntallBestillinger(Tid)**
Ansvar	- Skal finne hvor mange bestillinger som er blitt gjort ut i fra gitt dato
Type	System
Kryssreferanser	Fra Use Caset Rapport
Unntak	
Prebetingelser	En dato må være valgt En rapport type må være valgt
Postbetingelser	

Navn	**FinnSolgtePizzaer()**
Ansvar	- Skal finne antall solgte pizzaer - Skal finn hvilke pizzaer
Type	System
Kryssreferanser	Fra Use Caset Rapport
Unntak	
Prebetingelser	En dato må være valgt En rapport type må være valgt
Postbetingelser	

Navn	**FinnSolgteVarer()**
Ansvar	- Skal finne antall solgte varer - Skal finne hvilke pizzaer
Type	System
Kryssreferanser	Fra Use Caset Rapport
Unntak	
Prebetingelser	En dato må være valgt En rapport type må være valgt
Postbetingelser	

Navn	**FinnSolgteIngredienser()**
Ansvar	- Skal finne hvor mye ingredienser som er blitt brukt
Type	System
Kryssreferanser	Fra Use Caset Rapport
Unntak	
Prebetingelser	Dato og rapport type må være valgt
Postbetingelser	

Navn	**SkrivUt()**
Ansvar	- Skal skrive ut en rapport
Type	System
Kryssreferanser	Fra Use Caset Rapport
Unntak	
Prebetingelser	
Postbetingelser	


: DagligLeder
: MenyOppDat
: Pizza
: Varer
FinnSisteVarenr()
SlettVare()
LeggInnNyVare()
OppdaterVare()
FinnPizzanr
SlettPizza
LeggInnNyPizza()
OppdaterPizza
KobleIngTilPizza()

Figur 2.3.4.E

Navn	**FinnSisteVarenr()**
Ansvar	- Skal finne et nytt varenummer.
Type	System
Kryssreferanser	Fra Use Caset Oppdatere
Unntak	
Prebetingelser	Det må være valgt å legge inn en ny vare
Postbetingelser	Klar til å legge inn ny vare

Navn	**SlettVare()**
Ansvar	- Skal slette en vare fra databasen
Type	System
Kryssreferanser	Fra Use Caset Oppdatere
Unntak	
Prebetingelser	En vare må være valgt
Postbetingelser	

Navn	**LeggInnNyVare()**
Ansvar	- Skal finne vare data - Skal legge varen inn i databasen
Type	System
Kryssreferanser	Fra Use Caset Oppdatere
Unntak	Hvis ikke et varenummer er tildelt
Prebetingelser	Vare data må være skrevet
Postbetingelser	

Navn	**OppdaterVare()**
Ansvar	- Skal finne hvilken vare det er snakk om - Skal finne data som skal legges inn
Type	System
Kryssreferanser	Fra Use Caset Oppdatere
Unntak	Hvis ikke varenummer er kjent
Prebetingelser	Varedata må være gitt
Postbetingelser	

Navn	**FinnPizzanr()**

Ansvar	- Skal finne et pizza nummer
Type	System
Kryssreferanser	Fra Use Caset Oppdatere
Unntak	
Prebetingelser	
Postbetingelser	Ny pizza kan legges inn

Navn	**SlettPizza()**
Ansvar	- En pizza skal slettes - Sjekke om et pizza nummer er registrert
Type	System
Kryssreferanser	Fra Use Caset Oppdatere
Unntak	Hvis ikke pizza nummer er registrert
Prebetingelser	
Postbetingelser	

Navn	**LeggInnNyPizza()**
Ansvar	- Skal legge inn en ny pizza - Tildele et pizza nummer
Type	System
Kryssreferanser	Fra Use Caset Oppdatere
Unntak	Hvis ikke et pizza nummer er blitt tildelt
Prebetingelser	Data om pizza må være skrevet
Postbetingelser	

Navn	**OppdaterPizza()**
Ansvar	- Skal oppdatere data til en pizza - Skal finne pizza nummer
Type	System
Kryssreferanser	Fra Use Caset Oppdatere
Unntak	Om ikke pizza nummer er kjent
Prebetingelser	En pizza må være valgt
Postbetingelser	

Navn	**KobleIngTilPizza()**
Ansvar	- Skal koble ingredienser opp mot forskjellige pizzaer
Type	System
Kryssreferanser	Fra Use Caset Oppdatere
Unntak	Hvis ikke pizza nummer er kjent
Prebetingelser	En pizza må være valgt En ingrediens må være valgt
Postbetingelser	

2.4 Begrensninger

2.4.1 Software Design Begrensninger

2.4.1.1 Software standarder og språk

Programvare
Programvaren som skal utvikles er ment til bruk på en stasjonær PC, og skal kommunisere med CAPI protokollen til ISDN.

Navngiving
Navngiving av klasser, funksjoner og variabler er fast gjennom hele dokumentet. Klasser, Funksjoner og Variable har stor bokstav først og for hvert nytt ord i navnet. Det brukes ikke mellomrom mellom ordene i navnet.

Dokumenter
Prosjektgruppen har laget en dokumentmal som brukes gjennomgående for alle dokumenter i prosjektet.

2.4.1.2 Software grensesnitt

Mellom moduler/klasser utveksles informasjon ved parameter overføring og funksjonskall.

2.4.1.3 Software pakker/verktøy

Under kravspesifikasjonsfasen blir RationalRose brukt for å utvikle Use casene.
For tekstdokumenter som dokumentasjon og rapporter brukes Microsoft Word.

2.4.1.4 Software kommunikasjonsstandarder og grensesnitt

Mellom applikasjonen og ISDN kortet skal vi benytte CAPI som grensesnitt.

Hva er CAPI?

CAPI står for COMMON-ISDN-API og er et programmeringsgrensesnitt for systemer som benytter seg av ISDN for overføring av data. De aller fleste drivere til ISDN-kort som er tilgjengelige i dag har støtte for CAPI. I Windows benyttes en DLL-fil (Dynamic Link Library) som leveres med ISDN-kortet. En DLL-fil er en kompilert modul som brukes sammen med en exe-fil for å tilby funksjoner og lignende til en applikasjon. Windows applikasjoner kan kjøre kall mot funksjonene i denne CAPI-DLL-fila og overføre blant annet data, fakser og telefonsamtaler over ISDN-linjer.

Bakgrunnen til CAPI var at tre tyske produsenter av ISDN-kort gikk sammen for å utvikle et felles produktuavhengig programmeringsgrensesnitt for PC baserte ISDN løsninger. Dette var i 1989. Etter hvert bidro flere produsenter og programmerere til denne standarden og en gruppe kalt ”the CAPI association” ble grunnlagt. Den versjonen av CAPI som brukes mest i dag er CAPI V2.

CAPI og oppgaven.

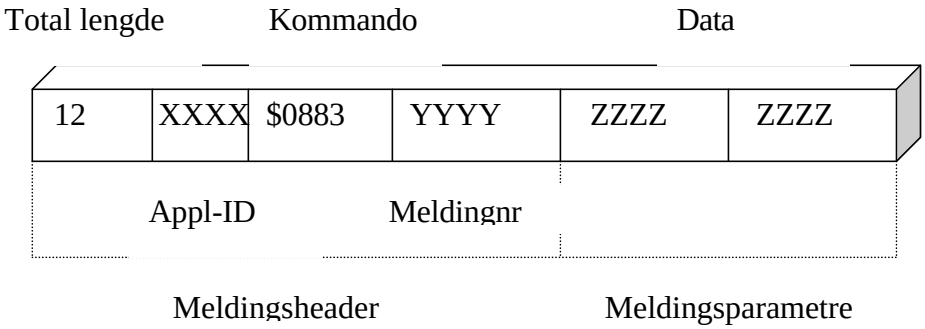
Applikasjonen skal lese inn telefonnummeret til kunden automatisk. Siden Kikkis har ISDN-linje til sine lokaler, er bruk av et ISDN-kort og CAPI den beste måten å få lest inn nummeret.

Det første applikasjonen må gjøre er å registrere seg hos ISDN-kortet. ISDN-kortet sender en forbindelse-ID som applikasjonen fra nå av må bruke. For å utføre dette kalles CAPI-kommandoen: CAPI_REGISTER. Når en forbindelse er registrert kan man benytte seg av kommandoene: CAPI_GET_MESSAGE eller CAPI_PUT_MESSAGE for å motta eller sende meldinger til kortet. Alle meldinger til og fra kortet/applikasjonen må svares på. En melding fra applikasjonen til kortet kalles en forespørsel og kortets svar kalles en bekreftelse. En melding fra kortet kalles en indikasjon og applikasjonens svar en respons. Når applikasjonen ikke lenger har bruk for data fra kortet kalles kommandoen CAPI_RELEASE for å bryte forbindelsen.

I et operativsystem som Windows 98, som denne applikasjonen skal kjøres på, bruker man kommandoen CAPI_WAIT_FOR_SIGNAL i en separat tråd for å ”lytte” til kortet. Denne tråden til kortet opprettholdes til kortet sender en indikasjon.

CAPI-meldinger er områder i systemets minne som blir "okkupert". Når Applikasjonen skal sende en melding til kortet klargjøres et minneområde og det sendes en peker til dette området. Når applikasjonen får responsen tilbake kan området frigjøres. Det skjer på samme måte når meldinger blir sendt fra kortet.

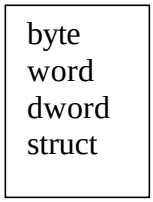
Innholdet i en CAPI-melding er en header med konstant størrelse etterfulgt av et dataområde som kan ha variabel størrelse. Headeren inneholder info om den totale størrelsen på meldinga, forbindelse-ID, CAPI-kommando og meldingsnummer. Meldingsnummeret brukes for å validere svaret på meldingen.



Figur 2.4.1.4.A

Figuren 2.4.1.4.A viser hvordan en CAPI-melding ser ut i minnet. Den totale lengden av meldinga er oppgitt i byte. Figuren bruker en melding som kalles av kommandoen CAPI_INFO_RESP. Meldingen er et svar på en indikasjon fra kortet. Applikasjons-ID og meldingsnummer kommer fra kortet. Kommando $08 sier at det er en type INFO melding, $83 forteller at meldingen er en respons.

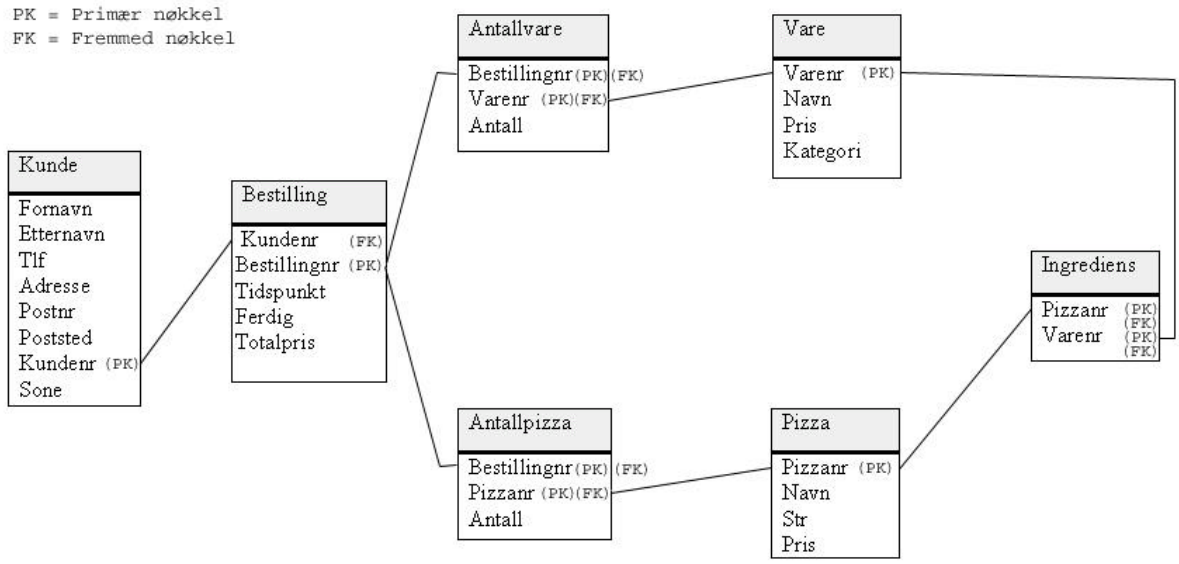
CAPI opererer med fire forskjellige datatyper for data.



Figur 2.4.1.4.B

2.4.1.5 Database

ER - diagram over databasen



Figur 2.4.1.5

Krav til databasen:
Informasjon fra postene skal kunne komme fram i løpet av max 1 sekund.
Det skal være en SQL basert database.
Applikasjonen skal utføre spørringene mot databasen.
Databasen skal være open source på grunn av lisenser.

2.4.1.6 Operativsystem

Systemet er utviklet for å virke på Windows 98. Dette fordi Windows 98 fungerer bra sammen med Interbase databaser. Hvis ansatte på Kikki's har noen erfaringer med et operativsystem så er det høyst sannsynlig Windows.

2.4.1.7 Toleranser, marginer og muligheter/tilfeller

Det er ikke satt noen begrensninger da dette systemet skal gå på en enkel stasjonær PC. Databasen er satt til å tåle å inneholde minst 20000 kunder, men dette tallet er i realiteten mye høyere.

2.4.2 Hardware design begrensninger

2.4.2.1 Hardware krav og omgivelser

ISDN kortet må støtte CAPI programmerings brukergrensesnittet

2.4.2.2 Hardware grensesnitt

ISDN linjen må være oppe og ISDN kortet koblet til.

2.4.3 Bruker Design Begrensninger

Brukeren er en person med alt fra ingen til mye datakunnskap. Programmet skal være lett å forstå med minst mulig knapper og god oversikt.

2.5 Aspekter Omkring Livssyklus

2.5.1 Dokumentasjon

Det er lagt inn en Windows help funksjon i programmet som forklarer gangen i hver enkelt funksjon som operatøren involveres i .
I hjelpen er det tatt for seg hvert eneste vindu og hver eneste knapp, og forklart hva disse gjør.
Det vil også bli dokumentert i rapporten, men ikke like detaljert som i hjelp funksjonen.

2.5.2 Modul- og integrasjonstesting

De forskjellige modulene ble testet etterhvert som nye funksjoner ble implementert.
Spesielt har vi lagt vekt på testing av systemkritiske funksjoner som kommunikasjon med databasen, og kommunikasjon med ISDN-kortet via CAPI.
Det ble også foretatt tester etter at programmet var ferdig.

Krav til hardware: Kikki's Pizza har datamaskiner de ønsker å bruke. Disse maskinene er

mer enn raske nok til å kjøre programmet. Det som mangle på datamaskinene er ISDN kort. Dette må kjøpes inn og installeres for at programmet skal fungere riktig. Det er meningen at det skal benyttes touch-scrren til programmet, og programmet er derfor bygget opp rundt dette (store knapper etc). Kikki's vil avvente situasjonen før de bestemmer seg om de vil gå til innkjøp av touch-screen, siden dette ikke er en ubetydelig investering i en forholdsvis liten bedrift. Men det er tatt høyde for en eventuell senere installering av dette og skal ikke by på problemer.

2.5.3 Krav til support, service og vedlikehold

Vedlikehold av systemet vil bli gjort av daglig leder.
Support på hardware blir gjort av leverandørene til disse.

2.5.4 Krav til utvidelser

Systemet er bygd opp slik at det uten større problemer kan legges til flere moduler.
Systemet skal i fremtiden kunne støtte bestilling over internet, Wap og SMS.

2.6 Aspekter omkring installasjon

2.6.1 Hardware installasjon

Systemet skal installeres på kjøkkenet hos Kikki's. Terminalen skal sett opp etter Kikki's ønsker. En ekstra skjerm skal settes opp slik at kokken kan ha oversikt over alle bestillingene fra der det lages pizza.

2.6.2 Opplæring

Det er lagt med en hjelp fil som viser gangen i programmet.
Videre opplæring vil skje ved hjelp av gruppemedlemmene.

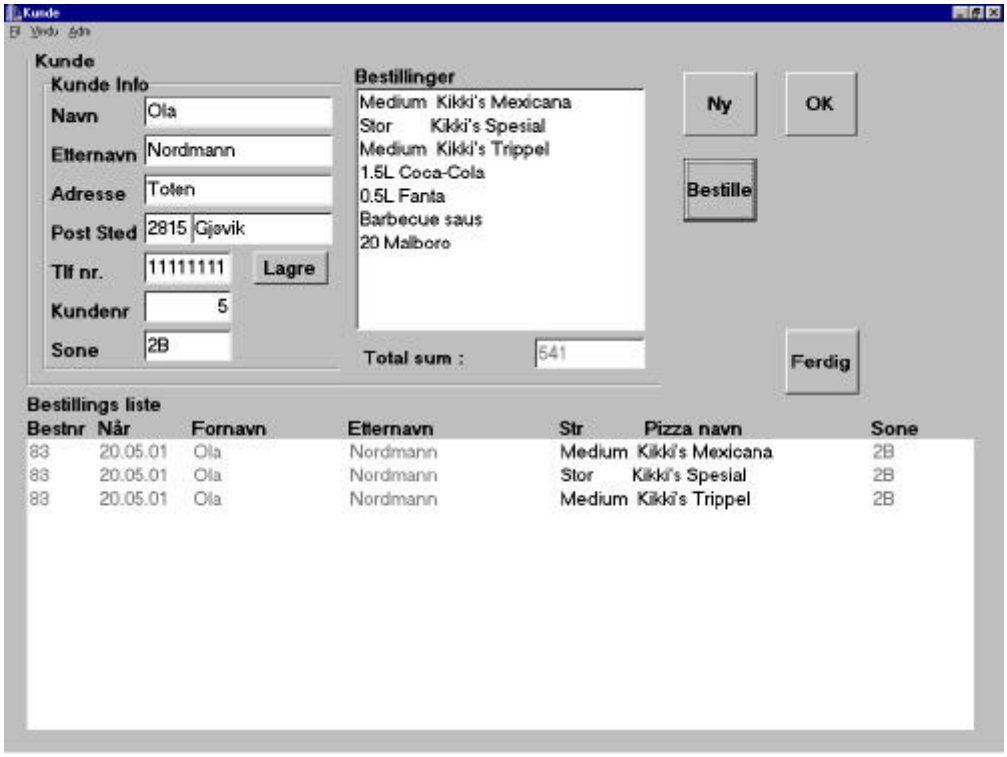
3 Design

Designdokumentet ligger som **vedlegg A; Designdokument.**
Designdokumentet var en del av prosjektet vi var usikre på. Ingen av oss hadde noen utviklingserfaring til å sette så konkrete retningslinjer for utviklingen på forhånd. Da vi laget designdokumentet, var vi allerede så å si ferdig med utviklingen. Vi følte at vi trengte å skissere løsningen før vi skrev designdokumentet.

Vi brukte mange tegninger på ark før vi fant en løsning vi følte vi kunne gå for. Dette var et fint medium for å dele ideer, og samtidig få et mer helhetlig syn selv.

Vi valgte et design der det skulle være mins mulig knapper. De knappene som er, skal være store slik at de er lette å treffe ved bruk av en touch screen. Vi valgte å få inn mest mulig informasjon på de enkelte sidene slik at brukeren kan forholde seg til få sider. Sidene skal ha en god oversikt med stor skrift.

Her viser vi et eksempel på skjermbilde av hovedsiden der kundeinformasjon bestillingen og ventelisten vises.



Figur 3

4 Implementering

4.1 Verktøyvalg

På et tidlig tidspunkt i hovedprosjektet fant vi ut at *C++* som programmeringsspråk var best egnet for denne oppgaven. Vi valgte å bruke Borland C++ Builder 5 som utviklingsverktøy. Detter fordi vi hadde erfaringer med Borlands utviklings verktøyer fra før.

Vi valgte å bruke Interbase 6 til databaseutviklingen fordi database systemet er open source. Det gir ingen lissensutgifter for oppdragsgiver.

4.2 Implementeringsteknikker

4.2.1 Standarder

Vi fulgte en mal over variabel navn og windows komponenter som vi fikk utdelt i faget Grafiske brukergrensesnitt. Vi har gjort visse avvik spesielt på definisjon av lokale variable. Ref **vedlegg B; Prefiks-forkortelser.**

Eksempel på koding:

```
Void LeggInnNyVare(String Kategori, String Navn, int Pris) {
Int Nr;
FrmDatabase->qryVareNr->Close();
FrmDatabase->qryVareNr->Open();
Nr=frmMenyOppDat->dbEdtVaeNr->Text.ToInt();
Nr++;
FrmDatabase->qryLeggInnNyVare->Insert();
FrmDatabase->qryLeggInnNyVare->FieldByName(”Varenr”)->AsInteger=Nr;
FrmDatabase->qryLeggInnNyVare-> FieldByName(”Navn”)->AsString=Navn;
FrmDatabase->qryLeggInnNyVare-> FieldByName(”Pris”)->AsInteger=Pris;
FrmDatabase->qryLeggInnNyVare-> FieldByName(”Kategori”)->AsString=Kategori;
FrmDatabase->qryLeggInnNyVare->Post();
FrmDatabase->qryLeggInnNyVare->ApplyUpdates();
FrmDatabase->qryLeggInnNyVare->Close()
FrmDatabase->qryLeggInnNyVare->Open()
Oppdater();
}
```

frm…..=Form
qry….=Qurry (Spørring)

4.2.2 Koding

Under selve programmeringen var det en evolusjonær fremgangsmåte. Dette kom av at vi har for lite erfaring med systemutvikling til å kunne forutse hele utviklingsforløpet. Vi måtte dermed ta for oss læringen under selve utviklingsprosessen. Dette førte til at det ble mye prøving og uttesting underveis.
For å kunne gjennomføre kodingen måtte vi tilegne oss ny kunnskap. På jakt etter dokumentasjon som kunne føre oss et steg videre i riktig retning ble det gjort oppslag i bøker, ref **punkt 9; Litteraturliste**, samt søk på Internett.

Selve programmeringen ble gjort ved at vi utviklet en funksjon, for så å sjekke om den fungerte på en tiltenkt måte, for så å utvikle en ny del. Dette vil si at vi utførte mye av funksjonstestingen underveis.

5 Testing og Kvalitetssikring

5.1 Testing

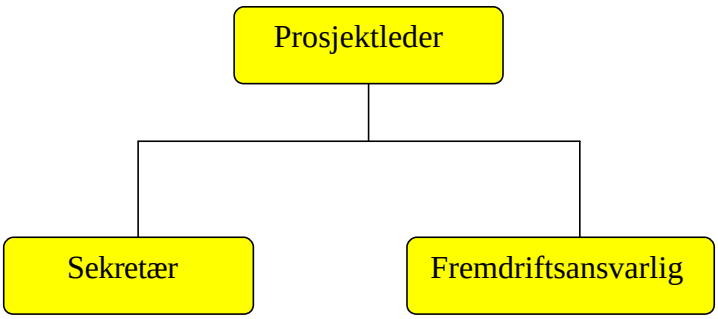
5.1.1 Modultesting

Vi valgte å kjøre feiltesting samtidig som programmet ble utviklet. Dette var fordi vi lagde et skall i begynnelsen, som seinere ble utviklet videre til en fullstendig versjon.

Hver programkode og funksjoner ble testet etter hvert som de ble laget. Vi satte opp hjelpe editor felter på skjermbildet som viste svaret på det funksjonene kom fram til. Disse feltene ble senere fjernet da funksjonene var ferdig testet og kom med riktig output.

5.2 Kvalitetssikring

Siden vi er tre på gruppen så vi det nødvendig å fordele ansvaret, og klargjøre hva de forskjellige postene innebar av ansvar innad i gruppen. Det ble delt inn i ansvarsområder på følgende måte: prosjektleder, sekretær, fremdrifts ansvarlig.



Figur 5.2

Vi så for oss at en av suksess faktorene til en god kvalitets sikring var å ta en risikoanalyse av prosjektet. Ref **vedlegg C; Risikoanalyse.**
Det ble laget normer og regler på hvordan dokumentasjonen skal se ut. Ut ifra dette ble det laget en dokument mal i Word som alle skulle bruke.

6 Diskusjon av resultater

6.1 Systemets virkemåte

Når systemet starter opp sjekkes det om programmet ble riktig avsluttet sist gang. Ble det ikke det, kommer det en melding om dette og det er mulighet til å hente inn siste backup av databasen hvis det skulle være ønskelig.

Applikasjonen som utvikles for Kikkis skal først sette ISDN-kortet i lyttemodus. Hvis det kommer inn en samtale skal telefonnummeret til innringeren sendes til applikasjonen, samtalen skal ikke besvares av kortet. Etter at kortet har sendt telefonnummeret skal det settes tilbake til lyttemodus.

Brukeren ar da klar til å ta imot en bestilling. Brukeren skal bekrefte at informasjonen om kunden stemmer slik at kvitteringen blir riktig skrevet. De bestillingene som blir gjort bli lagt direkte in i databasen ettersom de bli gjort. Velger kunden å angre på en bestilt vare, så må denne slettes fra databasen.

En kvittering blir skrevet ut med kunde informasjon om hva kunden har bestilt denne gangen.

Administrator eller daglig leder har mulighet til å oppdatere, slette og legge til nye varer og pizzaer. Dette gjøres på en passord beskyttet side(Form). Her er det også mulighet til å få skrevet ut forskjellige rapporter på hva som er blitt bestilt i en valgt periode.

Systemet har en backup funksjon som iverksettes hvert 5 minutt. Den kan og skal tas backup manuelt hver kveld. Det er mulighet til å lagre databasen manuelt på ett annet media.

6.2 Ønskelige krav

Oppgaven vi har fått tildelt er en frittstående oppgave uten definerte krav fra oppdragsgiveren. Gruppa kom med forslag til oppdragsgiver over hvilke krav enn skulle sette til systemet. Oppdragsgiver har da sagt ja/nei.

Når en kunde ringer inn, skal telefonnummeret letes opp i databasen.
Adressene skal deles opp i soner over Gjøvik området.
Programmet skal kunne skrive ut forskjellige rapporter på hva som er blitt solgt i et valgt tidsområde.
Det skal skrives ut kvitteringer til hver bestilling.

6.3 Avvik fra ønskelige krav

Adressene blir ikke automatisk delt inn i soner. Sonen må manuelt skrives inn på lik måte som gjøres med annen kunde informasjon. For eksempel da en kunde ringer inn for første gang og telefonnummeret ikke ligger lagret i databasen.

Per dags dato 22. Mai virker ikke CAPI koblingen mot ISDN kortet som den skal. Dette er på grunn av mangel på testmuligheter.

6.4 Videre arbeid

I framtiden kan systemet utvides til å kunne ta imot bestillinger over internet og Wap SMS.

6.5 Alternative muligheter, valg under veis

Fra før av har Kikki's pizza installert AnyWeb telefoner fra Samsung. Disse telefonene har innebygd nettleser og har mulighet for å lese E-mail. Telefonen har også nettverks tilkoblingskontakter. Vi vurderte å koble PC med applikasjonen mot denne telefonen. Det var ikke tilgjengelig noen drivere eller annen grensesnitt mot denne telefonen. Vi valgte derfor å gå bort fra denne telefonen, og heller hente telefonnummeret gjennom et ISDN kort. Her ble det brukt CAPI programmerings grensesnittet mot applikasjonen.

7 Konklusjon

7.1 Evaluering av gruppas arbeid

Hovedmålet for denne oppgaven har hvert å utvikle et produkt for Kikki's pizza i henhold til de krav som er utarbeidet i kravspesifikasjonen. I denne prosessen har både gruppa som helhet og enkeltpersonene blitt stilt ovenfor mange ulike oppgaver og gruppemedlemmene har hatt ulike roller gjenom prosjektet.

Arbeidet med prosjektet har gitt oss god erfaring med å utvikle et produkt for en oppdragsgiver. Vi har erfart at å gjøre systemutvikling i praksis har stor forskjell fra teorien. Det var ikke lett å planlegge i store detaljer, til dette hadde gruppen alt for liten erfaring.

7.2 Gruppemedlemmers konklusjon

7.2.1 Fredrik Lysakerud

Det har vært lærerikt å jobbe med et prosjekt i en gruppe. Jeg har sett at forskjellige personer har helt forskjellig syn på en og samme ting. Kommunikasjonen i gruppa har hvert bra og tydelig. Det er blitt gitt klare arbeidsoppgaver slik at alle viste hva de skulle gjøre til enhver tid.
Har lært mye nytt om programmering og det å jobbe sammen i gruppe.

7.2.2 Morten Kristoffersen

Det har vært interessant å jobbe med et reelt prosjekt. Jeg har fått en god del ny kunnskap i programmering og databaser. Det var også interessant å få innsikt i mulighetene man har til å overføre data over ISDN med hjelp av CAPI-grensesnittet.

Gruppa har etter min mening fungert bra. Alle medlemmer har gjort jobben sin. Det har vært perioder da progresjonen har gått veldig sakte og vi møtte "veggen", men de fleste problemene fant vi en løsning på eller vi fant andre muligheter.

Ting som kunne vært bedre er etter min mening kommunikasjonen med oppdragsgiveren. Det har vært vanskelig å få tak i hva slag forventninger og krav oppdragsgiveren har hatt til applikasjonen.

7.2.3 Ole Jørgen Reierstad

Hovedprosjektet har for min del vært spennende og lærerikt. Det var gøy å jobbe med en reel og omfattende oppgave å få en "smak" av arbeidslivet. Synes gruppen har fungert bra og kommunikasjonen innad i gruppen har vært topp.
Det som ikke har vært like bra er kommunikasjon med oppdragsgiver.
Det har til tider vært vanskelig å få kontakt med de. De hadde heller ikke så mange krav til prosjektet, noe som førte til at vi stod forholdsvis fritt i forhold til kravspesifikasjon og lignende.
Utover dette føler jeg at prosjektet alt i alt har gått bra.

8 Litteraturliste

Henning Johansen. Retningslinjer for Hovedprosjekter ved AT, August 2000

Robert Lafore, Object-oriented Programming in C++, Secomd edition 1994

Kent Reisdorph, Teach Yourself Borland C++ Builder 4 in 24 Hours, 1999

John Miano, Tom Cabanski, Harold Howe, Borland C++ Builder How-To, The Definitive C++ Builder Problem-Solver, 1997

Marco Cantu, Mastering Delphi 3. Second Edition 1997

Joakim Dalby, Databasen Håndbogen fra design til brug, 2. Utgave 1994

Kent Reisdorph, Ken Henderson, Teach Yourself Borland C++ Builder in 21 Days, First Edition 1997

Craig Larman, Applying UML And Patterns, An Introduction to Object-Oriented Analysis and design, 1998

Jim Mischel, Jeff Duntemann, Borland C++ Builder Programming Explorer The Hands-On Guide to Mastering Point-and-Click C++ Development, 1997

Kent Reisdorph, Borland C++ Builder 4 Unleashed The Comperehensive Solution, First Edition 1999

Hjemmesiden til Peter Zwosta, som har laget Capi komponenter til C++ Builder og Delphi.
http://home.t-online.de/home/Peter.Zwosta/capiidx.htm

Hjemmesiden til Capi Asosiation
http://www.capi.org/

Hjemmesiden til Borland og Interbase
http://www.borland.com/

9 Vedlegg

Innholdsfortegnelse Vedlegg