Larmprogram för grannar

LUNDS
UNIVERSITET
Lunds Tekniska Högskola

LTH Ingenjörshögskolan vid Campus Helsingborg
Computer engineering / Datateknik

Examensarbete:

Filip Göransson

© Copyright Filip Göransson

LTH Ingenjörshögskolan vid Campus Helsingborg
Lunds universitet
Box 882
251 08 Helsingborg

LTH School of Engineering
Lund University
Box 882
SE-251 08 Helsingborg
Sweden

Tryckt i Sverige
Media-Tryck
Biblioteksdirektionen
Lunds universitet
Lund 2012

Sammanfattning

I examensarbetet analyseras och utvecklas en alternativ process vid utlösning av larm i en bostad, där grannar uppmanas samverka med hjälp av ett larmprogram. Larmföretaget Trex, uppdragsgivaren till arbetet undrar om det är möjligt att med hjälp av Skype skapa en tjänst som liknar denna alternativa process.

Den tilltänkta tjänsten inleds med att en kund får vetskap om att larmet i dennes bostad utlösts. Kunden väljer därefter att delge informationen till sina kontakter/grannar eller att strunta i det. Informationsförmedling görs genom att ringa ett telefonsamtal till ett online-nummer som tillhör ett Skype-konto. Kontakterna till kunden rings sedan upp via ett larmprogram som innehåller kod för kommunikation med Skype och en databas. Vid svar på samtal spelas ett ljudklipp upp med adressen till bostaden. Kontakterna i närheten kan därefter agera och titta till den specificerade bostaden.

Analysen som görs i arbetet berör framförallt Skype API eftersom huvudfrågan i arbetet är: Går det att använda Skype till att larma? Vilka datastrukturer som är lämpliga för kommunikation med databasen analyseras också.

Resultatet av examensarbetet ger indikationer på att det går att använda Skype för att larma. Det utvecklade programmet saknar dock uppspelningsfunktion vilket innebär att det inte kan användas helt och hållet ännu. Utförligare testning och vidareutveckling av slutgiltigt program krävs följaktligen innan programmet kan användas på ett tillfredsställande och meningsfullt vis.

Utvecklingen av programmet gjordes med programmeringsspråket Visual Basic och ramverket .NET tillsammans med Microsoft SQL Server.

Nyckelord: Skype, .NET, Visual Basic, larmprogram, databas, Trex

Abstract

This thesis is about analyzing and developing an alternative response process when an alarm is triggered in a home. Neighbors are in this alternative urged to cooperate with the help of an alarm program. The alarm company Trex, the author of this assignment, is wondering if it would be possible to create a service that resembles the alternative described above with the help of Skype.

The process commences with a customer being given the knowledge that the alarm in his or hers home has been triggered. The customer then chooses whether or not to share this information with the contacts/neighbors. If the customer chooses to share the information he or she calls an online number connected to a Skype account. Via an alarm program containing code used to communicate with Skype and a database, the contacts are then called. If a call is answered an audio clip plays with the customer's address information. The contacts available in the vicinity can then act upon hearing this information and check out the specified home.

The analyzing that is done in this thesis especially concerns Skype API, this is because of the main question that is asked in this thesis, which is: Is it possible to use Skype to alarm? The rest of the analyzing concerns the data structure to use when managing data to and from the database.

The result of this thesis gives indications that Skype can be used to alarm. The developed program is however lacking the ability to play audio clips in calls, which means it can't be fully used yet. More thorough testing and development are needed of the final program to be able to use the program in a satisfying and meaningful way.

The programming language and database used in the development of the program were Visual Basic.NET and Microsoft SQL Server.

Keywords: Skype, .NET, Visual Basic, alarm program, database, Trex

Förord

Jag vill inledningsvis tacka Trex för möjligheten att få utföra ett intressant och lärorikt examensarbete.

Jag vill även tacka Christian Nyberg och Mats Lilja för tips och rättning av examensarbetet.

Till sist vill jag tacka utvecklarna på Skypes forum för kunskaperna de delade med sig av.

Innehållsförteckning

1 Inledning

1.1 Bakgrund

Larmföretaget Trex med verksamhet i Båstad installerar larm hos företag och privatpersoner. Företaget är litet men har funnits länge och under samtliga år samarbetat med väktarbolag. Nu har man ställt sig frågan om grannsamverkan kan användas i larmprocessen istället för väktare. Denna alternativa larmprocess beskrivs nedan.

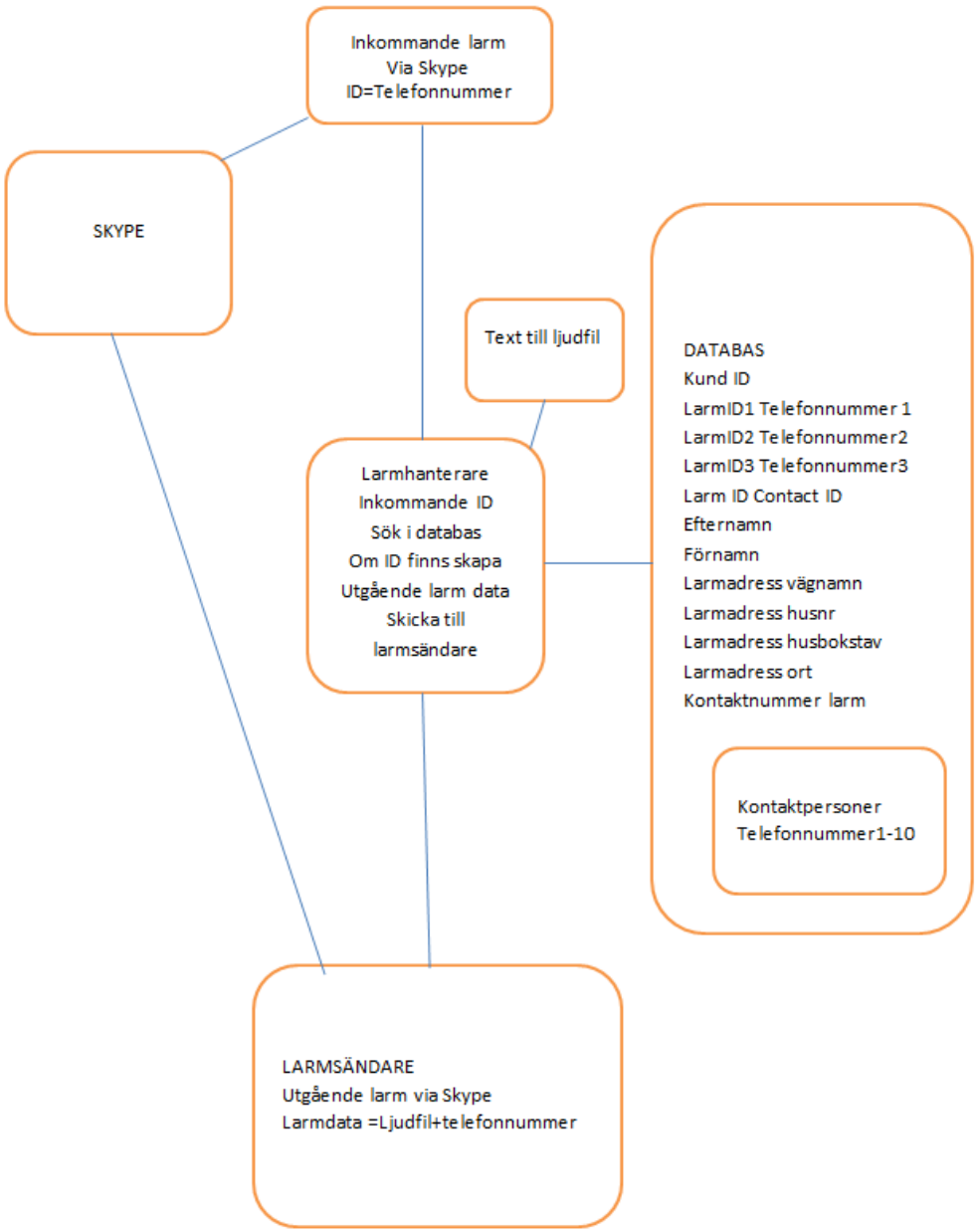
När ett larm går i ett hus eller lägenhet behövs det ibland väktare som kan rycka ut. Responstiden kan i vissa fall vara för stor för de kunder som bor avlägset. Ett kostnadseffektivt och tidseffektivt alternativ till väktare vore om grannar inom rimligt avstånd ifrån varandra kunde hjälpas åt när larm går. För att detta ska kunna förverkligas måste man som kund när ett larm går, genom en enkel tjänst, kunna underrätta sina grannar. För att grannarna ska kunna få information så snabbt som möjligt krävs det att tjänsten inte har tidsfördröjningar. Det krävs även att grannarna får informationen samtidigt för en effektiv samverkan. Detta kan möjligtvis uppfyllas genom att kunden kan ringa ett nummer kopplat till ett Skype-konto som sedan ett program använder för att automatiskt ringa upp grannar i det närliggande området. För en automatiserad tjänst likt denna krävs ett program som kan kommunicera med en databas och Skype.

En tjänst som uppfyller kraven ovan finns i dagsläget inte men test gjorda av Trex med programspråket Visual Basic.Net tillsammans med Skype API har gett resultat som pekar på att en sådan tjänst kan utvecklas.

Två punkter som gör det möjligt att använda tjänsten:

- De flesta kan nås närsomhelst och varsomhelst idag tack vare mobiler.
- Dagens bredbandshastigheter och teknologier har gjort Skype effektivt.

1.2 Larmprogrammets uppbyggnad



Figur 1: Hur larmprogrammet ska fungera

Varje kund som väljer att använda larmtjänsten ska läggas in i en databas. En kontaktlista ska därefter skapas åt kunden med grannar som finns i databasen. För att tjänsten ska vara så effektiv som möjligt bör grannar som bor närmast väljas som kontakter.

Det tjänsten ska erbjuda kunderna i databasen är att vid larm ge dem möjligheten att enkelt förmedla information om larmet till deras kontaktlista.

Att skicka SMS är inget bra alternativ eftersom det i vissa fall kan dröja innan det når fram, dessutom kan ett SMS missas av mottagaren. Trex anser istället att uppringning av kontakter är ett bättre alternativ.

För en uppringningsfunktion krävs ett Skype-konto med ett online-nummer och ett program som automatiskt kan ringa och spela upp larmmeddelande skapat från en kunddatabas.

1.2.1 Processen

I figur 1 kan man se att telefonnumret är det som ska användas för att hitta en kund i databasen vid inkommande larm. Ett ljudklipp som innehåller information om vilket hus och adress det rör sig om ska därefter skapas. Detta kan göras automatiskt t.ex. med hjälp av biblioteket ”Microsoft Speech Object Library”. I det finns en text-till-röst-funktion som kan integreras i ett program. När ett ljudklipp skapats ska det sedan skickas ut med varje samtal.

1.3 Problemformulering

Arbetet kommer undersöka om det går att utveckla ett larmprogram som använder sig av Skype samt ta reda på vilken typ av databashanterare och grafiskt användargränssnitt som är bäst lämpat för detta ändamål.
Huvudsakliga frågan i projektet kommer vara:

- Går det att använda Skype till att larma?

Delfrågor

- Blir tjänsten tidseffektiv med Skype API?
- Är kostnaden rimlig?
- Fungerar samtal med flera kunder samtidigt?
- Blir tjänsten tillförlitlig?

1.4 Syfte

Syftet med projektet är att analysera och vid möjlighet utveckla ett alternativ som kan effektivisera och ändra på den nuvarande processen vid larmutlösning för företaget Trex. Resultatet ska ge indikation på huruvida Skype kan ingå i processen eller inte.

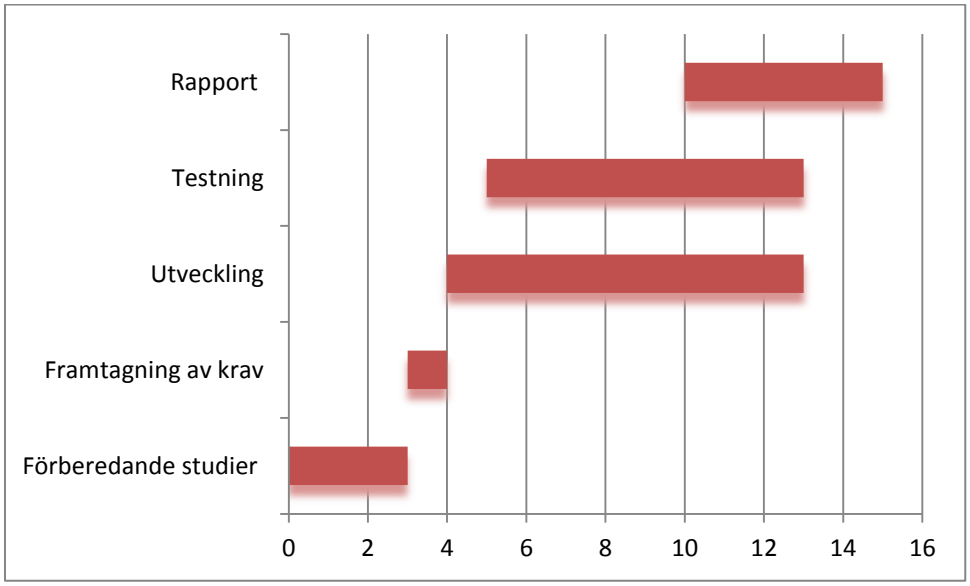
1.5 Avgränsningar

Projektet ska ge svaret på om det finns möjlighet och om det är lämpligt att använda Skype i en larmapplikation. Att utveckla ett helt fungerande larmprogram är inte rimligt. Funktionaliteten i programmet är beroende av vad analysen ger för resultat och hur mycket som hinns med de veckor då utveckling sker.

2 Metod

Utvecklingen av larmprogrammet började med att bestämma krav. De krav som prioriterades var de krav som tillsammans utgjorde en enkel larmapplikation d.v.s. en applikation innehållandes databas, grafiskt gränssnitt och nödvändiga klasser för integration med funktioner i Skype API.

Ingen speciell programutvecklingsmetod användes under projektet. Arbetet delades upp i olika faser som överlappade varandra. Utveckling och testning skedde parallellt. Även kod kommenterades och testresultat antecknades för användning i rapporten under dessa faser, vilket innebär att dokumentationen delvis skedde parallellt med testning och utveckling. Rapportskrivandet enligt tidplanen nedan är således inte helt och hållet representativ för hur processen såg ut, men om man bortser från denna punkt och att några extra veckor användes för rapportskrivande så kan man säga att tidplanen följdes.



Figur 2: Tidplan

2.1 Förberedande studier

Den första veckan ägnades åt att välja programspråk, testa olika databaser och utvecklingsmiljöer. Programmeringsspråken som passade bäst till projektet var Visual Basic och C#. Detta berodde på att de testprogram som redan fanns gjorda av Trex använde sig av .NET framework och Skypes API.

Fördelen med att välja C# hade varit att det liknar Java mer än VB.NET, vilket datateknikutbildningen handlat till stor del om. Valet föll trots det på VB.NET eftersom Trexs kunskap och testprogram då kunde användas bättre, vilket var

till projektets fördel. Med VB.NET som programspråk kunde även projektets utveckling diskuteras enklare med Trex och framtida vidareutveckling av programmet göras mer okomplicerat.

För att få kunskap om språket användes övningar på internet. När det gäller databasutveckling i Visual Basic användes ”Murach’s ADO NET 4 database programming with VB 2010” där man kan läsa om Visual Basic 2010 och dess alternativ när det gäller databasutveckling. Till boken finns det övningar att hämta på en hemsida.

3 Teknisk Bakgrund

3.1 Skype

Skype är ett program där man kan kommunicera med hjälp av IP-telefoni. Samtal kan ske mellan datorer, datorer och telefoner, datorer och mobiler. Det finns även chat tillgänglig där man kan chatta och dela filer[1]. Skillnaderna mellan de olika samtalsalternativen förutom de uppenbara är kostnaderna.

3.1.1 Samtalskostnader

Om man ska ringa med sitt Skype-konto till ett annat Skype-konto är det gratis, men om man ska ringa till mobiler eller telefoner måste man betala in pengar med hjälp av kreditkort eller annat betalningssätt. Vid samtalskredit kan man åt gången sätta in 100 kr eller 250 kr[2].

Öppningsavgiften för samtal ligger på mellan 50 öre och en krona beroende på om man ringer till mobil eller telefon. Minutavgiften ligger på 21 öre/minut för uppringning av fasta telefoner och 2,63 kr för mobiler.

För månadsabonnemang kan man ringa obegränsat mycket i Sverige till fasta telefoner för kostnaden 55 kr. Om man behöver ett månadsabonnemang för uppringning av både mobiler och fasta telefoner finns det inget obegränsat alternativ, då måste man abonnera på ett visst antal minuter, t.ex. 120 minuter för 172 kr eller 400 minuter för 241 kr i månaden[3].

För köp av abonnemang för online-nummer för vanliga telefoner och mobiler att ringa till finns det två alternativ. Det ena är 3-månaders för 185 kr och 12-månaders för 500 kr[4].

3.1.1.1 Lämpligaste alternativen för larmprogrammet

För just denna applikation är det bäst att köpa abonnemang för ett visst antal minuter i månaden, helst max som är 400 min. Minutpriset vid uppringning av mobiler blir då endast 60 öre/minut jämfört med 2,63 öre/minut med Skype-

kredit. Detta abonnemang kan man naturligtvis avsluta om man senare upptäcker att det inte finns behov för just det antalet minuter och då istället ta ett billigare abonnemang. När det gäller ett online-nummer för kontot, räcker 3-månadersabonnemang.

Eftersom den påtänkta tjänsten ska kunna nå en kunds kontakter på bästa sätt kommer troligtvis de flesta att ange sina mobilnummer istället för fasta telefonnummer som kontaktnummer i databasen. Därför är ett obegränsat månadsabonnemang där man endast kan ringa fasta telefoner inte passande för kontot till larmprogrammet, även om möjligheten att köpa kredit upptill för samtal med mobiler finns.

3.1.2 Skype P2P

Skype använder sig av Skype Protocol, ett hemligt protokoll där P2P-teknik används[5]. Datorer kopplas samman i ett datornätverk där varje enskild dator kan agera klient eller server för någon av de andra datorerna i nätverket; i P2P sammanhang brukar man kalla detta för att datorerna blir noder. Fördelen med att använda P2P nätverk jämfört med att använda klient-server- nätverk är att kostnaderna blir mindre eftersom man kan utnyttja slutanvändarens processor- och nätverkskraft. Om en ny nod kopplas till nätverket bidrar den med mer processorkraft och bandbredd till hela nätverket.

3.1.3 Tekniker för effektivisering av Skype

För att Skype ska bli effektivt används olika tekniker:

1. Ett av de största problemen man stöter på i P2P nätverk är brandväggar, men i Skype P2P är detta inget problem. Användare som begränsas av brandväggar kan ta hjälp av klienter som inte har brandväggar eller av klienter med publika dirigeringsbara (routningsbara) IP-adresser vid kommunikation med andra klienter bakom brandvägg. Detta görs genom dirigering av samtal. Säkerheten äventyras inte av detta eftersom trafiken samtidigt är krypterad. Man har dessutom utvecklat olika tekniker för att förbigå inställningar av brandväggar eller gateways (nätverksnoder) som i P2P sammanhang hade gjort kommunikation i Skype omöjlig. Tack vare dessa tekniker fungerar således Skype bakom de flesta brandväggar och gateways.

2. Många program där man använder snabbmeddelande använder sig av centraliserade bibliotek, vilka håller reda på IP-adress och status för en viss användare. Detta alternativ har visat sig kräva stora resurser när antalet användare blir många. Istället för centraliserade bibliotek använder Skype ett globalt decentraliserat användarregister för att hålla reda på vilka användare som är tillgängliga att upprätta anslutning med. Detta gör man genom

supernoder. Det som skiljer sig mellan noderna och supernoderna i Skype är att supernoderna inte dirigerar några samtal, de håller istället reda på information om andra användare[6].

3.1.4 Skype4Com

Skype4Com är en COM DLL[7], med andra ord en Active X komponent. Den kan användas i utvecklingsmiljöer med stöd för Active X som t.ex. Visual Studio. När Skype4Com blivit importerat i ett projekt är Skype API tillgängligt. Det representeras av ett bibliotek. I biblioteket finns det klasser med möjlighet att skapa objekt, använda egenskaper och hantera händelser, vilka sedan kan användas till att kommunicera med Skype och hantera telefonsamtal och andra viktiga funktioner[8].

Mycket av Skype4Com är händelsebaserat, vilket betyder att utvecklare kan skriva kod som hanterar händelser för de flesta funktioner i Skype.

3.2 .NET Framework (Dotnet Framework)

.NET Framework är ett mjukvaruramverk. Det körs huvudsakligen på Microsoft Windows. Ramverket förser en utvecklare med komponenter som kan användas tillsammans med egen kod. De två komponenter man framförallt kan använda för att beskriva ramverket är klassbiblioteket ”Base Class Library” och exekveringsmotorn ”Common Language Runtime”[9].

3.2.1 Base Class Library (BCL)

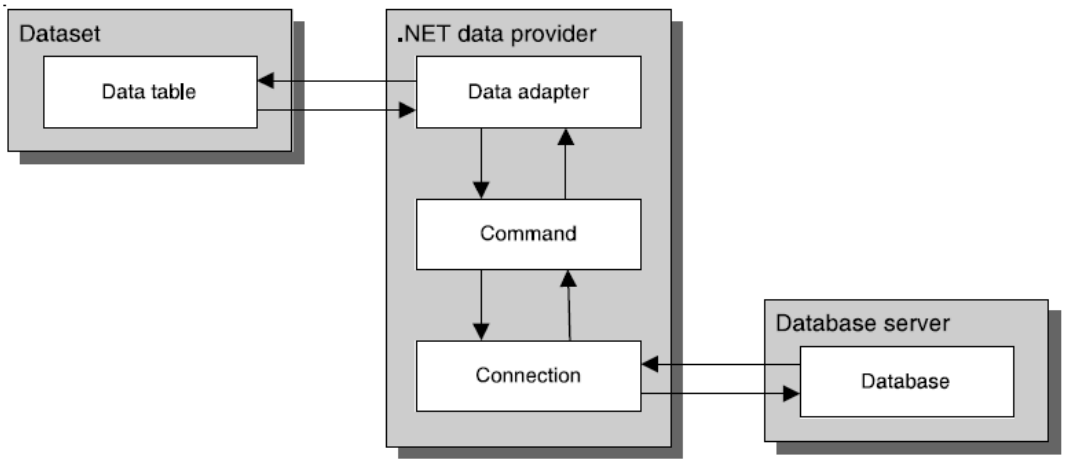
Klassbiblioteket BCL innehåller testad och återanvändbar kod framförallt för vanliga operationer inom programmering. Tack vare det blir programmeringsprocessen i dotnet mer tidseffektiv. När det gäller webbutveckling t.ex. för utveckling av dynamiska hemsidor, program och tjänster, kan man använda sig av ramverket ASP.NET.
Biblioteket innehåller även klasser för interaktion med databashanterare som t.ex. Microsoft SQL Server och MYSQL, dessa klasser kallas ADO.NET-klasser[10].

3.2.2 ActiveX Data Objects.NET (ADO.NET)

ADO.NET är API:n för dataåtkomst för databasapplikationer i dotnet. För att utveckla databasapplikationer i dotnet använder man ADO.NET-klasserna. När det gäller att arbeta med data i databaser finns det två olika alternativ att välja mellan. Alternativen är att arbeta med dataset eller direkt med databasen[11].

3.2.2.1 Dataset

Klasserna i ADO.NET ger utvecklaren alternativ när det gäller datahantering. Ett av alternativen är att använda dataset. Ett dataset används för att lagra data från databasen. Arbete med dataset innebär att man inte arbetar direkt med databasen. Med detta tillvägagångssätt behövs ingen anslutning med databasen förutom när man måste hämta och uppdatera data. Denna uppbyggnad av en databasapplikation kallas för en frånkopplad dataarkitektur. Samverkande klasser och dataflöde vid arbete med dataset är illustrerat i figur 3.



Figur 3: Dataflöde mellan klasser vid användning av dataset

I figuren ser man att dataset består av tabeller. De motsvarar databasens tabeller. Lagret mellan applikation och databas är ”.NET data provider”, för SQL server är det ”SQL server data provider”. De klasser som tillhandahålls av den sköter hämtning och lagring av data.

Processen kan förklaras på följande vis: ”Data adapter” utför SQL-frågor som sparats i objekt av klassen ”Command”. Command-objekt använder sedan objekt av klassen ”Connection” till att hämta eller spara data i databasen. Vid uppdatering av data i databasen d.v.s. när borttagning eller insättning skett i ett dataset, kan man med hjälp av egenskapen ”Update” i ”Data Adapter” automatiskt uppdatera ändringarna till databasen.

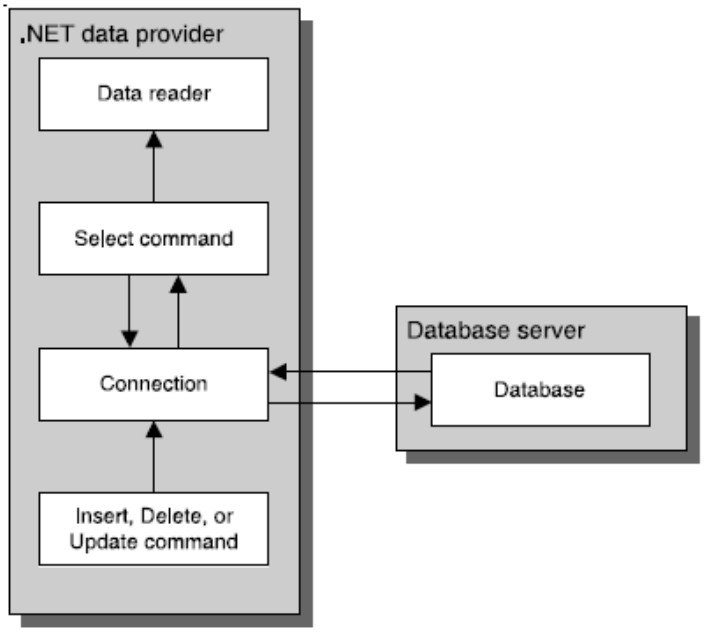
En fördel med att använda dataset i applikationer är att man använder mindre datorresurser vid datamanipulation eftersom man inte behöver vara uppkopplad till databasen om man vill jobba med data i den. Uppdatering av databasen blir även den mindre komplex eftersom den sköts automatiskt genom jämförelse mellan dataset och databas.

Dock kan fördelen beskriven ovan bli till en nackdel om många arbetar samtidigt med databasen och många försöker uppdatera data. Data kan då bli inaktuell eller skrivas över oavsiktligt. Detta samtidighetsproblem kan man åtgärda med frekvent uppdatering av databas eller undantagshantering, men detta motverkar då till viss del syftet med dataset eftersom man då behöver tillgång till databasen mer frekvent trots den frånkopplade dataarkitekturen [12].

3.2.2.2 Arbeta direkt med databas

Om man inte vill använda dataset kan man istället arbeta med databasen direkt. Den största skillnaden är att dataflödet inte styrs av en data-adapter.

Varje Command-objekt exekveras var för sig. Resultaten av frågorna hanteras även de enskilt. Vid varje insättning, borttagning eller annan operation öppnas och stängs en anslutning till databas. Hur flödet ser ut kan man se i figur 4.



Figur 4: Dataflöde vid arbete direkt med databasen

Ett Command-objekt innehållande SQL-kod exekveras och hämtar data från databasen genom klassen ”Connection”. Resultatet kan sedan läsas med hjälp av klassen ”Data reader”. Objektet läser rad för rad, vilket innebär att man måste anpassa den datastruktur som sparar resultatet efter detta.

Fördelen med att inte använda dataset är att man får mer frihet att bestämma och anpassa sitt program. När man använder dataset är det mycket kod som

genereras, vilket innebär att det kan vara svårt att få en överblick och känsla av kontroll över koden. Som följd blir det svårare att felsöka koden.
Utan dataset kan man även skapa sina egna klasser för hantering av data, dessa kan sedan ingå i klassbibliotek som kan användas i flera olika applikationer.

De nackdelar man stöter på utan dataset är bl.a. att mycket kod måste skrivas för att utföra det som är simpelt med dataset. Med mer kod krävs mer tid och mer kunskap av databaser, SQL, programspråk, felsökning och felhantering[13].

3.2.3 Common Language Runtime (CLR)

CLR är en exekveringsmotor ”Runtime Environment”. Den exekverar och kompilerar koden för dotnetspråk. Den är resultatet av en implementation av ”Common Language Infrastructure” förkortat CLI, som är en specifikation skapad av Microsoft, över hur en miljö ska vara för att kunna köra flera högnivåspråk på datorer med olika hårdvaruarkitekturer och ramverk utan att språken behöver anpassas[14]. Miljön beskrivs i nästa stycke.

Kompilatorer för olika programspråk ger CLR metadata. Metadata baseras bland annat på typer och referenser för programkoden skriven. CLR använder sedan informationen till att hantera och lokalisera klasser, instanser och generera maskinkod. Eftersom kompilatorer och verktyg har detta samarbete med CLR och följer CLI:s riktlinjer blir språkinteroperabilitet en möjlighet. Objekt skapade i ett av språken kan utan problem kommunicera med objekt i andra språk[15].

All kod som exekveras av CLR kompileras först till CIL (Common Intermediate Language) bytekod. En JIT-kompilator i CLR översätter därefter koden till maskinkod som sedan processorn utför[16].

För språk som Visual Basic innebär CLR att språket blir mer objektorienterat. Några exempel är att man kan skapa hierarkier genom att klasser kan ärva, att programmeraren kan skapa interface och att man kan få en mer strukturerad undantagshantering[17].

3.2.4 Visual Basic .NET (VB.NET)

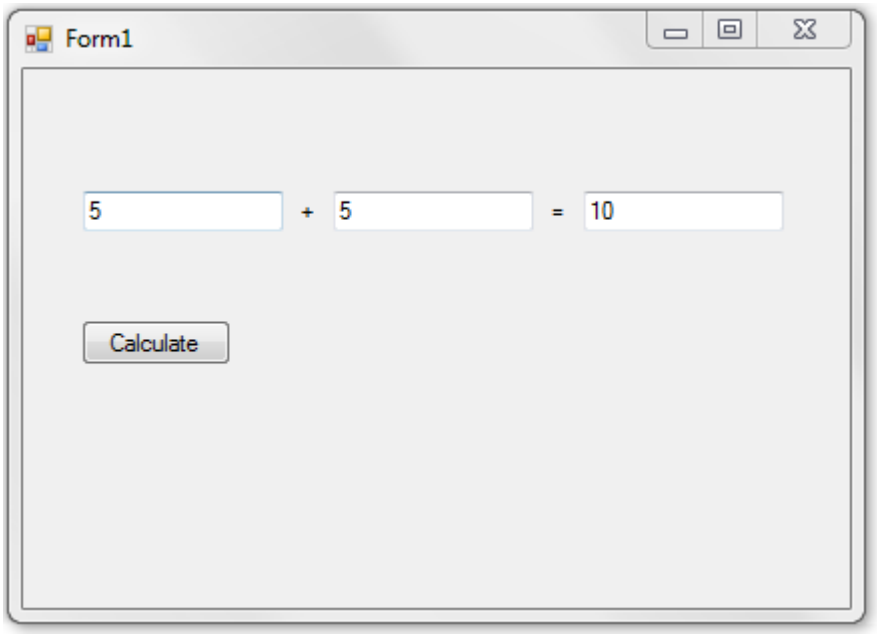
”Visual” i Visual Basic står för att det är språk där man utvecklar applikationer med hjälp av ett grafiskt gränssnitt. Detta gränssnitt brukar användas för att skapa prototyper, inte bara av .NET-utvecklare utan även av utvecklare för andra språk. Det är just lättheten med att testa programfunktioner genom ”drag-and-drop” av komponenter och kontroller som gör det till ett bra språk för utveckling av grafiska användargränssnitt[18].

”Basic” i Visual Basic är en förkortning av ”Beginner’s All Purpose Symbolic Instruction Code”. Det är ett språk utvecklat på 60-talet som skulle göra det enkelt att förstå programmering och göra applikationer. Med BASIC kunde man inte utveckla sitt program på ett visuellt sätt, det var endast textbaserat[19].

Vad som skiljer sig mest mellan VB och VB.NET är att VB.NET är mer objektorienterat och kan kommunicera med andra språk.

Vid utveckling av program i VB.NET brukar man använda Visual Studio. I ”Microsoft Visual Basic 2010”, en del av Visual Studio 2010, använder man sig av olika menyer där man kan dra och släppa komponenter och kontroller på en form. En ”Windows form” är en yta på vilket information presenteras visuellt och interaktion med användare kan ske. Kontroller som t.ex. textrutor kan läggas på ytan, användaren kan sedan använda dessa för att ge input till programmet. Input kan ske med hjälp av en knapp, även det en kontroll. Hur programmeraren bestämmer vad som ska göras och när det ska göras, görs genom att hantera och utnyttja de händelser som skapas för Form-objektet och varje kontroll när de blir aktiva. Det är därför man brukar säga att VB.NET är ett händelsedrivet språk.

I exemplet nedan adderas innehållet i de två första rutorna vid knapptryck. Resultatet visas sedan i den tredje rutan.



Figur 5: Exempelprogram med additionsräkning

Koden för programmet ser ut på följande sätt:

```
Public Class Form1

    Private Sub Button1_Click(sender As System.Object, e As System.EventArgs)
    Handles Button1.Click
        Dim number1, number2, result As Integer
        number1 = TextBox1.Text
        number2 = TextBox2.Text
        result = number1 + number2
        TextBox3.Text = result
    End Sub

End Class
```

”Sub” står för subrutin. Just denna subrutin hanterar knappens klickhändelse. När man deklarerar en variabel använder man ”Dim” som står för ”Dimension”. Typen av variabel skriver man sist. I detta fall är typen integer. För att få tillgång till textrutornas innehåll används punknotation följt av egenskapen ”Text”. Addition sker sedan precis som i andra programmeringsspråk med operatorn +. Resultatet av additionen sparas därefter i en variabel som slutligen tilldelas den tredje textrutan.

4 Utveckling

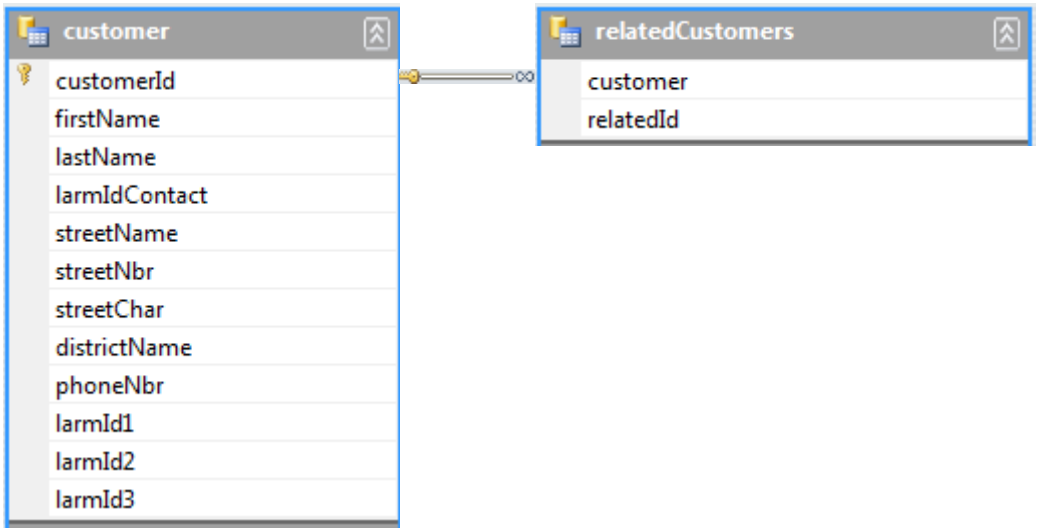
Utvecklingen inleddes i slutet av de förberedande studierna. Övergången mellan dessa faser underlättades med hjälp av de exempelprogram som utvecklats med hjälp av övningar och Trexs exempelprogram.

4.1 Databas

Microsofts SQL server var den databashanterare som använts i övningarna från boken ”Murach’s ADO.NET database programming with VB 2010 ”. Enkelheten med denna hanterare med VB.NET-kod gjorde att valet av SQL-server föll på den.

4.1.1 Skapandet av databas

Som beskrivet i avsnitt 1.2 behövdes två tabeller för att skapa en kunddatabas där varje kund har en unik kontaktlista, en för kunderna (customer) och en för kontakterna (relatedCustomers).



Figur 6: Tabeller i databasen

I figur 6 kan man se tabellerna och deras kolumner. Relationen mellan tabellerna är mittpartiet där nyckel och oändlighetstecknet motsvarar en-till -många-relation.

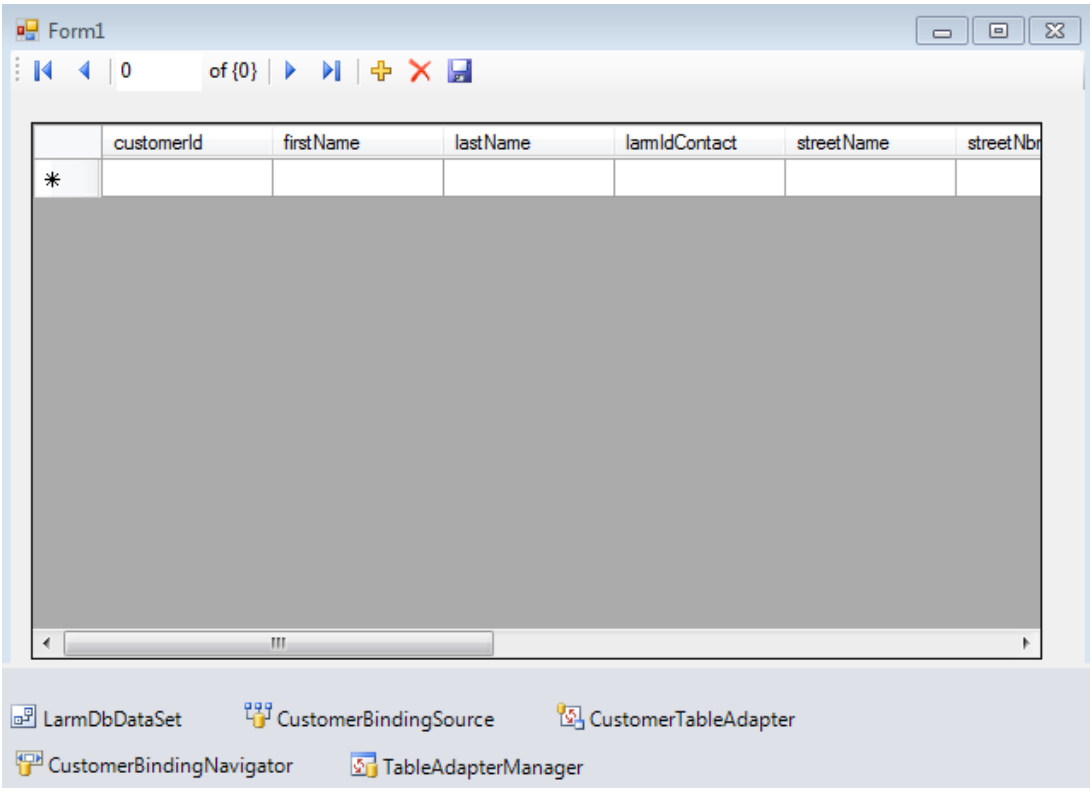
Primärnyckeln i customer är customerId. Detta id tilldelas varje ny kund automatiskt. I tabellens script är primärnyckel-raden inställd på ökning med ett för varje ny kund, vilket resulterar i att alla kunder får unika id-nummer. Den främmande nyckeln i relatedCustomer är raden customer. Den innehåller med andra ord samma id som i customerId. Något man tyvärr inte urskiljer i figuren är att raderna customer och relatedId tillsammans bildar en primärnyckel för att förhindra dubbletter i kontakter.

För att skräddarsy databasen på enklast sätt användes ”Microsoft SQL Server Management Studio”. Det är ett tilläggsprogram till ”Microsoft SQL Server” där man bl.a. kan skapa en databas helt utan ett script eller med ett script. Det alternativ som passade bäst för detta projekt låg mittemellan alternativen.

För att vara försäkrad om att vissa inställningar i databasen inte skulle ställa till problem och slösa tid om de implementerades fel, användes några av tilläggsprogrammets funktioner. Efter generering av ett script med dessa inställningar, kompletterades det med egen kod. Det slutgiltiga databasscriptet kördes till sist i tilläggsprogrammet för att skapa en databasfil för användning i Visual Studio.

4.1.2 Val av struktur för hantering av data

När databasfilen blivit importerad i programprojektet var det dags att bestämma hur databaskommunikationen skulle ske och med hjälp av vilka datastrukturer. Några av exempelprogrammen från den förberedande fasen använde sig av dataset samt en inbyggd funktion i Visual Studio som skapar en grafisk sammanställning av raderna i en tabell genom att man drar en av databasens tabeller från en sidomeny och släpper på ett form-objekt. I figur 7 har kundtabellen customer, vilken i det här exempelprogrammet är en del av ett dataset tillsammans med relatedCustomers, släppts på ritytan Form1.



Figur 7: ”Drag-and-drop” av tabell på ett form-objekt

En kontroll som hanterar representationen av data skapar ett rutnät, detta kan ske p.g.a. att Visual Studio använder sig av komplex databindning, vilket

innebär att kontrollen blir bunden till alla element i rutnätet d.v.s. alla rader och kolumner. Till följd av rutnätet skapas fyra nya objekt för att hantera automatisk navigation, tilläggning, borttagning, uppdatering av kundtabellen i databasen. Deras namn kan man se nedanför rutnätet. Med rutnätet kan man sedan lägga till sql-frågor som bestämmer hur data ska laddas in och var.

Denna enkla funktion till att hantera dataset verkade lockande till en början eftersom erfarenheten av VB.NET var begränsad, men ju längre tiden gick, tills dess att förberedelsefasen och utvecklingsfasen började överlappa varandra, kom insikten av att ju mer förenklat det grafiska användargränssnittet samt databaskommunikationen gjordes desto mer begränsat och komplicerat blev utvecklingen av resterande delar av programmet. Det var dock inte bara denna inbyggda funktion för att hantera dataset som den negativa erfarenheten av dataset baserades på. Flera av de klasser som gjordes med dataset i förberedelsefasen var svåra att felsöka och göra användbara i andra klasser.

I avsnitt 3.2.2.1 och 3.2.2.2 tas nackdelar och fördelar upp om dataset. De nackdelar som var mest märkbara i programmen var svårigheterna med att felsöka och göra samarbetande klasser. Nackdelen när det gällde samtidighet var inget argument mot dataset i detta projekt eftersom databasen inte ska användas av flera samtidigt.

Istället för dataset användes listor i databasklassen samt ett grafiskt användargränssnitt bestående av flera olika klasser (se nästa avsnitt).

4.2 Klasser

KundDb

Klassen som kommunicerar med databasen heter KundDb. I den finns metoder för att ta bort kund, lägga till kund och ändra kunddata i databasen. Det finns även metoder för att lägga till och ta bort kontakter i en kunds kontaktlista. De beskrivna metoderna var metoder som redan var planerade i den förberedande fasen. De oplanerade men förväntade metoderna som tillkom i utvecklingsfasen var en sökningsmetod som söker på för- och efternamn och två andra sökmetoder, vilka försöker hitta en kund med ett visst id eller telefonnummer. Till sist gjordes metoder för hämtning av kunder och kontakter för visualisering av databasen. Dessa returnerar listor av kundobjekt. Exempel på en sådan metod är metoden GetKunder()

```
Public Shared Function GetKunder() As List(Of Kund)
    ..................
    Return kundLista
```

Beslutet att inte använda dataset innebar att metoder för datahämtning och sparning behövdes. Anledningen till att just liststrukturen blev datastrukturen

för sparning av data var för att den visade sig passa bra för att spara kundobjekt i på ett simpelt sätt och som gjorde visualisering av databasdata enklare i GUI-klasserna.

Hjälpklass för GUI:

Kund

Kundklassen är en hjälpklass till GUI-klasserna. Med klassen skapar man kundobjekt. Dessa objekt har egenskaper som motsvarar kundtabellen i databasen. Eftersom dataset inte används måste svaren på SQL-frågorna i databasklassen som rör kunder tas omhand på ett annat sätt. För att förenkla denna process används objekt som motsvarar en kund i databasen.

Varje egenskap sparas ner i ett objekt med hjälp av en dataläsare (se avsnitt 3.2.2.2). Eftersom raderna i tabellen läses av sekventiellt måste motsvarande egenskap i kundobjektet ha samma ordning i koden. När objekten fått alla egenskaper läggs de i en lista deklarerat för kundobjekt. Denna liststruktur gör det sedan enkelt för klasser såsom DatabaseGUI, AddNModify, AddContact och huvudklassen MultiCaller att hämta och använda kundlistor med sina egna metoder.

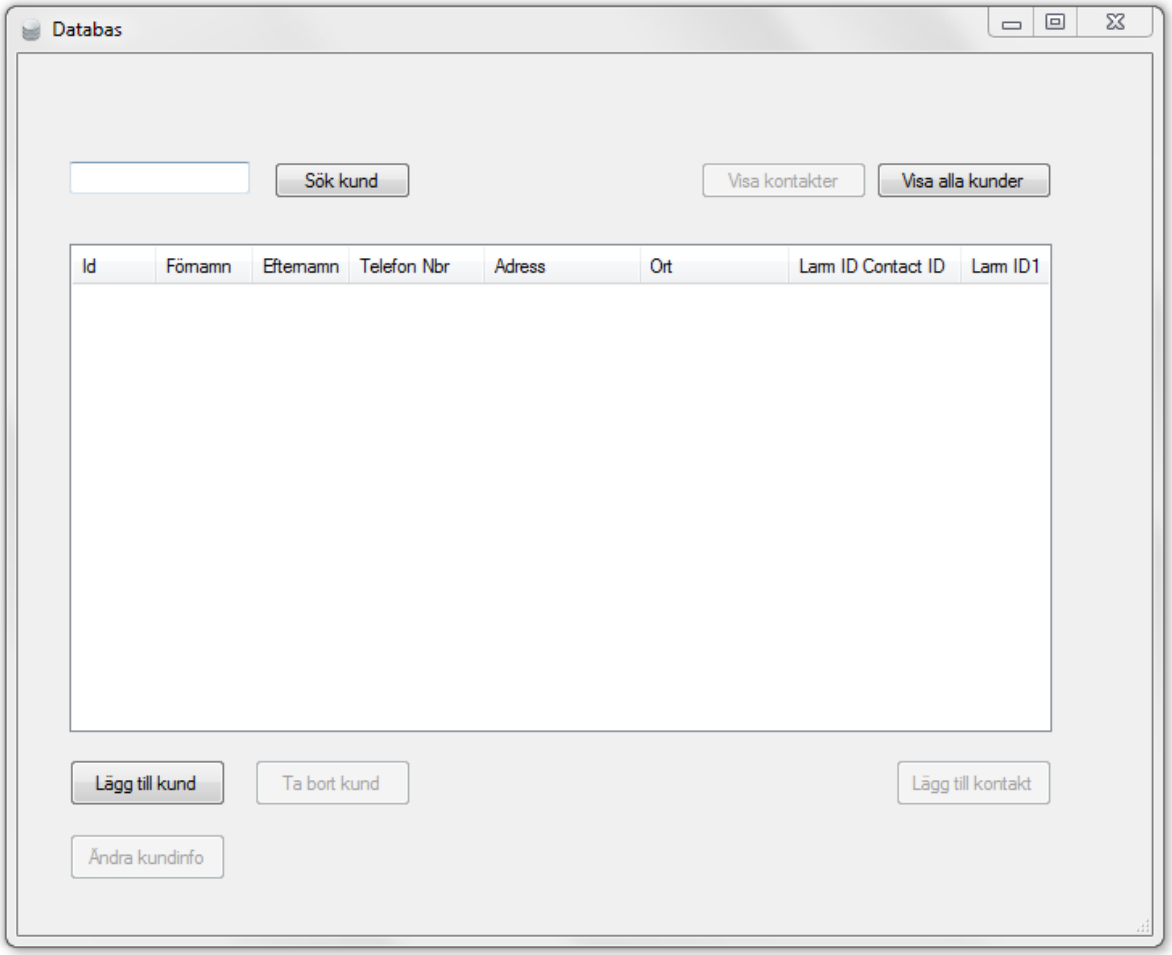
Egenskaperna för id och förnamn för en kund läggs till på följande sätt. Varje egenskap har Get- och Set-metoder.

```
Public Property KundFNamn() As String
        Get
            Return d_KundFNamn
        End Get
        Set(ByVal value As String)
            d_KundFNamn = value
        End Set
End Property



Public Property KundId() As Integer
        Get
            Return d_KundId
        End Get
        Set(ByVal value As Integer)
            d_KundId = value
        End Set
    End Property
End Property
```

De klasser som utgör GUI är följande:

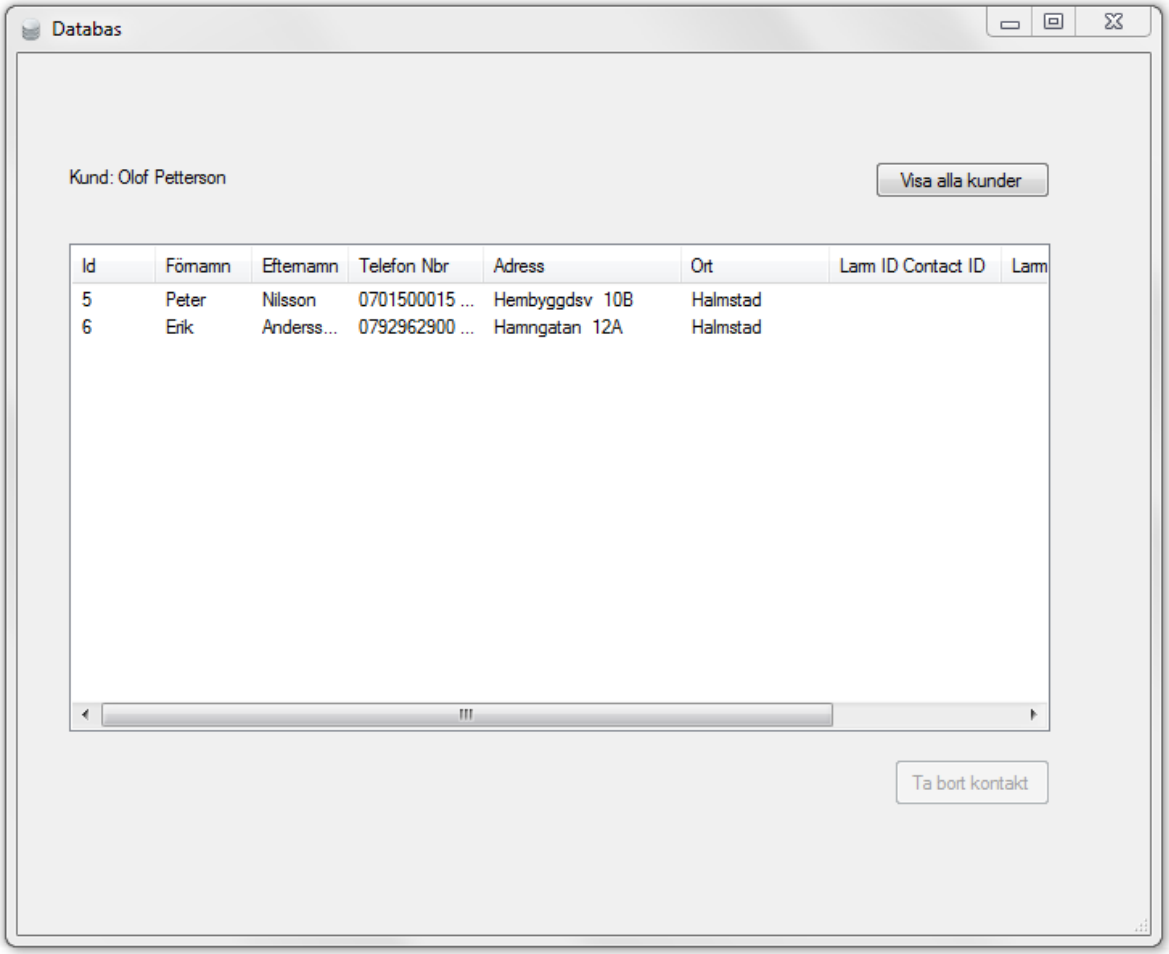
DatabaseGUI
I den här klassen sköts visualisering av databasen och CRUD (Create, Retrieve, Update, Delete). Rutnät likt det somt togs upp i 4.1.2., fast utan dataset, har lagts på ett form-objekt. Det engelska namnet på denna kontroll är ”listview”, på svenska blir det listvy och det stämmer bra överens med vad det är. Valet att presentera data på detta sätt var för att det passade bra ihop med listorna från databasklassen och för att det representerade tabellen på ett tillfredställande sätt.

En av egenskaperna en listvy har är att man kan lägga till kolumner. Ifyllning av dessa kolumner görs genom att använda listor skapade av databasklassens metoder. Listor som returneras är antingen kundlistor eller kontaktlistor. De olika listorna skiljer sig inte åt sett till de objekt de innehåller. Kontaktlistorna innehåller kundobjekt de med eftersom egenskaperna för en kontakt är samma som för en kund och kommer från samma tabell, nämligen kundtabellen.



Figur 8. DatabaseGUI

I klassen finns knappar för tilläggning, borttagning, ändring, sökning och visning (se figur 8). De knappar på bilden som inte är tillgängliga beror på att ingen kund i listvyn valts. När knapptryck av ”Visa alla kunder” sker och man väljer en kund blir de utsuddade knapparna synliga. Denna avaktivering och aktivering av knappar är till för att minimera risken att fel kund väljs.
Hur det ser ut när man väljer ”Visa kontakter” kan man se i figur 9.



Figur 9: Visualisering av kontaktlista

Kontakter visas i samma ruta som kunder. Det som händer vid ”Visa kontakter” är att listvyn töms på kunder och ersätts med kontakter och att kundknapparna, söknappen och knappen man tryckt på försvinner. Under utvecklingen hände det ofta att någon kund var i fokus som inte skulle vara det eller att ingen kund var i fokus vid knapptryck. För att slippa dessa fel implementerades kod som släckte eller avaktiverade knappar när de inte behövdes.

Skälet till att lägga ”Lägg till kontakt” i GUI-läget i figur 8 istället för i figur 9 är att man då har tillgång till alla kunder i listan, vilket kan vara behövligt om man vill se adresserna till kunderna man vill lägga till som kontakter. AddContact-klassen visar endast namnet på personen man vill lägga till.

AddContact

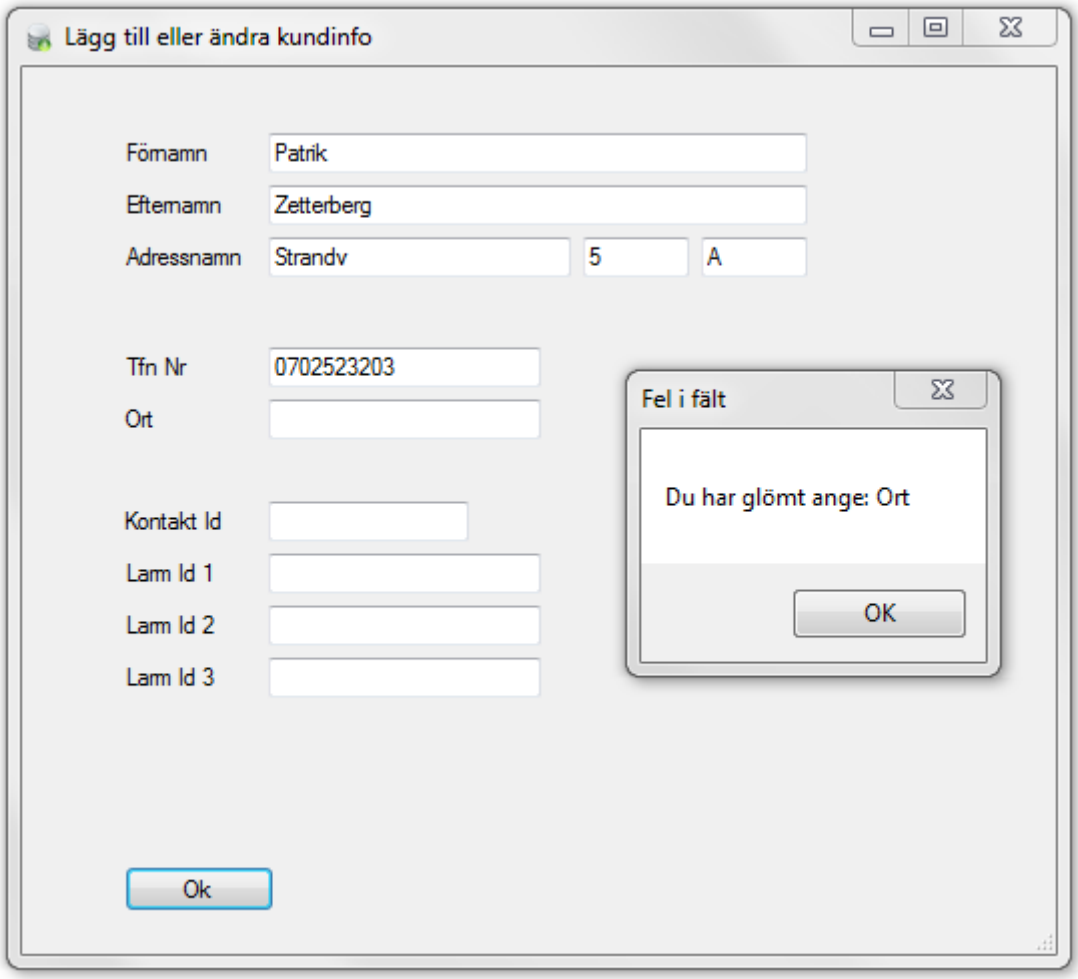
Med AddContact-klassen bestående av ett form-objekt kan man lägga till kontakter för en kund. På ytan finns en ”combobox” vilket är en kontroll där en lista över kunder som kan läggas till som kontakter visas vid fokusering. I klassen används databasklassens metod för att lägga till kontakter.

Figur 10: Tilläggning av kontakt

När man lagt till en kontakt dyker det upp ett fönster. I bilden ovan är tilläggningen lyckad. Om en kontakt redan finns i kontaktlistan eller om man försöker lägga till sig själv som kontakt meddelas detta istället.

AddNModify

Om man vill lägga till eller ändra information för befintlig kund används denna klass. I figur 11 på nästa sida kan man se hur det går till.



Figur 11: Fönster för tilläggning eller ändring av kundinfo

Denna klass kan man använda om man vill ändra en kunds information eller lägga till en ny kund. Om en kund är i fokus i DatabaseGUI-fönstret så vet klassen att en ändring av en befintlig kund ska göras i stället för tilläggning av ny kund. Om en ändring ska göras så hämtar den information om vald kund osv.

Vid tilläggning av ny kund kan man stöta på det scenario som figur 11 visar. Här glömdes ort att fyllas i och med hjälpklassen ValidData beskriven nedan visas passande felmeddelande.

ValidData

Det är väldigt viktigt att input som ges i textrutor i AddNModify är korrekt eftersom databasen har bestämda typer i tabellerna. ValidData-klassen är en hjälpklass. Den har som syfte att se till så att textrutor som behövs fyllas i blir ifyllda och att input för en viss ruta inte är felaktig. Felaktigt blir det t.ex. om man försöker ge strängvärden där det ska vara integer, ValidData anger då med hjälp av en popup-ruta på vilken rad felet finns.

Larm

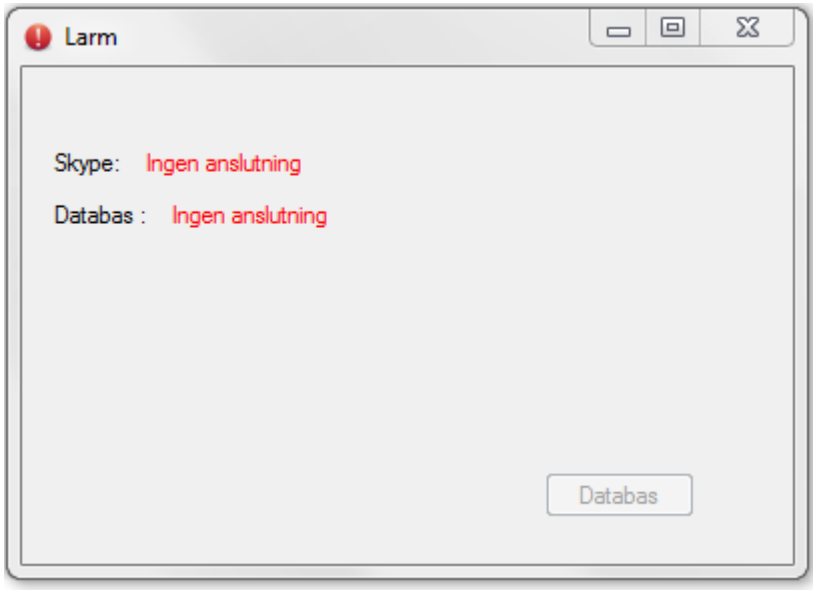
Larmklassen är en hjälpklass till GUI-klasserna och MultiCaller-klassen. Den sköter anslutningen med databasen. Genom att anropa dess enda metod GetConnection() får man anslutningen till databasen i form av en ”connectionstring”. Anslutningsträngen innehåller databasens instansnamn, i vilken mapp man hittar den och namnet man gett den.

4.3 Huvudprogrammet

Integration av klasser gjordes i huvudprogrammet MultiCaller. Det är en klass som påbörjades av Trex. I den fanns från början exempel på hur man kunde ringa upp telefoner med hjälp av Skype4Com och metoder som hanterade olika händelser för funktioner i Skype.

MultiCaller

Varje form-objekt har en händelse som heter ”Load”. Denna händelse inträffar vid start av ett fönster. MultiCaller-klassen hanterar dess load-händelse med hjälp av en subrutin. Just att välja denna klass som startfönster föreföll sig logiskt eftersom det var i denna klass inkommande samtal skulle hanteras och kontakter ringas upp med hjälp av databas. Med andra ord behövde denna klass vara i fokus hela tiden i väntan på samtal. I load-händelsen behövdes därför en check som kollade om Skype-klienten är igång och om databasen är tillgänglig. Innan Skype är igång och databasen är tillgänglig ser fönstret för den här klassen ut som i figur 12.

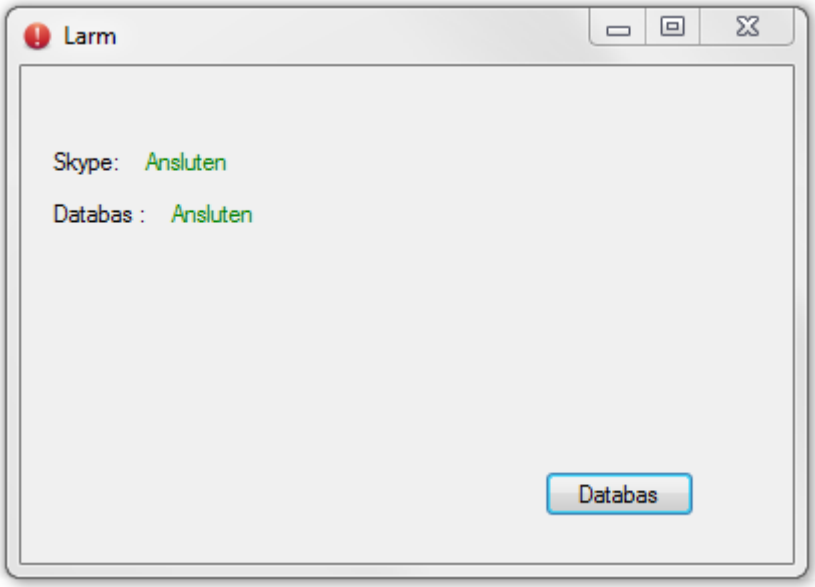


Figur 12: Ingen anslutning till Skype eller databas.

Om Skype inte är igång, startar MultiCaller upp Skype-klienten med kodraden ”skypeClient.Client.Start(true)”. Denna startfunktion är en del av Skype4Com-biblioteket.

Däremot blir det större problem om inte databasen är tillgänglig, för det kan inte larmprogrammet åtgärda. Då krävs manuell start eller konfiguration av databasen.

Om anslutningarna till Skype och databas fungerar ser det ut som i figur 13.



Figur 13: Databas och Skype tillgängliga.

När databasen blir tillgänglig fungerar även databasknappen nere i högra hörnet. Om man klickar på denna kommer man till DatabaseGUI.

Genom att studera och experimentera med Skype4Com-biblioteket, kom förståelsen efter en tid om vilka funktioner och händelser som skulle användas i samband med samtal. Ett samtal kan befinna sig i olika tillstånd, ett av dem är ”clsRinging” i samtalsstatusklassen ”TCallStatus”. I koden på nästa sida hanteras inkommande samtal i larmprogrammet.

```
Public Sub skypeClient_CallStatus(ByVal pCall As SKYPE4COMLib.ICall, ByVal Status As
SKYPE4COMLib.TCallStatus) Handles skypeClient.CallStatus

If Status = SKYPE4COMLib.TCallStatus.clsRinging And _
            (skypeClient.Convert.TextToCallType("INCOMING_P2P") = pCall.Type Or _
             skypeClient.Convert.TextToCallType("INCOMING_PSTN") = pCall.Type) And _
                Counter = 0 Then
                Try
                    ' Answer the Call.
                    pCall.Answer()
                    ' Make counter 1 to handle double signal
                    Counter = 1
                    ' Thread sleeps so functions don't collide
                    Thread.Sleep(1000)
                    ' Finish the Call.
                    pCall.Finish()
                    ' Thread sleeps so calls don't collide
                    Thread.Sleep(1000)
                    ' Call Contacts
                    callContacts(pCall)
                    ' Display call information.
                Catch ex As Exception
                        MsgBox("Unable to answer call: " + ex.ToString,
                        MsgBoxStyle.Information)
                End Try

            Else
                Counter = 0
End If

End Sub
```

"INCOMING_PSP" står för inkommande samtal från ett Skype-konto och "INCOMING_PSTN" från en fast telefon eller mobil. För att svara använder man funktionen answer() och för att avsluta Finish(). "pcall" är ett call-objekt som skickas med CallStatus-händelsen som parameter i subrutinen.

callContacts() är en metod i MultiCaller klassen som använder sig av databasklassens metoder för att ringa upp en kunds kontakter. Den har ett call-objekt som parameter. Olika metoder i databasklassen används beroende på typen av det inkommande samtalet. Koden nedan är från callContacts().

```
      Dim kund As New Kund
      Dim kontaktLista As List(Of Kund)
            ' Decides whether the incoming call is placed
            ' from a telephone or a Skype account
      If (skypeClient.Convert.TextToCallType("INCOMING_P2P") = Pcall.Type) Then
             kund = KundDb.GetKundByName(Pcall.PartnerHandle.ToString)
             kontaktLista = KundDb.GetKontakter(kund.KundId)
      ElseIf (skypeClient.Convert.TextToCallType("INCOMING_PSTN") = Pcall.Type) Then
             Dim pstn As String
             pstn = "0" + Pcall.PstnNumber.Substring(3, Pcall.PstnNumber.Length - 3)
             kund = KundDb.GetKundByTeleNbr(pstn)
             kontaktLista = KundDb.GetKontakter(kund.KundId)
      End If
```

När en kund hittats ringer Skype upp alla kontakter i kontaktlistan. Uppringningen av en kunds kontakter görs genom koden nedan.

```
        ' Gets the contacts for a specific customer
 kontaktLista = KundDb.GetKontakter(kund.KundId)

If kontaktLista.Count > 0 Then
   Dim kontakt As Kund
  For i As Integer = 0 To kontaktLista.Count - 1
   kontakt = kontaktLista(i)
   myCall = skypeClient.PlaceCall("+46" + kontakt.TeleNbr)'Calls the contacts
  Next
End If
```

4.4 Testning

4.4.1 Databas

Databasen testades genom script och funktioner i ”Microsoft SQL Server Management Studio”. Ett exempel på testscript är inläggningen av många kunder i databasen. Restriktioner (constraints) kollades också genom olika tester för input.

4.4.2 GUI

DatabaseGUIs sökfunktion och AddNModifys textrutor testades för olika input.

4.4.3 Huvudprogram

4.4.3.1 Uppringning

Uppringningen av en kunds kontaktlista testades till en början med hjälp av två Skype-konton. Detta för att det var smidigast och kostnadsfritt. Genom att göra ett konto i databasen med samma namn som Skype-kontot för larmprogrammet gick det att från ett annat konto på en annan dator ringa upp i larmprogrammets konto. Larmprogrammets konto hittade då det andra kontots kontaktlista i databasen och ringde upp den. Kontakterna för kontot var kopplade till telefonnummer tillhörandes tre mobiler och en fast telefon.

Vid svar på samtal upptäcktes att programmet fick felmeddelande om man svarade och avslutade inkommande samtal för tätt inpå. Detta löstes genom att låta programmet sova en sekund mellan answer() och finish(). Tråden fick även sova mellan finish() och utåtgående samtal till kontakter för att förhindra att samtal inte krockade (se avsnitt om MultiCaller).

I MultiCaller kan man även hitta en räknare som sätts till ett vid varje inkommande samtal. Vid varje mottagning av samtal innan denna åtgärd, skickades en dubbelsignal som inte gick att svara på. Detta fel gick inte att

hitta i koden. De felsökningar som gjordes gav inga svar på var denna signal kom ifrån, men tack vare forumet för Skype API på Skypes hemsida, hittades en tråd om detta fel. Det visade sig av inläggen att felet var ett känt problem. En utvecklare till API:n skrev där att man genom egen kod var tvungen att hantera denna signal.

Lösningen blev att vid varje nytt inkommande samtal sätta en räknare till ett och sedan använda räknaren i en if-sats för att filtrera bort den felaktiga signalen. Innan nästa riktiga signal inkommer nollställs räknaren (se kod för svar av samtal avsnitt 4.3). Den felaktiga uppringningssignalen som inte går att svara på går aldrig in i if-satsen och krockar således inte med pågående samtal.

När dubbelsignalen och krockande samtal hanterats blev samtalen tidseffektiva. Tidsfördröjningen för uppringning av kontakt ett (mobil ett) var precis som vid normala samtal d.v.s. samtal inte hanterade av Skype API. Tidsfördröjningen kontakter emellan var inte heller stor. Ungefär en halv sekund till en sekund var tidsfördröjningen mellan signalerna för mobil ett och mobil två, likaså mellan mobil två, tre och den fasta telefonen.

För att vara säker på att uppringning fungerade i riktiga situationer, skaffade Trex ett online-nummer på Skype. Sedan lades testkunder och testkontakter in i databasen med riktiga telefonnummer. Vid uppringning av numret från en av testkunderna, ringdes tillhörande kontaktlista upp. Detta fungerade utan problem.

4.4.3.2 Uppspelning

Kod för uppspelning och testning av ljudklipp för samtal gjordes sist i projektet. Anledningen till detta var att uppspelningen var beroende av att uppringningen fungerade. Testningen av uppspelning kunde inte göras på ett bra sätt annars.

Försök till att använda eget inspelat klipp för uppspelning i ett samtal fungerade till en början inte, men inlägg från utvecklare på forumet för Skype API skrev att endast klipp med 16 bitars sampling och 16khz mono kunde spelas upp. Efter att klippet anpassats till detta format fungerade uppspelningen för ett samtal.

Tyvärr fungerade detsamma inte för flersamtal. Vid test med två mobiler, fungerade uppspelning i mobil ett tills mobil två svarade. Då avbröts uppspelningen för mobil ett och klippet spelades upp för mobil två. Klippet fortsatte för mobil ett först när mobil två avslutade sitt samtal. Felsökning av koden gav inget svar på problemet, men som så många gånger innan under

test- och utvecklingsfasen fanns svaret på Skypes forum. Svaret visade sig vara att man inte kan spela upp ljud för mer än ett samtal åt gången. Endast ett samtal kan ges ljud-input.
Den ursprungliga idén om att spela upp ljudklipp för varje enskilt samtal slopades. En annan diskussionstråd tog dock upp konferenssamtal d.v.s. samtal där man kopplar ihop flera olika personer. Utvecklarna påpekade att ett konferenssamtal räknades som ett samtal.

Testning gjordes av uppspelning i konferenssamtal med två telefoner. Uppspelningen lyckades men inte utan problem. Nya problem som uppstod var t.ex. att den kontakt som svarar först får höra meddelandet från början, den andra lite senare.

4.5 Problem under utvecklingen

4.5.1 Information

Dokumentationen var bristfällig när det gäller vissa delar av Skype API. Detta ledde till att man fick förlita sig på Skypes forum. På forumet fanns utvecklare för Skype API, där de skrev om lösningar och fakta som inte gick att hitta i Skype4Com-dokumentationen eller i annan dokumentation.

4.5.2 Programrelaterat

De flesta fel som uppstod i programmet gick oftast att lösa utan hjälp från Skype-forumet eller internet genom användning av Try/Catch-satser och vanliga felsökningsmetoder. Samtalskrockar och dubbelsignaler var dock undantag. För att visualisera dubbelsignalen och överblicka samtalskrockar användes ett verktyg avsett för Skype som heter Tracer. Med Tracer kan man se samtalen i textform i realtid och skriva in kommandon direkt till Skype-klienten.

5 Resultat

Det utvecklade programmet har tillgång till en kunddatabas. CRUD (Create, Retrieve, Update, Delete) i denna görs möjligt genom GUI-klasser som använder sig av en databasklass innehållandes metoder med SQL-frågor. Ytterligare funktioner som finns tillgängliga tack vare dessa klasser är hantering av kontakter och sökning av kunder. En funktion som hanterar uppringning av kunder hanteras i en särskild samtalsklass vid namn MultiCaller. Den fungerar delvis för uppringning men behöver anpassas efter upptäckter gjorda gällande uppspelning av ljudklipp i samtal.

Någon funktion för uppspelning av ljudklipp i samtal finns i nuläget inte. Anledningen till att detta inte finns är oförutsedda omständigheter i slutet av projektet och tidsbrist.

Utgångspunkten var från början att det för varje enskilt samtal skulle spelas upp ljudklipp vid larmsignal. Denna samtida uppspelning i flera samtal kunde emellertid inte implementeras i programmet eftersom Skype API inte stöder det. Endast ett samtal kan vara aktivt och spela upp ljudklipp.

Fokus flyttades sista veckan i utvecklingsfasen från flersamtal till konferenssamtal. Ett konferenssamtal är endast ett samtal, men med flera deltagare. Test av uppspelning i konferenssamtal med två deltagare gav ett positivt resultat.

Sammanfattningsvis kan man säga att projektet inte resulterade i ett larmprogram som fungerar, utan i ett program och lösningar som kan användas i framtagningen av ett.

6 Slutsats och kommentarer

Projektet hade som syfte att analysera om Skype går att använda i ett larmprogram. Enligt det program och tester som gjorts så verkar den möjligheten vara god. De problem som stöttes på i slutet av projektet gällande Skype borde man med mer tid kunna lösa.

Delfrågorna ställda i problemformuleringen kan besvaras positivt. Låga samtalskostnader vid abonnemang gör att den färdiga tjänsten kan erbjudas billigt. Tidsfördröjning för uppringning blir minimal och kan ignoreras. Man kan således dra slutsatsen att den färdiga tjänsten kan bli effektiv, både kostnadsmässigt och tidsmässigt

Användandet av konferenssamtal istället för flera samtal kan ses som negativt i den bemärkelsen att det tillför ytterligare problem som måste lösas, t.ex. loopning av ljudklipp i samtalet. Tillförlitligheten i programmet blir däremot bättre med konferenssamtal eftersom samtalskrockar kan minimeras i och med att flera samtal sammanslås till ett och samma samtal.

Tillräckligt många test har dock inte gjorts av nuvarande program för att säga om Skype API och programmet är tillförlitliga i larmsammanhang. Dessutom kommer ändringar av programmet med b.la. konferenssamtal fordra än mer testning av programmet. Först när larmprogrammet genomgått ändringar och tester kan man dra en slutsats gällande denna delfråga.

6.1 Framtida vidareutveckling

När uppspelningsfunktionen implementerats och testats behövs integration av text-till-röst i programmet. Eftersom Get- och Set-metoder finns för varje kundegenskap borde skapandet av ljudklipp inte bli komplicerat.

I början av projektet diskuterades samtalshistorik med Trex. Det var ingenting som till sist blev ett krav. Anledningen var att historik inte ansågs vara en nödvändig del av larmprogrammet. Om man ändå i framtiden känner för att lägga till samtalshistorik kan detta göras tämligen lätt genom att studera de existerande klassernas kod och kommentarer, detsamma gällande databasens script.

Den funktion historiken hade fyllt hade varit att man då kunde sett över vilka kunder som svarar och inte svarar. Kunder vars statistik upplevts som dålig hade då kunnat få tillsägelser eller i värsta fall blivit uteslutna. D.v.s. för att få hjälp av andra måste man själv vara hjälpsam. Utan hjälpsamhet grannar emellan, kvittar det om programmet fungerar. För den färdiga tjänstens effektivitet bestäms framförallt av kunderna själva, inte av programmet.

6.2 Källkritik

De flesta källor (se avsnitt 8) är från hemsidor som tillhör Microsoft. Det finns därför inga skäl till att tro att de inte är trovärdiga. De referenser som är länkar till definitionssidor anses trovärdiga de med. Vid snabb jämförelse av definitionerna på andra sidor verkar de stämma bra överens med de sidor som är refererade.

I delen ”andra referenser” i referenser finns där en länk till ett forum. I testningsavsnittet (se avsnitt 4.4) står det att inlägg från utvecklare av Skype API använts för att lösa problem. Även om det är ett forum finns det ingen anledning till att betvivla korrektheten av inläggen, för de har varit till stor hjälp vid jakten på lösningar och förklaringar på de problem som uppstod under utvecklingen av programmet.

7 Terminologi

API – Application Programming Interface. Kommunikationsinterface för mjukvarukomponenter

COM – Component Object Model. Gränssnitt för objekt att kommunicera i program.

DLL – Dynamic-link libray. Filer innehållandes programfunktioner.

SQL – Structured Query Language

Form – Klass i .NET Framework. Används för utvecklande av GUI

P2P – Peer To Peer. Datorer (noder) sammankopplade till ett nätverk.

GUI – Graphical User Interface.

CRUD - Create, Retrieve, Update, Delete. Svensk översättning: Skapa, hämta, uppdatera och ta bort.

Runtime Environment - Miljö där kod körs. Svensk översättning: Exekveringsmotor

Runtime - Tillstånd då kod exekveras

VB - Visual Basic

.NET Framework – Mjukvaruramverk för .NET-språk

8 Referenser

[1] Skype. (2012). *What is Skype?*. https://support.skype.com/en/faq/fa6/what-is-skype (2012-11-11)

[2] Skype. (2012). http://www.skype.com/intl/sv/prices/skype-credit/. (2012-11-11)

[3] Skype. (2012). http://www.skype.com/intl/sv/features/allfeatures/call-phones-and-mobiles/ (2012-11-11)

[4] Skype. (2012). http://www.skype.com/intl/sv/features/allfeatures/online-number/ (2012-11-11)

[5]Bernadette, Shell. & Clemens, Martin. (2006). *Webster's New World Hacker Dictionary*. Wiley Publishing . Sida 243.

[6]Skype. (2012). *What are P2P communications?*. https://support.skype.com/en/faq/fa10983/what-are-p2p-communications (2012-11-11)

[7]Skype. (2011-11-16). *Skype4Com: What is it and what can it do?*. http://devforum.skype.com/t5/Desktop-API/Skype4COM-What-is-it-and-what-can-it-do/m-p/8155/highlight/true (2012-11-11)

[8]Skype. (2012). *Skype4Com* http://developer.skype.com/accessories/skype4com (2012-11-11)

[9]Microsoft Developer Network (MSDN). (2012). *Getting started with the .NET framework* http://msdn.microsoft.com/en-us/library/vstudio/hh425099.aspx (2012-11-11)

[10] Microsoft Developer Network (MSDN). (2012). *.NET Framework Class Library Overview*. http://msdn.microsoft.com/en-us/library/hfa3fa08.aspx (2012-11-11)

[11]Boehm, Anne. & Mead, Ged. (2011). *murach's ADO.NET 4 database programming with VB 2010*. Mike Murach & Associates Inc. Sida 34

[12] Boehm, Anne. & Mead, Ged. (2011). *murach's ADO.NET 4 database programming with VB 2010*. Mike Murach & Associates Inc. Sid 34-37, 41-40.

[13] Boehm, Anne. & Mead, Ged. (2011). *murach's ADO.NET 4 database programming with VB 2010*. Mike Murach & Associates Inc. Sid 38-39.

[14] Miller, S James. & Ragsdale, Susanne (2003). *Common Language Infrastructure Annotated Standard.* Addison-Wesley Professional.

[15] Microsoft Developer Network (MSDN). (2012). *Common Language Runtime Overview.* http://msdn.microsoft.com/en-us/library/ddk909ch%28v=vs.71%29.aspx (2012-11-11)

[16] Richter, Jeffrey. MSDN Magazine. (2000). *Microsoft .NET Framework Delivers the Platform for an Integrated, Service Oriented Web* http://msdn.microsoft.com/en-us/magazine/bb985004.aspx (2012-11-11)

[17] Microsoft Developer Network (MSDN). (2012). *What's New in the Visual Basic Language.* http://msdn.microsoft.com/en-us/library/y17w47af%28v=vs.71%29.aspx
(2012-11-11)

[18]TechTarget. Margaret Rouse. (2007). *Visual Basic (VB.)* http://searchwindevelopment.techtarget.com/definition/Visual-Basic
(2012-11-11)

[19]TechTerms. (2012). *BASIC* http://www.techterms.com/definition/basic
(2012-11-11)

Andra referenser:

Diskussionsforum där utvecklare av Skype API svarat. *Forumlänk.*
http://devforum.skype.com/
(2012-11-11)

Figur 3 och 4. Boehm, Anne. & Mead, Ged. (2011). *murach's ADO.NET 4 database programming with VB 2010*. Mike Murach & Associates Inc. Sidorna 35 och 39.