**Projektuppgift**

*DT207G backend-programmering*

**Projektuppgift**

DT207G

Niklas Borg

**MITTUNIVERSITETET  
Avdelningen för informationssystem och -teknologi**

**Författare:** Förnamn Efternamn, [xxxxx0000@student.miun.se](mailto:xxxxx0000@student.miun.se)  
**Utbildningsprogram:** Webbutveckling, 120 hp  
**Huvudområde:** Datateknik  
**Termin, år:** XX, 20XX

Sammanfattning

Sammanfattningen fungerar som en beskrivning av rapportens innehåll. Den ska underlätta en snabb genomgång av dokumentet och därför utgöra ett koncentrat av rapporten i sin helhet, det vill säga rymma allt från syfte och metod till resultat och slutsats. Exempel: ”Målet med denna undersökning har varit att besvara frågan… . Undersökningen har genomförts med hjälp av…. Undersökningen har visat att…”. Nämn inget stoff som inte behandlas i rapporten. Sammanfattning skrivs i ett stycke. 200-250 ord är en rekommendation. Hänvisningar till rapportens text, källor eller bilagor är inte tillåtet, utan sammanfattningen ska ”stå på egna ben”. Undvik såväl formler och matematiska symboler som kursiv och fet stil. Sammanfattningen kan avslutas med en uppräkning av nyckelord, som kan underlätta sökande efter rapporten i biblioteksdatabaser. Exempel:

**Nyckelord:** Människa-dator-interaktion, XML, Linux , Java.

Innehållsförteckning

Sammanfattning iii

Förord iv

Terminologi vi

1 Introduktion / Inledning 1

1.1 Bakgrund och problemmotivering 1

1.2 Övergripande syfte / Högnivåproblemformulering 2

1.3 Avgränsningar 2

1.4 Konkreta och verifierbara mål / Detaljerad problemformulering 2

1.5 Översikt 3

1.6 Författarens bidrag 3

2 Teori / Bakgrundsmaterial 4

2.1 Definition av termer och förkortningar 4

2.1.1 Exempel på rubriknivå 3 5

2.2 Att referera eller citera 5

2.3 Källförteckning och källhänvisningar 5

2.4 Automatiskt numrerade källhänvisningar 6

2.5 Illustrationer 7

2.6 Matematiska formler 8

3 Metod 9

4 Konstruktion / Lösningsalternativ 10

5 Resultat 11

6 Slutsatser / Analys / Diskussion 12

Källförteckning 13

Bilaga A: Dokumentation av egenutvecklad programkod 14

Exempel på underrubrik 14

# Introduktion / Inledning

I tidigare utbildning har du huvudsakligen kommit i kontakt med relativt väldefinierade problemställningar som formulerats av lärare och läromedel. Under projektkurser och examensarbete måste du som student själv lägga ned en hel del tankearbete på att definiera vad som egentligen är uppgiften. Denna analys redovisar du i rapportens inledande kapitel. Genom att beskriva vilket problem eller problemområde som du har valt att studera och ditt motiv för detta, ger du dessutom läsaren en allmän introduktion till rapporten.

Det inledande kapitlet motsvarar innehållet i den *projektplan* som ska presenteras några veckor efter det att examensarbetet påbörjats. Projektplanen kan även innehålla en tidplan för arbetet, och omnämna några av de källor du har för avsikt att läsa och referera (i kapitel 2), samt några ord om vilken metod (se kapitel 3) du har valt för att angripa problemet med.

Inledning eller Introduktion (välj en av rubrikerna) utgör kapitel 1, och kan t ex rymma nedanstående underrubriker. Försök att komma till ”kritan” så snart som möjligt. För att behålla intresset behöver läsaren få veta vad ditt arbete handlar om redan inom några få meningar. Tänk på att den som snabbt vill skapa sig en uppfattning om arbetet ofta läser endast rapportens sammanfattning, inledning samt slutsatser, eftersom dessa delar vanligen skrivs på en hög nivå utan alltför tekniska och matematiska detaljer.

## Avgränsningar

Fokus kommer ligga på backend-delen av hemsidan. Därav kommer grafiska element inte vara av vikt då det används för att demonstrera funktioners som finns tillgängliga i backend-delen.

Läsare kommer förutsattas att de besitter kunskap inom front-end programmering.

Front-end kommer baseras på Angular men kommer ej att beskrivas mer ingående då detta inte är fokus för rapport.

## Konkreta och verifierbara mål / Detaljerad problemformulering

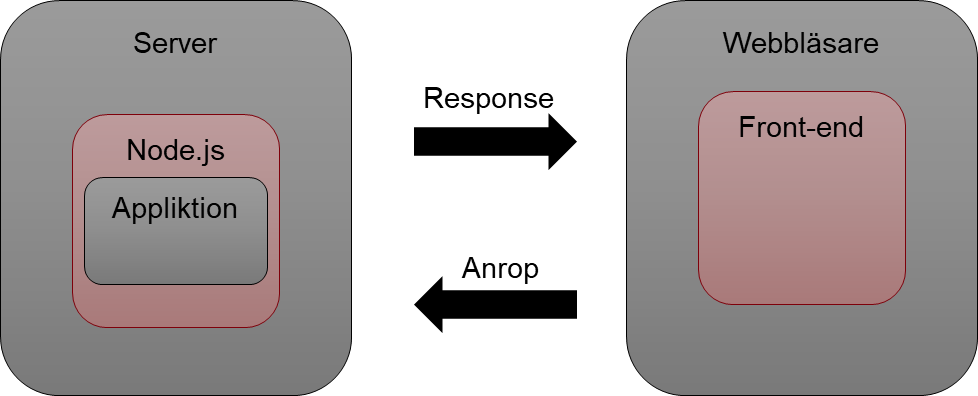
Målet med uppgiften är att skapa en serverfunktion som tillåter användare att läsa, korrigera samt uppdatera eller ta bort data från en databas. I detta fall kommer fokus att vara för en restaurangverksamhet där följande funktioner ska finnas.

* Skapa en meny
  + Radera data
  + Läsa data
  + Uppdatera data
  + Lägga till data
* Boka bord
  + Radera data
  + Läsa data
  + Uppdatera data
  + Lägga till data

# Teori / Bakgrundsmaterial

## Webbserver

I vanligaste fall består en hemsida av två huvudelar, en front-end och en back-end. Back-end exekveras på en server med någon form av exekverare som tex Node.js. Node.js kan tillåta utvecklare att köra kod på en server, dvs utanför en webbläsare (node.js, 2024).



Detta öppnar upp för möjligheter att skapa applikationer som kan förse och ta emot data från front-end och skicka denna information vidare till en databas eller liknande.

## Express

Express är ett ramverk som hjälper utvecklaren att bygga webbapplikationer och API. Detta ramverk erbjuder utvecklaren olika funktioner som beskrivs i nedanstående avsnitt (Express, 2024). För att använda detta så installeras paket tex via NPM och sedan importeras i den tänka javascriptfilen enligt nedan. (BYT UT BILD)

En bild som visar text, skärmbild, Teckensnitt, nummer

Automatiskt genererad beskrivning

### Middleware

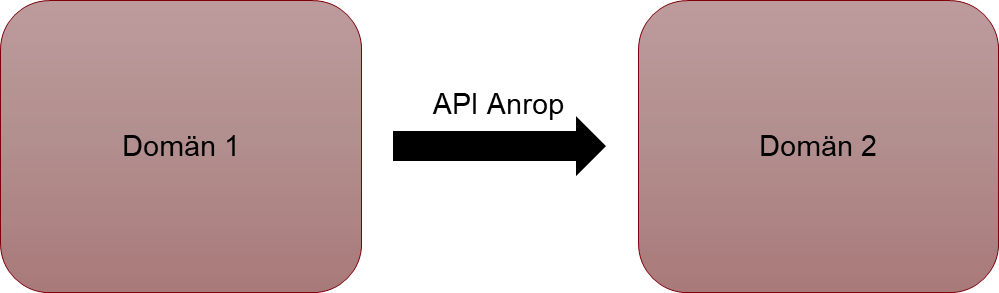
I Express kan utvecklaren anropa moduler från ramverket för att unna hantera olika funktioner, detta kan vara allt från inloggning på en hemsida till loggning (Express, 2024). Nedanstående avsnitt går igenom några av de middleware som finns tillgängliga i express.

#### JSON

Express erbjuder en funktion som konvertera JSON till javascript objekt. Detta är användbart för att snabbt och smidigt kunna hantera data i API anrop som kommer från användaren (Express, 2024).

#### Cors

Cross-Orgin Resource Sharing(Cors) ä ren function inbyggt i express som skyddar mot anrop över olika domäner. Cors tillåter utvecklare att kontrollera hur och vilka som ska tillgång till en server-resurs (Express, 2024). Detta skulle tex kunna vara att domän https//nagontin.se ska ha tillgång till https//någontinannat.se/api.



### Models

För att kunna skapa logik i backend applikationer hanteras detta i en så kallad model. Här så finns struktur för all data som hanteras, likt den som skapas i databas schema. Till detta så skapas också regler hur ett API anrop ska gå till. Detta är tex validering av inkommande data och liknande.

### Routing

För att kunna kontrollera vilken information eller funktion ett anrop till servern ska utföra så används routing. Detta tillåter utvecklaren att skapa funktioner som aktiveras vid ett API och specifika http-funktioner, se avsnitt CRUD gällande http funktioner.

Ett anrop till en back-end applikation kan vara genom ett

En bild som visar text, Teckensnitt, skärmbild, linje

Automatiskt genererad beskrivning

En bild som visar text, skärmbild, Teckensnitt

Automatiskt genererad beskrivning

## CRUD

Create, Read, Update och Delete(CRUD) är en “standard/norm” inom uteckling av API. Dvs ett mönster som är väldigt vanligt att förekommande för ett API. Detta kan samman kopplas med de http-metoder som finns och är beskriva nedan.

#### Get

Används för att hämta data från back-end.

#### Post

Används för att lägga till data i back-end.

#### Put

Används för att uppdatera befintlig data i back-end.

#### Delete

Används för att ta bort data från back-end.

## Säkerhet/Autentisering

För att kunna skydda information och resurser i back-end. För att göra detta finns flera metoder. Principen för dem är desamma, hantera känslig information så som lösenord eller personuppgifter.

### Grundläggande autentisering

En användare skickar lösenord samt användarnamn via ett http-anrop där header innehåller denna information. Det är viktigt att skydda dessa uppgifter under anropet då de är exponerade för nätverks ”sniffning” om dessa inte blir krypterade via tex HTTPs i anropet.

### Kryptering av lösenord

För att kunna hantera och skydda lösenord när dessa ska sparas i en databas behöver dessa krypteras för att skydda dem. Detta är för att skydda känslig information ifall någon obehörig skulle få tillgång till den.

Meningen med krypteringen av lösenord innan lagring är att dessa blir för ögat osammanhängande text. Det finns olika krypteringsmetoder en av dem är ”bcrypt” som bygger på att man har en krypteringsnyckel.



Att notera är att dekryptering inte är möjligt utan krypteringsnyckel.

### JWT's

Jason Web Token(JWT) används för att kunna styra om en klient ska ha tillgång till en resurs. För detta så är en JWT uppdelad i tre delar, headers, payload samt signatur.

#### Headers

Beskriver meatadata som skickas med i anropet till server från användaren. Detta kan tex vara…………………………

……………

#### Payload

Payload innehåller data som skickas till servern. Detta kan tex vara användarnamn, övrig data osv. detta kan se ut enligt nedan tex.

{

"tableID":"nikl3as2222",

"username":"niklas",

"password":"test",

"bookingDate":"2024-06-15T14:03:24Z",

"bookingId":"663e79613c42f4cc900ecde7"

}

#### Signature

För att kunna säkerställa att det är en behörig användare som anropar servern används en signatur som generas när en användare veriferar sig mot servern. När en användare veriferar sig så sker detta oftast genom att lösenord samt användarnamn skickas med i payload. Server tar emot dessa och om dessa är korrekta så svarar servern med en sträng som sedan kan användas av klienten för att verifera sig mot resurser i servern.

# Metod

För att kunna skapa denna sida kommer följande program/mjukvaror att användas.

Figma – design och utformning.

Angular cli – För att kunna skapa en angular applikation. Detta är ett kommando baserat tillägg som kan användas för att skapa delar till en Angular applikation.

Visualstudio code – Kod editor

# Konstruktion / Lösningsalternativ

## Design

Grundtanken med hemsidan har varit att hålla det enkelt och neutralt. Uppdelningen är enligt nedan med en klassik meny knapp upp till vänster med dropdown funktion. Utöver detta så är den en main del där innehåll visas upp genom en container. Sedan en footer längst ner.

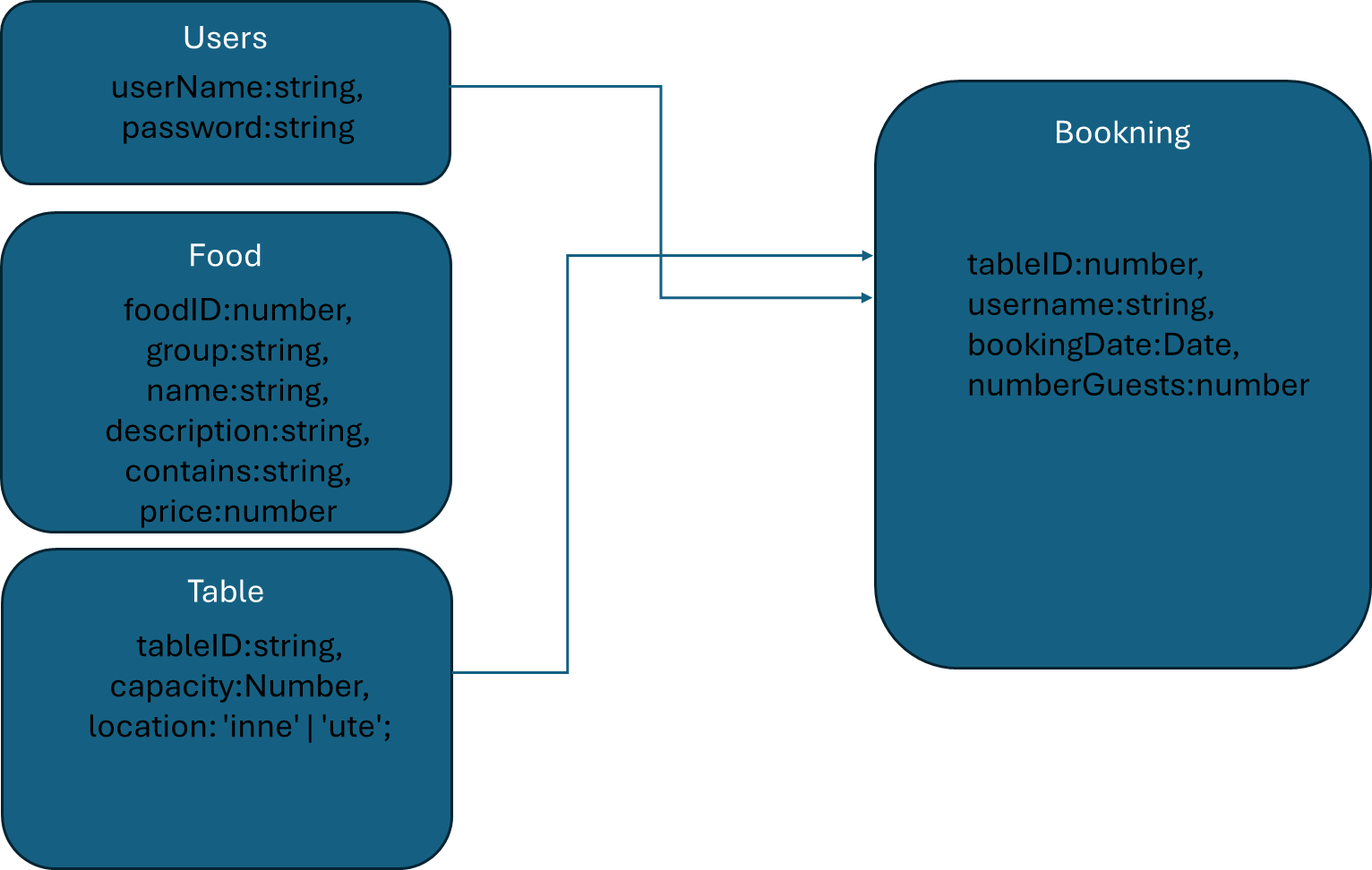
En bild som visar text, skärmbild, Rektangel, design

Automatiskt genererad beskrivning

## Databas

Databasen har blivit baserad på en NoSQL databas via MongoDB. Detta har blivit uppdelat i totalt 3 stycken dokument med referenser enligt nedan. Det finns en relation mellan Users och table till bookning för att man ska kunna binda en kund mot en bokning. Bord refereras mot, eftersom en bokning måste blockera upp andra användare från att boka samma bord under den tiden.

Food är till för att enkelt kunna visa meny till kunder online.



Varje del har ett eget schema och beskrivs nedan.

User

const userSchema = new mongoose.Schema({

    username:{

        type:String,

        required:true,

        unique:true,

        trim:true

    },

    password:{

        type:String,

        required:true

    }

});

Table

//skapar schemat för bord/table.

const tableSchema = new mongoose.Schema({

    tableID:{

        type: String,

        required: true,

        unique: true,

        trim: true

    },

    capacity:{

        type:Number,

        required:true,

        min:1

    },

    location:{

        type:String,

        required:true,

        enum:['inne','ute']

    }

});

Food

const mongoose = require("mongoose");

//skapar schemat för maträtter

const foodSchema = new mongoose.Schema({

    foodID:{

        type: Number,

        required: true,

        unique: true,

        trim: true

    },

    group:{

        type:String,

        required:true,

        enum:['förrätt','huvudrätt','efterrätt']

    },

    name:{

        type:String,

        required:true,

        unique: true

    },

    description:{

        type:String,

        required:true

    },

    contains:{

        type:[String],

        required:true

    },

    price:{

        type:Number,

        required:true

    }

});

Bokning, notera referens till andra delar av databas

//skapar schemat för bokning. med referat till TABLE ID samt USER ID

const bookingSchema = new mongoose.Schema({

        tableID:{

            type: String,

            ref: 'Table',

            required: true

        },

        username:{

            type:String,

            required:true,

            ref:'authprojekt'

        },

        bookingDate: {

            type: Date,

            required: true

        },

        numberGuests: {

            type: Number,

            required: true,

            min: 1

        }

    },

    {

        timestamps: true

    }

);

### Databas anslutning

Anslutning till databas som hostas av mongodb(Atlas) sker genom att ansluta enligt nedan.

const mongoose = require("mongoose");

mongoose

    .connect(uri)

    .then(() => {console.log("Ansluten till mongodb");})

    .catch((error)=> {console.error("nåt gick fel"+error);})

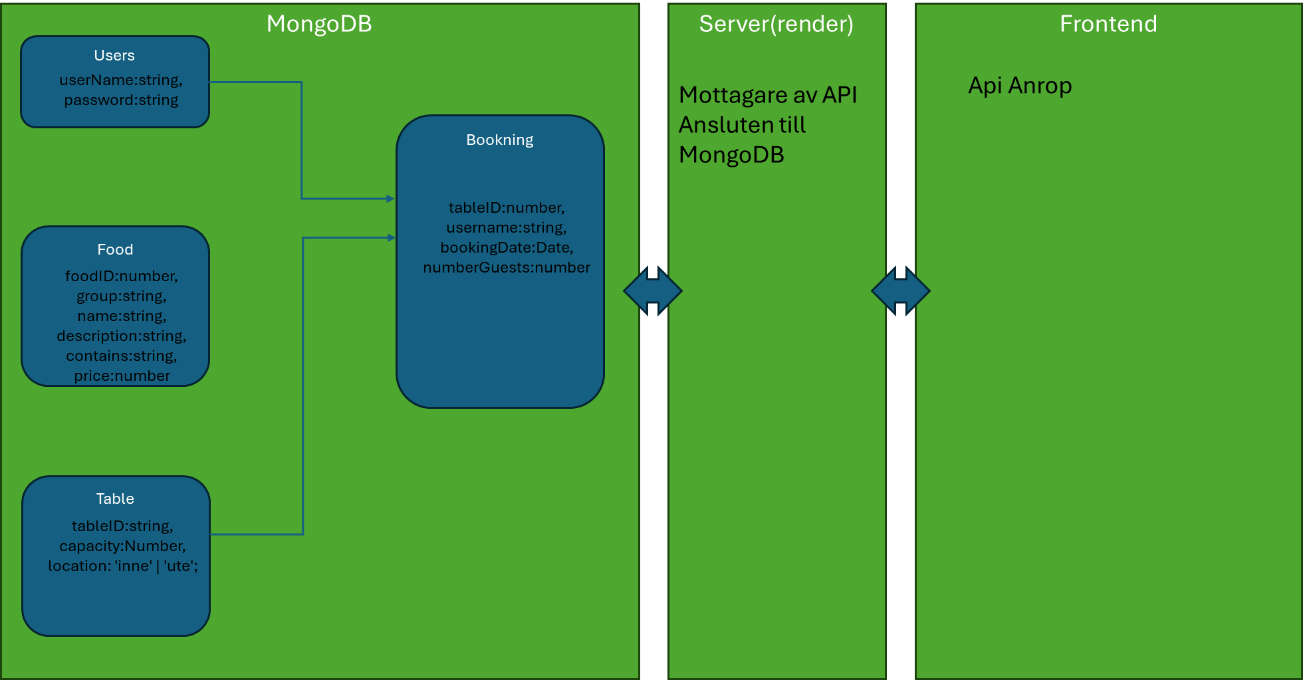
URL i detta fall är en länk till databas innehållande inloggning och referens till vilken .

## Backend server

Backend server är publiserad via Render. Render är en server som möjliggör för front-end delen att anropa funktioner(API). koden som finns på server har dessutom möjligheter att ansluta till tex databaser.

Denna har i sin tur kontakt med API anrop från Frontend.

samt kommunicerar med mongodb Databasen för att hämta och skriva information.



### Funktioner API

Varje API anrop är enligt CRUD när det kommer till anrop. Funktioner har blivit uppdelade i två delar, en ”models” och en ”routers”. Routers innehåller validering och anropar funktioner i models där datan hanteras.

En bild som visar text, skärmbild, Teckensnitt

Automatiskt genererad beskrivning

#### User

User anropet består av två delar. Det ena är till för at kunna lägga till nya användare, dvs i detta fall ”register”. Det andra anropet är för att användare ska kunna logga in och tex boka ett bord.

Anropet:login bygger på en hashning av lösenord och är krypterade i databasen genom bcrypt. Vid en inloggning jämförs det krypterade lösenordet med det medskickade lösenorder i https-anropet. Vid en lyckad inloggning returneras en JWT till användaren som sparas undan i local-storage.

#### Table

Denna funktion är till för att skapa resurser som används i tidsbokning/bordsbokning. Varje bord definieras med en placering inne/ute i detta fall samt ett ID och max antal gäster.

API anrop är: table

Detta anrop är skyddad och kräver inlogg som är admin.

#### Bookning

Bookning bygger på resureser som skapas i ”Table”. Detta för att garantera att ingen bokning sker som inte har ett bord tillförsett. Servern tar emot API anropet och kontrollerar innan bokning slutförs att ingen annan har utfört en bokning på samma tid vilket skulle resultera i en dubbelbokning.

#### Food

# Resultat

Resultatkapitlet ingår när du har genomfört en systematisk undersökning, t ex en utvärdering av ett datorprogram som du har utvecklat, vilket krävs inom examensarbeten på C- och D-nivå. I resultatkapitlet redovisas objektiva resultat av en empirisk undersökning., t ex en sådan utvärdering som nämns ovan. Tänk på att eventuella kommentarer i detta kapitel endast får vara av förtydligande art. Dina egna synpunkter och subjektiva (personliga) kommentarer hör hemma i kapitlet Slutsatser/Analys/Diskussion.

Sträva efter att redovisa resultaten, till exempel enkät-, test-, mät-, beräknings- och/eller simuleringsresultat, så överskådligt och lättbegripligt som möjligt. Resultaten presenteras med fördel i diagram- eller tabellform. Redovisning av intervjuer kan bestå av sammanfattningar, eventuellt kompletterade med några konkreta exempel.

Omfattande resultat, till exempel fullständiga sammanställningar av enkätresultat, stora tabeller och långa matematiska härledningar, placeras med fördel i bilagor.

# Slutsatser / Analys / Diskussion

Efter de objektiva resultaten följer kapitlet Slutsatser/Analys/Diskussion (välj en rubrik), där du presenter dina egna slutsatser, din subjektiva uppfattning, samt kritiskt analyserar resultatens tillförlitlighet och generaliserbarhet.

Om denna del är omfattande kan den indelas i flera kapitel eller under-kapitel, t ex ett analys- eller diskussionskapitel med förklaringar till och kritisk granskning av resultaten, ett slutsatskapitel där de viktigaste resultaten och slutsatserna presenteras, samt ett avsnitt med förslag på fortsatt arbete inom området.

Att återknyta till undersökningens syftes- och målformulering hör till det viktigaste i detta kapitel.

Ge gärna utrymme åt svaren på följande frågor: Vad är projektets nyhetsvärde och viktigaste bidrag till forskningen eller teknikutvecklingen? Har projektets mål uppnåtts? Har uppdraget utförts? Vad är svaret på den inledande problemformuleringen? Har resultatet blivit det väntade? Är slutsatserna generella, eller gäller de bara under vissa förutsättningar? Vilken betydelse har metod- och modellvalet för resultaten? Har nya frågor väckts på grund av resultatet?

Den sista frågan inbjuder till möjligheten att ge förslag till andra, anknytande undersökningar, d.v.s. förslag dels till åtgärder och rekommendationer, dels till fortsatt forskning eller utveckling för den som vill bygga vidare på ditt arbete.

I tekniska rapporter på uppdrag av företag presenterar du här den rekommenderade lösningen på ett problem. Du kan då göra en konsekvensanalys av lösningen ur tekniskt såväl som lekmannaperspektiv, till exempel i fråga om ekonomi, miljö och förändrade arbetsrutiner. Kapitlet innehåller då rekommenderade åtgärder samt förslag på vidare utveckling eller forskning, och utgör således beslutsunderlag för uppdragsgivaren.

Källförteckning

Här följer exempel på hur en källförteckning kan utformas enligt Vancouver-systemet. Den är automatiserad enligt metoden numrerad lista och korsreferenser, som beskrivs i kapitel 2.4.

1. Monash University, ”Citing, Vancouver Style” <http://www.lib.monash.edu.au/tutorials/citing/vancouver.html> Publicerad 2006-04-13. Hämtad 2012-01-25 (Exempel på referens till webbsida)
2. Svenska Datatermgruppen, ”Information om datatermer”,  
   http://www.nada.kth.se/dataterm/   
   Publicerad 1998-08-20. Hämtad 2005-04-11. (Exempel på referens till webbsida)
3. M. Eriksson, ”Dynamic Single Frequency Networks”, *IEEE Journal on Selected Areas in Communications (J-SAC)*, vol. 19, nr. 10, 2001, pp. 1905-1914. (Exempel på referens till artikel i vetenskaplig tidskrift)
4. S. Strömquist, *Skrivboken*. 4 uppl. Malmö: Gleerups., 2000 (Exempel på referens till bok)
5. L. Forsslund, ”Rapportering av forskningsresultat - ett rationaliserings-  
   objekt”. *Industriell teknik*, 22, 1969, s. 361-363. (Exempel på referens till artikel i tidskrift)
6. N. Bie, “Minspel på Internet”. *Dagens Nyheter* 7 juli, 1997, page 5. (Exempel på referens till artikel i dagstidning)
7. K. Berg, ”Inledning till kunskap”, Fiktiva högskolan, rapport nr XYZ-102, 2003, 120 sidor. (Exempel på referens till annan rapport)
8. AB Benzlers. *Växelboken. Kuggväxlar, kuggremmar, kilremmar etc*. Mecman. Hydraulik, SKF. Huvudkatalog. (Exempel på referens till katalog)
9. Microsoft Corporation, 1992: *Microsoft Word Användarhandbok. Ordbehandlingsprogram för Macintosh Version 5.0*. (Exempel på referens till manual)
10. Wikipedia, ”H.264/MPEG-4 AVC”, http://en.wikipedia.org/wiki/H.264/MPEG-4\_AVC   
    Hämtad 2010-04-21. (Exempel på referens till webbsida)

Bilaga A: Dokumentation av egenutvecklad programkod

Exempel på underrubrik