Fakulteten för ekonomi, kommunikation och IT
Datavetenskap

Daniel Mester Pirttijärvi

Hampus Skystedt

Beställningsklient i surfplattor för restauranger

Restaurant order client for tablets

Examensarbete 15 hp
Dataingenjörsprogrammet

Datum:	2011-06-09
Handledare:	Donald F. Ross
Examinator:	Stefan Lindskog
Löpnummer:	C2011:07

Karlstads universitet 651 88 Karlstad
Tfn 054-700 10 00 Fax 054-700 14 60
Information@kau.se www.kau.se

Beställningsklient i surfplattor för restauranger

Daniel Mester Pirttijärvi och Hampus Skystedt

Denna uppsats är skriven som en del av det arbete som krävs för att erhålla en kandidatexamen i datavetenskap. Allt material i denna uppsats, vilket inte är mitt eget, har blivit tydligt identifierat och inget material är inkluderat som tidigare använts för erhållande av annan examen.

Daniel Mester Pirttijärvi

Hampus Skystedt

Godkänd 2011-06-09

Handledare: Donald F. Ross

Examinator: Stefan Lindskog

Sammanfattning

De flesta har vi nog varit på en fullservicerestaurang, en sådan som har serveringspersonal som kommer med menyer när man anländer, som tar beställningen vid borden, som serverar mat och dryck, som kommer med notan vid betalning. I sådana restauranger måste serveringspersonalen traditionellt sett springa många vändor mellan gäster, kök, bar och kassaterminal.

Under projektet har en lösning utvecklats som gör det möjligt för serveringspersonalen att ta gästernas beställningar redan vid borden och därmed spara ett par vändor till kassan. Lösningen består av maskinvara i form av en surfplatta och ett program som fungerar som kassaklient. Programmet förmedlar automatiskt beställningarna till kassasystemet via en trådlös nätverksförbindelse. Användargränssnittet har anpassats särskilt för att användas med pekskärm.

Denna uppsats beskriver på vilka grunder valet av maskinvara gjordes och hur utvecklingen av programmet gick till.

Abstract

Most of us have been in a full service restaurant, a restaurant which has waiters who hand out menus, take orders at the tables, serve food and beverages and bring the bill. In such restaurants the waiters usually have to make several trips between guests, kitchen, bar and point of sale station.

In this dissertation a solution has been developed that enables the servers to take orders at the tables and thus eliminating a couple of trips to the point of sale station. The solution consists of hardware, in this case a tablet pc, and an application. The application automatically transfers the orders to the point of sale system. The user interface has been especially customised for touch screen use.

This dissertation describes the grounds on which the hardware decision was based and how the application was developed.

Tack till

Vi vill tacka Daniel Höglund och Per Lindhé, våra uppdragsgivare på AffärsIT i Skandianavien AB, som har varit behjälpliga under hela projektet. Vi vill även tacka vår handledare Donald F. Ross som har granskat uppsatsen och gett råd under projektets gång.

Innehåll

Figurer

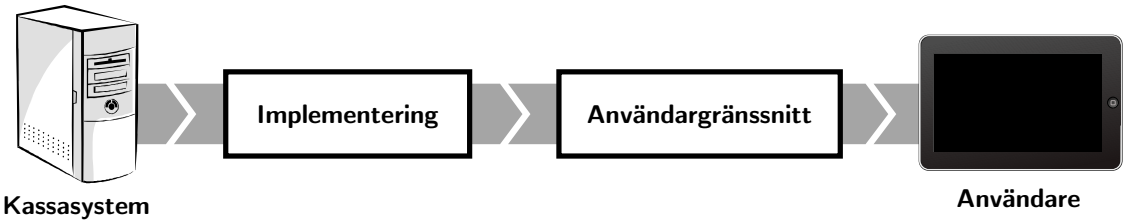
Kapitel 1

Inledning

På vissa restauranger har man serveringspersonal som kommer med menyer när man anländer, tar beställningar vid borden och serverar maten. I sådana restauranger skriver personalen vanligtvis ned beställningarna i ett block. Om restaurangen har ett datoriserat kassasystem förs beställningen därefter in vid närmaste kassaterminal. I vissa fall förmedlar kassasystemet automatiskt beställningen till kök och bar.

1.1 Syfte

Med lösningen som utvecklats under projektet kan några av dessa vändor elimineras genom att serveringspersonalen, med hjälp av en surfplatta preparerad med programmet som utvecklats under projektet, kan ta en beställning direkt vid bordet. Programmet ser därefter till att beställningen automatiskt förmedlas till köket och baren. Serveringspersonalen behöver därmed inte längre ta beställningen på ett block, för att därefter uppsöka en kassaterminal och där digitalisera beställningen. Produkten kan därmed sägas vara ett modernt alternativ till papper och penna.



Figur 1.1. Projektöversikt.

1.2 Översikt

Det första som gjordes under projektet var att fastställa en specifikation i samråd med uppdragsgivaren. Lösningen som specificerades har huvudsakligen tre beståndsdelar: det befintliga kassasystemet Isupos, en surfplatta samt ett program på surfplattan som fungerar som kassaklient (se Figur 1.1).

KASSASYSTEMET Isupos har utvecklats av uppdragsgivaren AffärsIT i Skandinavien AB. Kassasystemet är utvecklat i .NET-miljö och är anpassat för att användas med pekskärm. I projektet har endast kassasystemets API använts (Lindhé, 2011).

SURFPLATTOR. Under projektet gjordes ett urval av plattor som kan vara lämpliga att använda tillsammans med kassaterminalen. Flera kriterier har legat till grund för urvalet; plattan måste vara hållbar då den kommer att utsättas för hård behandling och plattan måste vara i ett behändigt format, då den ska bäras omkring av serveringspersonalen. Även vilka utvecklingsplattformar som finns tillgängliga för plattan har beaktats.

KASSAKLIENTEN, programmet som kommer att köras på surfplattorna, har utgjort huvuddelen av projektet. Klienten, som utgörs av implementeringen och användargränssnittet i Figur 1.1, används för att ta gästernas beställningar vid borden – betalningen sker fortfarande i kassan. Programmet ansvarar för att beställningarna automatiskt förmedlas till kök och bar via en trådlös nätverksanslutning till kassasystemet.

Programmet ska dock kunna användas även då nätverksanslutning tillfälligt saknas, till exempel då personalen befinner sig på en uteservering utanför det

trådlösa nätverkets räckvidd. Därför lagras alla ändringar i en meddelandekö som programmet successivt ser till att tömma och skicka till kassasystemet då möjlighet finns. Eftersom beställningarna kan ändras även från andra surfplattor och kassaterminaler finns det även en synkroniseringsenhet i programmet som hämtar uppdaterad information om aktuella beställningar från kassasystemet. För att undvika att två personer ur personalen samtidigt ändrar på en och samma beställning blir beställningen låst till den person som först öppnar beställningen.

Kassaklienten har skrivits i C# och .NET Framework 4.0.

1.3 Disposition

Uppsatsen är i stort sett strukturerad efter det arbetsflöde som har följts under projektet.

BAKGRUND. Uppsatsen börjar med att i Kapitel 2 presentera bakgrundsinformation till projektet. I detta kapitel presenteras serveringspersonalens arbete som det vanligtvis utförs idag samt befintliga lösningar som på ett eller annat sätt liknar den lösning som utvecklats under projektet.

VAL AV SURFPLATTA. Det första arbetsmomentet är att välja vilken typ av surfplatta som programmet ska köras på. Valet av surfplatta kommer att behandlas i Kapitel 3.

ANVÄNDARGRÄNSSNITT. I Kapitel 4 beskrivs utformningen av det användargränssnitt som används i surfplattorna och de tankar som lett fram till dagens prototyp.

IMPLEMENTERINGEN av kassaklienten består dels av användargränssnittet, dels av en modell som representerar de entiteter som existerar i kassasystemet, dels synkroniseringen som håller kassasystemet uppdaterat om händelser i kassaklienten och vice versa (till exempel till följd av ändringar gjorda på andra plattor). I Kapitel 5 presenteras var och en av dessa beståndsdelar.

UTVÄRDERING. I Kapitel 6 utvärderas den prototyp som framställts. I kapitlet presenteras problem som uppstått under projektet, sådant som skulle ha kunnat göras bättre samt sådant som inte hunnits med under projektet.

SLUTSATS. Uppsatsen avslutas med Kapitel 7, där arbetet med projektet sammanfattas och utvärderas. I detta kapitel jämförs även de planer som inledningsvis ställdes upp med den slutliga prototypen på en icke-teknisk nivå. Dessutom presenteras vad som framledes kommer att hända med prototypen.

Kapitel 2

Bakgrund

Detta kapitel kommer att presentera bakgrunden till projektet. Först presenteras uppbyggnaden av ett kassasystem och hur man använder det i restaurangmiljö. Därefter undersöks befintliga lösningar som liknar den lösning som utvecklats. Slutligen kommer uppdragsgivaren AffärsIT i Skandinavien AB att presenteras kort.

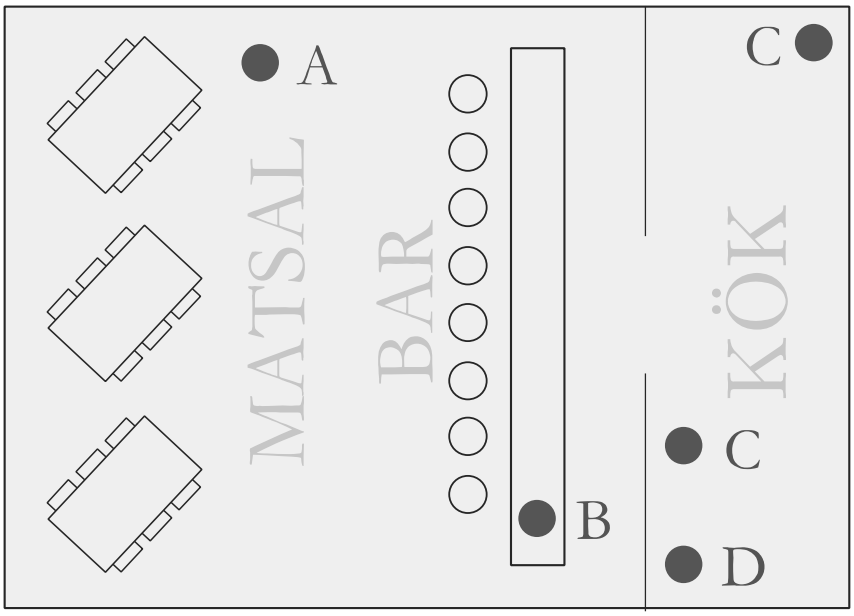
2.1 Kassasystem

I dagens samhälle finns flera typer av restauranger med varierande grad av service. Å ena sidan finns snabbmatskedjorna som låter gästerna själva lägga sina beställningar direkt när de kommer in i restaurangen och låter dem också hämta maten själva när den är klar. Å andra sidan finns restaurangerna där gästerna sätter sig vid ett bord och blir uppassad av serveringspersonal, som både tar beställningen och serverar maten. Fortsättningsvis kommer de senare kallas för fullservicerestauranger och det är dessa som uppsatsen huvudsakligen fokuserar på.

Detta avsnitt kommer att ta upp två typer av kassasystem: stationära kassasystem och kassasystem utökade med bärbara kassaterminaler. Huvudfokus kommer att ligga på datoriserade kassasystem, då endast dessa är relevanta för ändamålet.

2.1.1 Stationära kassasystem

Idag har datoriserade stationära kassasystem blivit ett självklart val för många restauranger, såväl för snabbmatskedjor som för fullservicerestauranger (Malison, 2003). Datoriserade kassasystem avsedda för restauranger består vanligen av ett flertal sammanlänkade komponenter: i matsalen och i baren har man vanligtvis kassaterminaler, där personalen kan slå in beställningar, i köket finns skrivare eller skärmar som ger personalen en överblick över aktuella beställningar, samt en server för komponenterna (se Figur 2.1) (Dahmer & Kahl, 2009).



Figur 2.1. Exempel på komponenter i ett kassasystem i restaurangmiljö.

A. En handterminal som en servitör bär omkring. B. En stationär kassaterminal. C. Skrivare som skriver ut en bong eller en monitor som visar aktuella beställningar. D. Server som hanterar alla beställningar.

Det datoriserade kassasystemet har flera fördelar: att komponenterna kommunicerar med varandra innebär att köket automatiskt kan underrättas när en beställning läggs av serveringspersonalen, att beställningarna lagras centralt på en server innebär att kassasystemet kan hålla reda på beställningens status. Det sistnämnda kan i snabbmatskedjor kombineras med digitala nummerlappar, som i folkmun bland annat kallas *puckar* (Språkrådet, 2010), som piper eller på annat sätt automatiskt meddelar gästerna när deras mat är klar att hämtas (Dahmer & Kahl, 2009).

Figur 2.2. Exempel på en bong.

Såhär beskriver (Dahmer & Kahl, 2009) personalens interaktion med kassasystemet:

1. En person ur serveringspersonalen går fram till gästen. Under tiden som gästen gör sin beställning antecknar serveringspersonalen beställningen med papper och penna.
2. Servitören går till närmaste kassaterminal, där notan förs in i kassasystemet. Kassasystemet håller reda på bordsnummer, när beställningen lades och beställningens innehåll. Beställningen hålls öppen under tiden gästen är kvar i restaurangen, vilket innebär att det är enkelt för personalen att lägga till och ändra i gästens beställning.
3. I restaurangens kök och bar finns skärmar eller skrivare utplacerade. Då serveringspersonalen matar in en beställning i kassasystemet, fördelar kassasystemet automatiskt de olika artiklarna på respektive avdelning i restaurangen: i köket skrivs en så kallad bong (se Figur 2.2) (Wikipedia, 2010a) ut med de maträtter som beställts och i baren skrivs en bong ut med enbart vilken dryck som ska serveras.
4. När gästen ber om notan söker servitören upp en kassaterminal och låter kassasystemet summera och skriva ut notan. Personalen har i detta skede också möjligheten att dela notan på flera personer.
5. Gästen får notan och betalar för sig. Beställningen har därmed slutförs i kassasystemet.

Ur beskrivningen ovan kan man dra slutsatsen att serveringspersonalen gör beställningen i två steg: först skrivs beställningen ned på papper, och därefter matas beställningen in i kassasystemet. Förutom dubbelarbetet och den tidsförlust det innebär finns två nackdelar med förfarandet: dels måste servitören mellan dessa moment lämna bordet och uppsöka en terminal, dels kan servitören inte ta flera beställningar åt gången, eftersom de första gästerna då skulle få vänta längre på sin mat (Malison, 2002).

En nackdel med datoriserade kassasystem är att kassasystemen blir obrukbara vid strömavbrott om batteribackup saknas.

2.1.2 Handterminaler

För att angripa problemet med dubbelarbetet beskrivet i föregående stycke kan man använda trådlösa handhållna kassaterminaler som kommunicerar med kassasystemet. De första handterminalerna kom redan för 25 år sedan och använde infraröd överföring för att kommunicera med kassasystemet. Beställningar togs genom att mata in PLU-koder[1] på handterminalens numeriska tangentbord (Manion & DeMicco, 2004).

Sedan dess har mycket hänt. Idag finns det handterminaler som kommunicerar med kassasystemet via en trådlös nätverksanslutning, har pekskärmar istället för numeriska tangentbord för inmatning av artiklar, och är speciellt anpassade för att tåla fuktiga miljöer och att tappas från viss höjd (Flytech, 2010). Vissa handterminaler imiterar de stationära kassaterminalernas knappbaserade gränssnitt, medan andra försöker imitera det traditionella sättet att ta beställningar med penna och papper, genom att låta serveringspersonalen skriva in beställningen med penna på enhetens pekskärm (Dahmer & Kahl, 2009) (Malison, 2002).

I *Replacing Pencil & Pad in Full Service Restaurants* nämns flera fördelar med de handhållna terminalerna (Malison, 2002). Först och främst behöver serveringspersonalen inte längre söka upp en stationär kassaterminal för att

[1] PLU, price lookup code, är ett slags korta artikelnummer som butiker slår in i kassan (Wikipedia, 2010c)

lägga in gästernas beställningar. Vid rusningen i en stor restaurang kan det betyda att personalen inte behöver köa med andra servitörer vid terminalerna.

Utöver effektiviseringen som enheterna utgör, tillför enheterna ett mervärde genom att mer information om rätterna kan visas i enheterna, vilket betyder att serveringspersonalen inte behöver memorera hela menyn och dessutom kan svara på specifika frågor som gästerna ställer om rätterna utan att behöva lämna bordet.

Såhär beskriver (Dahmer & Kahl, 2009) personalens interaktion med kassasystemet när handhållna terminaler används:

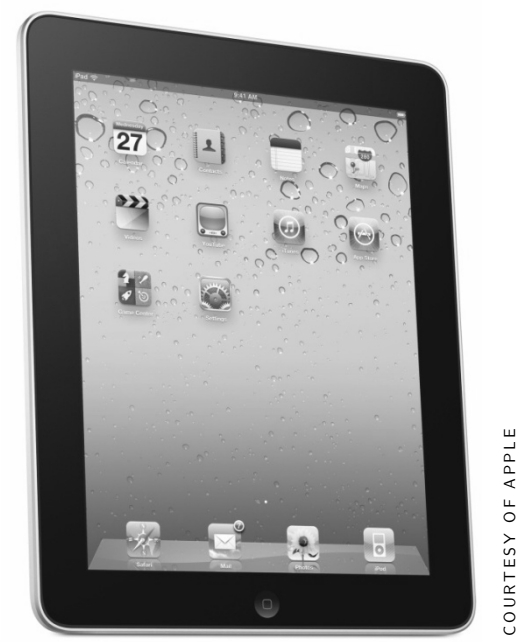
1. En person ur serveringspersonalen går fram till gästen. Servitören tar fram sin handterminal och öppnar en ny nota för gästen.
2. Under tiden som gästen gör sin beställning matar servitören in den i terminalen. Hela tiden syns vad som beställts för att servitören ska kunna vara säker på att inmatningen blir korrekt.
3. När beställningen är klar väljer servitören att skicka beställningen. Beställningen hålls dock öppen under tiden gästen är kvar i restaurangen, vilket innebär att det är enkelt för personalen att lägga till och ändra i gästens beställning direkt i handterminalen eller valfri annan terminal.
4. Kassasystemet ansvarar sedan för att hålla reda på beställningen och för att underrätta köket och baren om beställningen (som beskrivet i föregående avsnitt).
5. När gästen ber om notan summerar kassasystemet beställningen och skriver ut notan. Detta kan antingen ske i handterminalen om den har en skrivare inbyggd eller ansluten, eller vid en stationär terminal.
6. Gästen får notan och betalar för sig. Om handterminalen har en inbyggd kortläsare behöver inte personalen lämna bordet för att genomföra eventuell kortbetalning.

Serveringspersonalen kan spara några vändor till en stationär kassaterminal genom att använda sig av handterminaler, men trots detta ser man i realiteten dem väldigt sällan. Några möjliga orsaker till att handterminalerna inte fått en

sådan stor spridning är att enheterna har för dålig batteritid, är tunga jämfört med papper och penna, att systemen kan vara för omständiga att använda (det vill säga att det krävs för många tryck för att göra en beställning), och att handterminalerna är flera gånger dyrare än dess största konkurrent: ett block och en penna. (Malison, 2003)

2.2 Existerande produkter

En surfplatta är en dator som består av en enda enhet där all inmatning sker på dess pekskärm (PC Magazine, u.å.).



Figur 2.3. Exempel på en surfplatta, en Apple Ipad.

Surfplattorna som vi ser i affärerna idag är relativt nya – det var främst efter Apples lansering av Ipad (Figur 2.3) under 2010 som de blev populära (Apple, 2010). Väldigt likt surfplattor finns även läsplattor vars huvudfunktion är att visa elektroniska böcker (Wikipedia, 2010b).

2.2.1 Historik

Handhållna datorer utan tangentbord har funnits i flera årtionden. Under sent 1980-talet var det mycket uppmärksamhet kring handhållna penndatorer, men

det var först i början på 1990-talet som utvecklingen började ta fart (Wikipedia, 2011j). Det blev inte det genombrott som industrin hade hoppats på utan det dröjde längre, ända tills Microsoft återintroducerade sina Tablet PCs under 2002 som enheterna började bli förhållandevis framgångsrika (Wikipedia, 2011i).

Intresset för enheterna ökade när Apple släppte sin Iphone 2007 med sin nya multipekskärmsteknik (Gartner, Inc., 2008) (Apple, 2007) (Block, 2007). Marknaden för mobila handhållna pekskärmsenheter beräknades passera 362,7 miljoner enheter år 2010 (Gartner, Inc., 2010). Under 2011 förväntas 57,6 miljoner surfplattor säljas (Abelson, 2011).

2.2.2 Handterminaler

Handterminaler med pekskärm är små kompakta handburna enheter, där inmatning på skärmen sker genom att använda finger eller penna. Det finns huvudsakligen två olika gränssnitt på systemen som används ute i restauranger: det första är ett menybaserat knappsystem och det andra är baserat på igenkänning av skrift (Malison, 2002).

Det menybaserade knappsystemet är i princip samma gränssnitt som finns på stationära kassaapparater. Skillnaden är att det är anpassat till skärmens storlek som där i de flesta fall är avsevärt mindre.

Action Systems, Inc. har ett system som är baserat på igenkänning av skrift och började utvecklas redan 2002 (Malison, 2002). Fokuset med gränssnittet har varit att göra det så likt det traditionella upplägget med användning av penna och papper som möjligt. I detta gränssnitt är skärmen uppdelad i tre delar: den första delen visar alla föremål i beställningen, den andra delen skriver servitören i och den tredje delen visar upp olika förslag som resultat av det som skrivits in.

2.2.3 Ipadmeny för kunder

På senare tid har det rapporterats om flera restauranger som har bytt ut sina klassiska menyer mot Ipads (Carmody, 2010). The Global Mundo Tapas bar i norra Sydney, Australien har helt och hållet bytt ut sina menyer mot Ipads (Mundo Global Tapas, u.å.) som kör en skräddarsydd applikation där gästerna

kan kolla på menyn och själva beställa in den mat och dryck de vill ha (The Sydney Morning Herald, 2010). Även på Bone's restaurang i Atlanta, USA har ett liknande koncept använts (Bone's Restaurant, u.å.). Där får gästerna tillgång till restaurangens hela vinlista på 1 350 artiklar genom en applikation på en Ipad. Redan första veckorna märktes direkt en ökning av försäljningen utan annan tänkbar förklaring än att det berodde på det nya användandet av surfplattor (Sack, 2010).

Produkten kommer främst att rikta sig till serveringspersonal men det kan i framtiden även tänkas komma till användning mer för gästerna själva genom att de får surfplattor vid bordet och kan välja från dem och göra egna beställningar.

2.2.4 Övriga tillämpningar

Det är inte bara inom restaurangbranschen som man har provat att använda sig av surfplattor. I Georgia, USA har Gwinnet Hospital System köpt in surfplattor till över 4 000 anställda på bland annat Gwinnet Medical Center och tre sjukhus. Plattorna köptes in för att ersätta delar av organisationens IT-infrastruktur bestående av stationära och bärbara datorer samt övriga bärbara enheter (LANDesk, 2007).

Plattorna medför att sjukhuspersonalen har direktåtkomst till röntgenbilder och journaler. Tester kan beställas direkt via plattan och resultaten presenteras i desamma (Cooper & Lee, 2008).

Surfplattor kan också användas för att ersätta utskrivna dokument. I Årjängs kommun har man gått över till att handlingar skickas elektroniskt och är åtkomliga via kommunens intranät. För att undvika att tjänstemännen skriver ut handlingarna har man nyligen köpt in läsplattor till de elva ledamöterna i kommunstyrelsen. Plattorna har mottagits med stort intresse och inköpskostnaden beräknas ha sparats in på ett år (Magnusson, 2011).

2.3 AffärsIT i Skandinavien AB

Projektet har utförts på plats hos företaget AffärsIT i Skandinavien AB. AffärsIT har sålt kassasystem och tillbehör på nätet sedan 2004 och säljer också

två egenutvecklade kassasystem: EasyCashier för mindre företag och sedan 2010 kassasystemet Isupos (Figur 2.4) som är anpassat för större företag och restaurangbranschen.

Figur 2.4. Isupos kassasystem med kvittoskrivare och kortterminal.

Maskinvaran för Isupos består av en PC med integrerad pekskärm som är framtagen för att användas i butiksmiljö. Tillhörande programvara, ISU-kassa, är utvecklad i .NET-miljö och är specialanpassad för att användas med pekskärm. Programvaran är mycket dynamisk och kan anpassas på många plan för att passa varje specifik kund (Lindhé, 2011).

2.4 Sammanfattning

Datoriserade kassasystem kan, utöver den funktionalitet som de traditionella kassaapparaterna erbjuder, även användas för att förmedla beställningar från serveringspersonalen till restaurangköket. Med en handterminal med trådlös anslutning till kassasystemet kan serveringspersonalen besöka fler gäster utan att

behöva uppsöka en stationär kassaterminal. Försök har gjorts att använda surfplattor i restaurangmiljö, bland annat som meny för gästerna. Även utanför restaurangbranschen har surfplattor använts kommersiellt, bland annat inom sjukvården.

Kapitel 3

Val av surfplatta

Kassaapplikationen som har utvecklats kommer att säljas som en del av en helhetslösning där maskinvara i form av en surfplatta ingår. Då lösningen kommer att användas i restauranger, kommer den tidvis bli utsatt för hårda tag, något som ställer hårda krav på surfplattorna. I detta avsnitt kommer de kriterier som har används för att utvärdera surfplattorna att beskrivas och de valda surfplattorna kommer att presenteras.

3.1 Bakgrund

Arbetsmiljön på restauranger kommer att vara krävande för surfplattorna; serveringspersonalen kommer stundtals att bära dem med sig och stundtals att lägga ifrån sig dem. De kommer att befinna sig i närvaro av både mat och dryck, så det är inte helt orimligt att anta att surfplattorna kommer att utsättas för fukt, spill och stötar. De kommer förmodligen även i vissa fall att bli tappade i golvet. Det är därför viktigt att de surfplattor som väljs ut är välanpassade för ändamålet och kan tåla att användas i denna miljö. Det finns flera olika krav som måste uppfyllas och flera olika egenskaper som behöver vara anpassade för ändamålet så gott som möjligt.

3.2 Krav och egenskaper

För att den maskinvara som valts ska hålla för det hårda klimat som den kommer att utsättas för har ett antal surfplattor valts ut och jämförts enligt ett antal kriterier. De utvalda plattorna har hittats dels genom produktlistor på Prisjakt (www.prisjakt.nu), dels genom sökningar på Google (www.google.se).

De kriterier som ställdes upp har delats in i två kategorier: en kategori för krav som måste uppfyllas för att plattan ska komma i fråga, samt en kategori för allmänna egenskaper som har tagits i beaktning. Även plattor som uppfyller kraven har gallrats bort för att hålla platturvalet i rimlig omfattning. De utvalda plattorna anges i avsnittet Resultat.

3.2.1 Krav

TRÅDLÖS NÄTVERKSÅTKOMST. Serveringspersonalen kommer att bära plattan med sig runt i restaurangen. Av praktiska skäl måste plattan vara sladdlös. Då programmet dessutom måste kommunicera med kassasystemet, måste plattan ha en trådlös nätverksanslutning.

SKÄRMSTORLEK. Plattan måste ha en skärmstorlek i intervallet 7-11 tum. Minde skärmar anses rymma för liten mängd information. Detta skulle innebära att vyer måste delas upp och personalen måste scrolla mer för att nå rätt information, vilket leder till att personalen måste rikta en större del av uppmärksamheten på enheten än på gästerna. Enheter med större skärmar anses bli för klumpiga för att bäras omkring.

3.2.2 Egenskaper

PLATTFORM. I dagsläget är det på den svenska surfplattemarknaden vanligast att surfplattorna kör operativsystemet Microsoft Windows 7, Googles Android eller Apples iOS (Prisjakt Sverige AB, 2011). Varje plattform har sina för- och nackdelar. Plattformarna kommer att granskas närmare i avsnitt 3.6.

BATTERITID. Plattorna bör optimalt kunna användas en hel dag utan att behöva laddas. Ett sådant krav kan dock inte anses vara realistiskt utifrån de specifikationer som granskats (se exempel på specifikationer i figur 3.5). Dock är en lång batteritid givetvis att föredra framför en kort.

HÅLLBARHET. Plattan kan komma att utsättas för hård behandling och bör tåla detta. Plattorna bör i synnerhet tåla väta och att tappas i golvet. Tanken är att en lite dyrare men hållbarare modell kan bli billigare i längden än en billig modell som ofta går sönder och måste ersättas. Detta kommer att beskrivas närmare i sektion 3.3.

UTSEENDE. En av de bakomliggande faktorerna för produkten är att den ska ge ett modernt och fräscht intryck. Det ställer krav på surfplattans utseende. I detta fall finns det givetvis inga objektiva kriterier för vad som är passande, utan denna bedömning har gjorts helt subjektivt.

SKÄRMTYP. Platta skärmar finns i flera utföranden, både med avseende på hur bilden framställs och hur trycken registreras. Dessa beskrivs närmare i avsnitt 3.4 och 3.5.

STORLEK OCH VIKT. Förutom skärmstorleken har hänsyn även tagits till plattornas storlek i övrigt. I synnerhet tjockleken kan vara viktig att kontrollera, då denna kan avgöra plattans greppvänlighet. Enheter som är så tunga att det är orealistiskt att personalen orkar bära dem med sig under en längre period har exkluderats från jämförelsen.

TILLGÄNGLIGHET. Några plattor har exkluderats på grund av att de antingen inte släppts och saknar information om lanseringsdatum, eller att plattan saknar svenska leverantörer och enkla importalternativ.

KOSTNAD. Uppdragsgivaren har uppgett att plattornas kostnad spelar en mindre roll, men viss hänsyn har ändå tagits för kostnaden.

3.3 Hållbarhet

Surfplattan som kommer att vara värd för slutgiltiga produkten kommer troligen att utsättas för hårda prövningar; de kan komma att tappas i golvet, de kan komma att användas med smutsiga händer, de kan komma att utsättas för fuktiga miljöer eller rentav sköljas under rinnande vatten. Frågan är om det finns surfplattor som klarar av denna hårda miljö.

Figur 3.1. Exempel på en robust dator på en bilverkstad.

Sveriges Televisions konsumentprogram Plus testade den 10 mars 2011 de två konsumentplattorna Apple Ipad och Samsung Galaxy Tab. SVT lät Patrik Norqvist, fysiker vid Umeå universitet, göra ett antal tester av plattornas hållbarhet. Det visade sig att plattorna klarade av att tappas från drygt en meters höjd in i ett bordshörn och klarade av att utsättas för 45 kg vikt fördelat på några kvadratcentimeter, vilket enligt Norqvist motsvarar trycket under ett bildäck. Skärmglaset sprack dock då en järnkula med en vikt på ungefär 1 kg släpptes från 30 cm höjd på plattorna. (Sveriges Television, 2011)

Man kan dela in bärbara datorer i kategorierna robusta, semirobusta och icke-robusta datorer (Cooper & Lee, 2008). De robusta datorerna (se exempel i

Figur 3.1) är speciellt anpassade för mycket hårda klimat och ska hålla för att användas i miljöer med bland annat sand och smuts, och ska hålla för slag och vibrationer. Exempel på användningsområden är fordonsmonterade datorer och datorer som används på byggarbetsplatser och oljeriggar.

På andra sidan av skalan finns de icke-robusta datorerna, dit merparten av de kommersiella bärbara datorerna hör. Hållbarhetskraven på dessa datorer är lägre, men bör vara hållbara nog för att kunna ersätta stationära datorer i kontorsmiljö.

Mellan de robusta och icke-robusta datorerna finns semirobusta datorer, som bland annat bör klara av bruk inom sjukvård och av fältsäljare som är på resande fot en hel del (Cooper & Lee, 2008).

Många tillverkare som säger sig bygga robusta datorer använder sig av standarden MIL-STD-810F (US Department of Defence, 2000) som har utvecklats av den amerikanska militären. Standarden specificerar ett antal miljötester som testar variabler som luftfuktighet, höga och låga temperaturer, vibration och smuts. Dock certifierar militären inga enheter, vilket innebär att det är upp till tillverkaren att testa om enheten uppfyller ett specifikt test. Vidare föreskriver inte standarden någon certifiering, något som vissa tillverkare utnyttjar, genom att i sin marknadsföring ange att deras enheter är MIL-STD-810F-kompatibla, när de i själva verket endast uppfyller ett eller ett fåtal av de tester som finns angivna i standarden (Cooper & Lee, 2008).

3.4 Skärmtekniker

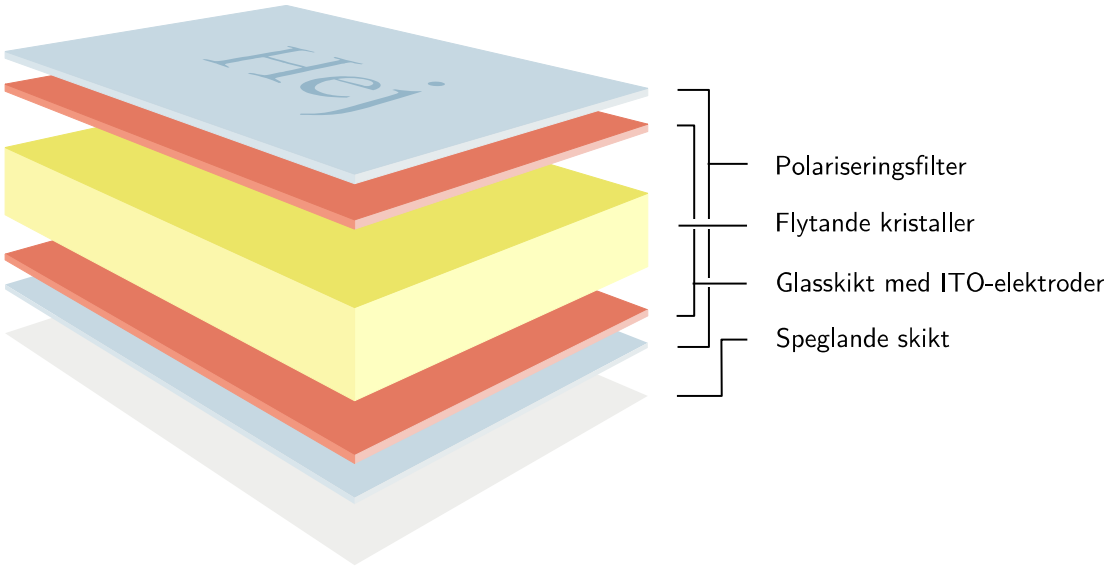
I fallet surfplattor är endast platta skärmar av intresse. I det här avsnittet kommer plattskärmstekniken LCD och olika utföranden av denna skärmtyp att presenteras.

Den första LCD-skärmen uppfanns 1971 av de två schweiziska forskarna Martin Schadt och Wolfgang Helfrich. Uppfinningen baseras på upptäckten av flytande kristaller som gjordes redan 1888. (VDMA, 2008)

Plattskärmstekniker kan delas in i självemitterande skärmar, som själva fungerar som ljuskälla, och icke-emitterande skärmar, som behöver en extern

ljuskälla för att fungera. LCD-skärmarna tillhör den senare kategorin (VDMA, 2008).

LCD-skärmarnas funktion kan liknas vid en OH-bild (overheadbild) (se Figur 3.2), där vissa partier släpper igenom ljus medan andra inte gör det. I varje cell i skärmen finns flytande kristaller, som kan påverkas på ett sådant sätt att de släpper igenom ljus oförändrat då de är inaktiva, men vrider ljusets polarisering ungefär 90 grader då de aktiveras genom att utsättas för ett elektriskt fält. Genom att polarisera ljuset på väg in i skärmen och filtrera ljuset med avseende på dess polarisering på väg ut från skärmen, uppnås en effekt som liknar den i OH-bilder (VDMA, 2008).



Figur 3.2. Uppbyggnad av en reflektiv LCD-skärm.
(Wikipedia, 2007)

Det finns tre vanliga tekniker som används för att tillgodose LCD-skärmarnas behov av en extern ljuskälla:

TRANSMITTIVA SKÄRMAR har bakgrundsbelysning som vanligen utgörs av CCFL-lysrör eller LED-dioder[2]. Nackdelen med denna typ av skärmar är att bakgrundsbelysningen lätt konkurreras ut av omgivande ljus, såsom strakt

[2] Det man i allmänhet kallar LED-TV, är egentligen en LCD-skärm med LED-belysning. Även LED-skärmar existerar, men har relativt stora pixlar och måste således ses på större avstånd. De används dock ofta vid sportevenemang och konserter (Schultz, 2011).

solsken, samt att bakgrundsbelysningen alltid måste vara på vilket leder till ökad strömförbrukning (Zhu et al., 2005).

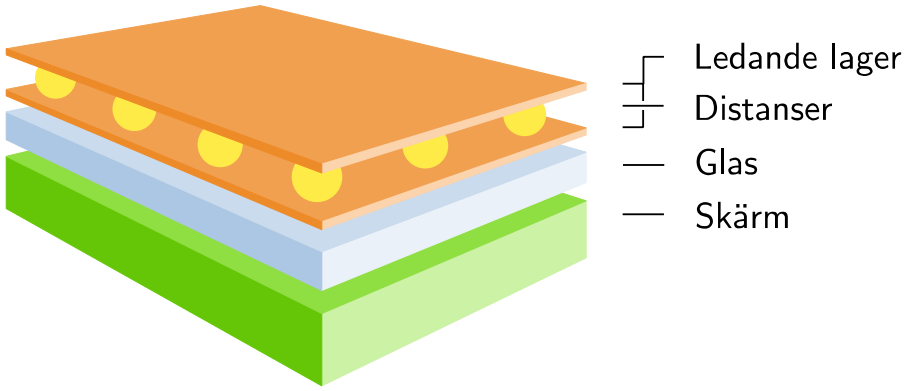
REFLEKTIVA SKÄRMAR saknar egen ljuskälla, utan förlitar sig istället på omgivande ljus som ljuskälla. Detta uppnås genom att baksidan av skärmen täcks med spegelfilm som reflekterar det omgivande ljuset (VDMA, 2008). Fördelen är att skärmens innehåll syns tydligt även i starkt ljus utomhus, och att avsaknaden av en intern ljuskälla leder till minskad strömförbrukning. Avsaknaden av en intern ljuskälla kan också vara en nackdel i oupplysta miljöer (Zhu et al., 2005).

TRANSFLEKTIVA SKÄRMAR har såväl en intern ljuskälla som ett reflekterande lager på baksidan, och har utvecklats för att ta tillvara på det bästa ur båda världarna. I ljusa miljöer kan ljuskällan inaktiveras för att spara ström, och fungerar då som en reflektiv skärm. Vid mörkare förhållanden kan ljuskällan aktiveras för att komplettera det omgivande ljuset (Zhu et al., 2005).

En transflektiv skärm skulle därför vara önskvärd i surfplattan som används för att köra produkten, dels för att öka läsbarheten då den används utomhus, dels för att minska strömförbrukningen.

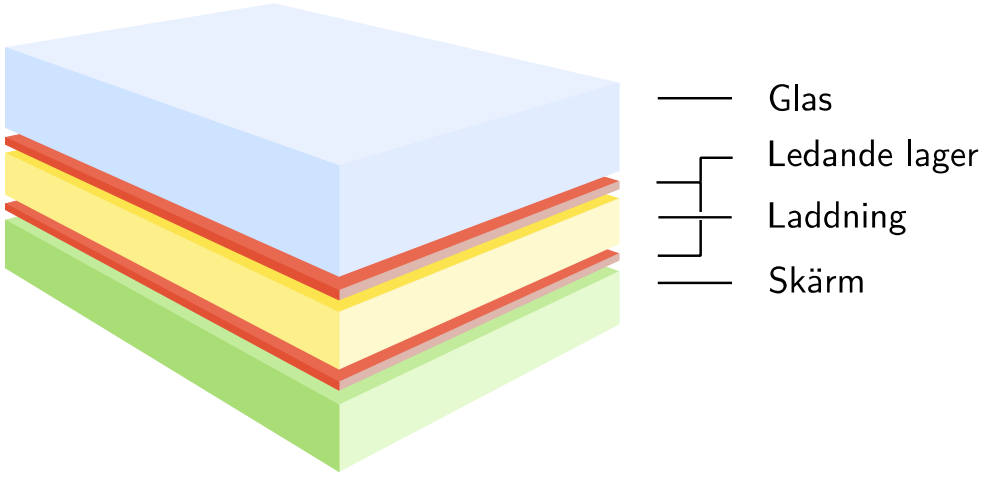
3.5 Pekskärmstekniker

I skrivande stund finns surfplattor med kapacitiva och resistiva pekskärmar för jämförelse på Prisjakt (Prisjakt Sverige AB, 2011). Idag domineras pekskärmsmarknaden av de resistiva skärmarna enligt marknadsinstitutet DisplaySearch (Semenza, 2010). Enligt DisplaySearch 2010 Touch Panel Market Analysis förväntas dock att de resistiva skärmarnas betydelse kommer att minska till förmån för de kapacitiva skärmarna (Semenza, 2010).



Figur 3.3. Uppbyggnad av en resistiv pekskärm.

RESISTIVA SKÄRMAR är beklädda med två flexibla ledande lager som är separerade med ett tunt lager av distanser som håller lagren nätt och jämnt isär (se Figur 3.3). Då man trycker på skärmen kommer det övre lagret i kontakt med det nedre och elektroner kommer att flöda mellan lagren. Genom att beräkna resistansen mellan lagren kan tryckpositionen bestämmas med hög precision (Baker & Fang, 2007). Den höga precisionen gör de resistiva skärmarna lämpliga att använda med penna (Hessel, 2010).



Figur 3.4. Uppbyggnad av en kapacitiv pekskärm.

Det finns KAPACITIVA SKÄRMAR (se Figur 3.4) som kan registrera flera tryck samtidigt och sådana som endast kan registrera ett tryck åt gången. Konstruktionerna på dessa skiljer sig något åt, men båda använder sig av kapacitans för

att detektera tryck. Beskrivningen i detta avsnitt avser de skärmar som har stöd för att registrera flera tryck åt gången.

En kapacitiv skärm är beklädd med två lager av elektriskt ledande mönster, som är separerade av ett tunt isolerande skikt. Genom att sätta en spänning mellan lagren bygger man upp en liten laddning mellan lagren. Dessa två lager bildar på så sätt en mängd transparenta kondensatorer. Då ett finger närmar sig en av dessa kondensatorer kommer fingret att stjäla en del av dess laddning, vilket detekteras av enheten (Barrett & Omote, 2010).

Det är kroppens kapacitans som gör det möjligt att detektera tryck på en kapacitiv skärm. Detta medför att tryck med pennor eller med handskar inte registreras av en kapacitiv skärm. Dock finns det pennor och handskar som är specialutvecklade för att fungera med kapacitiva skärmar (Hessel, 2010).

De kapacitiva skärmarna drar mer ström och är dyrare att tillverka än resistiva skärmar. På plussidan kan nämnas att de kapacitiva skärmarna upplevs som mer responsiva eftersom de inte erfordrar lika hårda tryck som resistiva skärmar (Hessel, 2010).

3.6 Plattformar

De vanligaste plattformarna som används i surfplattorna idag kan efter en kontroll på Prisjakt antas vara iOS, Android och Microsoft Windows 7. Dessa kommer att presenteras i detta avsnitt.

3.6.1 iOS

iOS, tidigare känt som Iphone OS, släpptes tillsammans med Apples Iphone under sommaren 2007, men används idag även i Ipod touch och Ipad (Wikipedia, 2011f). Operativsystemet är baserat på Mac OS X, vilket gör iOS till ett Unix-liknande operativsystem.

Inledningsvis hade operativsystemet inget stöd för tredjepartsapplikationer. I februari 2008 släppte Apple dock ett utvecklarpaket (SDK), som utvecklare kunde använda för att utveckla applikationer för iOS. Applikationerna, som utvecklas i programspråket Objective-C, kan testköras i medföljande simulator

eller i en enhet som kör iOS. Att föra över applikationen till en enhet är dock endast möjligt om man är med i Apples iOS Developer Program (Wikipedia, 2011f), vilket kräver att man har en Intel-baserad Mac som kör Mac OS X Snow Leopard (Apple Inc., 2011a) och betalar en årsavgift på 99 amerikanska dollar (Apple Inc., 2011b). Ett medlemskap i iOS Developer Program ger också möjligheten att publicera applikationer i App Store, Apples appbutik (Apple Inc., 2011b).

Apples iOS anses av vissa vara en instängd plattform (Payne, 2010). En av anledningarna är att man endast kan installera appar som hämtats från App Store, där alla appar godkänns av Apple (Sutter, 2010). Vissa typer av appar godkänner inte Apple, vilket gör att användare inte kan installera dessa utan att först låsa upp enheten genom att *jailbreaka* den. Då iOS-enheten låses upp kan man installera appar som inte är godkända av Apple, till exempel sådana som hämtats från Cydia – en appbutik som drivs av tredje part (Sutter, 2010).

Enligt Ny Teknik är det lagligt att jailbreaka sin iOS-enhet i USA. Daniel Westman, doktorand i rättsinformatik vid Stockholms universitet, säger till Ny Teknik att rättsläget i Sverige är oklart: »Det går inte att säga med säkerhet att det är förbjudet« (Dahlström, 2010).

3.6.2 Android

Android utvecklades av gruppen Open Handset Alliance – där företag som Google, HTC, Motorola och Ebay ingår – som en öppen och omfattande plattform för mobila enheter (Open Handset Alliance, 2007b). Android består av ett Linuxbaserat operativsystem och tillhörande nyckelapplikationer som tillhandahåller grundläggande funktionalitet såsom e-post, surfning, kartläsning och dylikt.

Androidutvecklare har full åtkomst till de API:er[3] som de inbyggda programmen använder. Utvecklingen av applikationer sker vanligtvis i programspråket Java och körs i separata processer med den virtuella maskinen Dalvik som värd (jämför med vanliga Javaprogram som körs i JVM, Javas virtuella maskin).

[3] Application Programming Interface, ett programmeringsgränssnitt som kan användas av program för att kommunicera med andra program (Wikipedia, 2011).

Operativsystemet har stöd för att ha flera instanser av Dalviks virtuella maskin igång samtidigt vilket möjliggör multikörning.

I Androids dokumentation rekommenderas Eclipse som utvecklingsmiljö (Google, 2011b), vilken somliga är bekanta med från utveckling av Javaapplikationer. I Android SDK ingår en emulator som emulerar en mobil enhet med Android. Emulatorn kan användas för att testköra Androidappar om man saknar en riktig Androidenhet att testköra på (Google, 2011a). Stöd finns idag också för att köra operativsystemet på en vanlig dator med x86-baserad processorarkitektur (Android-x86, 2011).

En fördel med Android-plattformen är att den homogeniserar gränssnittet till maskinvaran i en mängd mobila enheter. Utvecklaren kan utveckla ett program utan att bekymra sig om skillnaderna mellan de hårdvaruplattformar som programmet kommer att köras på.

Hela plattformen är baserad på öppen källkod. Alla kan utveckla för plattformen utan att behöva betala för sig (Open Handset Alliance, 2007a). Att plattformen är öppen innebär också att man kan utveckla Androidapplikationerna i valfri utvecklingsmiljö – det räcker med att någon har implementerat stöd för Android i utvecklingsmiljön.

3.6.3 Windows 7

I slutet av 1985 släppte Microsoft första versionen av Windows, Microsoft Windows 1.0. Versionen hade stöd för multikörning och innehöll redan då flera systemprogram som vi är vana vid idag, såsom Kalkylatorn, Anteckningar och MS Paint (Wikipedia, 2011l).

I början av 1990-talet gav sig Microsoft in i branschen för penndatorer med »Windows for Pen Computing«, ett tillägg för Windows 3.1 som bland annat gjorde skriftigenkänning möjlig (Wikipedia, 2011m) (Blickenstorfer, 2005). Branschen ansågs gå en ljus framtid tillmötes, där penndatorerna kunde ersätta de stationära datorerna med mus och tangentbord. Förväntningarna infriades dock inte då folk ansåg att penndatorerna var för krångliga att använda och att skriftigenkänningen inte fungerade tillfredställande. I mitten av 1990-talet ansågs

konsumentmarknaden för penndatorer vara utdöd, men penndatorerna levde vidare inom industrin (Blickenstorfer, 2005).

2002 gjorde Microsoft ett nytt försök med operativsystemet Windows XP Tablet PC Edition (Wikipedia, 2011i). Vid den tidpunkten fanns bättre teknik tillgänglig vilket ledde till att försöket lyckades bättre än under 90-talet (Blickenstorfer, 2005).

I och med lanseringen av Windows Vista upphörde Tablet PC som separat utgåva, då pekfunktionerna istället infördes i den vanliga utgåvan av Windows. Nytt för Windows Vista var också stöd för att peka på skärmen med fler än ett finger, samt förbättrat stöd för skriftigenkänning. Detta stöd förbättrades ytterligare i Windows 7 (Wikipedia, 2011i).

Eftersom det finns så pass många alternativ för utvecklingsplattformer för Windows har .NET Framework valts ut som den plattform som presenteras i denna uppsats. .NET Framework är en grund för program och webbtjänster som är oberoende av var koden körs och lagras (Microsoft, 2011a).

.NET Framework innehåller ett objektorienterat klassbibliotek som kan användas för att bygga grafiska användargränssnitt, såväl i webbapplikationer som i program som körs lokalt. Program skrivna för .NET Framework exekveras av CLR, Common Language Runtime, som fungerar som en virtuell maskin (jämför Javas klassbibliotek och virtuella maskin). CLR tillhandahåller bland annat en skräpsamlare som frigör minne som ockuperas av dataobjekt som inte längre används (Microsoft, 2011a).

.NET Framework stödjer en mängd programspråk, däribland C#, VB.NET och F# för att nämna några (Wikipedia, 2011g).

Några nackdelar med plattformen är att plattorna för Windows generellt verkar vara dyrare enligt tabellen i Figur 3.5, samt att Windowsplattor med skärmar mindre än 10 tum verkar vara obefintliga efter en kontroll på Prisjakt (Prisjakt Sverige AB, 2011).

3.7 Resultat

Bakgrunden till valet av surfplatta har nu presenteras. I detta avsnitt kommer de utvalda surfplattorna och den valda plattformen att presenteras.

3.7.1 Surfplattor

Ett antal surfplattor har valts ut som skulle kunna vara intressanta att användas tillsammans med programmet. Vid urvalet togs hänsyn till de krav och egenskaper som tidigare presenterats. Att presentera varje surfplatta som har granskats skulle ta alldeles för mycket plats, så i detta avsnitt presenteras endast de surfplattor som anses kunna vara lämpliga för ändamålet. Notera att urvalet av plattor gjordes före valet av plattform, vilket innebär att det finns utvalda surfplattor för alla plattformar. Plattorna som har granskats har hittats via huvudsakligen Prisjakt, men även genom sökningar på Google.

Se information om de utvalda plattorna i Figur 3.5. Den platta som ansågs vara mest lämpad för ändamålet är Motion CL900, som är tillverkad för att vara extra tålig.

3.7.2 Plattform: Windows 7

Valet av plattform föll på Windows 7. Det finns flera orsaker till att Windowsvaldes framför de andra plattformarna. Här är några:

BEKANT. Både vi som arbetat med projektet och uppdragsgivaren är mest bekanta med Windowsplattformen för utveckling. Användarna är bekanta med Windows som operativsystem.

SUPPORTVÄNLIGHET. Med Windowsplattor kan supporten koppla upp sig till en enskild platta via Remote Desktop Protocol och se och styra vad som händer på surfplattan (Wikipedia, 2011k).

BEPRÖVAD PLATTFORM. Utvecklingen av Windows påbörjades redan 1981 (Wikipedia, 2011l) och har haft många år på sig att mogna.

BRED PLATTFORM. Det finns en mängd programspråk, utvecklingsplattformar och verktyg som kan användas i Windowsmiljö (Wikipedia, 2011h).



	Galaxy Tab [1]	Ipad (utan 3G) [1]	Internet Tablet [1]
Tillverkare	Samsung	Apple	Archos
[2] **Operativsystem**	Android 2.2	iOS	Android 2.2
Skärmstorlek	7″	9,7″	10,1″
Upplösning	1024 x 600	1024 x 768	1024 x 600
Pekskärmteknik	Kapacitiv	Kapacitiv	Kapacitiv
[3] **Batteritid**	4,5 h	10 h	10 h
Vikt	385 g	680 g	480 g
Mått (mm)	190 x 120 x 12	243 x 190 x 13,4	270 x 150 x 12
[4] **Pris**	3 300 kr	2 800 kr	2 300 kr
Källor	(Samsung, u.å.)	(Apple Inc., 2010)	(Archos, u.å.)
Övrigt		Kräver en Intel-baserad Mac för utveckling samt ett utvecklarkonto hos Apple.	

Figur 3.5. Jämförelse av de utvalda surfplattorna.




ViewPad 10	WindPad	Paddy-T V2	Stylistic Q550	Motion CL900
Viewsonic	MSI	Paddytek	Fujitsu	Motion Computing
Win 7 P/HP + Android 1.6	Win 7 HP	Win 7 HP	Win 7 P/HP	Win 7 P/HP
10,1″	10,1″	10,1″	10,1″	10,1″
1024 x 600	1024 x 600	1280 x 720	1280 x 800	1366 x 768
Kapacitiv	Kapacitiv	Kapacitiv	Kapacitiv	Kapacitiv
4 h	6 h	4 h	8 h (3 h)	8 h (2 h)
875 g	800 g	880 g	760 g	950 g
275 x 170 x 16,5	274 x 173 x 18,5	275 x 170 x 14	275 x 192 x 16,2	277 x 179 x 15,4
2 700 kr	3 800 kr	5 600 kr	6 500 kr	7 000 kr
(ViewSonic Corporation, u.å.)	(Webhallen, 2011) (Thörnqvist, 2011)	(PCB Distribution AB, 2011)	(Fujitsu, 2011)	(Motion Computing, 2011)
	Testning av prototypen skedde på denna platta under utvecklingen.	Svensk tillverkare.	Lanseras 2 maj i Sverige.	Lanseras 2:a kvartalet i USA. Transflektiv skärm som tillval. Testad enligt MIL-STD-810G för fall från 1,2 m, IP52, extratåligt glas.

[1] Dessa enheter finns med för jämförelse med de relevanta Windowsplattorna.

[2] Win 7 P = Windows 7 Professional, Win 7 HP = Windows 7 Home Premium.

[3] Laddtid anges inom parentes.

[4] Priserna anges exklusive moms och är hämtade från Prisjakt den 13 mars 2011, eller tillverkarens riktpriser om plattan ännu inte släppts.

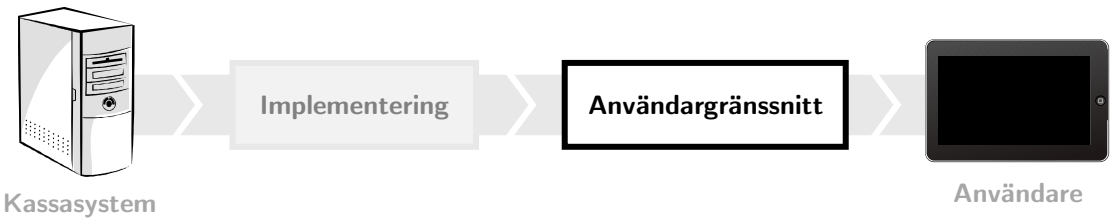
3.8 Sammanfattning

Hänsyn har tagits för flera egenskaper då urvalet av surfplattor gjordes. Bland annat bör plattan ha lång batteritid, tåla hård behandling och vara lämplig att bära omkring, både med avseende på vikt och storlek. Skärmen är en annan faktor som har spelat in i urvalet.

Windows 7 har valts som plattform som programmet kommer att köras på, där .NET Framework används som grund för programmet.

Kapitel 4

Användargränssnitt



Figur 4.1. Användargränssnittet i relation till projektets indelning.

Då arbetet inleddes fanns ingen uttalad specifikation för projektet. Den första uppgiften var att ta fram en specifikation i samråd med uppdragsgivaren. I samband med att specifikationen togs fram gjordes också överskådliga skisser över användargränssnittet.

4.1 Utformning av grafiskt gränssnitt

Utformningen av det grafiska användargränssnittet i prototypen gjordes av oss själva. Gränssnittet är till för att underlätta interaktionen mellan kassasystemet och användaren.

4.1.1 Önskvärda egenskaper

Innan arbetet med att skissera gränssnittet påbörjades bestämdes några önskvärda egenskaper för gränssnittet:

PEKVÄNLIGT. Programvaran kommer i produktionsmiljö uteslutande att användas med pekskärm. Ett krav är därmed att det ska vara lätt att använda med endast fingrar.

INTUITIVT. Gränssnittet ska i största möjliga mån utvecklas enligt rådande uppfattningar om hur grafiska användargränssnitt fungerar. Gränssnittet i stationära kassasystem kommer efterliknas till viss del för att användaren ska känna sig bekant med gränssnittet. Det är viktigt att användaren är medveten om statusen på beställningar.

LÄTTANVÄNT. Det ska gå lätt och smidigt att använda gränssnittet. När serveringspersonalen använder gränssnittet ska uppmärksamheten kunna ligga på gästerna och inte på gränssnittet.

TILLTALANDE. Gränssnittet är förvisso huvudsakligen avsett för serveringspersonalen i restauranger, men även gästerna eller andra personer kan komma i kontakt med gränssnittet. Därför bör programmet vara estetiskt tilltalande, utan att för den delen dra till sig onödig uppmärksamhet.

UPPLÖSNINGSOBEROENDE. Gränssnittet bör inte vara beroende av en specifik pixeltäthet. Vissa surfplattor har skärmar med högre pixeltäthet än det normala för datorskärmar (96 dpi, punkter per kvadrattum) (Wikipedia, 2011d), vilket kan resultera i att detaljerna i gränssnittet blir för små. Därför är det bra om gränssnittet skalas om automatiskt efter bildskärmens pixeltäthet för att alla detaljer ska behålla sina fysiska dimensioner.

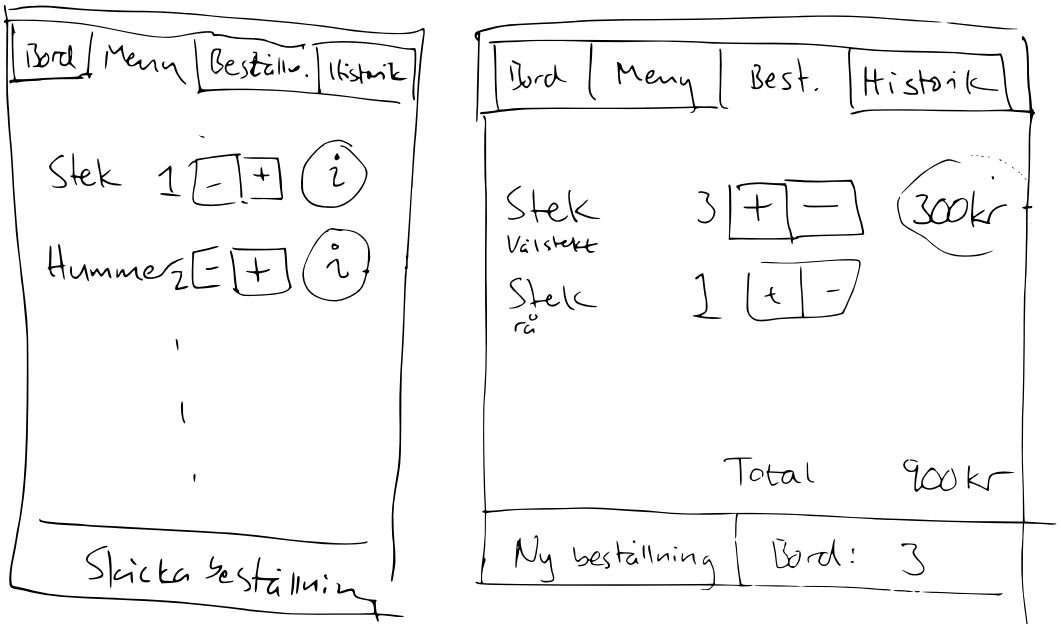
De inledande skisserna gjordes med papper och penna efter diskussioner med uppdragsgivaren. Skisser gjordes för de vyer i programmet som inledningsvis var planerade. Skisserna diskuterades sedan med uppdragsgivaren, innan några vyer designades digitalt utefter resultatet från diskussionerna.

4.1.2 Gränssnitt för pekskärmar

När det gäller pekskärmar har Microsoft listat en del principer som är bra att följa för att gränssnittet ska uppfylla de önskvärda egenskaper som satts upp (Microsoft, 2009).

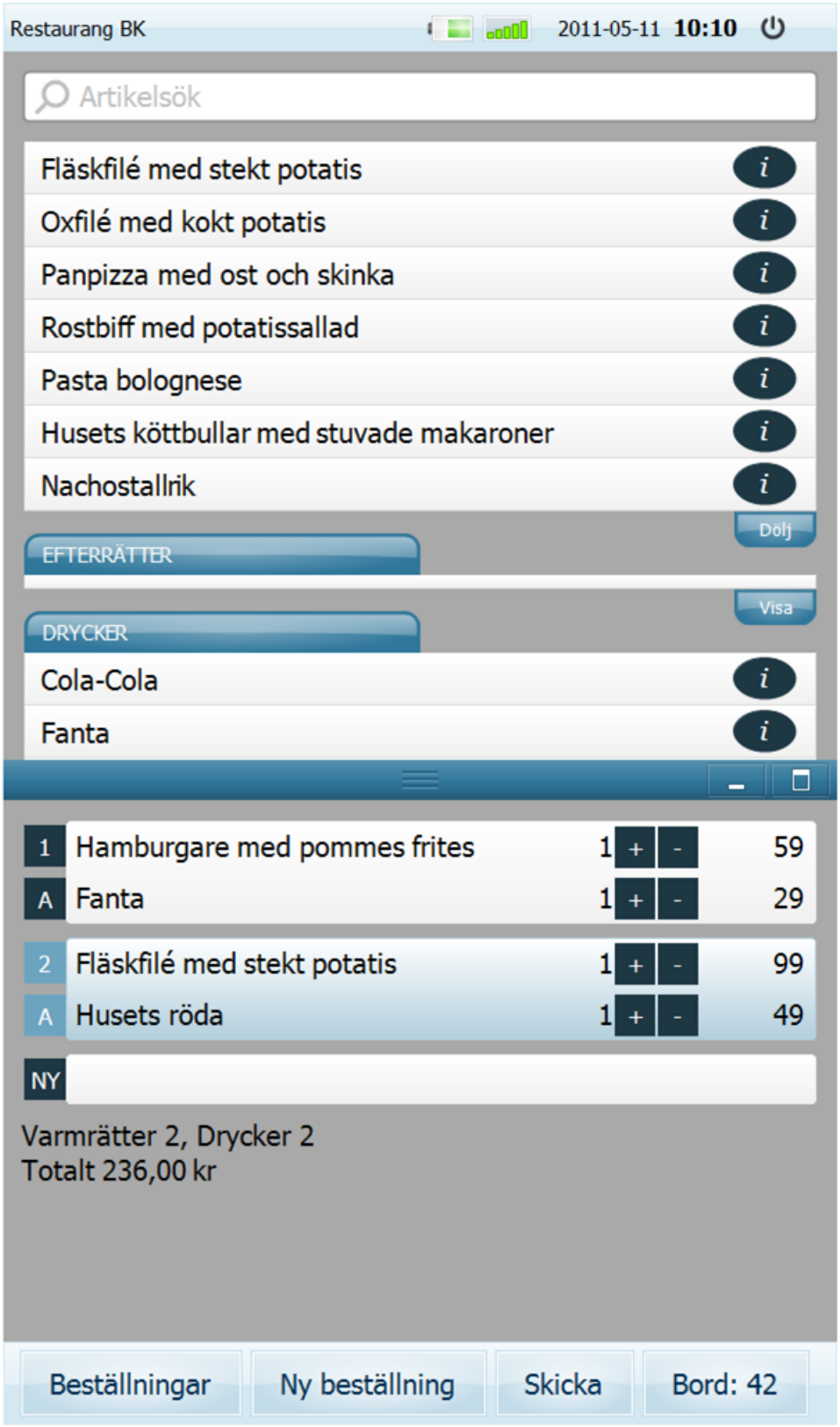
- Minimera antalet tryckningar, för att göra gränssnittet så smidigt som möjligt att använda och för att snabba upp användandet.
- Knappar och andra tryckbara komponenter måste vara stora nog för att det ska vara enkelt att träffa rätt (knappstorlekar vid 96 dpi bör vara minst 6x6 millimeter (Microsoft, 2009), gärna större än 10x10 millimeter för ökad effektivitet och träffsäkerhet).
- Det ska inte finnas några små hål mellan tryckbara komponenter som inte reagerar på tryckningar. Komponenterna ska antingen placeras så att de ligger intill varandra, så att det är ett tillräckligt stort mellanrum mellan dem (minst 1,4 millimeter vid 96 dpi)[4], eller göra så att tryckningar även mellan komponenterna leder till att den närmaste komponenten blir tryckt.
- Gränssnittet behöver ritas om snabbt när scrollning sker så att det ger ett smidigt och följsamt intryck.
- Undvik att kräva tryck långt ifrån varandra. När man använder en mus ger en 3 centimeters förflyttning av musen under normala förhållanden en förflyttning av pekaren på drygt 14 centimeter, medan man med en pekskärm måste flytta fingret de 14 centimetrarna i stället (Microsoft, 2009).
- Underlätta vid textinmatning. Att skriva in text utan ett tangentbord är krävande och hjälpmedel för att underlätta är rekommenderat.

[4] Uträknat värde.



Figur 4.2. Skiss av fristående meny- och beställningsvyer.

Figur 4.3. Skärmbild av kombinerad meny- och beställningsvy.



Figur 4.4. Skärmbild av beställningsvyn i vertikalt läge.

4.2 Vyer

Gränssnittet kommer att bestå av flera olika vyer som fyller olika funktioner. Huvudvyn som alla delar utgår ifrån är menyn, varifrån det går att på ett eller annat sätt nå alla övriga vyer. Det är den mest centrala delen i gränssnittet. Alla vyer är anpassade så att de kan användas oavsett vilken orientering plattan befinner sig i (antingen liggande eller stående, jämför Figur 4.3 och Figur 4.4).

4.2.1 Menyvy

I menyns övre eller vänstra del (beroende på orientering) listas alla artiklar som finns inlagda i kassasystemet, indelade i kategorier (se B i Figur 4.3). En kategori kan komprimeras eller expanderas för att ge plats åt andra kategorier genom att trycka på kategorins rubrik eller på minimera knappen i kategorins nedre högra hörn. Användaren scrollar i menyn genom att svepa med fingret i höjdled.

Figur 4.5. Informationsdialog med detaljer om en artikel.

Varje artikel har en informationsknapp (C), vilket leder användaren till en informationsdialog för just den artikeln (se Figur 4.5). Informationsdialogen visar en mer utförlig beskrivning med eventuell tillhörande bild av artikeln.

Figur 4.6. Sökning efter pommes frites.

Användaren kan också snabbt hitta rätt artikel genom att söka i artikellistan. Detta sker genom att användaren trycker i sökrutan (A) som finns högst upp i vyn, varpå ett skärmtangentbord kommer fram, som används för att ange ett sökord. Sökordet kan antingen vara en del av eller hela artikelns titel, eller vara en akronym[5] för orden som ingår i artikelns titel. Sökresultatet uppdateras under tiden man skriver (se Figur 4.6).

I vyns nedre eller högra del (beroende på skärmens orientering) visas den aktuella beställningen. Beställningen utgörs av ett antal *delbeställningar*, där varje delbeställning representerar en stolsplats. Varje delbeställning anger vad en enskild person har beställt. Mer specifikt visas:

- de artiklar som personen beställt (F)
- antal av varje artikel (G)
- pris för artiklarna (J)

Delbeställningar används av användaren för att snabbt ange vilka som ska betala tillsammans när notan efterfrågas. Det går att gruppera (E) olika delbeställningar tillsammans för att underlätta när inte alla gäster betalar var för sig. Även här scrollar användaren i menyn genom att svepa med fingret i höjdled.

Det finns alltid en delbeställning som är markerad, och det är däri som artiklar hamnar när de läggs till. Man kan markera en annan delbeställning genom att trycka på den. En ny delbeställning läggs till genom att trycka på den

[5] En akronym är en förkortning som bildas av den inledande bokstaven i flera ord (Wikipedia, 2011a). Till exempel OMP för **o**xfilé **m**ed **p**epparsås.

tomma delbeställningen som alltid finns längst ner (K), som då automatiskt blir markerad.

Plus- och minusknappar (H) för varje artikel i beställningen kan användas för att ändra antal av artikeln, eller för att ta bort en artikel från beställningen.

Figur 4.7. Alternativdialog.

Om man dubbelklickar[6] eller högerklickar[7] på en artikel i en delbeställning kommer en alternativdialog (Figur 4.7) att visas. Här finns möjlighet att lägga till en notering och det finns möjlighet att flytta artikeln till en annan delbeställning.

Nedanför listan med delbeställningar finns en informationsrad som talar om det totala antalet artiklar per kategori som beställts och det totala priset för hela beställningen.

Mellan meny- och beställningsdelen finns en avdelare. Denna är anpassningsbar och kan justeras för att efter behov kunna ge större plats åt antingen meny-

[6] Dubbelklick på pekskärmen görs genom att trycka med fingret två gånger på skärmen.
[7] Högerklick åstadkoms genom att antingen vila med fingret på skärmen några sekunder eller vila med ett finger på skärmen och samtidigt trycka med ett annat finger bredvid.

eller beställningsdelen. Det går även att maximera en del så att enbart denna syns.

Längst ner i vyn finns fem knappar, *Beställningar*, *Ny beställning*, *Skicka*, *Fram kök* och *Bord*. Då beställningen är färdig, trycker användaren på knappen Skicka, vilket initierar en utskrift av en bong i köket och eventuellt även en utskrift i baren. Om beställningen består av exempelvis förrätt, varmrätt och efterrätt, kan användaren använda knappen Fram kök för att skicka ett meddelande till köket att det är dags att påbörja tillagningen av nästa rätt.

För att påbörja en ny beställning trycker användaren på Ny beställning. Då sparas den aktuella beställningen utan att skickas till köket och beställningsdelen töms så att det bara finns en ny tom delbeställning. Den sparade beställningen går att komma åt från beställningsvyn. Knappen Beställningar öppnar beställningsvyn, och knappen med bordsnummer visar en bordsväljare (se Figur 4.8).

Figur 4.8. Dialog där personalen väljer vilket bord en beställning hör till.

I bordsväljaren går det att välja vilket bord som en beställning hör till. Väljaren består av en lista och en textruta. I listan kommer bordsnummer 1 till och med

99 att vara listade och det kommer även att finnas alternativ för att välja bar eller bord 0. Bord 0 kommer att användas för beställningar som saknar ett specificerat bordsnummer. Det går även att ange valfri bordsbeteckning i textrutan.

Inledningsvis planerades att meny- och beställningsdelen skulle visas som två separata vyer, men den idén slopades till förmån för en mer lättöverskådlig kombinerad vy.

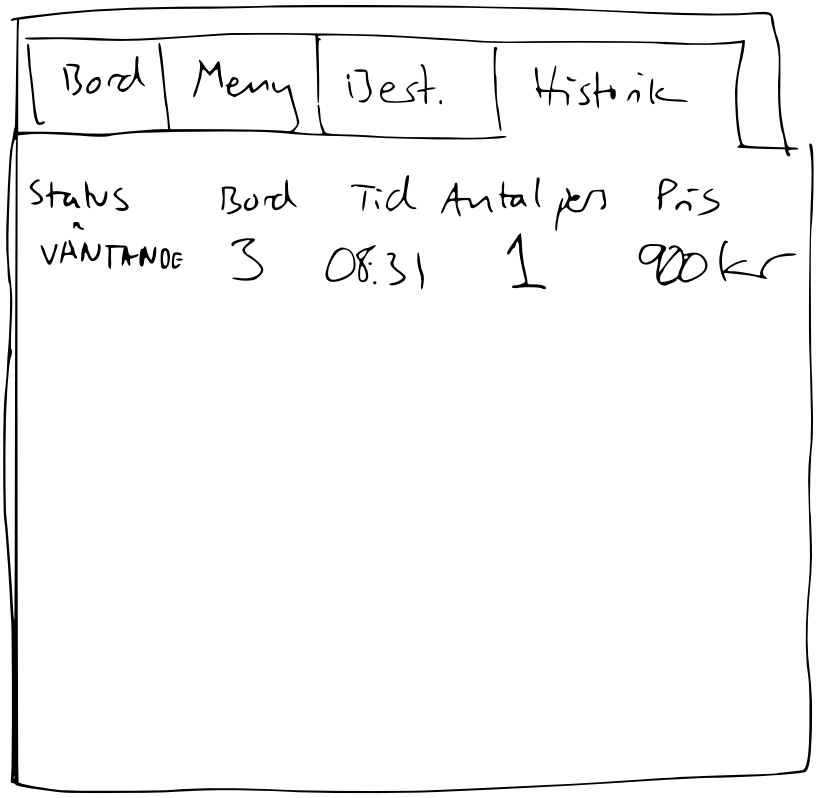
4.2.2 Beställningsvy

I beställningsvyn presenteras en översikt av alla beställningar som ännu inte har betalats. Vyn är uppdelad i två delar: en del där användarens egna beställningar visas och en del där beställningar gjorda av annan serveringspersonal visas.

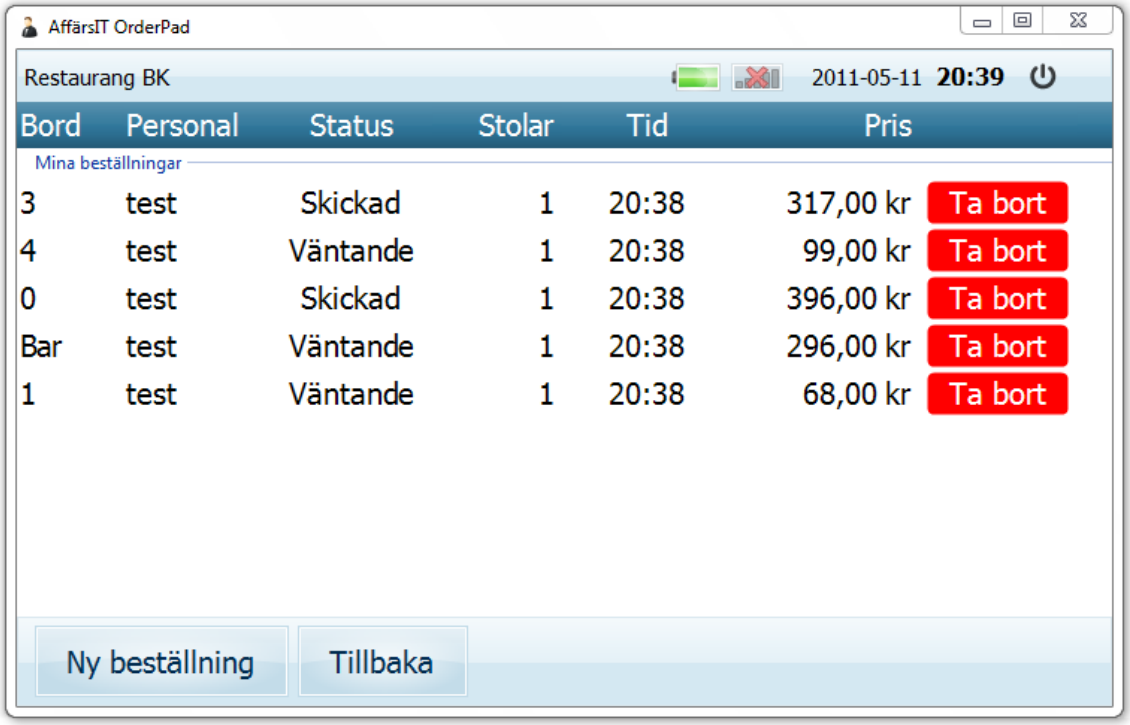
För varje beställning visas bordsnummer, namnet på den personal som har skapat beställningen, status, antal stolar, tiden då beställningen gjordes, beställningens totala pris och en knapp för att ta bort beställningen (endast egna beställningar). Statusen talar om ifall beställningen har blivit skickad till kök och bar eller om den är väntande.

Genom att trycka på en beställning kommer man till menyn, där beställningen öppnas i beställningsdelen. Det går att öppna beställningar gjorda av annan personal så länge som beställningen är stängd eller inte har ändrats under de senaste 60 sekunderna.

Längst ner i vyn finns två knappar – en för att skapa en ny beställning och en för att gå tillbaka. Tillbakaknappen leder till menyvyn där den senast aktiva beställningen öppnas. Om beställningen har tagits bort öppnas en tom beställning.



Figur 4.9. Ursprunglig skiss av beställningsvyn.



Figur 4.10. Skärmbild av beställningsvyn.

4.2.3 Inloggningsvy

Figur 4.11. Inloggningsvyn i två utföranden.

Det första en användare ser när den startar gränssnittet är en inloggningsvy (se Figur 4.11). Kunden kan välja mellan två inloggningsvyer: en av enklare typ där en numerisk kod både identifierar och autentiserar en användare, och en som identifierar användaren med ett användarnamn och kräver lösenord för autentisering. I det senare fallet används skärmtangentbordet för inmatning.

4.2.4 Statusrad

Figur 4.12. Statusraden som är synlig i alla vyer.

Allra högst upp i varje vy finns en statusrad (se Figur 4.12) som innehåller diverse användbar information. Denna rad är alltid synlig. Längst till vänster står restaurangens namn. På högersidan finns indikatorer för batteri- och

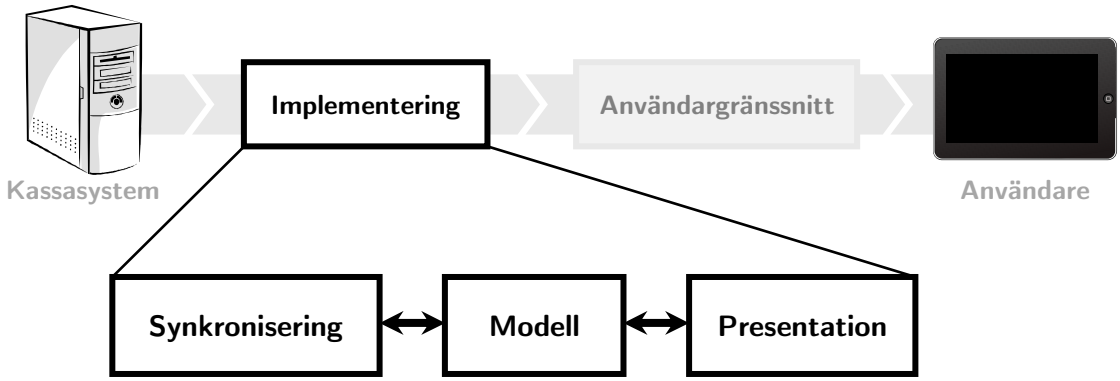
nätverksstatus, dagens datum och aktuell tid samt en knapp för att logga ut och stänga av.

4.3 Sammanfattning

Det grafiska användargränssnittet underlättar interaktionen mellan kassasystemet och användaren och har flera önskvärda egenskaper: det ska vara pekvänligt, intuitivt, lättanvänt, tilltalande och upplösningsoberoende. När gränssnittet utformas är det viktigt att tänka på att det kommer att användas på pekskärm och därför tillämpa vissa principer som finns för att underlätta användandet. Gränssnittet kommer bestå av olika vyer: menyvyn som används för att ta emot och ändra beställningar, bordsväljaren, beställningsvyn som används för att se aktuella beställningar och inloggningsvyn som används vid inloggning.

Kapitel 5

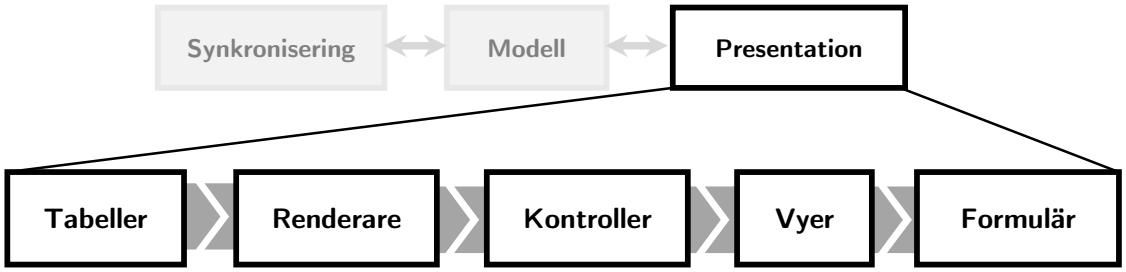
Implementering



Figur 5.1. Indelning av implementeringen.

Förra kapitlet visade hur programmets användargränssnitt ser ut och fungerar utifrån användarens perspektiv. Detta kapitel börjar med att presentera de olika lager som bygger upp användargränssnittet och översiktligt hur dessa är implementerade och används ur en programmerares perspektiv. Därefter gås modellen igenom, som representerar entiteter i kassasystemet, där all information från kassasystemet cachas. Slutligen presenteras synkroniseraren, som håller den lokala modellen och kassasystemet synkroniserade och vilka strategier som används för att realisera denna.

5.1 Presentation



Figur 5.2. Översikt av presentationslagrets beståndsdelar.

I förra kapitlet presenterades hur användargränssnittet är upplagt. I det här avsnittet kommer implementeringen av användargränssnittet att presenteras. Implementeringen har delats upp i flera lager för att särskilja utseendet från beteendet (se Figur 5.2). Det gör det enkelt att designa om hela eller delar av programmet, eller låta användaren välja utseende på användargränssnittet.

5.1.1 Formulär

Programmet innehåller enbart ett formulär, `MainForm`, och det är formulärets uppgift att visa en vy, hantera vybyte och att starta synkroniseraren. Det första som visas vid start är en inloggningsdialog.

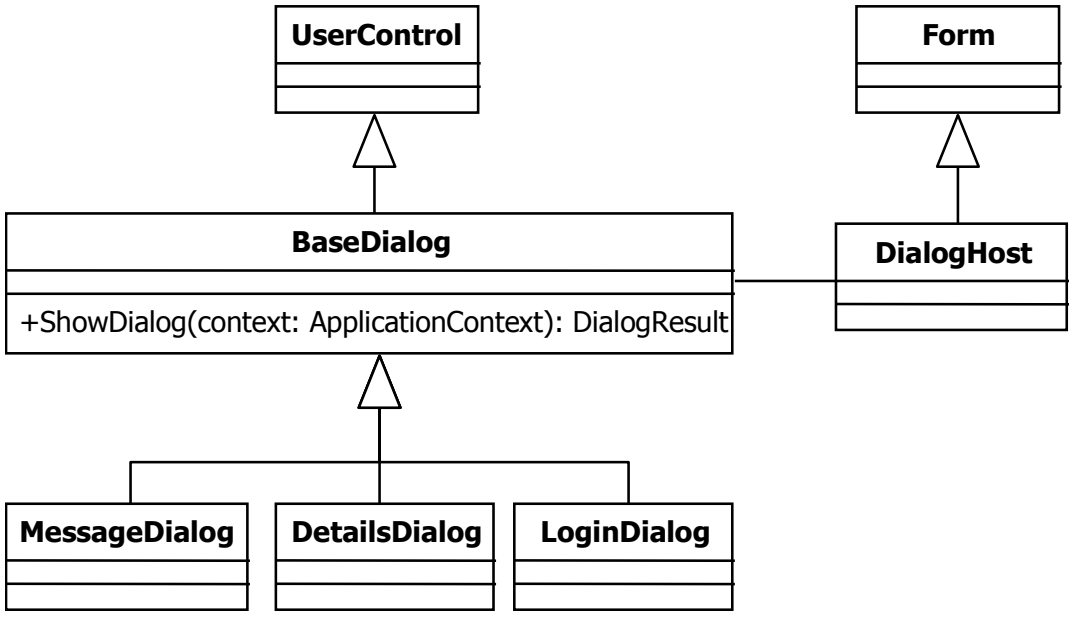
5.1.2 Vyer

Programmet innehåller ett antal skärmar som rymmer olika information. Var och en av dessa skärmar representeras av en vy i programmet. Vyernas uppgift är förutom att specificera innehållet i en skärm, att koordinera sina kontroller och renderare med modellen. När interaktion sker med en kontroll, till exempel i form av ett tryck, skickar kontrollen detta vidare till vyn som gör ändringar i modellen beroende på vad som ska hända. De olika vyerna ärver från `BaseView` som är en `UserControl`, vilket gör det lätt att designa vyer i designläget i Visual Studio.

5.1.3 Dialoger

Flera delar av användargränssnittet ansågs passa bättre som dialogrutor istället för separata vyer. Några exempel är inloggningsskärmen (se Figur 4.11) och meddelanderutor. För att programmet skulle få ett enhetligt utseende skapades ett bibliotek för dialoger med anpassad rendering. Notera att dialogen, till skillnad från en vy, inte täcker hela skärmen, och att den bakomliggande vyn syns nedtonad i bakgrunden för att påminna användaren om var han eller hon kom ifrån.

Varje dialog ärver från klassen `BaseDialog`, som i sin tur är en `UserControl` (se Figur 5.3 för en översikt av dialogklasserna). Det innebär att även dialogerna kan skapas i designläget i Visual Studio. `BaseDialog` omfattar dialogrutans innehåll exklusive dess nedtonade bakgrund.



Figur 5.3. Klasshierarki för dialoger.

Klassen `DialogHost` fungerar som värd för dialogerna. `DialogHost` skapar ett fönster som precis täcker den bakomliggande vyn. I fönstret ritas en bild av den bakomliggande vyn upp, varpå en renderare ansvarar för att rita effekter på bakgrunden. Standardrenderaren täcker hela bakgrunden med en halvgenom-

skinlig svart färg och ritar ut en skugga bakom dialogen. `BaseDialog` placeras därefter centrerat i fönstret.

Då en dialog ska visas används alltid den dialogklass som ärver från `BaseDialog` – `DialogHost` används endast av `BaseDialog` och undantagsvis av klasser som ärver från `BaseDialog`. `BaseDialog` ser till att dialogen som öppnas är modal, det vill säga att användaren inte kan använda den bakomliggande vyn förrän dialogen har stängts.

5.1.4 Hjälpklasser

Några klasser har skapats som hjälpklasser till andra kontroller. I detta avsnitt beskrivs några av dessa hjälpklasser.

ANIMERINGAR

Vissa kontroller animeras vid vissa operationer. För att förenkla hanteringen av animeringar och för att hålla koden ren, skapades ett mindre bibliotek för att hantera animeringar som baseras på en förflyttning mellan två punkter i planet. Biblioteket kan användas för att flytta en bild eller för att anropa en valfri kodsekvens varje gång en bildruta ska renderas.

Positionen vid renderingen av varje bildruta beräknas med hjälp av en integrerad andragradskurva. Effekten av att använda en sådan funktion är att animeringen startar och slutar mjukt, i motsats till det abrupta start och slut som en linjär funktion skulle ha resulterat i.

SCROLLABLECONTROL

Flera av de kontroller som används i programmet är scrollbara, det vill säga att om hela innehållet inte ryms i den synliga arean, kan användaren skjuta på innehållet i höjd- och sidled. Ett alternativ för dessa kontroller är att låta dem ärva från kontrollen `ScrollableControl` i .NET Framework. Dock påträffades ett problem med denna kontroll: om kontrollens automatiska scrollhantering är

aktiverad går det inte att dölja rullningslisten som automatiskt målas ut av `ScrollableControl` (Microsoft, 2008). Flera försök har gjorts enligt (MSDN Forums, 2010) att kringgå denna begränsning utan framgång. Därför beslutades att en motsvarande kontroll skulle utvecklas för programmet.

Den egenutvecklade klassen `ScrollableControl` saknar rullningslist och måste därför kunna användas på annat vis. I detta fall kan man scrolla på två sätt, dels med hjälp av rullhjulet på en mus (fungerar både vertikalt genom att rulla hjulet och horisontellt genom att luta på hjulet på möss som stödjer det), dels genom att använda fingret för att dra innehållet åt något håll (om programmet körs på en enhet med pekskärm och Windows 7 eller senare). Kontrollen använder Windows API för att hantera pekrörelser.

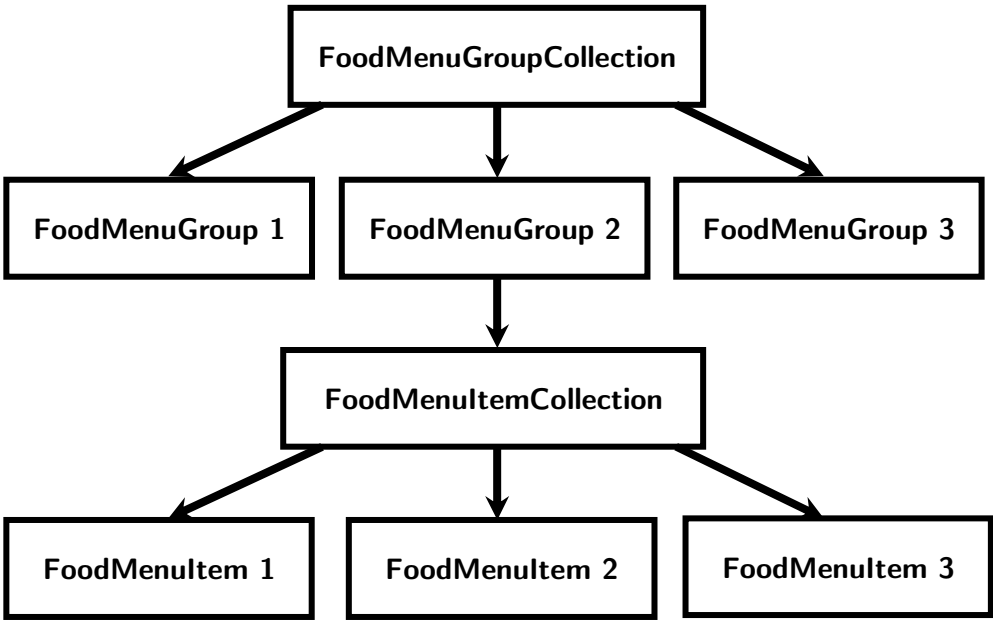
5.1.5 Kontroller

Varje komponent i det grafiska användargränssnittet har implementerats som en egen kontroll. Kontrollens ansvarsområden är layouten av kontrollens innehåll och att delegera renderingen till respektive stil (gås igenom i sektion 5.1.6). Det är även kontrollen som sköter dess funktion.

I denna sektion presenteras några av de kontroller som har implementerats, däribland menyn, som presenterar alla artiklar i kassasystemet och beställningskontrollen, som används för att redigera en beställning.

MENY

Kontrollen `FoodMenu` används för att presentera de artiklar som finns inlagda i kassasystemet. Artiklarna är indelade efter de kategorier som fås genom kassasystemets API.



Figur 5.4. Menystruktur.

Innehållet i menyn specificeras med hjälp av de två hjälpklasserna `FoodMenuItem` och `FoodMenuGroup` (se Figur 5.4). En `FoodMenuGroup` representerar en kategori i kassasystemet och innehåller information om kategorins rubrik och alla artiklar (av typen `FoodMenuItem`) som kategorin innehåller. Gruppen håller också reda på huruvida den har skjutits ihop. Artiklarna representeras av klassen `FoodMenuItem`, som innehåller information om artikelns titel och om artikeln har ytterligare information.

Kontrollen ritas ut manuellt, och baseras inte på någon av de färdiga kontrollerna i .NET Framework. För renderingen används ett antal renderare från den aktuella stilen (se avsnitt 5.1.6) som är specialiserade på ett visst visuellt element – i det här fallet ett listelement, en infoknapp och en flik. Eftersom alla dessa element renderas sekventiellt uppifrån och ned beräknas positionen enkelt genom att addera elementets höjd till en pekare som pekar på den position där nästa element kommer att placeras.

Då användaren väljer att skjuta ihop en grupp i menyn kommer en animering att spelas upp som visuellt skjuter ihop gruppen. För att animeringen ska spelas upp mjukt renderas hela den area som ska skjutas upp (eller ned om användaren expanderar gruppen) till en buffrad bild. Denna bild målas sedan ut

30 gånger per sekund i olika positioner tills animeringen är avslutad. Animeringen sker med hjälp av tidigare beskrivet animeringsbibliotek.

BESTÄLLNING

Kontrollen som har hand om beställningar heter `OrderControl`. Den ärver från `ScrollableControl` och är en del av menyvyn. Dess jobb är att visa och möjliggöra ändringar i den nuvarande beställningen för ett visst bord.

Kontrollen håller reda på vilken beställning som är aktuell och vilken delbeställning som är markerad. Innehållet i den aktuella beställningen hämtas från modellen.

Delbeställningar och artiklar har olika delar som ska målas ut (se Figur 4.3). För att alla olika delar ska hamna på rätt plats räknas det ut rektanglar för var och en av dem. En rektangel håller reda på delens storlek och placering. När delen ska målas ut tar en renderare hand om det och fyller rektangeln med rätt innehåll. Även om en renderare får en rektangel behöver inte innehållet som den fyller ut den med vara rektangulärt; det är upp till renderaren att bestämma innehållets form, till exempel om den kommer vara av rund, eller fyrkantig form eller om det ska vara text.

Metoderna som kontrollen består av kan delas in i tre olika grupper: en grupp som räknar ut storleken på de olika rektanglarna, en grupp som håller reda på rektanglarnas olika placeringar och en grupp som tar hand om tryck och kollar vilken rektangel som träffats.

Rektanglarnas storlek är kopplad till skärmens pixeltäthet samt kontrollens storlek och innehåll. Metoderna för att räkna ut rektanglars storlek används i de övriga metodgrupperna.

Placeringen av rektanglar utgår från det övre vänstra hörnet och placerar rektanglarna relativt varandra. För att spara prestanda kontrolleras att endast delar som helt eller delvis syns på skärmen renderas och om allt som syns på skärmen har renderats går den inte vidare med nästkommande delar. Kontrollen börjar med första delbeställningen. Om delbeställningens rektangel har någon del som kommer att synas på skärmen skickas delbeställningens olika delar till renderare som målar ut dem. Nästa delbeställning kommer att hamna nedanför

den första och med den sker samma koll på dess rektangel om den ska målas ut. På samma sätt gås alla delbeställningar igenom.

När tryck kontrolleras går det till på liknande sätt som vid placeringen av rektanglarna, men i stället för att måla ut delarna undersöks delens rektangel för att kontrollera om trycket är inom den.

SKÄRMTANGENTBORD

Eftersom det inte finns något tangentbord till plattan sker all inmatning genom tryck på skärmen. Alla textrutor använder därför ett skärmtangentbord för att underlätta textskrivning. Detta sker genom användandet av *Tablet PC Input Panel* som är Windows inbyggda skärmtangentbord (se Figur 5.5), vilket har visat sig medföra vissa problem.

Figur 5.5. Skärmtangentbordet i Windows 7.

Normal användning av skärmtangentbordet går till så att när användaren trycker på en textruta kommer en ikon upp som behöver tryckas på för att ta fram tangentbordet. Det placeras då på en position relativt textrutan och har samma storlek som när det senast användes (Microsoft, 2011d).

Skärmtangentbordet bör i programmet användas på ett annorlunda sätt; när användaren trycker på en textruta ska tangentbordet automatisk komma fram och placeras kant i kant med skärmens nedre del, detta för att minska antalet tryckningar som behövs och för att det inte ska täcka vissa delar av programmet. Detta betyder att både placering och storlek på tangentbordet behöver kunna

ändras, så att det hamnar på rätt plats och med rätt storlek. Att använda tangentbordet på detta sätt medför vissa problem, se avsnitt 6.2.5.

AVDELARE

Avdelaren **`ViewSplitter`** ärver från .NET Frameworks **`SplitContrainer`** och finns i både menyvyn och beställningsvyn. Förutom utseendemässigt finns det två stora skillnader mellan **`ViewSplitter`** och basklassen: avdelaren kan justeras efter plattans orientering och den har en maximera-, en minimera- och en återställknapp.

Avdelarens orientering bestäms av åt vilket håll man håller plattan. När plattan är liggande ska vyns två olika delar vara placerade horisontellt bredvid varandra och när plattan är stående ska delarna vara placerade vertikalt över varandra (se Figur 4.3). Det är avdelarens jobb att se till att denna placering blir korrekt. Avdelaren är den enda kontrollen som ser olika ut beroende på plattans orientering.

Det syns alltid två knappar och det är olika knappar beroende på vilket tillstånd avdelaren befinner sig i. Det finns tre olika tillstånd för avdelaren:

- NORMALT TILLSTÅND Avdelaren kan röras fritt fram och tillbaka och maximera- och minimeraknapparna syns.
- MAXIMERAT TILLSTÅND Den första delen är maximerad och återställ- och minimeraknapparna syns.
- MINIMERAT TILLSTÅND Den första delen är minimerad och maximera- och återställknapparna syns.

INDIKATORER

Det finns två indikatorer: **`IndicatorBattery`** för batteristatus och **`Indicator-Network`** för nätverksstatus, där båda ärver från **`ToolStripItem`**. Det är kontrollerna själva som renderar symbolerna, detta på grund av att de inte påverkas av den valda stilen utan renderas likadant oavsett stil.

Batteriindikatorn läser av batteristatusen direkt via .NET Framework och renderar en bild av ett batteri som visar i procent hur mycket batteri plattan har kvar.

Nätverksindikatorn hämtar nätverksstatusen från Windows API, genom att gå igenom alla tillgängliga trådlösa nätverksanslutningar och kontrollera statusen på den första som är ansluten till ett nätverk. Utifrån detta renderas fem staplar som är antingen gråa eller fyllda, där antalet fyllda beror på statusen. Om ingen anslutning hittas renderas ett rött kryss över symbolen.

5.1.6 Stilar

Ett krav från uppdragsgivaren var att det ska vara enkelt att ändra utseendet på programmet. För att göra detta möjligt har ytterligare två lager införts i koden för användargränssnittet: ett som ansvarar för att rita upp enskilda visuella element, och ett som ansvarar för att leverera information om färger och typsnitt som elementen ska använda sig av. I detta avsnitt kommer det förstnämnda att presenteras. För enkelhetens skull refererar ordet stil till en uppsättning sådana element.

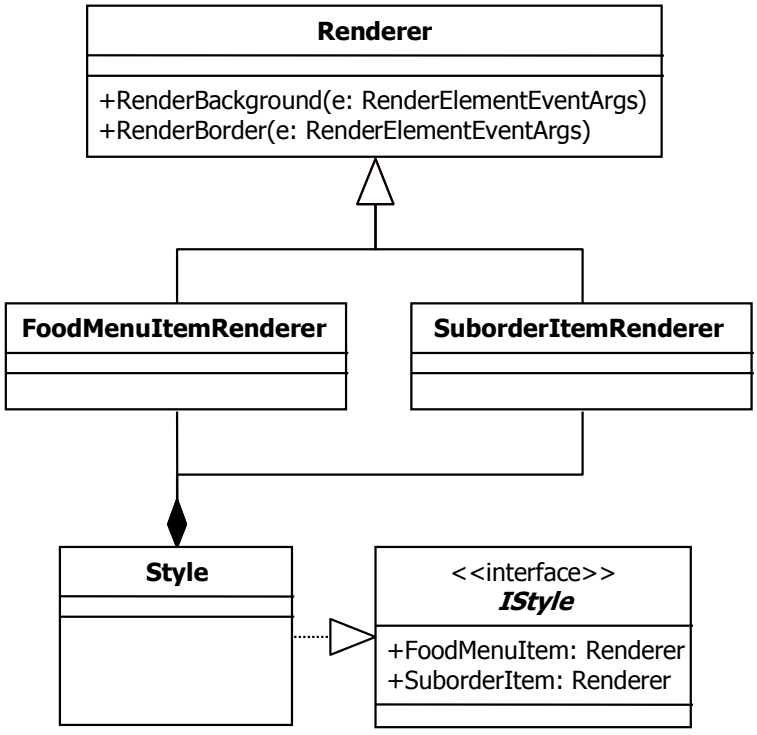
En stil består av ett antal renderare som alla är specialiserade för att rita ut en viss typ av visuella element med en likartad stil, därav benämningen. Varje stil måste bland annat tillhandahålla renderare för att rita ut flikar, olika typer av knappar och avdelare. Alla renderare har en metod för att rita ut ett elements bakgrund och en metod för att rita ut elementets kantlinje.

Fördelen med denna struktur är dels att det är enkelt att ändra i renderingen av ett visst visuellt element, då alla renderare är samlade på samma ställe, dels att programmet är förberedd för att kunna innehålla flera stilar, vilket skulle kunna presenteras för kunden som en möjlighet att själv ändra stil på programmet.

Implementeringstekniskt sett består varje stil av en egen klass som implementerar gränssnittet `IStyle` (se Figur 5.6). Gränssnittet anger vilka typer av renderare som alla stilklasser måste innehålla. Alla renderare ärver från basklassen `Renderer`, och returneras även som sådana i stilen, vilket gör det möjligt för en stil att returnera samma renderare för flera olika typer av element.

I standardstilen används exempelvis samma renderare för statusradens och verktygsradens bakgrund.

Renderarnas basklass `Renderer` anger att alla renderare måste ha minst två metoder: `RenderBackground` för att rita ut elementets bakgrund och `RenderBorder` för att rita ut elementets kantlinje (se Figur 5.6). Notera dock att metoderna inte nödvändigtvis behöver göra något – om elementet i en viss stil saknar kantlinje behöver `RenderBorder` inte utföra något arbete. Båda metoderna kräver en parameter av typen `RenderElementEventArgs`. Denna klass innehåller bland annat information om inom vilket område som elementet ska ritas ut, vilken färg som ska användas (om inte standardfärgen används) och elementets orientering i de fall elementet kan ritas i olika riktningar (avdelaren kan exempelvis ritas såväl horisontell som vertikal).



Figur 5.6. Förenklat klassdiagram över stilar och renderare.

Det kan tyckas onödigt att förmedla parametrarna till renderaren i ett eget objekt, men det finns en viktig anledning till detta. Anledningen grundar sig i att koden bör vara enkel att bygga ut och att det bör vara möjligt för en tredjepart att i framtiden skapa stilar som ligger i separata filer. Om `Render-`

Background och **RenderBorder** hade haft ett antal hårdkodade formella parametrar, hade ett problem varit att om en parameter skulle läggas till, hade ingen av de gamla renderarna, som potentiellt sett kan vara utom vår kontroll, vara kompatibla med programmet förrän utvecklaren av renderarna lagt till denna parameter till renderarnas metodsignaturer. Även för egenutvecklade renderare hade det varit omständigt att lägga till parametrar, då alla renderare måste redigeras manuellt för att passa den nya signaturen.

Ytterligare en annan nackdel som ett antal hårdkodade formella parametrar medför är att alla renderare är begränsade till en och samma uppsättning parametrar.

Genom att förmedla alla parametrar genom objekt av typen **RenderElementEventArgs** löses dessa problem. I det första fallet behöver inga renderare ändra sina metodsignaturer, då ytterligare parametrar kan läggas i klassen **RenderElementEventArgs** istället för i renderarnas metodsignaturer. Även om klassen **RenderElementEventArgs** utökas, kan renderare som ligger i separata filer fortfarande användas, då dess funktion inte påverkas av att klassen **RenderElementEventArgs** har fler egenskaper än vid kompileringstillfället.

Renderare är med denna lösning inte längre begränsade till en uppsättning hårdkodade parametrar, utan om en specifik renderare behöver en specifik parameter, kan en specialiserad klass som utökar klassen **RenderElementEventArgs** skapas, där ytterligare parametrar kan läggas till. I och med polymorfism kan instanser av den utökade klassen skickas till alla typer av renderare, utan att hänsyn behöver tas till om renderaren har stöd för de utökade parametrarna.

Viktigt att notera är att stilarna med tillhörande renderare endast bestämmer utformningen av de visuella elementen – information om färger och typsnitt levereras av underliggande färg- och typsnittstabeller och elementens funktion bestäms av kontrollerna.

5.1.7 Tabeller

Vid utvecklingen av prototypen gjordes bedömningen att utformningen av en stil med tillhörande renderare endast behöver kunna göras av programmerare. Val

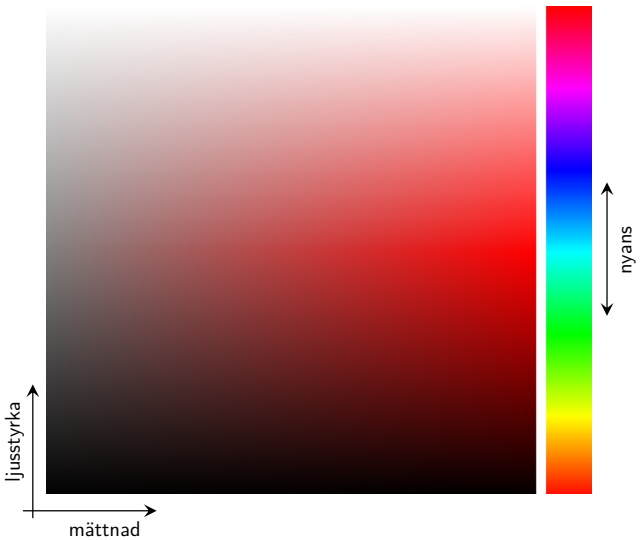
av färg och typsnitt är dock en uppgift som till viss del behöver kunna göras av kunden själv. Därför skapades ytterligare ett lager som innehåller färg- och typsnittstabeller. Dessa kommer att presenteras i denna sektion.

Det finns en signifikant skillnad mellan färg- och typsnittstabellerna som ska noteras innan detaljer om dessa presenteras. Medan färgtabellerna generellt används av stilarna, används typsnittstabellerna av kontrollerna i lagret ovanför stilarna. Anledningen är att det är kontrollen som ansvarar för den övergripande layouten som beror på innehållets, det vill säga textens, storlek.

Målsättningen med färg- och typsnittstabellerna är att de ska vara oberoende av stilarna – alla stilar ska dock ta hänsyn till den information som tabellerna levererar.

Implementeringen av tabellerna är strukturerad på liknande sätt som stilarna. Gällande färgtabellen finns gränssnittet `IColorTable` som definierar vilka färger alla färgtabeller ska innehålla. Det är upp till färgtabellens implementation att bestämma hur färgerna bestäms.

HUECOLORTABLE



Figur 5.7. Färgrepresentation enligt HSL.

Den färgtabellsimplementering som används som standard är `HueColorTable`, som beräknar alla färger med utgångspunkt i en grundfärg. Tanken med detta är att oavsett vilken färg kunden väljer som grundfärg, ska ingen färgkombination som gör gränssnittet oläsligt kunna förekomma.

HueColorTable bygger på färgrepresentationsmodellen HSL. För att representera en färg i HSL anges tre komponenter: nyans (färgens kulör), mättnad (hur intensiv färgen är) och ljusstyrka (anges på en skala från svart till vitt) (Wikipedia, 2011e). Se Figur 5.7.

Utifrån den grundfärg som kunden kan välja beräknas färgens nyans. Alla färger som används i gränssnittet kommer sedan att ha samma nyans som grundfärgen – endast mättnaden och ljusstyrkan varieras för att åstadkomma de färger som behövs.

TYPSNITTSTABELL

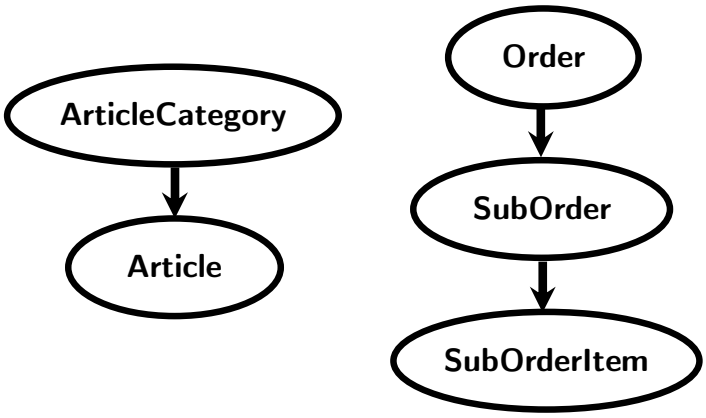
Liknande färgtabellerna finns tabeller som anger vilka typsnitt och typstorlekar som ska användas i gränssnittet. Typsnittstabellen som används som standard använder nästan enbart typsnittet Tahoma i olika storlekar.

5.2 Modellen

Modellen innehåller en representation av de entiteter som existerar i kassasystemet. Några av de mest centrala entiteterna är `Order`, `SubOrder`, `SubOrderItem`, `Article` och `ArticleCategory` (se Figur 5.8). Dessa innehåller information om existerande artiklar, alla aktuella beställningar och dess aktuella statusar i kassasystemet samt förhållanden mellan entiteter.

Den lokala modellen är en cachad representation av den information som tros existera i kassasystemet vid varje tidpunkt. Eftersom såväl datan i kassasystemet som datan i den lokala modellen kan ändras individuellt krävs någon form av synkronisering. Denna synkronisering kommer att beskrivas i nästa avsnitt.

Alla entiteter i modellen är instanser av klasser som alla ärver från `ModelEntity` och implementerar gränssnittet `IModelEntity`. Varje entitet är själv ansvarig för att hålla reda på vilka attribut som har förändrats och således måste skickas till kassasystemet. Denna funktionalitet tillhandahålls huvudsakligen av basklassen `ModelEntity` – entitetsklasserna behöver endast delegera åtkomsten till sina attribut via `ModelEntity`.

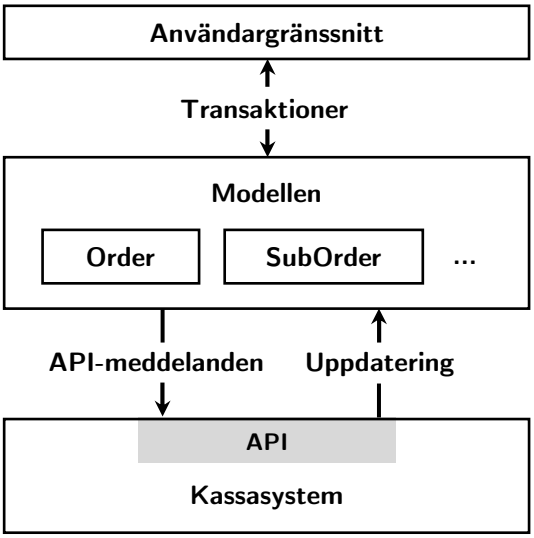


Figur 5.8. Entiteterna i modellen representerad som en riktad graf.

Till detta kommer relationer till andra entiteter. En beställning kan exempelvis ha en 1-N-relation till ett godtyckligt antal delbeställningar, som i sin tur också kan ha en 1-N-relation till ett antal artiklar. 1-N-relationer modelleras i prototypen med hjälp av listor som innehåller de entiteter som är refererade. Dessa listor ärvs från basklassen `ModelEntityCollection`, som är ansvarig för att notera vilka entiteter som läggs till, ändras och tas bort. Detta ansvar har delegerats till klassen `TransactionContext` som kommer att presenteras i nästa avsnitt.

Några entiteter, till exempel `Order`, som representerar en beställning, ligger högst i sin entitetshierarki och saknar därmed en förälder. Därför krävs ytterligare en nivå ovanför modellen som håller reda på vilka toppnivåentiteter som ingår i den lokala avbildningen av kassasystemets innehåll. Denna nivå utgörs av en klass som implementerar gränssnittet `IStorage`. `IStorage` innehåller listor på aktuella beställningar och artiklar. `IStorage` kommer även den att presenteras närmare i nästa avsnitt.

5.3 Synkronisering



Figur 5.9. Översikt för synkroniseringen.

Som tidigare nämnts finns möjligheten dels att den lokala modellen ändras, vilket innebär att kassasystemet måste informeras om ändringen, dels att kassasystemets databas ändras exempelvis till följd av en begäran från en annan klient, vilket innebär att den lokala modellen måste få uppdateringar från kassasystemet (se Figur 5.9 för en översikt av synkroniseringen).

5.3.1 API

All kommunikation med kassasystemet sker genom kassasystemets API, som låter klienterna ställa frågor om kassasystemets aktuella tillstånd och skicka begäran till kassasystemet. Kassasystemets API är utformat som en SOAP-tjänst, där alla meddelanden skickas över HTTP. SOAP är enligt standardiseringsorganisationen W3C ett »protokoll avsett för att utbyta strukturerad information i en decentraliserad, distribuerad miljö [egen översättning]« där varje meddelande är i XML-format (W3C, 2007). Användningen av SOAP-tjänster förenklas i Visual Studio, där användaren kan lägga till referenser till en SOAP-tjänst. Visual Studio genererar då automatiskt proxyklasser för tjänsten. I programkoden använder man sedan dessa proxyklasser utan att behöva tänka på vilken typ av kommunikation som sker mellan klient och server (Microsoft,

2011e). Därför läggs ingen mer vikt vid att närmare beskriva SOAP-protokollet i denna uppsats.

5.3.2 Synkroniserare

Komponenten `Synchronizer` är ansvarig för att hålla den lokala modellen och informationen i kassasystemet synkroniserade. Synkroniseringsprocessen är som tidigare konstaterat beroende av kassasystemets API, som kontaktas via en nätverksanslutning, något som inducerar en fördröjning i processen. För att undvika att programmets grafiska användargränssnitt blockeras (»hänger sig«) då synkroniseringsprocessen pågår har denna flyttats ut till en separat tråd. Synkroniseringstråden ligger vanligtvis vilande, men vaknar med bestämda tidsintervall för att hämta uppdateringar från kassasystemet, samt när UI-tråden (tråden som användargränssnittet körs i) har gjort en ändring i modellen som behöver skickas till kassasystemet.

Kommunikationen mellan trådarna sker med hjälp av klassen `EventWaitHandle` i .NET Framework som kan användas för att skicka signaler mellan trådar (Microsoft, 2011b). Då synkroniseringstråden har slutfört sina aktuella uppgifter anropas en metod i `EventWaitHandle` som blockerar tråden tills antingen UI-tråden skickar en signal till `EventWaitHandle` eller en bestämd tidsgräns uppnås (tidsgränsen bestäms utifrån när nästa automatiska uppdatering ska ske).

5.3.3 Strategier

Under designen av synkroniseringslagret i prototypen undersöktes och testades olika strategier för att hålla modellen uppdaterad. Strategierna som undersökts kan grovt placeras på en skala mellan två ytterligheter:

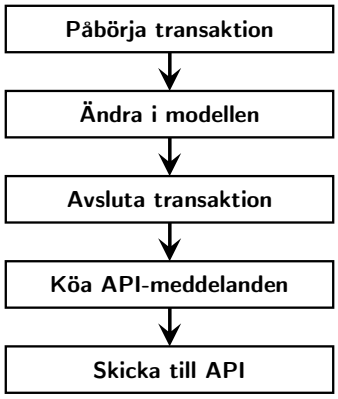
- Varje gång en ändring sker i modellen skapas ett diffmeddelande som innehåller skillnaden mellan tillståndet före och tillståndet efter ändringen. Meddelandet läggs i en meddelandekö som en synkroniserare läser ifrån och skickar till kassasystemets API. Nackdelen är att meddelandekön kan fyllas med redundanta meddelanden (exempelvis om en användare ändrar på

beställning X och sedan tar bort den, vilket resulterar i ett onödigt meddelande, nämligen det som anger att beställning X har ändrats). Fördelen är att datatrafiken minskas då endast ändringarna skickas till kassasystemet.

- En synkroniserare laddar med jämna mellanrum ned all information om artiklar och aktuella beställningar från kassasystemet. Det nedladdade innehållet jämförs med modellen. Svårigheten med denna strategi är att fastställa om det är informationen från kassasystemet eller från den lokala modellen som är korrekt om en skillnad hittas.

I prototypen används en kombination av ovanstående strategier. Vid replikeringen av ändringar gjorda i kassasystemet till den lokala modellen används en strategi som liknar den andra ytterligheten. Vid replikering i motsatt riktning används en strategi som mestadels liknar den förstnämnda ytterligheten.

5.3.4 Skicka ändringar till kassasystemet



Figur 5.10. En ändrings väg till kassasystemet.

Användargränssnittet, som körs i UI-tråden, kommunicerar inte direkt med synkroniseringstråden, utan denna kommunikation går genom modellen och något som i prototypen kallas transaktioner (se Figur 5.10). Transaktionen säkerställer att inga andra trådar ändrar i modellen samtidigt och ansvarar även för att samla ihop ändringarna som sker under transaktionens livslängd för att sedan skapa meddelanden som kan skickas till kassasystemet. Ett exempel på ett sådant meddelande är `EditOrderApiMessage` som skickas för att underrätta kassasystemet om en ändring i en beställning.

Transaktionerna som sker i prototypen liknar databastransaktioner i flera avseenden. Här är några viktiga egenskaper hos en transaktion i prototypen:

- Modellen kräver att den anropande tråden befinner sig i en transaktion för att få ändra data i modellen. Om en tråd försöker ändra i modellen utan att först skapa en transaktion kastas ett `NotInATransactionException`.
- Om en tråd försöker skapa en transaktion samtidigt som en annan tråd befinner sig i en transaktion blockeras den förstnämnda tråden tills den andra tråden lämnar transaktionen. Denna och ovanstående punkt motverkar att två eller flera trådar samtidigt arbetar med samma data.
- Då en transaktion har skapats kan objekt i modellen hanteras utan vidare hänsyn till att den faktiskt avbildar motsvarande information i kassasystemet.
- Inga ändringar som sker i modellen under en transaktions livslängd skickas till kassasystemet förrän transaktionen är avslutad. Detta liknar en *commit* i en databastransaktion, som garanterar att ändringarna gjorda under transaktionen är sparade (Asplund, 2010). Prototypen lämnar dock inga sådana garantier, utan ändringarna läggs istället i en kö som synkroniseringstråden hämtar meddelanden ifrån vid nästa synkroniseringstillfälle.
- Prototypen saknar möjlighet att rulla tillbaka en transaktion, vilket normalt kan göras i en databastransaktion. (Asplund, 2010)
- Till varje transaktion hör en `TransactionContext` som noterar alla förändringar som görs i modellen under transaktionens livslängd. När tråden lämnar transaktionen kommer ändringarna i `TransactionContext` att läggas i synkroniserarens meddelandekö och skickas till kassasystemets API vid nästa synkroniseringstillfälle. Detta reducerar antalet redundanta meddelanden, eftersom man vid transaktionens slut vet att om en entitet tagits bort, behöver inga övriga ändringar som avser den entiteten skickas till kassasystemet.
- Nästlade transaktioner är tillåtna förutsatt att de skapas för samma synkroniserare. De nästlade transaktionerna körs i samma `TransactionContext` som yttertransaktionen.

Det finns vissa problem som har beaktas under utvecklingen av synkroniseringen:

- När ett objekt tas bort från kassasystemet är det raderat. Livslängden på ett objekt i modellen är dock längre. Om en beställning tas bort från modellen kan orderobjektet återanvändas i andra sammanhang, eftersom objektet inte tas bort förrän det inte längre finns några referenser till objektet (C# har en skräpsamlare (Microsoft, 2011c)). Därför har varje entitet som ingår i den lokala avbildningen av kassasystemet en lokal identifierare som genereras då entiteten läggs till i den lokala avbildningen och är giltigt till och med att entiteten tas bort från den lokala avbildningen.
- Kassasystemet använder heltal, hädanefter kallade serveridentifierare, som primärnyckel för varje objekt. I modellen används primärt objektreferenser och sekundärt lokala identifierare. `IStorage` håller reda på mappningar mellan lokala identifierare och serveridentifierare.
- Mappningarna läggs till då en lägg till-operation utförs mot kassasystemets API, och tas bort då en ta bort-operation utförs. Notera att mappningarna inte har samma livslängd som entiteterna i modellen, eftersom alla meddelanden som ska skickas till kassasystemet köas tills en nätverksanslutning finns tillgänglig.
- Det är möjligt att en entitet ändras i modellen som ännu inte finns i kassasystemet, och därför saknar ett serveridentifierare. Detta kan ske om ett objekt ändras då ett lägg till-meddelande för samma objekt ligger köat i synkroniserarens meddelandekö. Ett komplett API-meddelande kan således inte sparas i meddelandekön, utan istället sparas ett partiellt meddelande tillsammans med entitetens lokala identifierare. Vid synkroniseringstillfället slås det lokala identifieraren upp i `IStorage`, som då ska innehålla en mappning från det lokala identifieraren till en serveridentifierare.
- På klienten är det möjligt att exempelvis skapa en delbeställning först varefter en beställning skapas som innehåller delbeställningen. Denna ordning på operationerna tillåts inte av kassasystemets API, som kräver att föräldraobjektet existerar för att ett barnobjekt ska kunna skapas. För att lösa detta ritas entitetshierarkin upp som en riktad graf (se Figur 5.8). Entiteterna sorteras därefter topologiskt, vilket innebär att de entiteter som

saknar föräldrar (har ingraden 0) väljs ut först och tas bort från grafen, varefter processen upprepas tills grafen är tom (Weiss, 2006). Detta är ordningen som alla lägg till- och ändringsoperationer som sker i en transaktion kommer att utföras i då de skickas till kassasystemet. Borttagningsoperationer sker därefter i omvänd ordning.

5.3.5 Hämta uppdateringar från kassasystem

Hittills har synkroniseringen av ändringar på klientsidan presenteras. Kvar är delen av synkroniseringen som hämtar ändringar gjorda på andra klienter eller av kassasystemet och uppdaterar den lokala modellen med denna information. I detta avsnitt presenteras algoritmen som används för att spegla dessa ändringar.

VARIABLER OCH FUNKTIONER:

S : tupel innehållande alla element av aktuell typ som för närvarande existerar i kassasystemet

T : tupel innehållande alla element av aktuell typ som för närvarande finns lagrade i den lokala modellen, där alla nya entiteter ligger sist

L : $T \longrightarrow \mathbb{N}$

$T \longmapsto i$ sådant att t_i är sista t i T som vid något tillfälle har existerat i S

ALGORITM:

1. $\forall i \in [1, |S|]$:
   Hitta ett j sådant att $s_i \Leftrightarrow t_j$ $(s_i \in S,\ t_j \in T,\ j \in [i,\ L(T)]\ )$
   a. Om ett sådant j existerar:
      i. Uppdatera t_j efter s_i
      ii. Byt plats på t_i och t_j om $i \neq j$
   b. Om ett sådant j inte existerar:
      i. Sätt in $t \Leftrightarrow s_i$ på index i i T
2. Ta bort $t_i \in T$, där $i \in (|S|\,,\ L(T)]$

Algoritmens huvudfunktion är att spegla ändringar som hittas i datan returnerad från kassasystemet i den lokala modellen. Om en ändring hittas har informationen från kassasystemet alltid företräde framför informationen i den lokala modellen. Detta kan göras säkert då ändringar gjorda i modellen alltid måste skickas till kassasystemet innan en uppdatering åt andra hållet kan ske.

5.3.6 Noteringar

När personalen tar en beställning bör det vara möjligt att lägga till en notering för varje artikel i beställningen, till exempel om biffen ska vara blodig eller om brödet ska vara glutenfritt. I kassasystemets API finns dock ingen möjlighet att ange en notering på artikelnivå utan endast på order- och subordernivå. För att lösa detta problem används ett specialutformat format för att lagra noteringar för varje artikel i orderns noteringsfält. Några kriterier vid utformningen av formatet var att det skulle vara lätt att läsa för en människa (det är den här noteringen som köket ser på bongen), och att det bör gå att ändra i noteringen genom kassasystemet utan att det stör programmet på plattorna.

Så här är formatet konstruerat:

```
[Artikelnamn] ([Antal])
[Notering]
#[Subordernr]*[Artikelnr]
```

Den första raden anger sådant som kökspersonalen måste veta för att avgöra vad noteringen avser, och inget som prototypen kommer att försöka tolka. Den sista raden, som börjar med ett brädgårdstecken, anger för kassasystemet vilken artikelrad i beställningen som noteringen avser. Allt däremellan är textinnehållet i noteringen. Noteringen får innehålla flera rader, men om en notering innehåller ett brädgårdstecken, tas detta tecken bort från noteringen för att den inte ska misstolkas som den rad som anger delbeställning- och artikelnummer.

Exempel på en ordernotering där två av artikelraderna har en notering:

```
Oxfilé med potatis och kantarellsås (2 st)
Råstekt
#2354335*56332656

Oxfilé med potatis och kantarellsås (1 st)
Glutenfritt
#2354332*56332657
```

5.4 Sammanfattning

Implementeringen kan delas upp i tre delar: presentation, modell och synkronisering. Presentationen är implementeringen av användargränssnittet och består av flera lager – formulär, vyer, kontroller, renderare och tabeller – där de övre lagren hanterar översiktliga uppgifter och kommunikation med modellen och de nedre lagren har mer hand om visuella uppgifter. Modellen innehåller artiklar och beställningar som bygger upp en lokal kopia av informationen i kassasystemet. Synkroniseringen skickar och tar emot information mellan modellen och kassasystemet för att hålla dem båda informerade om ändringar som sker.

Kapitel 6

Utvärdering

I detta kapitel utvärderas de tekniska aspekterna av projektet; de verktyg som använts presenteras och bedöms och de tekniska problem som uppkommit under projektet sammanfattas. Dessutom listas sådant som är kvar att implementera.

6.1 Utvecklingsplattform

Som primär utvecklingsmiljö har Visual Studio 2010 valts tillsammans med C# och .NET Framework 4.0. Utöver Visual Studio har ett antal verktyg och insticksprogram såsom AnkhSVN och Inkscape använts under projektets gång. I detta avsnitt utvärderas språket C#, ramverket .NET Framework, utvecklingsmiljön Visual Studio med tillhörande insticksprogram och verktyg.

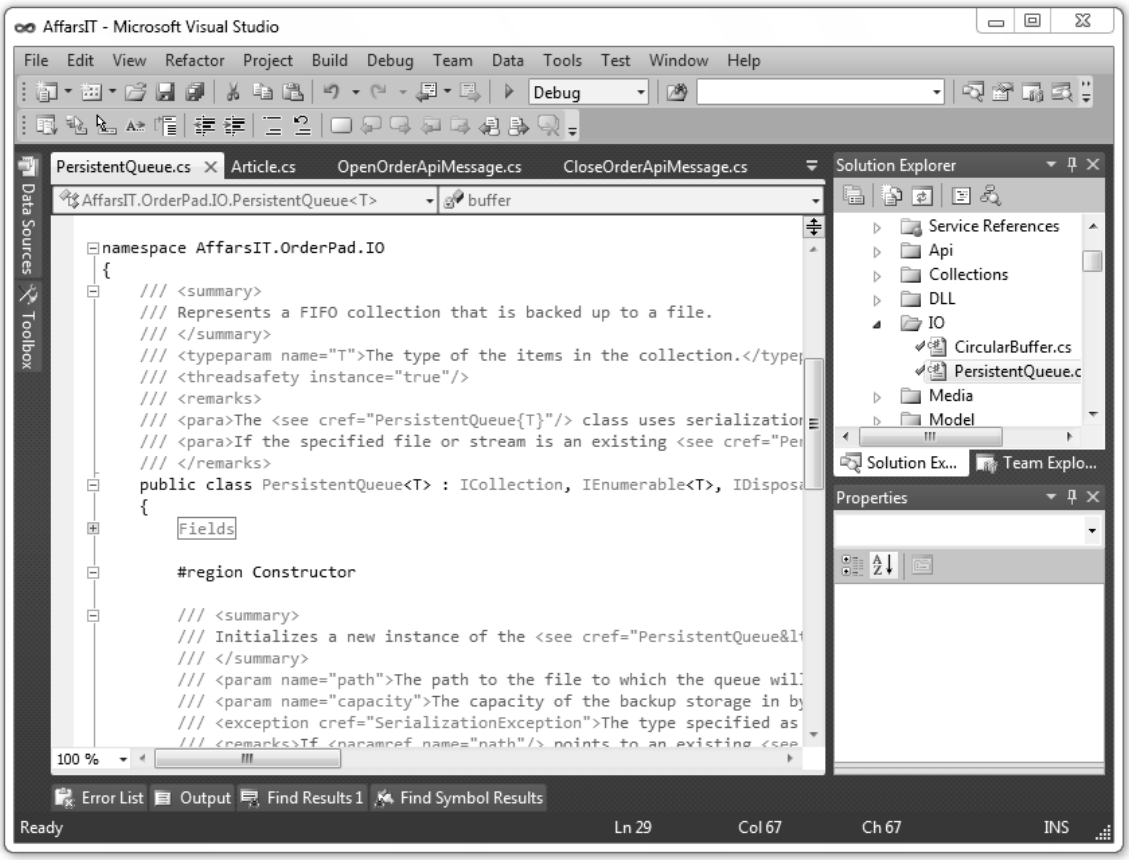
6.1.1 C# och .NET Framework

Prototypen har till fullo utvecklats i C# för .NET Framework 4.0. För konstruktionen av användargränssnitt har Windows Forms använts tillsammans med GDI+ för rendering av anpassade kontroller. Utvecklingen av C# har flutit på bra och vi gillar språket. Att valet föll på Windows Forms istället för det nyare WPF (Windows Presentation Foundation) var huvudsakligen för att förenkla för de utvecklare som tar över utvecklingen till sommaren som inte är bekanta med WPF. Det hade dock funnits flera fördelar med att använda WPF,

bland annat integrerat stöd för pekgester (Rodriguez, 2009). Pete Faraday och Brad Becker på Microsoft rekommenderar WPF för produkter som innehåller ett komplext textinnehåll och som har ett anpassat utseende (Faraday & Becker, 2007), vilket passar bra in på prototypen.

6.1.2 Visual Studio

All utveckling av prototypen har skett i Microsofts utvecklingsmiljö Visual Studio (se Figur 6.1). Visual Studio förenklar utvecklingen med funktionen IntelliSense, som försöker förutspå vad man tänker skriva efter att man har skrivit några tecken på ett namn.



Figur 6.1. Visual Studio 2010.

Hanteringen av webbtjänster som använder SOAP är dessutom integrerad i Visual Studio, som automatiskt genererar proxyklasser som kan användas för att kommunicera med, i detta fall, kassasystemets API. Detta gör att API:t kan användas i princip som om det hade körts på samma maskin (Microsoft, 2011e).

Givetvis måste man fortfarande ta hänsyn till att alla anrop genom proxyklasserna lägger till en viss fördröjning eftersom de resulterar i anrop över nätverket.

Designern i Visual Studio är en funktion som normalt underlättar konstruktionen av användargränssnitt genom att man som utvecklare kan rita upp gränssnittet genom drag- och släpprincipen. Designern har förenklat utvecklingen även i detta fall, men har inte varit till lika stor hjälp som den normalt är, då flera av komponenterna renderas manuellt för att få fullständig kontroll över komponenternas utformning och funktion.

Visual Studios stöd för C# är välutvecklat och fungerar mycket bra i vår mening.

6.1.3 Kodhantering

För att hålla reda på källkoden har en Subversionserver använts för versionskontroll med insticksprogrammet AnkhSVN som klient. AnkhSVN integrerar sig i Visual Studio, där det automatiskt lägger nya projektfiler under versionskontroll. Som komplement användes TortoiseSVN, som integrerar sig i Windowsutforskaren. TortoiseSVN användes bara i sällsynta fall, bland annat då egenskaper skulle ändras på filer som inte inkluderats i projektet i Visual Studio.

```
/// <summary>
///     Stops the synchronizer thread.
/// </summary>
/// <param name="wait">
///     Indicates whether the calling thread should be blocked until the
///     synchronizer has terminated.
/// </param>
/// <exception cref="InvalidOperationException">
///     The synchronizer is not running.
/// </exception>
public void Stop(bool wait)
{
```

Figur 6.2. Exempel på dokumentation i Visual C#.

Insticksprogrammet GhostDoc har använts för att underlätta dokumentationen av koden. I Visual C# finns en XML-baserad konstruktion (Microsoft, 2011f) för att ange vad exempelvis en metod gör, vad den tar för parametrar och vad den returnerar i anslutning till subjektet som dokumenteras. Se Figur 6.2 för ett exempel på sådan dokumentation.

Vad GhostDoc gör är att generera skelettet för dokumentationen och fyller det med information som gissas utifrån metodens namn och parameternamn. Har man en metod som heter ClickButton gissar GhostDoc att dokumentationen blir »Clicks the button«. Den senare funktionaliteten har gett upphov till en hel del skratt under utvecklingen och programmet har huvudsakligen använts för att snabbt generera skelettet för dokumentationen.

6.1.4 Windows Installer

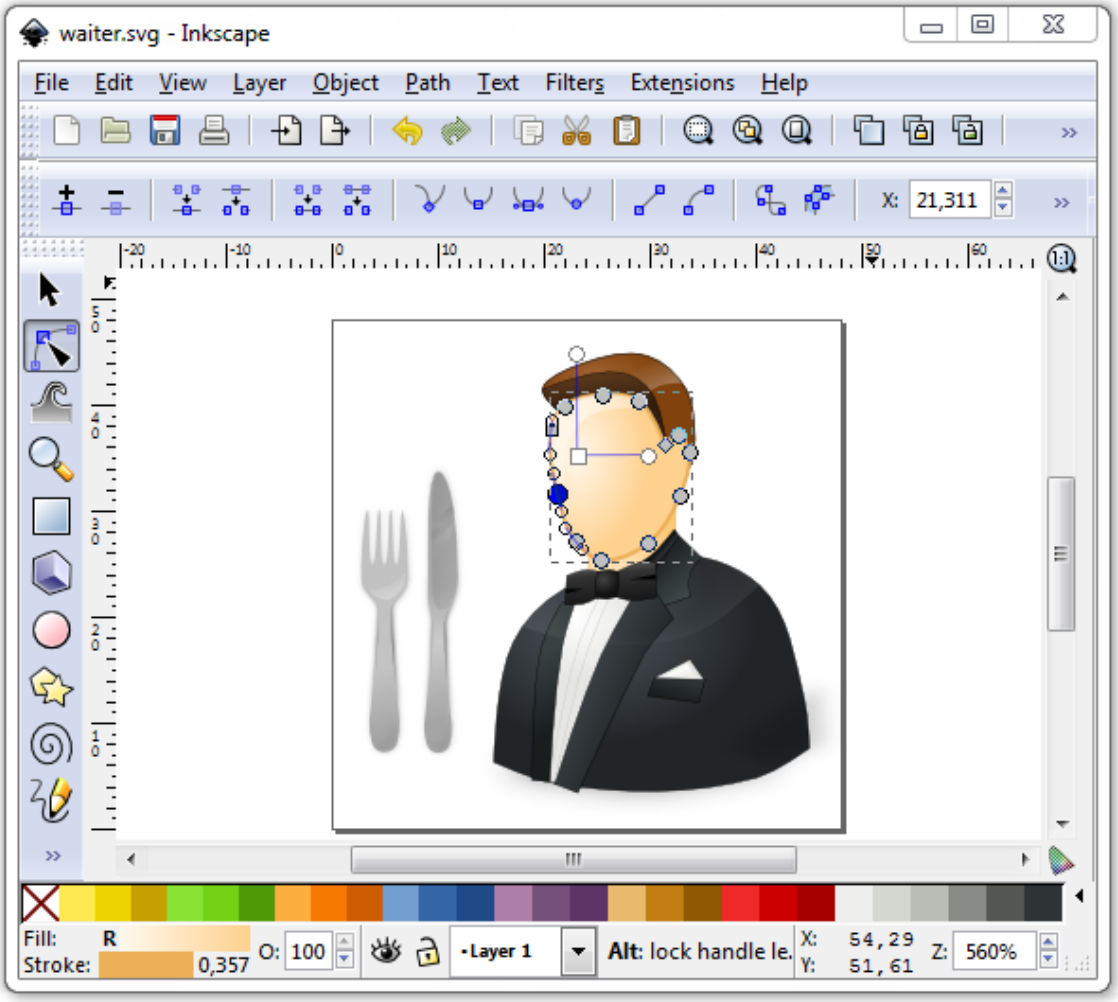
För att förenkla installationen av programmet på plattorna har projektet konfigurerats för automatisk generering av installationsfiler. För denna generering används WiX, Windows Installer XML, som är en open source-lösning utvecklad av Rob Mensching på Microsoft för att generera Windows Installer-paket (filer som slutar med .msi) utifrån XML-filer (Microsoft, 2007a). Som kuriosa kan nämnas att WiX var den första prototyp som Microsoft släppt som öppen källkod (Microsoft, 2007a). Fördelen med Windows Installer-paket är att de är felsäkra i den mening att installationerna sker i transaktioner som kan rullas tillbaka om ett fel uppstår under en installation och därmed lämna systemet i det tillstånd som det befann sig i innan installationen påbörjades (Microsoft, 2010).

6.1.5 SoapUI

För att testköra kassasystemets API har programmet SoapUI använts. SoapUI är ett program med öppen källkod för att skicka meddelanden till en SOAP-tjänst och granska meddelandet i retur. (Eviware, u.å.)

6.1.6 Inkscape

All mer avancerad vektorgrafik, såväl till programmet som till uppsatsen, har ritats i open source-programmet Inkscape (se Figur 6.3). Inkscape är ett alternativ till kommersiella verktyg för att rita vektorgrafik, såsom Adobe Illustrator. (Wikipedia, 2011c)



Figur 6.3. Inkscape.

Inkscape har fullständigt fyllt våra behov för ritandet, men har gett oss många gråa hår vid export till vissa filformat, däribland EMF, där alla former som innehåller gradienter verkar försvinna. Lösningen har dessvärre varit att undvika gradienter.

6.1.7 Microsoft Word

För rapportskrivningen har Microsoft Word använts. Microsoft Word valdes då vi är mest bekanta med det och att det håller reda på alla referenser som har använts automatiskt. Microsoft Word har dessutom använts för att skissera användargränssnittet.

6.2 Problem

Många dagar har gått till att lösa problem som uppstått. I detta avsnitt presenteras problemen kortfattat. I de fall problemet är mer omfattande har det beskrivits närmare i tidigare kapitel.

6.2.1 Android

Innan platturvalet var klart var siktet inställt på att använda en platta med Android, då Android verkar vara vanligast på nya surfplattor – den 2 maj 2011 var 62 procent av surfplattorna listade på Prisjakt Androidbaserade, jämfört med 21 procent som var Windowsbaserade (Prisjakt Sverige AB, 2011). Således studerades Androids plattform under några dagar innan beslutet togs att satsa på Windowsplattformen (se avsnitt 3.7.2).

6.2.2 Synkronisering

Ett moment som har tagit mycket tid i anspråk är utformningen av synkroniseringen. Detta kommer av beslutet att modellen i prototypen skulle vara fristående från synkroniseringen i kombination med att kassasystemet API är inriktat på operationer medan modellen avbildar tillstånd.

Flera lösningar för att hålla modellen uppdaterad övervägdes. Det första alternativet som övervägdes var att skapa ett API-meddelande vid varje ändring av modellen och lägga meddelandet i en kö. Kön skulle vara sparad på disk och automatiskt läsas in när programmet startades för att säkerställa att ingen information gick förlorad om programmet kraschade eller på annat sätt oväntat avslutades.

Denna idé övergavs av två anledningar: dels skulle detta tillvägagångssätt leda till en mängd redundanta meddelanden i meddelandekön (om användaren av modellen exempelvis hade ändrat stolsnummer och betalgrupp på en delbeställning hade det lett till att två meddelanden hade skapats för dessa ändringar, när det egentligen hade räckt med ett meddelande), dels att när kön hade återställts från disk hade relationerna mellan entiteterna i modellen gått förlorade. Det senare hade kunnat lösas genom att spara innehållet i modellen till

disk med jämna mellanrum, och läsa in avbildningen när programmet startade. Dock hade detta inneburit att om programmet hade kraschat till följd av ett fel som uppstått när modellen synkroniserats, hade detta fel säkerligen uppstått ånyo när programmet startats om, eftersom modellen då skulle befinna sig i samma tillstånd som innan programmet kraschade.

Dessutom hade denna säkerhetsåtgärd i många fall inte varit nödvändig, eftersom om programmet stängs, exempelvis till följd av att batteriet i plattan tagit slut, hämtar personalen mer troligen en ny platta istället för att vänta på att den ursprungliga plattan laddats klart, vilket skulle leda till att ändringarna som låg i meddelandekön på ursprungsplattan ändå hade gått förlorade. Då plattan senare ska användas, kanske en dag senare, vill ändringarna skickas till kassasystemet vilket kan leda till oförutsedda händelser.

I synkroniseringen ingår också en låsmekanism som låser en öppen beställning till en specifik klient för att förhindra att flera användare samtidigt ändrar i samma beställning. Detta är dessvärre mindre förenligt med en annan egenskap som prototypen har, nämligen att den ska kunna användas då nätverksanslutning tillfälligt saknas. Då låsningen kräver att klienterna direkt eller indirekt måste kommunicera med varandra är detta ett problem som inte har lösts. En möjlig lösning skulle kunna vara att plattorna kommunicerar med varandra för att öka nätverkets räckvidd (vilket i praktiken skulle innebära att man har skapat ett ad hoc-nätverk).

Vi har dock bedömt att avsaknaden av tillförlitlig låsning då nätverksanslutning saknas inte är ett stort problem eftersom en servitör med största sannolikhet står vid bordet där beställningen tas. Därmed ser andra servitörer att gästerna vid det bordet redan betjänas av en annan servitör.

6.2.3 Ikoner

En aspekt som har beaktats under utvecklingsarbetet av användargränssnittet är att alla delar ska skalas efter skärmens pixeltäthet. Detta innebär att såväl text som grafik måste vara skalbara. När det gäller texten är det inget problem då uteslutande vektoriserade typsnitt använts i prototypen, och kan därmed skalas

fritt. Värre är det när det gäller ikoner och illustrationer. I detta fall tillämpas lämpligen någon av dessa två metoder:

- Använda RASTERBILDER som har en upplösning som överskrider dimensionerna på ytan där de ska målas ut. Detta leder till att en marginal finns om ikonen skulle behöva renderas på en skärm med högre pixeltäthet än tänkt. Nackdelen är att detta leder till större filstorlekar och att man måste sätta en lämplig begränsning för hur mycket högre pixeltäthet som ska stödas utan att bilden blir otydlig.
- Använda VEKTORGRAFIK som kan skalas fritt.

I prototypen har båda metoderna använts. För merparten av ikonerna har vektorgrafik använts, däribland batteri- och nätverksindikatorerna samt minimera-, maximera- och återställknapparna. För större illustrationer och för ikoner som sällan används har rasterbilder använts.

6.2.4 Noteringar

Ett problem som uppstod då synkroniseringen implementerades var att kassasystemet saknar stöd för att lagra en notering för en specifik artikel, till exempel att oxfilén ska vara blodig eller att brödet ska vara glutenfritt. För att lösa problemet utformades ett specialformat för att lagra noteringarna på ordernivå, som både kan läsas maskinellt och inte vara främmande för människor. Se även avsnitt 0.

6.2.5 Skärmtangentbord

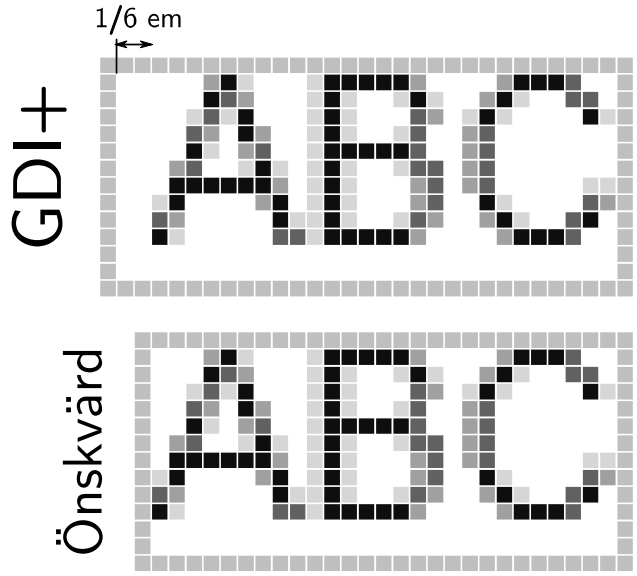
Det finns ingen inbyggd funktionalitet i Windows skärmtangentbord (se avsnitt 5.1.5 för mer information om skärmtangentbordet) för att ändra dess storlek, utan då måste Windows API användas. Vid test märktes att för att kunna ändra tangentbordets storlek krävs administratörsrättigheter (troligtvis körs skärmtangentbordet i en systemprocess), vilket är något som inte bör krävas. Utan att kräva administratörsrättigheter har ingen lösning på problemet hittats.

Det som istället gjordes var helt enkelt att låta tangentbordets storlek vara samma som när det användes senast.

Då programmeraren inte bestämmer storleken på tangentbordet behövs ett sätt att ta reda på dess storlek för att kunna placera ut det på rätt plats. Tangentbordet har inbyggd funktionalitet för detta men med problemet att innan tangentbordet visats för första gången anges storleken 0, vilket gör att tangentbordet placeras utanför skärmen första gången det ska visas. Detta löstes genom att använda Windows API för att ta reda på storleken, vilket ger rätt storlek även vid första visningen. För att endast ta reda på storleken krävs inga administratörsrättigheter.

6.2.6 Textrendering

När text med olika storlek renderades justerad utmed en vertikal linje hade de olika texterna olika stort mellanrum i vänsterkant (se Figur 6.4).



Figur 6.4. Textrendering i GDI+ och som önskas.

Efter en del efterforskning fastslogs att det är GDI+ som automatiskt fyller ut texten med 1/6 em (Microsoft, 2007b). Lösningen som används är tämligen simpel: vid rendering beräknas hur många pixlar 1/6 em motsvarar för det aktuella typsnittet och texten förflyttas samma antal pixlar åt vänster.

6.3 Förbättringsåtgärder

Trots att målet har varit att göra en så bra och välfungerande prototyp som möjligt finns det givetvis sådant som kan förbättras. Några förbättringsåtgärder:

- FÖRBÄTTRAD TRANSAKTIONSHANTERING. Transaktionshanteringen skulle kunna förbättras för att i högre grad undvika att redundanta meddelanden läggs i meddelandekön och skickas över nätverket. Detta skulle exempelvis kunna göras genom att transaktionen inte stängs direkt då klienten lämnar transaktionen, utan den hålls öppen några sekunder därefter för att fånga upp eventuella följande operationer. En nackdel med detta förfarande är att det lägger till ytterligare en fördröjning innan ändringarna appliceras i kassasystemet. En annan metod är att, när transaktionen är färdig, gå igenom meddelandekön och slå ihop nya meddelanden med eventuella gamla meddelanden som refererar till samma entitet.

- PERMANENT CACHELAGRING. Cachen där alla modellentiteter lagras lokalt skulle kunna sparas på disk tillsammans med meddelandekön för att behålla ändringar som annars går förlorade om programmet kraschar. Svårigheter med förfarandet är att om en ändring har gett upphov till programkraschen kommer programmet att krascha igen när det startas, samt att cachen snabbt blir inaktuell. Det är då tveksamt om man kan uppväga nyttan mot dessa nackdelar och den tid det tar att implementera.

- NY DESIGN AV BESTÄLLNINGSVYN. Beställningsvyn är i dagsläget uppbyggd med standardkontrollen ListView från .NET Framework. Kontrollens utseende skulle kunna anpassas bättre till programmets övriga utseende. Rent funktionellt uppfyller dock ListView de behov som finns.

6.4 Framtida tillägg

I detta avsnitt listas ett antal tillägg som kan implementeras i framtiden. Från början var det tänkt att vissa av dessa tillägg skulle ingå i projektet, men dessa prioriterades bort med anledning av tidsbrist.

- LÅS. En tanke är att man ska kunna låsa gränssnittet. Plattan skulle i det låsta läget kunna lämnas ut till en gäst som vill kolla vilka rätter som finns att välja på. I det låsta läget kan inga beställningar ändras eller skickas, utan det är bara artikellistan som kan visas.

- ANVÄNDARINSTÄLLNINGAR. Flera inställningar bör kunna sparas centralt i kassasystemet så att de följer med användaren oavsett vilken platta som används. I dagsläget ligger alla inställningar i en konfigurationsfil som lagras lokalt på varje klient.

- BORDSVY. Från början var tanken att prototypen skulle innehålla en separat bordsvy, där borden finns utplacerade rent visuellt som en grafisk modell av restaurangen för att snabbare kunna hitta rätt bord. Flera problem finns med idén, bland annat att modellen bör sparas som en bild, som restaurangen själv måste hålla uppdaterad. För uppdateringen bör det finnas ett separat program som möjliggör enkel redigering av modellen exempelvis enligt drag- och släpprincipen. De senast valda borden skulle också kunna presenteras för att göra valet av bord snabbare.

- ALTERNATIV. Varje artikel bör kunna ha ett antal förkonfigurerade alternativ, exempelvis *välstekt* och *blodig* för en biff. Detta är dock något som inte kunnat implementeras på grund av att kassasystemets API saknar stöd för sådana alternativ.

- IKONER. I matmenyn skulle ett antal ikoner kunna användas för att indikera om en rätt är vegetarisk, glutenfri, slutsåld eller dylikt.

6.5 Sammanfattning

Prototypen har utvecklats i C# och har dragit nytta av .NET Framework 4.0 och Windows Forms. Visual Studio har fungerat bra som utvecklingsmiljö för projektet och har i kombination med instickspogram som AnkhSVN och GhostDoc fyllt de behov som funnits. Windows Installer-paket används för att installera programmet och genereras med hjälp av WiX.

Under utvecklingen har flera problem uppstått. Bland annat har synkroniseringen med kassasystemet varit ett huvudbry som vållat flera problem, bland annat eftersom prototypen ska kunna användas trots att nätverksanslutning tidvis kan saknas, något som gör det omöjligt att helt tillförlitligt låsa en beställning till en specifik servitör.

Skärmtangentbordet har varit en annan komponent som vållat problem. Dels var det inledningsvis svårt att ange tangentbordets position på skärmen, dels kan inte storleken på tangentbordet bestämmas.

Kapitel 7

Slutsats

Detta kapitel sammanfattar projektet och reflekterar över hur arbetet i sin helhet har gått. I slutet av kapitlet presenteras även uppdragsgivaren AffärsIT:s framtidsplaner för prototypen.

Lösningen som utvecklats körs på surfplattor och används av serveringspersonal för att ta beställningar från gästerna i restauranger direkt vid borden utan att behöva uppsöka en kassaterminal. Programmet skickar automatiskt alla nya beställningar via en trådlös nätverksanslutning till kassasystemet, som vidarebefordrar dem till baren och köket, där beställningen skrivs ut på en så kallad bong. Kökspersonalen använder bongarna för att avgöra vilka maträtter som ska tillredas.

Fördelen med systemet är att serveringspersonalen inte behöver uppsöka en kassaterminal mellan varje kund. Det ger en tidsvinst som skulle kunna användas till att sänka priserna, läggas på att ge gästerna mer service eller öka vinsten. Dessutom ger programmet serveringspersonalen tillgång till mer information om rätterna som finns inlagda i kassasystemet.

7.1 Projektvärdering

Det har på senare tid blivit allt vanligare med pekskärmsförsedda produkter, och det har därför varit intressant och lärorikt att prova på att utveckla program för användning på pekskärm. Det har varit en utmaning att utveckla synkroniseringen med kassasystemet då den även ska kunna fungera utan nätverksanslutning och kunna hantera flera samtidigt inloggade användare.

Arbetssättet som har använts har fungerat bra och har medverkat till att målen har kunnat uppnås. Det har varit kul att se en prototyp som är redo att ingå i ett pilottest växa fram under den begränsade tid som har funnits till vårt förfogande.

Större delen av projekttiden har tillbringats på uppdragsgivarens kontor. Det har medfört att deras utvecklare har kunnat kontaktas när det har behövts, både personligen och via e-post, vilket har varit bra. Uppdragsgivaren har gett oss stor frihet att ta beslut angående prototypen, men de har hållits uppdaterade genom att det som gjorts demonstrerats med ungefär en månads mellanrum. Mellanrummen hade kunnat varit något kortare för att få mer återkoppling till utvecklingen.

7.2 Tillvägagångssätt

De första veckorna av projektet lades på att ta fram en specifikation för lösningen som skulle framställas (se Bilaga A). Det fanns inte så mycket specificerat av uppdragsgivaren, vilket ledde till att vi tog fram förslag för olika aspekter av lösningen som uppdragsgivaren sedan fick ta ställning till. Specifikationen diskuterades därefter fram tillsammans med uppdragsgivaren.

Utifrån specifikationen sattes en detaljerad tidsplan upp hur många dagar utvecklingen av varje enskild komponent skulle ta i anspråk. Denna tidsplan hölls i stora drag med vissa avvikelser. Den största avvikelsen är den tid som har krävts för att skriva denna uppsats. Från början avsattes en dag varannan vecka per person för uppsatsskrivning men utökades till en dag varje vecka.

Under projektet har arbetet strukturerats upp genom att varje vecka inletts med ett kort planeringsmöte där vad som ska göras under veckan planerats. I början av varje dag har en lista sammanställts på sådant som bör göras under dagen. I slutet av veckan har arbetet utvärderats och en veckorapport sammanställts.

7.3 Resultat

I detta avsnitt kommer resultatet för projektets huvudbeståndsdelar – val av surfplatta, kassaklient och kassasystem – att sammanfattas.

7.3.1 Surfplattorna

Det praktiska arbetet började med att valet av en lämplig surfplatta skulle göras. Flera kriterier och önskemål diskuterades fram, bland annat med avseende på plattornas hållbarhet och utseende, och utifrån dessa var tanken att en specifik surfplatta skulle väljas. Under letandet efter surfplattor upptäcktes att flera plattor kunde vara svåra att få tag på och att marknaden ändras mycket snabbt. Det ledde till att idén att välja en specifik platta förkastades, och istället var målsättningen att välja ut ett antal lämpliga kandidater, utifrån vilka en platta kunde väljas då tiden var inne för att lansera produkten.

Retrospektivt kan sägas att mindre tid borde ha lagts på valet av maskinvaran – den tiden hade kommit till mer nytta om den hade lagts på att skriva om valet av maskinvara i uppsatsen. Det hade varit mer optimalt om vi hade skrivit om maskinvaran i anslutning till att efterforskningen gjordes, men i början låg uppsatsen drygt en månad ur fas med det praktiska arbetet. Detta medförde dock att det hann komma information om lansering av några nya surfplattor, bland annat Fujitsu Q550, som togs med i platturvalet.

Dessutom hade det varit en fördel om valet av plattform hade gjorts innan platturvalet gjordes.

7.3.2 Kassaklienten

Nästa steg i processen var att börja implementera prototypen. I praktiken var detta steg till viss del sammanvävt med valet av maskinvara, vilket ledde till att plattformen inte var helt bestämd då implementeringen påbörjades. Därför påbörjades arbetet på plattformen för Android innan beslutet att fortsätta i Windows-miljö togs.

Det fortsatta utvecklingsarbetet bestod mestadels av att implementera det som bestämts i specifikationen. Enda större avvikelsen från den ursprungliga specifikationen är att matmenyn och beställningsredigeraren initialt hade tänkts vara två separata vyer. Denna idé slopades i ett tidigt skede i samråd med uppdragsgivaren och implementerades istället som en kombinerad mer överskådlig vy.

Mot slutet av projektet uppstod tidsbrist till följd av att uppsatsen hade tagit mer tid än beräknat, drygt hälften av projekttiden, vilket innebar att vi blev tvungna att ta bort vissa detaljer (se avsnitt 6.4) för att kunna presentera en fungerande prototyp i slutet av terminen. Även detta gjordes i samråd med uppdragsgivaren.

7.3.3 Kassasystemet

All kommunikation med kassasystemet har skett genom dess API. Detta API har utvecklats av en konsult som vi inte har haft någon kontakt med, vilket har lett till svårigheter att rapportera buggar och att komma med önskemål om förändringar i API:t. De ändringar som meddelats ska ske har dragit ut på tiden.

Överlag har API:t fungerat bra men vissa aspekter med API:t anser vi är ologiska och resulterar i extra arbete.

7.4 Framtida planer

Kassasystemet är i skrivande stund inte helt anpassat för dess nya applikation. Framledes kommer uppdragsgivaren att arbeta med att anpassa kassasystemet för den nya funktionalitet som den nya kassaklienten introducerat.

Uppdragsgivaren kommer att göra ett pilottest för prototypen i produktionsmiljö och om det visar sig fungera bra kan prototypen så småningom komma att lanseras på marknaden. Vi hoppas naturligtvis på att man i framtiden kan stöta på produkten när man äter ute.

Referenser

Abelson, J. (2011) *Tablet Gold Rush Strains Display Industry in 2011* [online]. Tillgänglig på: http://www.isuppli.com/Display-Materials-and-Systems/News/Pages/Tablet-Gold-Rush-Strains-Display-Industry-in-2011.aspx [hämtad 8 februari 2011].

Android-x86 (2011) *Android-x86* [online]. Tillgänglig på: http://www.android-x86.org/ [hämtad 14 mars 2011].

Apple Inc. (2010) *Teknisk information och tillbehör för iPad* [online]. Tillgänglig på: http://www.apple.com/se/ipad/specs/ [hämtad 23 mars 2011].

Apple Inc. (2011a) *iOS Dev Center* [online]. Apple Inc. Tillgänglig på: http://developer.apple.com/support/ios/ios-dev-center.html [hämtad 14 mars 2011].

Apple Inc. (2011b) *iOS Developer Program* [online]. Apple Inc. Tillgänglig på: http://developer.apple.com/programs/ios/ [hämtad 14 mars 2011].

Apple (2007) *Apple Reinvents the Phone with iPhone* [online]. Tillgänglig på: http://www.apple.com/pr/library/2007/01/09iphone.html [hämtad 8 februari 2011].

Apple (2010) *iPad Available in US on April 3* [online]. Tillgänglig på: http://www.apple.com/pr/library/2010/03/05ipad.html [hämtad 8 februari 2011].

Archos (u.å.) *Archos 10.1" Internet Tablet Tech Specs* [online]. Archos. Tillgänglig på: http://www.archos.com/products/ta/archos_101it/specs.html?country=se&lang=en [hämtad 23 mars 2011].

Asplund, K. (2010) *DAV B04 - Databasteknik* [online]. Tillgänglig på: http://www.cs.kau.se/cs/education/courses/dvgb04/lectures/Transaktionshantering1.ppt [hämtad 8 april 2011], arkiverat på http://www.webcitation.org/5xnSM7Fz1.

Baker, B.C. & Fang, W. (2007) *The Power Behind Resistive Touch Screens* [online]. Tillgänglig på: http://www.newarkinone.thinkhost.com/brands/promos/leading_edge/TI_Power_Behind_Resistive_Touch_Screens.pdf [hämtad 12 mars 2011].

Barrett, G. & Omote, R. (2010) Projected-Capacitive Touch Technology. *Information Display.* Nr 3/2010. ss.16-21. Hämtad från http://www.walkermobile.com/March_2010_ID_Projected_Capacitive.pdf.

Blickenstorfer, C. (2005) *A Brief History of Tablet PCs* [online]. BVTech, Inc. Tillgänglig på: http://www.biovisualtech.com/tablet-PC.htm [hämtad 14 mars 2011].

Block, R. (2007) *Live from Macworld 2007: Steve Jobs keynote* [online]. AOL Inc. Tillgänglig på: http://www.engadget.com/2007/01/09/live-from-macworld-2007-steve-jobs-keynote/ [hämtad 14 februari 2011].

Bone's Restaurant (u.å.) *iCellar* [online]. Tillgänglig på: http://www.bonesrestaurant.com/iCellar [hämtad 14 februari 2011].

Carmody, T. (2010) *Your Restaurant's Next Menu Is An iPad* [online]. Condé Nast Digital. Tillgänglig på: http://www.wired.com/gadgetlab/2010/09/your-restaurants-next-menu-is-an-ipad/ [hämtad 14 februari 2011].

Cooper, L.F. & Lee, S. (2008) *Assessing The Broad Business Case For Rugged Computing* [whitepaper]. Larstan Business Reports.

Dahlström, H. (2010) *Skalet som gör din Ipod till en Iphone* [online]. Tillgänglig på: http://www.nyteknik.se/nyheter/it_telekom/mobiltele/article3033367.ece [hämtad 28 mars 2011].

Dahmer, S.J. & Kahl, K.W. (2009) *Restaurant Service Basics*. 2nd ed. New Jersey, USA: John Wiley & Sons, Inc.

Eviware (u.å.) *soapUI. The Swiss-Army Knife of Testing* [online]. Eviware. Tillgänglig på: http://www.soapui.org/About-SoapUI/what-is-soapui.html [hämtad 2 maj 2011], arkiverat på http://www.webcitation.org/5yNS4UP8d.

Faraday, P. & Becker, B. (2007) [online]. Tillgänglig på: http://download.microsoft.com/download/e/4/3/e43ca6e6-8a17-42e4-984e-9aa45d7bddd3/WinFormsToWPF.DOCX [hämtad 26 april 2011].

Flytech (2010) *POS223 Mobile POS* [online]. Tillgänglig på: http://www.flytech.com/upload/brochure/64_doc.pdf [hämtad 8 februari 2011].

Fujitsu (2011) *Fujitsu STYLISTIC Q550 Slate PC (Data Sheet)* [online]. Fujitsu. Tillgänglig på: https://globalsp.ts.fujitsu.com/dmsp/docs/ds-stylistic-q550.pdf [hämtad 23 mars 2011].

Gartner, Inc. (2008) *Gartner Says Worldwide Mobile Phone Sales Increased 16 Per Cent in 2007* [online]. Tillgänglig på: http://www.gartner.com/it/page.jsp?id=612207 [hämtad 15 februari 2011].

Gartner, Inc. (2010) *Gartner Says Touchscreen Mobile Device Sales Will Grow 97 Percent in 2010* [online]. Tillgänglig på: http://www.gartner.co.uk/it/page.jsp?id=1313415 [hämtad 8 februari 2011].

Google (2011a) *Android Emulator* [online]. Tillgänglig på: http://developer.android.com/guide/developing/tools/emulator.html [hämtad 14 mars 2011].

Google (2011b) *Introduction* [online]. Tillgänglig på: http://developer.android.com/guide/developing/index.html [hämtad 14 mars 2011].

Hessel, D. (2010) Så funkar pekskärmarna. *M3*. nr 6/2010. ss.116 - 117.

LANDesk (2007) *Proactive Solutions for Today's Healthcare* [online]. LANDesk. Tillgänglig på: http://creekpointe.com/landesk/pdf/GwinnettHospitalSystem.pdf [hämtad 14 februari 2011].

Lindhé, P. (2011) Utvecklare på AffärsIT i Skandinavien AB [e-post] februari 2011.

Magnusson, J. (2011) Kommunen moderniserar. *Nya Wermlandstidningen*. 12 februari. s.72.

Malison, A. (2002) *Replacing Pencil & Pad in Full Service Restaurants* [online]. Tillgänglig på: http://actionsystems.com/whitepaper.html [hämtad 8 februari 2011].

Malison, A. (2003) *Benefits of Handheld Order Taking Systems in Full Service Restaurants* [online]. Tillgänglig på: http://actionsystems.com/whitepaper1.html [hämtad 7 februari 2011].

Manion, C. & DeMicco, F.J. (2004) Handheld Wireless Point of Sale Systems in the Restaurant Industry. 7(2)(2004).

Microsoft (2007a) *Open Source at Microsoft* [online]. Tillgänglig på: http://download.microsoft.com/download/B/6/8/B6811521-81F6-4F57-8009-6D9C60E5F744/Windows_Installer_XML_Toolset.pdf [hämtad 26 april 2011].

Microsoft (2007b) *Why text appears different when drawn with GDIPlus versus GDI* [online]. Tillgänglig på: http://support.microsoft.com/kb/307208 [hämtad 19 april 2011].

Microsoft (2008) *ScrollableControl.AutoScroll Property* [online]. Microsoft Corporation. Tillgänglig på: http://msdn.microsoft.com/en-us/library/system.windows.forms.scrollablecontrol.autoscroll(VS.90).aspx [hämtad 22 mars 2011].

Microsoft (2009) *Touch* [online]. Microsoft Corporation. Tillgänglig på: http://msdn.microsoft.com/en-us/library/cc872774.aspx [hämtad 15 mars 2011].

Microsoft (2010) *Rollback Installation* [online]. Tillgänglig på: http://msdn.microsoft.com/en-us/library/aa371370(VS.85).aspx [hämtad 26 april 2011].

Microsoft (2011a).*NET Framework Conceptual Overview* [online]. Microsoft Corporation. Tillgänglig på: http://msdn.microsoft.com/en-us/library/zw4w595w(VS.100).aspx [hämtad 14 mars 2011].

Microsoft (2011b) *EventWaitHandle Class* [online]. Microsoft Corporation. Tillgänglig på: http://msdn.microsoft.com/en-us/library/system.threading.eventwaithandle(v=VS.100).aspx [hämtad 10 april 2011].

Microsoft (2011c) *Garbage Collection* [online]. Microsoft Corporation. Tillgänglig på: http://msdn.microsoft.com/en-us/library/0xy59wtx(v=VS.100).aspx [hämtad 10 april 2011].

Microsoft (2011d) *TextInputPanel for Users of PenInputPanel* [online]. Microsoft Corporation. Tillgänglig på: http://msdn.microsoft.com/en-us/library/ms698524(v=vs.85).aspx [hämtad 19 april 2011].

Microsoft (2011e) *Web References in Visual Studio* [online]. Microsoft Corporation. Tillgänglig på: http://msdn.microsoft.com/en-us/library/tydxdyw9(v=VS.100).aspx [hämtad 10 april 2011].

Microsoft (2011f) *XML Documentation Comments (C# Programming Guide)* [online]. Microsoft Corporation. Tillgänglig på: http://msdn.microsoft.com/en-us/library/b2s063f7(v=VS.100).aspx [hämtad 27 april 2011].

Motion Computing (2011) *Motion CL900 Product Specifications* [online]. Motion Computing. Tillgänglig på: http://www.motioncomputing.com/resources/CL900/specifications_CL900.pdf [hämtad 23 mars 2011].

MSDN Forums (2010) *How to scroll without a scrollbar or make scrollbar invisible* [online]. Tillgänglig på: http://social.msdn.microsoft.com/Forums/en-SG/winforms/thread/6b9c2c72-e91a-40f0-a835-c12328490c0c [hämtad 23 mars 2011], arkiverat på http://www.webcitation.org/5xOl4rBm5.

Mundo Global Tapas (u.å.) *Mundo Global Tapas* [online]. Tillgänglig på: http://www.mundo.com.au/north-sydney [hämtad 14 februari 2011].

Open Handset Alliance (2007a) *Android FAQ* [online]. Tillgänglig på: http://www.openhandsetalliance.com/android_faq.html [hämtad 14 mars 2011].

Open Handset Alliance (2007b) *Industry Leaders Announce Open Platform for Mobile Devices* [online]. Tillgänglig på: http://www.openhandsetalliance.com/press_110507.html [hämtad 14 mars 2011].

Payne, A. (2010) *On the iPad* [online]. Tillgänglig på: http://al3x.net/2010/01/28/ipad.html [hämtad 14 mars 2011].

PC Magazine (u.å.) *Definition of tablet computer* [online]. Tillgänglig på: http://www.pcmag.com/encyclopedia_term/0,2542,i=52520,00.asp [hämtad 8 februari 2011].

PCB Distribution AB (2011) *Paddy-T V2 10,1 Inch Multitouch Tablet PC* [online]. Tillgänglig på: http://www.pcb.se/ItemInfo?itemId=488505860 [hämtad 23 mars 2011].

Prisjakt Sverige AB (2011) *Surfplattor* [online]. Prisjakt Sverige AB. Tillgänglig på: http://www.prisjakt.nu/kategori.php?k=1594 [hämtad 2 maj 2011].

Rodriguez, J. (2009) *Introduction to WPF 4 Multitouch* [online]. Tillgänglig på: Introduction to WPF 4 Multitouch [hämtad 26 april 2011].

Sack, K. (2010) *Choosing Wines at the Touch of a Screen* [online]. Tillgänglig på: http://www.nytimes.com/2010/09/15/dining/15ipad.html [hämtad 8 februari 2011].

Samsung (u.å.) *Galaxy Tab Spec.* [online]. Samsung. Tillgänglig på: http://www.samsung.com/se/consumer/mobile/mobilephones/mobilephones/GT-P1000MSANEE/index.idx?pagetype=prd_detail&tab=specification [hämtad 23 mars 2011].

Schultz, C.v. (2011) *Ny Teknik* [online]. Ny Teknik. Tillgänglig på: http://www.nyteknik.se/nyheter/it_telekom/tv/article3099815.ece [hämtad 27 april 2011].

Semenza, P. (2010) *Touch Screen Market Update* [online]. DisplaySearch: DisplaySearch. Tillgänglig på: http://www.displaysearch.com/cps/rde/xbcr/displaysearch/FlexTech_Sensor_Workship.pdf [hämtad 12 mars 2011].

Språkrådet (2010) *Digital nummerlapp blir gärna puck i folkmun* [online]. Tillgänglig på: http://www.språkrådet.se/8887 [hämtad 7 februari 2011].

Sutter, J.D. (2010) *Why people 'jailbreak' their iPhones* [online]. Turner Broadcasting System, Inc. Tillgänglig på: http://articles.cnn.com/2010-07-27/tech/why.jailbreak.iphone_1_app-store-iphone-popular-apps [hämtad 14 mars 2011].

Sveriges Television (2011) *Plus* [tv]. Del 6 av 8, sänt 10 mars 2011, Sverige.

The Sydney Morning Herald (2010) *Eatery gets iPad on the menu. and vice versa* [online]. Tillgänglig på: http://www.smh.com.au/technology/technology-news/eatery-gets-ipad-on-the-menu--and-vice-versa-20100604-xkir.html [hämtad 8 februari 2011].

Thörnqvist, J. (2011) *MSI Windpad 100W i januari* [online]. SweClockers AB. Tillgänglig på: http://www.sweclockers.com/nyhet/13340-msi-windpad-100w-i-januari [hämtad 23 mars 2011].

US Department of Defence (2000) *Department of Defence Test Method Standard for Environmental Engineering Considerations and Laboratory Tests* [online]. Tillgänglig på: http://www.everyspec.com/MIL-STD/MIL-STD+(0800+-+0899)/MIL_STD_810F_949/ [hämtad 13 juni 2011].

W3C (2007) *Introduction* [online]. W3C. Tillgänglig på: http://www.w3.org/TR/soap12-part1/#intro [hämtad 10 april 2011].

VDMA (2008) Flat Panel Display Technologies. *European Technology: Flat Panel Displays*. 6:e upplagan. ss.19-21.

Webhallen (2011) *MSI WindPad 100W* [online]. Webhallen Sverige AB. Tillgänglig på: http://www.webhallen.com/hardvara/128621-msi_windpad_w100-atom_z530-2gb-32gb_ssd-10.1_led-gma500-bt-win_7_hp [hämtad 23 mars 2011].

Weiss, M.A. (2006) Graph Algorithms. In *Data Structures and Algorithm Analysis in C++*. 3rd ed. Addison Wesley. ss.339-45.

ViewSonic Corporation (u.å.) *ViewPad 10 Tablet* [online]. Tillgänglig på: http://www.viewsonic.com/products/vpad10.htm [hämtad 23 mars 2011].

Wikipedia (2007) *File:LCD layers.svg* [online]. Tillgänglig på: http://commons.wikimedia.org/wiki/File:LCD_layers.svg [hämtad 12 mars 2011].

Wikipedia (2010a) *Bong* [online]. Tillgänglig på: http://sv.wikipedia.org/wiki/Bong_(restaurang)?oldid=11770143 [hämtad 7 februari 2011].

Wikipedia (2010b) *Läsplatta* [online]. Tillgänglig på: http://sv.wikipedia.org/wiki/Läsplatta?oldid=12997560 [hämtad 14 februari 2011].

Wikipedia (2010c) *Price look-up code* [online]. Tillgänglig på: http://en.wikipedia.org/wiki/Price_look-up_code?oldid=399016667 [hämtad 8 februari 2011].

Wikipedia (2011a) *Akronym* [online]. Tillgänglig på: http://sv.wikipedia.org/wiki/Akronym?oldid=13553540 [hämtad 23 mars 2011].

Wikipedia (2011b) *Application programming interface* [online]. Tillgänglig på: http://en.wikipedia.org/wiki/Application_programming_interface?oldid=433551501 [hämtad 13 juni 2011].

Wikipedia (2011c) *Comparison of vector graphics editors* [online]. Tillgänglig på: http://en.wikipedia.org/wiki/Comparison_of_vector_graphics_editors?oldid=430653649 [hämtad 26 maj 2011].

Wikipedia (2011d) *Dots per inch* [online]. Tillgänglig på: http://en.wikipedia.org/wiki/Dots_per_inch?oldid=420040785 [hämtad 24 mars 2011].

Wikipedia (2011e) *HSL and HSV* [online]. Tillgänglig på: http://en.wikipedia.org/wiki/HSL_and_HSV?oldid=419161788 [hämtad 23 mars 2011].

Wikipedia (2011f) *iOS (Apple)* [online]. Tillgänglig på: http://en.wikipedia.org/wiki/IOS_(Apple)?oldid=418431962 [hämtad 14 mars 2011].

Wikipedia (2011g) *List of CLI languages* [online]. Tillgänglig på: http://en.wikipedia.org/wiki/List_of_CLI_languages?oldid=414452272 [hämtad 14 mars 2011].

Wikipedia (2011h) *List of compilers* [online]. Tillgänglig på: http://en.wikipedia.org/wiki/List_of_compilers?oldid=420256117 [hämtad 23 mars 2011].

Wikipedia (2011i) *Microsoft Tablet PC* [online]. Tillgänglig på: http://en.wikipedia.org/wiki/Microsoft_Tablet_PC?oldid=411426043 [hämtad 14 mars 2011].

Wikipedia (2011j) *Pen computing* [online]. Tillgänglig på: http://en.wikipedia.org/wiki/Pen_computing?oldid=406813143 [hämtad 8 februari 2011].

Wikipedia (2011k) *Remote Desktop Services* [online]. Tillgänglig på: http://en.wikipedia.org/wiki/Remote_Desktop_Services?oldid=418935588 [hämtad 23 mars 2011].

Wikipedia (2011l) *Windows 1.0* [online]. Tillgänglig på: http://en.wikipedia.org/wiki/Windows_1.0?oldid=418371393 [hämtad 14 mars 2011].

Wikipedia (2011m) *Windows for Pen Computing* [online]. Tillgänglig på: http://en.wikipedia.org/wiki/Windows_for_Pen_Computing?oldid=413980175 [hämtad 14 mars 2011].

Zhu, X., Ge, Z., Wu, T.X. & Wu, S.-T. (2005) Transflective Liquid Crystal Displays. *IEEE/OSA Journal of Display Technology* 1(1), s.15.

Bilaga A. Specifikation

I den här bilagan presenteras den specifikation som togs fram i början av projektet.

A.1 Bakgrund

I restaurangbranschen tillämpas olika metoder för att ta emot beställningar från kunderna. Dels har vi det traditionella sättet där personalen kommer till kundens bord och tar beställningen från kunden. Personalen lämnar sedan beställningen till köket, som påbörjar tillagningen av de beställda rätterna. Dels har vi sättet som tillämpas i snabbmatsrestaurangerna; det första kunden gör är att lägga en beställning och betala för beställningen. Därtill finns det också andra varianter och kombinationer av dessa. Denna specifikation kommer fortsättningsvis fokusera på det traditionella sättet.

Ser vi till denna metod mer i detalj, innefattar serveringspersonalens roll bland annat

- att lämna ut menyer till kunderna,
- att ta beställningar från kunder,
- att göra eventuella ändringar i beställningar,
- att göra köket medvetna om vad som är beställt,
- att lämna ut maten till kunderna och
- att ta betalt.

Då personalen tagit en eller några beställningar, måste beställningarna lämnas till köket för att väntetiderna inte ska bli för långa. Beställningarna som

personalen har antecknat på lapparna måste sedan föras in i kassasystemet för att där parkeras tills betalning sker.

A.2 Mål

Med den nya tekniken finns mycket som skulle kunna förenklas i proceduren beskriven i föregående stycke. Momentet då serveringspersonalen måste gå till kassan och där föra in beställningen så fort som möjligt för att begränsa gästernas väntetider, kan elimineras med hjälp av en trådlös enhet som kan användas för att ta emot beställningar.

Detta medför att serveringspersonalen kan besöka flera kunder och ta in fler beställningar åt gången. Därmed går mindre tid till spillo för personalen och beställningarna når köket snabbare än tidigare.

A.3 Resultat

Projektet innefattar att vi kommer att leverera en komplett maskin- och programvarulösning för att mobilt kunna ta upp beställningar i restauranger. Maskinvaran kommer att vara en surfplatta som ska väljas ut efter det givna ändamålet och programvaran kommer bestå av en applikation till vald maskinvara.

Appen kommer att kommunicera med restaurangens befintliga kassasystem för att få uppdaterad information om meny, beställningar och status. I appen ska det finnas ett beställningsläge där det ska gå att ta emot beställningar på samma sätt som man gör traditionellt med papper och penna men det ska även finnas utökad funktionalitet. Det kommer gå att få en bordsöversikt där man välja ett bord för att göra en ny beställning eller se beställningar associerade till bordet, en utökad meny som kan visa mer detaljerad information med bilder på olika rätter och möjlighet att se information över olika aktuella beställningar.

Projektet kommer även innehålla dokumentation av koden för appen.

A.4 Arbetsuppgifter

A.4.1 Val av surfplatta

Det första som behöver göras är att välja vilken surfplatta som ska användas. De plattformar som finns tillgängliga på marknaden idag är framför allt Windows 7, iOS och Android. Enheterna kommer att kontrolleras mot ett antal kriterier som ställts upp:

- Robust, enheten bör tåla hård hantering i restaurangmiljö.
- Storlek, enheten ska vara lagom stor och lätt att greppa för att det ska vara enkelt för serveringspersonalen att bära den.
- Användarvänlighet, operativsystemet på enheten ska vara enkelt att använda om användaren behöver göra något utanför appen.
- Kostnad, både inköpskostnaden och kostnaden för installation och underhåll ska vara acceptabel.

Uppskattning: Vi beräknar att detta moment kan slutföras på en dag för två personer (2 persondagar).

A.4.2 Synkronisering av data

Datan som enheterna kommer att använda lagras i en MySQL-server i kassasystemet. Information om beställningar och matmenyn, som kassasystemet lagrar i denna databas, kommer att synkroniseras och lagras lokalt i varje platta för att personalen alltid ska ha en uppdaterad bild av läget i restaurangen.

För att hålla personalen uppdaterad om läget samt för att undvika att flera personer ur personalen gör ändringar i samma beställning kommer den lokala datan att uppdateras frekvent.

Hämtningen av data från kassasystemet sker genom ett SOAP-gränssnitt, som kommer att skapas av AffärsIT och ingår därmed inte i projektet. Appen kommer att lagra den nedladdade datan i en lokal databas, till exempel genom att använda SQLite.

I synkroniseringen ingår även momentet att hålla reda på när appen har anslutning till kassasystemet. Endast då kan data synkroniseras. I annat fall måste appen avvakta tills en anslutning finns tillgänglig.
Uppskattning: 16 persondagar (synkronisering av meny: 3 persondagar; synkronisering av historik: 3 persondagar; synkronisering av aktuell beställning: 6 persondagar; synkroniseringsstatus: 2 persondagar).

A.4.3 Grafiskt användargränssnitt

Surfplattan kommer att köra en app, där ett grafiskt användargränssnitt måste utformas. Gränssnittet kommer att ha ett antal så kallade vyer. Momenten som ingår i arbetet med varje vy är grafisk design, implementering, dokumentation och testning. Dessa vyer i gränssnittet är planerade:

INLOGGNINGSVY. Restaurangerna har alternativet att kräva att personalen måste logga in till kassasystemet innan det kan användas. Det grafiska gränssnittet måste möjliggöra inloggning av användare till kassasystemet.
Uppskattning: grafisk design: 0,5 persondag; realisering: 1 persondag (inkl. säker överföring av inloggningsuppgifter).

I varje vy ska en MENY finnas för att enkelt navigera mellan tillgängliga vyer. Då personalen är inloggad är de tänka alternativen Bord, Meny (för mat), Beställning och Historik. Både vilka val som finns tillängliga och dess namn kan komma att ändras under projektet.
Uppskattning: grafisk design: 0,5 persondag; realisering: 2 persondag.

BORDSVYN ger serveringspersonalen en lättöverskådlig grafisk representation av bordsplaceringen i restaurangen. Bordsvyn ska stödja flera plan med olika bordsplaceringar. När man trycker på ett bord visar en meny aktuella beställningar associerade till respektive bord. Eventuellt ska lediga och upptagna bord markeras med olika färg.
Uppskattning: grafisk design: 1 persondag; realisering: 5 persondagar.

MATMENYN listar rätterna som finns inlagda i kassasystemet. Menyn är uppdelad i två vyer; dels en som listar alla rätter, dels en som visar detaljerad information om en enskild rätt. Den detaljerade informationen innefattar bland annat en närmare beskrivning av rätten samt bilder.

Man kan direkt från båda vyerna lägga till rätten (eller ta bort den) i gästens beställning. Då detta görs får personalen olika val beroende på vilken rätt som valts, t.ex. om gästen önskar blodig eller välstekt köttstycke. Personalen ska i det här läget också ha möjlighet att skriva in andra önskemål i textform.

Personalen ska kunna skicka iväg beställningen direkt från menyn utan att gå via beställningsvyn beskriven nedan.

Menyn kommer eventuellt innehålla ett sökfält för att personalen snabbt ska kunna filtrera menyn efter vissa sökord. Om ingen rätt med sökordet hittas, ska man eventuellt kunna lägga till texten man sökt på i beställningen som en odefinierad artikel.

Uppskattning: grafisk design: 3 persondagar; realisering: 9 persondagar. Dessa tidsangivelser inkluderar även beställningsvyn nedan.

BESTÄLLNINGSVYN ska visa all information som hör till den nuvarande beställningen. Visar en lista över alla valda artiklars namn, antal, plus- och minusknapp och pris (ska kunna redigeras med ett tryck, då dyker ett numeriskt tangentbord upp). Beställningsvyn innehåller fält för att kunna ange bord för beställningen, antal personer, flytta hela eller delar av beställning till ny beställning (t.ex. delvis byta bord, dela på notan) och ta bort oavslutad beställning.

Uppskattning: se ovan.

HISTORIKVYN ska vara ett överskådligt sätt att kunna se alla aktuella beställningar och möjligtvis kunna ändras att visa även äldre beställningar. Kommer att vara en lista med kolumnerna: status på beställning, bord. tid, antal personer, pris. Status på beställning kommer att visa en av tre olika status: pausad (beställningen är påbörjad men har av någon anledning inte blivit slutförd), beställd (beställningen har blivit upptagen av personalen och är färdig att tillagas) eller betald (beställningen är slutförd och har blivit betalad).

Uppskattning: grafisk design: 1 persondag; realisering: 2 persondagar.

INDIKATORER. För att undvika att beställningar ska gå förlorade är det viktigt att det finns synliga indikatorer för batterinivå och ej inskickade beställningar. Personalen ska kunna se när batterinivån är låg så det är möjligt att sätta plattan på laddning och/eller ta en ny platta innan batteriet har tagit slut. Alternativt ska också en indikator visa nätverkstäckning, för att det ska vara möjligt att se att plattan har förlorat kontakten med nätverket och inte kunnat ta emot några uppdateringar.
Uppskattning: grafisk design: 0,5 persondag; realisering: 1 persondag.

A.4.4 Vad projektet inte omfattar

Detta är några aspekter som inte omfattas av projektet:

- Betalningar av notorna kommer att ske i det befintliga kassasystemet. Eftersom plattorna synkroniserar alla beställningar med kassasystemet, går det alltid att komma åt beställningarna från kassaapparaterna. Plattorna behöver inte skapa några kvitton, då detta görs i kassan.
- Vår del av projektet går ut på att förmedla beställningar till restaurangens kassasystem. Därefter måste beställningarna på något sätt förmedlas till kökspersonalen. I projektet ingår inte den del som förmedlar beställningarna till köket.
- Systemet kommer att ha möjlighet att associera beställningar till specifika bord, men reservation av bord kan inte hanteras av plattorna.

A.4.5 Övriga moment

- Läsa på om vald plattform; **uppskattning:** 10 persondagar.
- Möten med uppdragsgivare; **uppskattning:** 1,5 persondagar.

A.5 Omfattning

Den totala beräknade tidsåtgången är 56 persondagar. Den beräknade tillgängliga arbetstiden uppgår till 60 persondagar (6 heltidsveckor/person = 30

heltidsdagar/person = 60 persondagar). Den återstående tiden på 4 persondagar kan ses som en buffert.

A.6 Rimlighetsbedömning

Det finns en stor osäkerhet i tidsuppskattningarna som angivits, då varken plattformen som ska användas är känd eller om det tillkommer fler aspekter som inte har beaktats.

A.7 Resurser

För att kunna utföra arbetsuppgifterna som ingår i projektet behövs följande resurser:

- Internetanslutning
- En testserver som kör kassasystemets SOAP-gränssnitt
- Microsoft Visual Studio
- Microsoft Word
- Litteratur för vald plattform
- Eventuellt en platta att testköra på