TELAVOX

Samtalsstyrning i molnet

Av

Peter Hesslow och Aegir Atlason

Department of Electrical and Information Technology
Faculty of Engineering, LTH, Lund University
SE-221 00 Lund, Sweden

Sammanfattning

Genom dagens molnplattformar kan man enkelt få tillgång till flertalet tjänster som bland annat underlättar samarbete och datalagring. Detta examensarbete fokuserar på att med hjälp av en molnplattform utveckla och produktionssätta en webbapplikation, vars huvudsakliga funktion är att kunna söka igenom data för att kunna styra ett telefonsamtal. Examensarbetet utförs i samarbete med Telavox, som vill utöka den produktportfölj som de idag erbjuder sina *kunder* via sin kommunikationsplattform *Flow*. Produkten möjliggör att en *kund* utan vidare tekniska kunskaper eller resurser kan med hjälp av en enkel molntjänst sätta upp en personlig hänvisning för deras inringande *slutkunder*.

Det första steget i denna process var att avgöra vilken molnplattform som skulle användas. Detta gjordes genom att undersöka de olika tjänster som tillhandahålls i tre av de vanligaste molnplattformarna (Microsoft Azure, Amazon Web Services och Google Cloud). Eftersom *kunderna* har varierande IT-kunskaper låg stor vikt vid användbarhet i valet av molnplattform. Valet föll på Google Cloud, vars tjänster *Sheets* och *Apps Script* erbjöd ett lätthanterligt och familjärt användargränssnitt vid datalagring samt möjlighet att driftsätta direkt via Googles servrar.

Via *Sheets* skapades möjlighet för *kunder* att själva lägga in den data som sökningen skulle göras på, och denna data blev i sin tur lätt att nå för applikationer utvecklade i *Apps Script*. I *Apps Script* utvecklades en webbapplikation som kunde söka igenom ett *sheet* utifrån de inparametrar som skickats med i anropet. Applikationen anropades från Telavox *talsvar* när en *kund* fick ett inkommande samtal och gav respons som Telavox *talsvar* använde för att koppla samtalet vidare.

För att förbereda applikationen för produktionssättning var nästa steg att skissa på ett användargränssnitt i *Flow* för att göra det möjligt för *kunder* att själva aktivera funktionaliteten. Tillsammans med UX-teamet på Telavox togs två olika alternativ på användargränssnitt fram, dessa kan användas vid framtida utveckling. Arbetet för produktionssättning innehöll även framställning av en API-specifikation, tester och utvärdering av en fallstudie.

Resultatet av examensarbetet är en produkt med den *grundläggande funktionalitet* som gör det möjligt att söka igenom data med syftet att kunna styra ett telefonsamtal. Produkten är testad och fungerar vid författandet av denna rapport. Till detta har en grund lagts både för ett användargränssnitt och en mellanliggande service vars syfte är att hantera responsen från applikationen på ett standardiserat sätt.

Nyckelord: Samtalsstyrning, Google Apps Script, Sheets, cloud

Abstract

Through today's Cloud platforms, one can easily gain access to several services that eases collaboration and data storage. This thesis focuses on developing and deploying a web application, with the help of a cloud platform, which is able to search through data with the purpose of redirecting a telephone call. The thesis is done in collaboration with Telavox, who wants to broaden the product portfolio that they offer their customers today through their communication platform *Flow*. The product enables that a customer, without extensive IT-knowledge or resources, is able to use personal redirecting for their incoming phone calls with the help of a simple cloud service.

The first step in the process was to decide which cloud platform to use by examining the services provided by three of the most common cloud platforms (Microsoft Azure, Amazon Web Services and Google Cloud). Since the telephone number data is provided by customers with varying IT-knowledge, usability was paramount in deciding which platform to use. Google Cloud was chosen, whose services *Sheets* and *Apps Script* offers a familiar user interface for data storage and the possibility to deploy directly through Googles servers.

Through *Sheets* the customers themselves are able to insert the data which is to be searched through, and this data is easily accessible for applications developed with *Apps Script.* A web application was developed with the use of *Apps Script* which searches through a sheet based on parameters provided in a call to the application. This application can be called upon from Telavox's routing system when a customer receives an incoming phone call and responds with a phone number which Telavox's routing system uses to redirect the phone call.

To prepare the application for deployment the next step in the process was to draft wireframes for a user interface in *Flow* to make it possible for customers to sign up for the functionality. With the help of the UX-team at Telavox two different wireframes were made, which could serve as a base for future development. The work towards deployment also consisted of testing, producing an API specification and evaluating a case study.

The result of this thesis is a product with the basic functionality which makes it possible to search through data in order to redirect a telephone call. The product is tested and functional at the time of writing this report. A foundation has been laid for a user interface and an intermediate service which serves to handle the response from the application in a standardized way.

Keywords: Routing, Google Apps Script, Sheets, cloud

Förord

Vi vill rikta ett stort tack till Telavox och vår handledare Myad Tahajody som gjorde detta examensarbete möjligt. Genom sitt stora engagemang och bra feedback har arbetet ständigt styrts i rätt riktning. Vi vill även tacka vår handledare Christin Lindholm på LTH som varit involverad i processen från början till slut och givit oss mycket användbar feedback.

Innehållsförteckning

1. Inledning

I detta kapitel beskrivs bakgrunden till detta examensarbete med syfte, målformulering samt problemformulering.

1.1. Bakgrund

Telavox har som vision att ta fram en smart och effektiv kommunikationslösning som riktar sig till allt ifrån entreprenörsdrivna startups till globala jättar. Telavox är ett globalt IT-bolag med idag runt 230 anställda varav 60 utvecklare. Företaget har över 250 000 användare och fler än 12 400 företag anslutna till kommunikationsplattformen *Flow*. Då företaget varit molnbaserat sedan 2003 behöver Telavox ej anpassa sig till den digitala förändring som pågår i branschen utan istället fokusera på att bredda sin produktportfölj[18].

Flow, som är den molnbaserade kommunikationsplattform som företaget tillhandahåller, innehåller lösningar för bland annat telefoni, växel och chatt. Telavox har bytt ut alla abonnemang till licenser som inte är bundna till en specifik person utan kan flyttas runt och anpassas till deras *kunders* egna önskemål. All administration av dessa licenser sköter deras *kunder* själva genom *Flow Admin* [18].

Varje *kund* har en egen advisor som är en kontaktperson som hjälper till med allt från teknisk driftsättning till frågor kring fakturor och övrig support [18]. När en *kund* kontaktar Telavox på dess huvudnummer kopplas *kunden* direkt till rätt advisor som står som ansvarig.

Det är just denna funktion som examensarbetet tar avstamp i. Det står alla *kunder* fritt att implementera en sådan funktion som kopplar samtal rätt utifrån vilken *kund* som ringer in. Denna funktionalitet är emellertid dyr att implementera och blir således reserverad för de *kunder* med möjlighet att lägga kostsamma resurser, i form av tid och pengar, på sin kommunikationslösning.

Telavox önskan till författarna av detta examensarbete var att ta fram en prototyp, gärna genom att använda en *molnplattform*, som är kostnadsfri men också har ett gränssnitt som främjar användbarhet, för deras *kunder*. Det skulle kunna hjälpa Telavox och bredda sin produktportfölj ytterligare med funktionalitet som står högt på deras *kunders* önskelista till en låg kostnad.

Prototypen ska kunna koppla ett inringande samtal till rätt *anknytning* hos en *kund* baserat på ett *identifikationsnummer,* exempelvis personnummer eller ett inringande telefonnummer. På så sätt skulle samtalsflödet för en inringare kunna effektiviseras och upplevelsen förbättras. Prototypens funktionalitet kan beskrivas enligt följande flöde:

1. *Slutkund* ringer in till företag A som är *kund* hos Telavox.

2. Något av följande två scenarier sker:

    a. *Slutkund* får välkomstmeddelande och uppmaning att knappa in sitt *identifikationsnummer*

    b. *Slutkund* får välkomstmeddelande och information om att de nu kopplas till rätt anknytning.

3. Förfrågan skickas till Telavox som sedan söker fram data om *identifikationsnumret* på en *molnplattform*.

4. Baserat på den data som finns lagrad på denna *molnplattform* matchas *slutkunden* med rätt anknytning hos företag A.

5. Samtalet skickas till den rätta anknytningen hos företag A.

1.2. Syfte

Syftet med examensarbetet är att utveckla funktionalitet som förenklar kommunikation mellan företag och deras kunder. Det ämnar att ta fram en lösning som kan erbjuda funktionalitet som hittills krävt en djupare teknisk kunskap.

1.3. Målformulering

Målformuleringen har förändrats under arbetets gång och är därför uppdelad i två; initial målbeskrivning och nuvarande målbeskrivning. Anledningen till förändringen var att förmedling av information inte var relevant i fallstudien.

1.3.1. Initial målbeskrivning

Examensarbetet hade följande målbeskrivning vid uppstart:

- Förenkla för en *slutkund* genom att minska antal knapptryck och på så vis minska tiden för navigering.
- Förmedla information till operatör om uppringaren baserat på *identifikationsnumret*.

1.3.2. Slutgiltig målbeskrivning

Examensarbetet hade denna slutgiltiga målbeskrivning:

- Förenkla för en *slutkund* genom att minska antal knapptryck och på så vis minska tiden för navigering.

1.4. Problemformulering

Problemformuleringen har förändras under arbetets gång och är därför uppdelad i två delar; initial problembeskrivning och slutgiltig problembeskrivning. Anledningen till förändringen var att förmedling av information inte var relevant i fallstudien.

1.4.1. Initial problemformulering

Examensarbetet skulle behandla följande problem:

1. Hur kan Telavox telefonisystem exekvera kod på en *molnplattform*?
2. Vilka personuppgifter får lagras på en *molnplattform*?
3. Hur ska ett samtal kopplas till rätt *anknytning* beroende på *identifikationsnummer*?
4. På vilket sätt presenteras informationen om *slutkund* till *operatören*?
5. Kan rösttranskribering användas som komplement till knappinmatning?*

*Utvecklas i mån av tid

1.4.2. Slutgiltig problemformulering

Examensarbetet skulle behandla följande problem:

1. Vilken molnplattform är mest lämplig för ändamålet?
2. Hur kan Telavox telefonisystem exekvera kod på en *molnplattform*?
3. Vilka personuppgifter får lagras på en *molnplattform*?
4. Hur ska ett samtal kopplas till rätt *anknytning* beroende på *identifikationsnummer*?

1.5. Motivering av examensarbetet

Examensarbetet skulle möjliggöra för Telavox att ta större marknadsandelar genom att erbjuda ytterligare funktionalitet till sina *kunder*. Produkten skulle göra att även mindre företag får tillgång till funktioner som endast varit tillgängliga för större företag med stora resurser. Produkten som sådan skulle kunna hjälpa små företag utan teknisk kunskap att effektivt hantera inkommande kommunikation.

Produkten som sådan skulle kunna förenkla kommunikation mellan *kund* och *slutkund* i form av kortare väntetid och att *slutkunder* slipper knappa sig fram genom en knappvalsmeny.

För vår egen del så får vi möjlighet att arbeta hos ett spännande företag som befinner sig i en tillväxtfas. Funktionaliteten som vi har för avsikt att implementera är något som vi själva hade uppskattat vid företagskontakt och är något som Telavox *kunder* önskat i samband med kundundersökningar som gjorts.

2. Teknisk bakgrund

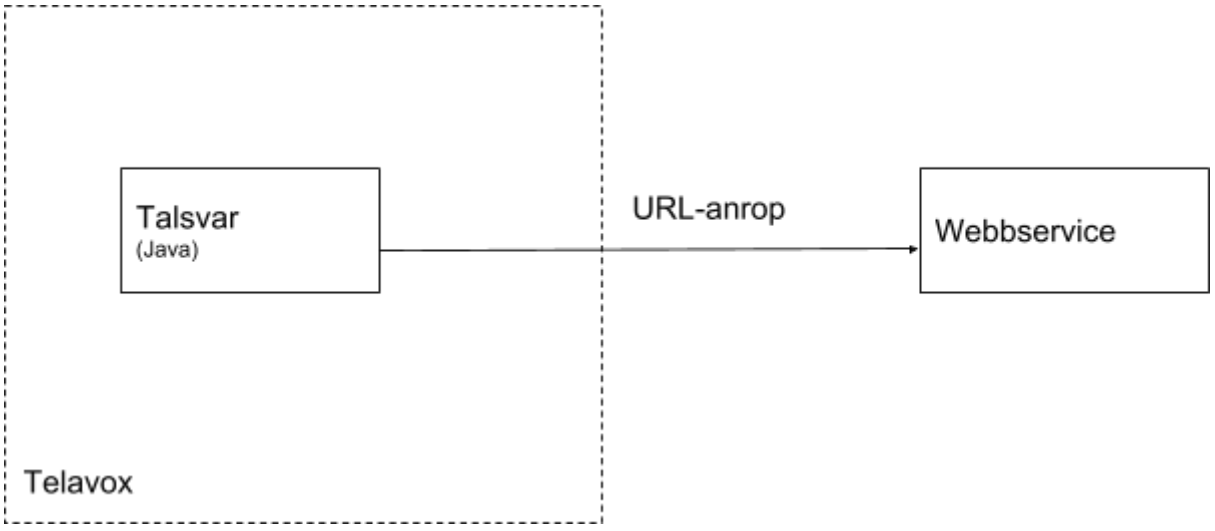
I detta kapitel beskrivs den tekniska bakgrund som fanns hos Telavox vid tidpunkten för detta examensarbete och funktionalitet som var relaterad till den produkt som utvecklats av författarna till denna rapport. Produkten kommer hädanefter i denna rapport att benämnas som webbtjänst. Kapitlet beskriver även de system som arbetet kom i kontakt med under arbetsprocessens gång. Därefter beskrivs den tekniska bakgrund och de tekniker som användes vid prototyp- och webbutveckling samt vid testning av webbtjänsten.

2.1. Teknisk bakgrund: Telavox

Detta delkapitel beskriver den tekniska bakgrund hos Telavox som var relevant till webbtjänsten.

2.1.1. Talsvaret

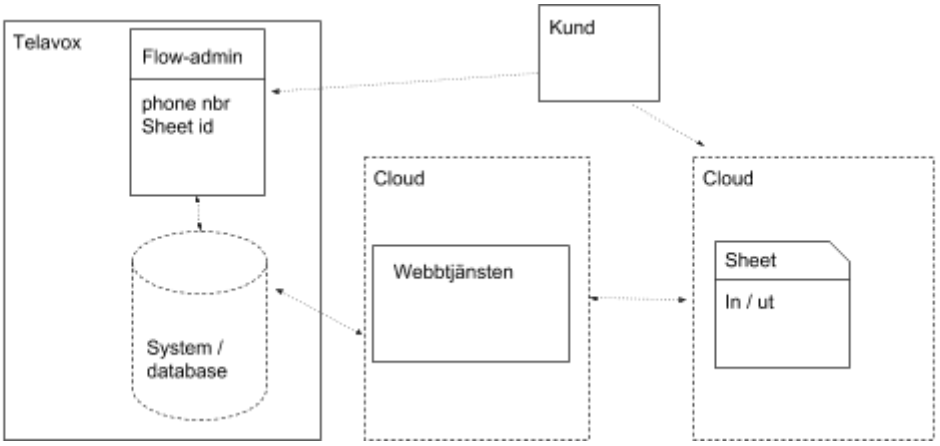
Vid författandet av denna rapport kan *talsvaret* beskrivas som en Java-applikation som har för uppgift att styra den funktionalitet som finns i Telavox växeltjänster. För ca tio år sedan togs en prototyp fram som gjorde det möjligt för *talsvaret* att anropa en URL. Syftet med denna var att kunna erbjuda *kunder* möjlighet att exempelvis föra statistik genom att räkna antalet anrop som gjordes vid någon händelse i *talsvaret* som genererade ett URL-anrop. Identifiering av vem som ringde baserades på att inringaren knappade in ett *identifikationsnummer* med hjälp av telefonens knappsats. Figur 1 nedan illustrerar hur java-applikationen anropar en webbservice via URL. Det förblev dock en prototyp och lanserades aldrig brett. Tanken var heller inte att *talsvaret* skulle ta emot respons från webbservicen utan var endast tänkt att användas åt ett håll som visas av den pil från *Talsvars*-boxen till webbservicen.



Figur 1: Talsvaret

2.1.2. Flow Admin

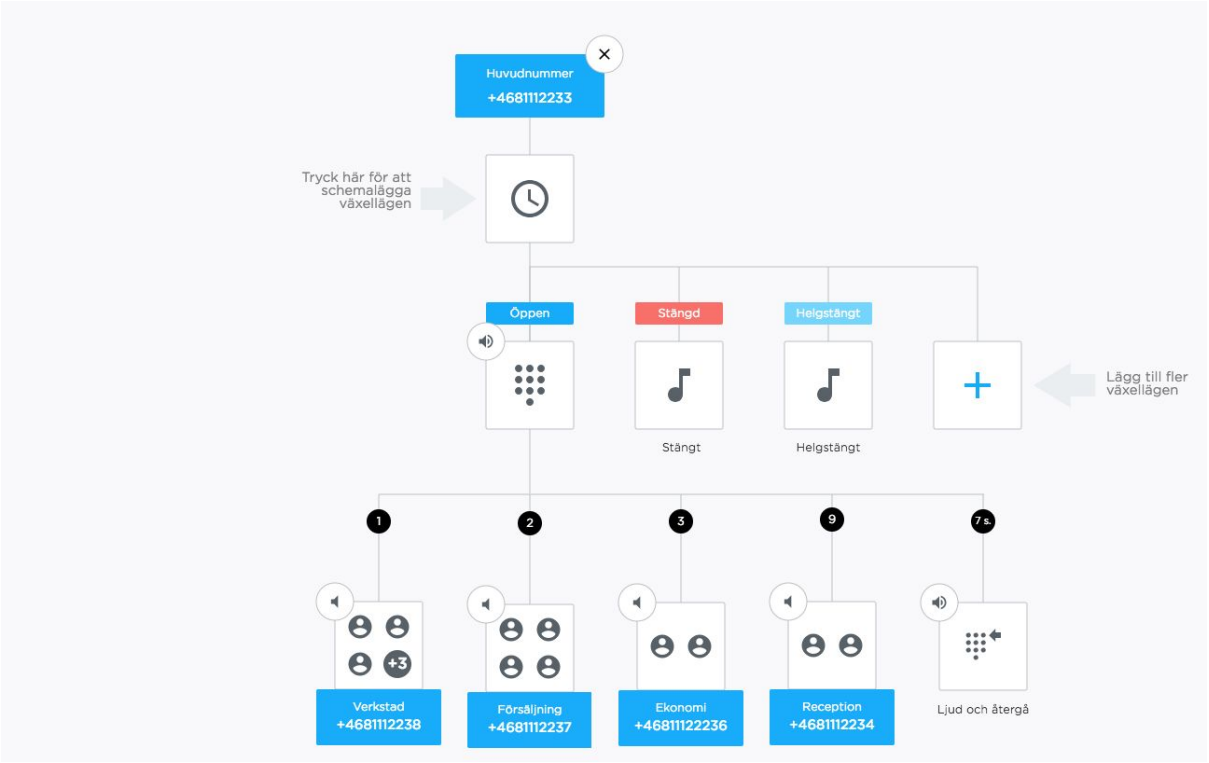
Flow Admin är det administrationssystem Telavox *kunder* vid skrivande av denna rapport använder för att konfigurera sina tjänster i *Flow*. I *Flow Admin* kan en *kund* bland annat konfigurera licenser, lägga till supportärenden och hantera telefonnummer. För att webbtjänsten ska kunna leta i rätt *sheet* behövs ytterligare funktionalitet tillföras i *Flow Admin* (se figur 2). Pilen mellan *Flow Admin* och system/database visar att den data en *kund* anger i *Flow Admin* lagras lokalt i Telavox databas. Den data som en *kund* ska kunna uppdatera är vilket telefonnummer webbtjänsten ska vara kopplat till och vilket *sheetId* som ska genomsökas för att kunna identifiera inkommande samtal. Pilen mellan *sheet* och webbtjänsten visar kommunikation mellan dessa två molnplattformar.



Figur 2: Beskrivning av flöde mellan Telavox och en *kund*

2.1.3. Växeltjänster

Växeltjänster hade vid författandet av denna rapport utseende enligt figur 3. Syftet med *växeltjänster* är att göra det möjligt för en *kund* att konfigurera sin telefoni. Längst upp i figuren visas *kundens* huvudnummer och under huvudnumret kan *kunden* lägga in sina öppettider. Beroende på klockslag dirigeras en *slutkund* till knappvalet eller meddelande, som i exemplet i figur 3, om stängt eller helgstängt. Om öppet hamnar *slutkunden* i knappvalet där ett välkomstmeddelande läses upp och därefter vilka knappval *slutkunden* har att välja mellan. I detta fall är det; 1 för verkstad, 2 för försäljning, 3 för ekonomi eller 9 för reception. Om *slutkunden* inte väljer något inom 7 sekunder spelas meddelandet upp igen. Dessa 7 sekunder syns i rutan längst ner till höger i figur 3. Detta är ett exempel på hur en *kund* kan lägga upp sin telefoni i *växeltjänster*. Det finns möjlighet att göra omfattande konfigureringar genom att koppla ett knappval till ett helt nytt träd och på så vis styra öppettider och nya knappval på nytt.



Figur 3: Växeltjänster

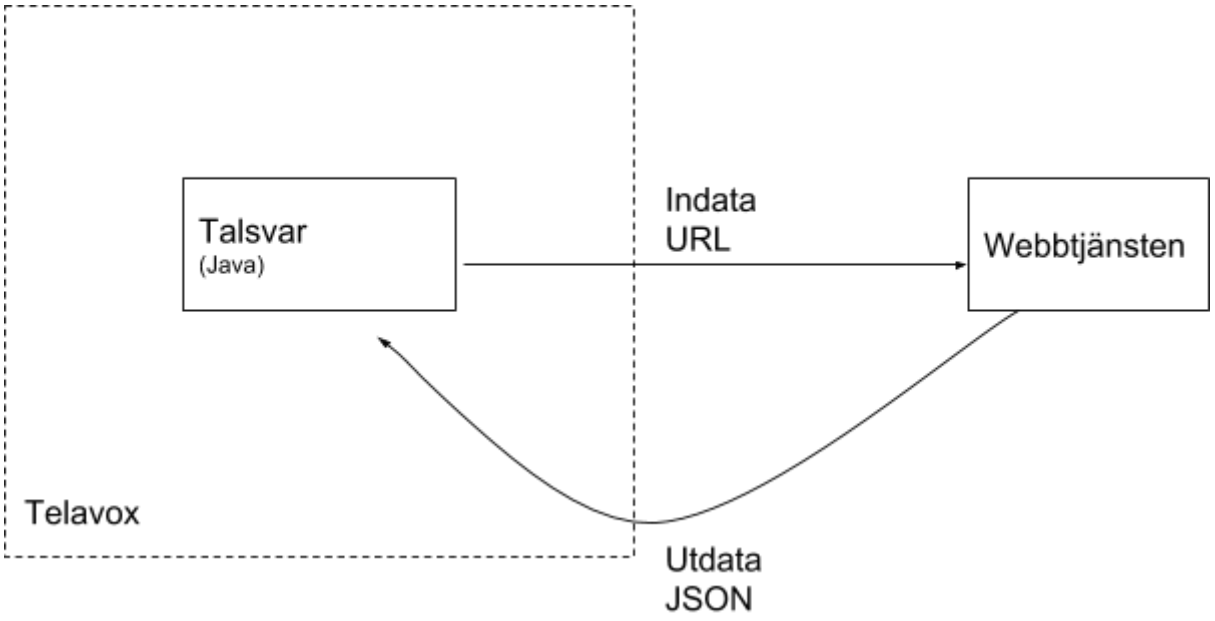
2.2. Teknisk bakgrund: Examensarbetet

Detta delkapitel beskriver den tekniska bakgrunden för webbtjänsten, hur webbtjänsten är tänkt att integreras med Telavox befintliga system, samt de tekniker som kommer att vara aktuella för framtagning av prototyp, utveckling och test.

2.2.1. Utveckling av befintlig funktionalitet

Då funktionalitet att anropa en URL redan existerade låg fokus istället på möjligheten att agera på den respons webbtjänsten skulle kunna returnera och vilken utökad produktportfölj Telavox skulle kunna erbjuda sina *kunder*.

Figur 4 visar hur den webbtjänst som skulle utvecklas anropar en URL som innehåller parametrar, exempelvis en sträng med ett telefonnummer. Därefter söker webbtjänsten igenom data och returnerar en sträng, exempelvis ett telefonnummer, som *talsvaret* sedan agerar på genom att koppla samtalet till en *anknytning* hos aktuell *kund*. Pilen mellan *Talsvars*-boxen och webbtjänsten visar *talsvarets* java-applikation anropar en URL varpå webbtjänsten sedan returnerar utdata i JSON-format.



Figur 4: Respons från webbtjänsten

2.2.2. Google Apps Script

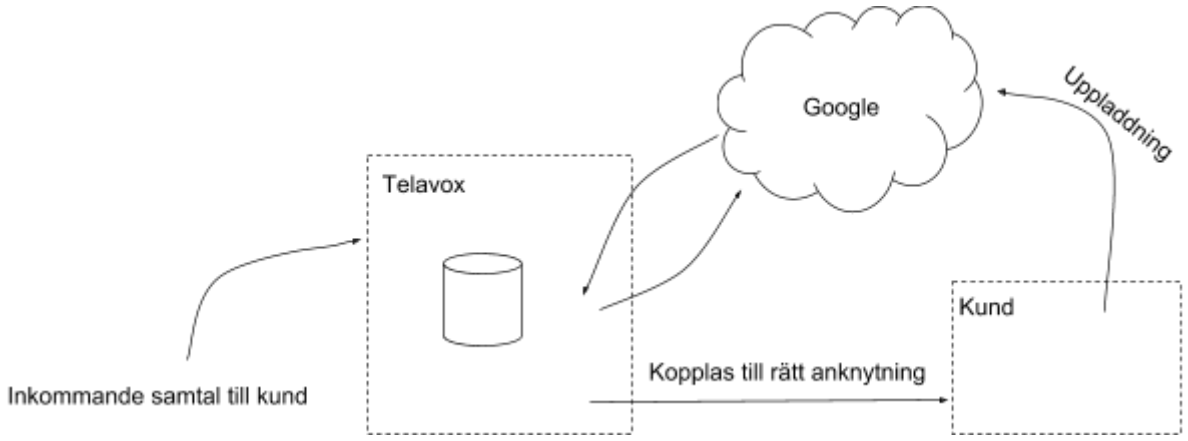
För att webbtjänsten skulle kunna förses med önskad funktionalitet valdes *Apps Script* från Google. *Apps Script* är en utvecklingsplattform för kod och applikationer som finns inbyggt i Googles olika applikationer och gör det möjligt att lägga till funktionalitet till bland annat *sheets* och *docs*. Det är exempelvis möjligt att utveckla och produktionssätta skript som skapar en ny meny till ett *sheet* via Google *Apps Script.* Dessa skript kan utvecklas fristående och lagras i Googles molnlagringstjänst *Drive* eller direkt kopplade till exempelvis ett *sheet*.

En av fördelarna med att använda Googles plattform är att autentisering redan finns inbyggt i form av Google-konton. Detta gör att man på ett enkelt sätt har möjlighet att dela dokument med andra Google-konton. På så sätt finns möjlighet att reglera behörigheter för Google-konton att kunna läsa och redigera dokument. Eftersom Google *Apps Script* är en del av Google erbjuds samma möjlighet att i realtid samarbeta i olika dokument. Det krävs inte någon flytt av filer mellan datorer samt att funktionalitet för backup, versionshantering och uppladdning finns inbyggt. Det är uppgifter som kan vara tidskrävande vid utveckling av applikationer[1].

2.2.3. Översiktlig beskrivning av webbtjänsten

Eftersom webbtjänsten är driftsatt som en *Google Web App* så är det möjligt att anropa den med en URL som innehåller ett nummer som identifierar ett inkommande samtal. webbtjänsten söker sedan efter en match på detta telefonnummer i ett *sheet* och returnerar den *anknytning* dit samtalet ska kopplas. Talsvaret kopplar sedan samtalet till denna *anknytning.* På så vis slipper den som ringer in till en *kund* att navigera själv i *talsvaret* genom olika menyval utan istället direkt bli kopplad till rätt *anknytning*.

Pilen mellan en *kund* och Google i figur 5 visar hur en *kund* till Telavox skulle själv kunna ladda upp data till sitt eget *sheet* på Googles *molntjänst.* Genom att Telavox får behörighet att söka i *kundens sheet* kan Telavox ta reda på vilken *anknytning* samtalet ska kopplas till. Pilen vid "inkommande samtal" är ett samtal till en *kund*. Pilen mellan Telavox och Google visar hur ett anrop till webbtjänsten görs varpå ett telefonnummer dit samtalet ska kopplas returneras. Pilen vid "kopplas till rätt anknytning" visar slutligen hur Telavox talsvar styr samtalet till det nummer som returneras från webbtjänsten från Google.

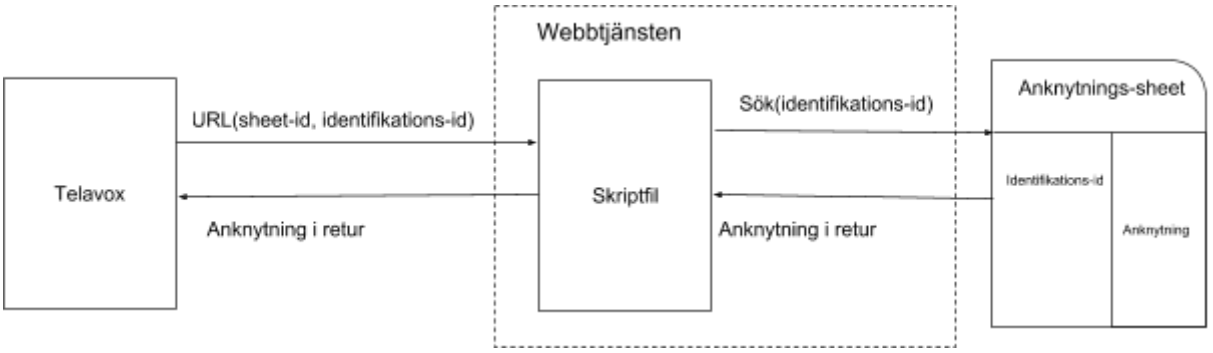


Figur 5: Översiktlig beskrivning av webbtjänsten

2.2.4. Teknisk beskrivning av webbtjänsten

Webbtjänsten består i grunden av en skriptfil som har för uppgift att söka igenom *sheet* som är publicerade via Google Cloud-plattformen.

I figur 6 visas de kopplingar till webbtjänsten i mer detalj. Pilen mellan Telavox och skriptfilen är när Telavox anropar webbtjänsten. Det sker med två parametrar; *sheetId* och *identifikationsnummer*. SheetId hämtas från Telavox databas medan identifikationsnummer erhålls genom det aktuella samtalet (telefonnummer eller nummer angivet med knappsats). Pilen mellan skriptfilen och ett *sheet* visar hur skriptfilen med hjälp av ett *sheetId* öppnar aktuellt *sheet* och söker därefter igenom den aktuella *kundens sheet* efter en cell där matchande anknytning till det *identifikationsnummer* som skickades med i anropet finns. Förutsatt att matchande *identifikationsnummer* finns i aktuellt *sheet*, returnerar webbtjänsten den *anknytning* dit samtalet skall kopplas. Detta illustreras av pilen mellan skriptfilen och Telavox. Därefter kan Telavox koppla samtalet till denna *anknytning*.



Figur 6: Teknisk beskrivning av webbtjänsten

2.2.5. Standalone Script

Webbtjänstens skriptfil är av typen *Standalone Scripts*. Det är *skriptfiler* som ej är kopplade till något bestämt Google-dokument som Sheets, Docs eller Slides. Dessa *skriptfiler* finns lagrade bland andra filer på den aktuella Google Drive-katalogen. *Skriptfilen* publiceras sedan som en Google *Web App* [2].

2.2.6. Google Web App

Eftersom ett *Standalone Script* kan publiceras som en Google *Web app* går det att anropa via en URL om *skriptfilen* uppfyller följande två krav:

1. *Skriptfilen* måste innehålla en `doGet(e)`- eller `doPost(e)`-funktion.
2. Funktionen måste returnera ett *HtmlService*-objekt eller ett *Contentservice*-objekt

När *skriptfilen* anropas med en HTTP GET, körs funktionen `doGet(e)` där e-argumentet kan innehålla information om någon parameter i anropet [3].

2.2.7. Postman

Innan webbtjänsten kopplades ihop med Telavox för att fungera i produktionsmiljö testades webbtjänsten med ett program som heter Postman. Postman är en API-klient som gör det möjligt för utvecklare att skapa, dela, testa och dokumentera API:er. Detta görs genom att möjliggöra för utvecklare att skapa och spara både enkla och komplexa HTTP-anrop samt tyda responsen[13]. Innan webbtjänsten kunde anropas från Telavox *talsvar* användes Postman primärt för att kontrollera att responsen är korrekt (se Appendix 9.2). Postmans funktion *Monitor* användes för att mäta svarstider vid anrop av webbtjänsten.

2.2.8. Sketch

För att en *kund* skulle kunna ges möjlighet att ange den data som är relevant för att kunna använda webbtjänsten togs en prototyp av ett användargränssnitt fram. Prototypen skulle

statuera ett exempel på hur en sida i *Flow Admin* skulle kunna se ut med den funktionalitet fanns inom ramen för detta examensarbete.

För att åstadkomma detta användes ett verktyg som heter Sketch, som är ett vektorbaserat designverktyg med möjlighet att göra en prototyp interaktiv[9]. Det går att göra statiska element klickbara och på så sätt visa det flöde en *kund* ska ta sig igenom för att komma igång med tjänsten.

3. Metod

Examensarbetet har till största del utförts i Telavox kontorslokaler i Malmö. Denna lokalitet har underlättat för kommunikation som skett löpande och med kortare varsel. Telavox har erbjudit arbetsstationer med den utrustning som krävs för att kunna utföra examensarbetet. Dessa arbetsstationer har stått i nära anslutning till den medarbetare på Telavox som utsetts till handledare för examensarbetet. Avstämningsmöten har hållits veckovis med handledaren och i vissa fall även andra medarbetare på Telavox. De veckovisa mötena har stått som grund i planeringen då utvärdering av veckan som gått gjorts och fokusområde för kommande vecka tagits fram.

Telavox är ett företag som arbetar efter agila processmodeller och uppmuntrar till att låta idéer snabbt ta form i det som kallas en *Minimum Viable Product (MVP)* för att visa på *Proof Of Concept (POC)*. Detta arbetssätt anammades i den arbetsgång som pågått under utformningen av detta examensarbete.

Utvecklingen av webbtjänsten som illustreras i figur 7 var uppdelad i fyra faser; analys, utveckling, test och produktionssättning, där produktionssättning låg utanför detta examensarbete. Iterationer uppstod även om utvecklingen kan ses som linjär. När den första prototypen togs fram i form av en *MVP*, togs arbetet vidare i nästa fas som bestod av tester och sedan tillbaka till utvecklingsfasen för att optimera webbtjänsten.

Figur 7: Faser

Hela examensarbetets arbetsprocess hade som syfte att göra den förberedande analys och utveckling som kunde stå till grund inför en produktionssättning av webbtjänsten där denna kunde testas mot en *kund*.

3.1. Analys

Första fasen av arbetet bestod till stor del av att utvärdera vilken *molnplattform* som skulle kunna användas för att uppnå syftet med examensarbetet. Genom efterforskningar i form av litteraturstudier och diskussion med Telavox beslutades att tre olika *molnplattformar* skulle kunna vara aktuella för en prototyp; Google Cloud, Microsoft Azure och Amazon Web Services. Utifrån varje enskild *molnplattform* undersöktes möjligheten att ta fram en *MVP* som skulle kunna uppnå *grundläggande funktionalitet*. Utöver valet av *molnplattform* utforskades även möjligheten att istället för en Google *Web App* att exekvera skriptet genom ett Google "körbart API". För att på ett effektivt sätt lyckas avgöra vilken *molnplattform* som skulle bli aktuell för *MVP* gjordes försök i att ta fram *MVP* för respektive *molnplattform.*

Eftersom det var möjligt att utveckla en *MVP* med hjälp av Google var en *MVP* framtagen redan i första fasen.

3.2. Utveckling

Utvecklingsfasen bestod av att förbättra den *MVP* som togs fram under analysfasen efter att tester gjorts. Fasen bestod även i att ta fram en prototyp-sida till *Flow Admin* för webbtjänsten och skriva en specifikation för webbtjänsten..

3.2.1. Webbtjänsten

Den *MVP* som togs fram i samband med analysen lyckades endast söka igenom ett *sheet* och returnera det värde som stod i cellen brevid det värde som skickades med som inparameter. På så sätt kunde *POC* bevisas. Efter möte med handledare och produkt- och utvecklingschef ställdes webbtjänsten inför frågor om hur svarstiden påverkades beroende på hur genomsökning av ett sheet gjordes vid uppskalning. Frågeställningar kring om det fanns någon cache-funktion inbyggd som skulle minska svarstiderna dök också upp. I samband med detta testades anrop av webbtjänsten med olika tidsintervaller och sökmetoder.

Under dessa möten diskuterades även användbarheten av webbtjänsten fram. Önskemål om ett *sheet* som har möjlighet att validera den data som en *kund* skriver in framkom. På så sätt ska varningar dyka upp och celler färgläggas i det *sheet* som en *kund* arbetar i. Detta skulle underlätta för en kund att hålla formatet korrekt, vilket var en förutsättning för att webbtjänsten skulle fungera i produktion.

Ytterligare krav för produktionssättning var att det skall finnas dokumentation för webbtjänsten. Det i form av en API-specifikation som specificerar de parametrar som skickas med i URL-anropet samt hur responsen ser ut.

I slutet av utvecklingsfasen lade produkt- och utvecklingschefen fram ett kundcase som skulle analyseras. Detta kundcase innefattade en *kund* till Telavox som uttryckt sitt intresse för den typ av webbtjänst som var under utveckling. Analyser gjordes i form av case-studies för att undersöka om webbtjänsten skulle leva upp till de önskemål som fanns hos denna kund.

3.2.2. Prototyp i Flow Admin

För att göra det möjligt för *kunder* att använda sig av webbtjänsten behövdes det upprättas en sida i *Flow Admin* där Telavox *kunder* kan ange det *sheetId* som är kopplat till deras *sheet* samt vilket telefonnummer webbtjänsten ska vara aktiverat på. När denna sida placerats i *Flow Admin* skulle det bli tillgängligt tillsammans med den övriga administration som en *kund* gör i *Flow Admin*.

Det första steget i denna process var att ta fram en klickbar wireframe för hur sidan skulle se ut. Detta för att komma fram till en layout som var i enlighet med resterande delar av *Flow Admin.* Prototypen togs fram med hjälp av *Sketch* och för att ha ett produktnamn för webbtjänsten valdes namnet "Routing". Framtagningen av prototypen var en iterativ process som bestod av samtal med UX-teamet, som arbetar med interaktionsdesignen av Telavox produkter.

3.2.3. Manual

En manual [appendix 9.4] togs fram i samband med att sidan till *Flow Admin* togs fram. Syftet med manualen var att kunna skicka ut till en *kund* som vill testa produkten i ett tidigt stadie. Den skulle även finnas länkad i från Routing-sidan i Flow Admin.

3.3 Test

Testfasen bestod i att iterativt testa den funktionalitet som byggdes på den *MVP* som tagits fram. Från början bestod testerna av enkla tester som att endast testa att det var rätt respons från webbtjänsten. Allt eftersom *MVP:n* utvecklades testades responstider vid uppskalning och tider för olika genomsökningsmetoder.

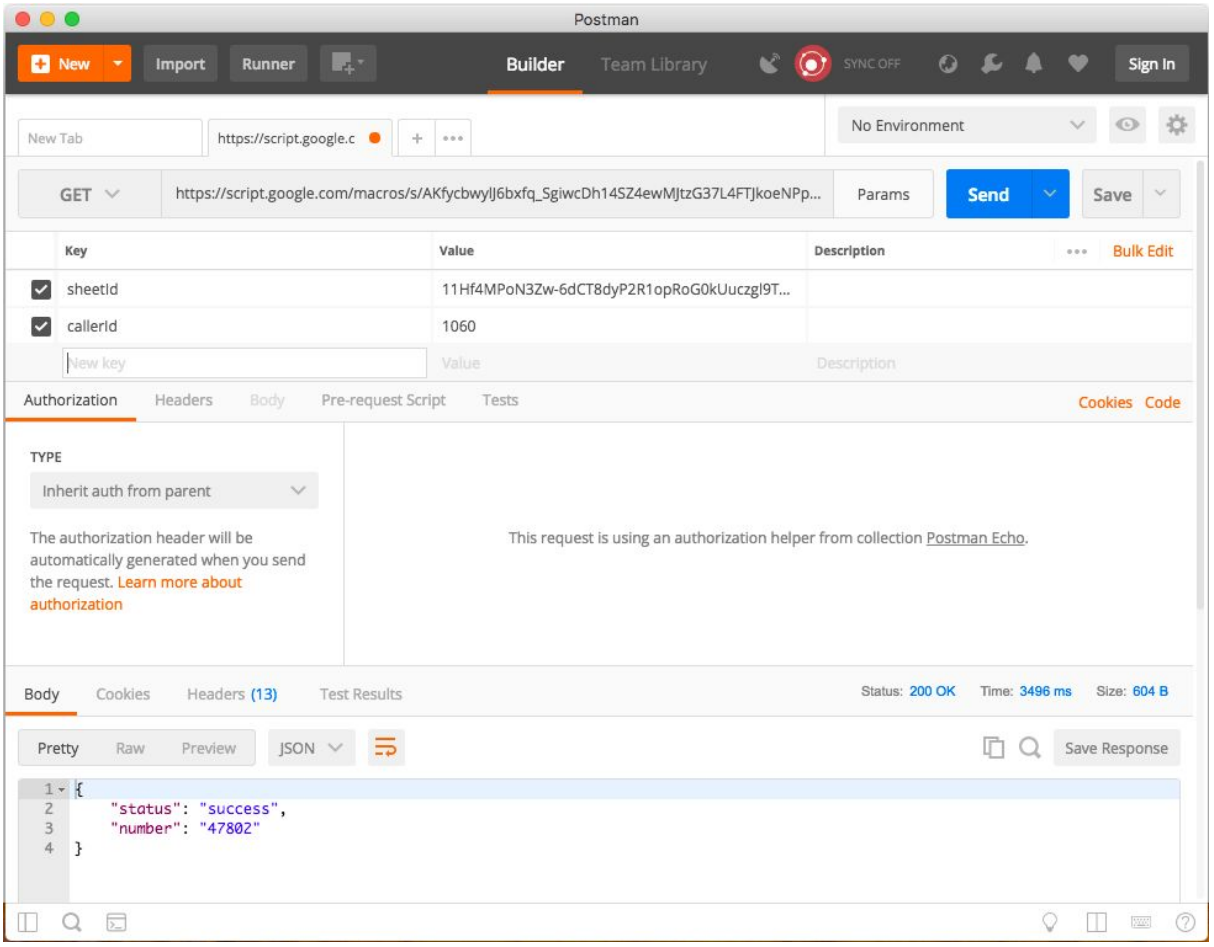
3.3.1. Funktionstest webbtjänst

A. Anrop med ogiltigt sheetId
B. Anrop med sheetId som ej är delat med webbtjänsten
C. Anrop med *identifikationsnummer* som ej finns i aktuellt *sheet*
D. Anrop med *identifikationsnummer* som finns i aktuellt *sheet*
E. Anrop med *identifikationsnummer* som finns duplicerat i aktuellt *sheet*

Resultat återfinns i appendix 9.5

3.3.2. Test av format och responstid

Med hjälp av Postman kunde URL-anrop simuleras för att kunna mäta responstid och kontrollera att korrekt respons ges när applikationen anropas. Ett anrop behöver innehålla en URL och om länken som anropas kräver parametrar måste även dessa vara specificerade [14]. I figur 8 visas en skärmdump över hur dessa parametrar anges i Postman.



Figur 8: Skärmdump från Postman

3.3.3. Test av ett sheet med självvalidering

Det *sheet* en *kund* ska arbeta i skulle kunna kunna validera den data som skrivs in i cellerna. På så vis gav det feedback om något format inte stämmer i form av längd eller tecken. Testerna gick ut på att testa de olika format som ska stödjas och gav röd färg i de celler som avvek från det rätta formatet.

3.4. Produktionssättning

Även om produktionssättning mot *kunder* låg utanför ramen för detta examensarbete involverades författarna i den process Telavox befann sig i för att få talsvaret att agera på den respons webbtjänsten svarade med. För författarna innebar detta arbete att förse Telavox med den data som behövdes för att kunna anropa webbtjänsten. Datan bestod av en URL till webbtjänsten samt hur de parametrar skulle specificeras vid anrop.

För att förbereda inför en produktionssättning utfördes även en fallstudie. Denna genomfördes då en av Telavox *kunder* uttryckt intresse för den funktionalitet som

examensarbetet arbetade med. Information om *kundens* förutsättningar erhölls genom kontakt med *kundens* kontaktperson på Telavox.

3.5. Källkritik

De källor som användes under detta examensarbete var främst i syfte att användas som bakgrundsinformation eller manual till de verktyg som användes. Motivering av de val författarna stått inför har ej påverkats av dessa källor utan har gjorts baserat på de erfarenheter som uppkommit i samband med analys, utveckling eller i samråd med Telavox då de besitter ett befintligt system som ska kunna interagera med webbtjänsten.

De källor som användes som manual eller instruktioner kom direkt i från verktygens utvecklare. I detta examensarbete handlade det om Google, Sketch, Microsoft, Postman. Författarna använde dessa källor för inhämtning av instruktioner för att undersöka om viss funktionalitet fanns vid försök att ta fram prototyper. Om dessa källor istället används för att motivera varför vi använt just dessa verktyg vid framtagning av prototyperna skulle trovärdigheten hos dem kunnats ifrågasatts. Detta eftersom de inte skulle ge en saklig bild av sina verktyg utan istället försöka sälja in dem. Trovärdighet hos dessa källor har därför ej ifrågasatts då de istället bekräftat vad författarna redan försökt åstadkomma vid användning av dessa verktyg.

3.5.1. Ferreira, James. Google Apps Script [1]

För att beskriva Google Apps Script från grunden valdes denna bok utgiven av O'Reilly Media. Genom att söka i diverse forum där flertalet rekommenderade boken föll valet på denna. Författaren är VD och grundare av Southwest Software Designs, LLC. Hans syfte är att göra Google Apps mer anpassningsbara. Han är aktiv inom Googles community och har därigenom hjälpt många genom att utveckla tillägg till Google för att få ut mer av produkterna. Boken är utgiven av O'Reilly Media som givit ut böcker relaterade till revolutionen kring internet sedan 1984. Många av deras böcker syns på hyllorna hos bland annat Telavox och Lunds Universitets bibliotek vilket bekräftade trovärdigheten hos boken.

3.5.2. Google Apps Script [2] [3] [4] [5] [6] [7] [8] [11] [12]

Googles produkter är ständigt under utveckling och det som är tryckt riskerar att bli för gammalt. Detta upptäcktes redan på många forum som studerades för att hitta information kring Googles produkter. Det som blev det mest naturliga för detta examensarbete var att använda Googles guider för att hitta den information som behövdes.

3.5.3. Sketch [9]

Sketch är det program som används av UX-designers på Telavox och är ett program som Telavox valt att köpa in till sina sina anställda. För att följa Telavox standard vad gäller utseende och kunna använda Telavox mallar valdes detta program även för examensarbetet.

Informationen som eftersöktes var ej inför vilket program som skulle användas utan i syfte att ta reda hur programmet används.

3.5.4. Microsoft Office [10]

När problem uppstod i att få till *grundläggande funktionalitet* i Excel online söktes information på Microsofts egna sida om hur detta kunde lösas. Eftersom Microsoft själva anger vilka begränsningar deras produkter har ansågs det inte nödvändigt att tvivla trovärdigheten hos källan.

3.5.5. BlazeMaster [13]

BlazeMaster är en hemsida som erbjuder lösningar för att testa bland annat API-anrop. De har kunder som Adobe, Atlassian och Walmart samt ett community med över 300 000 medlemmar. Författarna hittade sökte efter instruktioner i Postmans dokumentation utan framgång. Detta ledde till att BlazeMaster användes som en guide eller instruktion i hur användning av Postman för det specifikat fallet. Då funktionaliteten som efterfrågades gick att åstadkomma med hjälp av denna guide ansågs trovärdigheten vara tillräcklig för denna källa.

3.5.6. Postman [14] [15]

Postman är ett program som Telavox själva använder för att testa sina API-anrop. På så vis blev författarna till denna rapport introducerade till verktyget och använde Postmans hemsida som manual.

3.5.7. Techopedia Dictionary [16] [17]

För att hitta definitioner av *Minimum Viable Product* och *Proof of Concept* valdes Techopedia. Det var tydligare på denna sida jämfört med många andra. Egentligen hade författarna redan fått förklarat för sig av Telavox vad de betydde vilket gjorde att denna källa bekräftade detta vilket också gjorde trovärdigheten tillräcklig.

3.5.8. Telavox [18]

För inhämtning av bakgrundsinformation till Telavox användes deras egna hemsida.

4. Analys

Examensarbetet har ställts inför val, begränsningar och risker baserat på de grundkrav satta av Telavox:

- Att förse Telavox med en URL innehållande parametrar för att identifiera ett samtal och baserat på dessa parametrar returnera ett värde dit samtalet ska styras.
- URL-anropen ska göras till en *molnplattform* som ska vara enkel att använda för Telavox *kunder*.

De val som författarna av detta examensarbete stått inför var främst hur datan skulle lagras och vilken eller vilka *molnplattformar* som skulle användas. När prototypen väl tagit form stod författarna inför val gällande vilken riktning arbetet skulle ta, om fokus skulle ligga på utökad funktionalitet, UX eller produktionssättning. Författarna fick även möjlighet att göra en fallstudie på en av Telavox *kunder* som uttryckt intresse för denna typ av webbtjänst.

4.1. Val av molnplattform

För att hitta den eller de *molnplattformar* som skulle kunna vara aktuella för webbtjänsten gjordes jämförelser mellan Amazon AWS, Google Cloud samt Microsofts OneDrive. Andra val relaterade till *molnplattform* var sätt att publicera webbtjänsten.

4.1.1. Val av plattform: AWS, Cloud och OneDrive

I inledningsfasen skulle beslut tas om vilken molnbaserad lösning som skulle användas. De lösningar som diskuterades var Google Cloud, Microsoft Azure (OneDrive) och Amazon S3 (AWS).

Både Microsofts och Googles lösningar låg vid skrivandet av denna rapport nära varandra vad gäller funktionalitet att kunna ersätta en konventionell databas med ett kalkylblad som exempelvis Excel eller Sheets med körbara skript. I Amazons fall fanns inte någon motsvarande funktionalitet likt Google *Sheets* eller Microsofts Excel för hantering av data med ett användargränssnitt likt *Sheets* eller Excel. Både *Sheets* och Excel har ett användargränssnitt som en *kund* skulle känna igen sig i samt flexibla möjligheter för import och validering av data. Examensarbetets fokus hade i stor utsträckning legat på användbarhet för att uppnå ett användargränssnitt med funktionalitet som i *Sheets* eller Excel vilket i sin tur ändrat inriktning för detta examensarbete.

Med hjälp av Microsoft Excel finns funktionalitet likt Google Sheets att prenumerera på andra Exceldokument och köra skript för att söka igen och returnera med hjälp av makron. Det visade sig dock att Microsoft Excel Online inte har stöd för att köra makron vilket leder till att den *grundläggande funktionaliteten* inte gick att uppnå med Excel Online.[10].

På grund av att det saknades stöd för att köra makron online med Excel Online valdes Microsoft Azure bort. På grund av att ett användargränssnitt likt Sheets/Excel saknades i Amazon S3 och att det hade varit för tidskrävande att utveckla motsvarande funktionalitet, valdes Amazon S3 bort. Då *grundläggande funktionalitet* kunde uppnås med Google Cloud föll valet på denna molnplattform.

4.1.2. Val mellan Google API och Google Web App

När valet fallit på Googles *molnplattform* stod valet mellan att köra skriptet som en Google *Web App* eller som ett körbart API. *Google Web App* krävde vid skrivandet av denna rapport att rättigheter finns mellan de parter som ska ha tillgång till respektive *sheet*. Det hade i praktiken inneburit att det behövts en förbindelse mellan Telavox och en *kunds* Cloud-konto. På så vis skulle Telavox komma åt den data som lagras hos en kund.

För att undvika att Telavox får direkt tillgång till den data som lagras i *kunders sheet* undersöktes möjlighet att publicera skriptet som ett Google körbart API. Tanken var att man skulle kunna anropa ett API via URL och då få respons som sedan behandlades utan att Telavox fick insyn i *kundernas sheet* och datans integritet kunde behållas. Det visade sig dock att den anropande applikationen måste dela samma Cloud-konto om integriteten ska hållas intakt. Det går att skapa ett publikt API men då måste det *sheet* där datan lagras vara helt publikt vilket skulle innebära att integritetsskyddet inte existerar ändå.

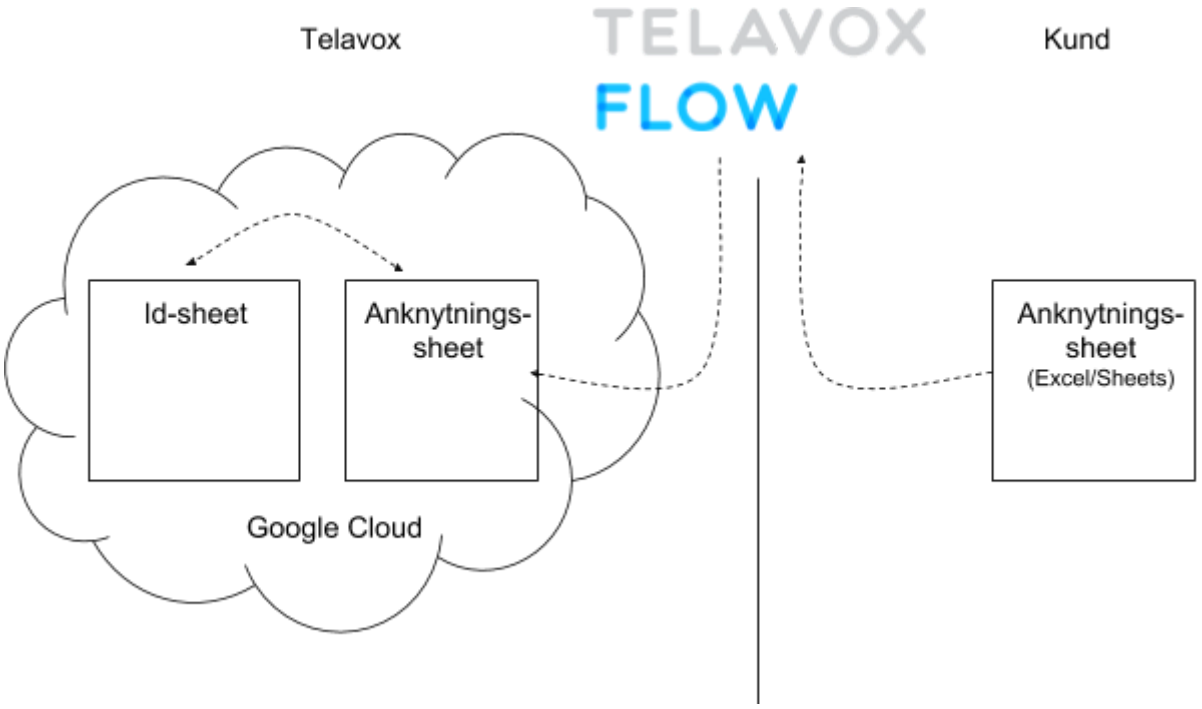
För att upprätthålla möjligheten för *kunder* att kunna hantera datan i *Sheets* föll valet på att publicera webbtjänsten som en Google *Web App*[12].

4.2. Val mellan sätt att hantera anrop och datalagring

Tre modeller togs fram och analyserades kring hur datan skulle delas och lagras mellan Telavox och dess *kunder*

4.2.1. *Id*- och *anknytnings-sheet* hos Telavox

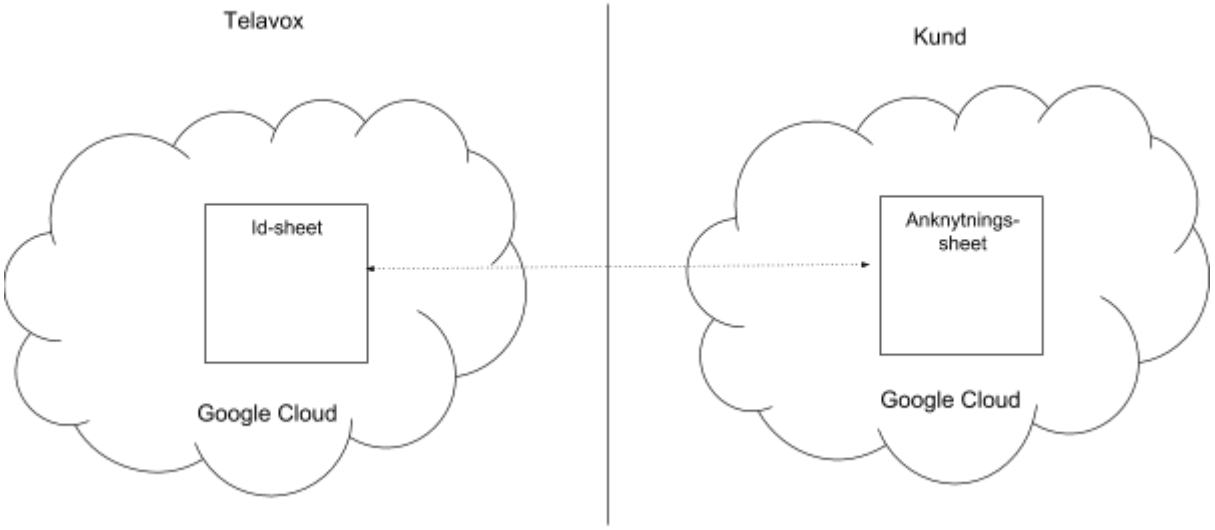
Figur 9 visar en första modell som bestod av att lagra både ett *id-sheet* och ett *anknytnings-sheet* på Telavox eget Google Cloud-konto. Pilarna i Google Could visar att ett *id*-sheet*,* som innehåller alla *kunder*, söker efter rätt *identifikationsnummer* och *anknytning* i *anknytnings-sheet*. På så vis skulle en *kund* inte behöva registrera sig för ett Google Cloud-konto utan med hjälp av funktionalitet i Flow-appen kunna ladda upp data i exempelvis CSV-format som illustreras av pilen mellan *anknytnings-sheet* och Telavox. Efter möte med Telavox beslöts det att inte gå vidare med denna idé då det skulle innebära att Telavox inte utnyttjar de fördelar som finns med Google Cloud, som funktionalitet för import och delning av dokument inte nyttjades.



Figur 9: *Id*- och *anknytnings*-sheet hos Telavox

4.2.2. *Id*- och *anknytnings-sheet* på olika Cloud-konton

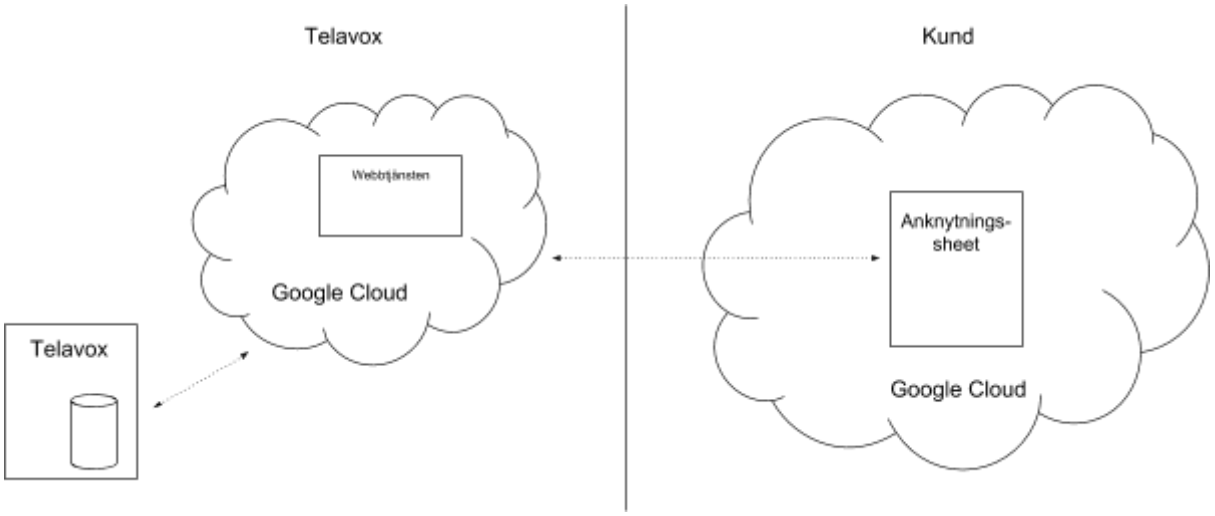
I figur 9 visas en en andra modell som bestod i att lagra alla *kunders sheetId* i ett separat *sheet* på Telavox Google Cloud-konto. Pilen mellan *id-sheet* och *anknytnings-sheet* visar att det först sker en sökning i ett *id-sheet* efter *sheetId* till rätt *kund*, därefter söks *kundens anknytnings-sheet* igenom efter *identifikationsnummer* och *anknytning*. Detta innebar emellertid dubbla slagningar i olika *sheet* vilket potentiellt gjorde det onödigt tidskrävande. Fördelen var att *kunder* själva kunde utnyttja Googles alla möjligheter för import och autosynkronisering gentemot *kunders* egna system. (se figur 10).



Figur 10: *Id*- och *anknytnings-sheet* på olika Cloud-konton

4.2.3. *Id*-sheet lokalt hos Telavox och *anknytnings-sheet* på *kunds* Cloud-konto

Figur 11 som var den tredje och slutliga modellen visar hur *id-sheet* ersatts med att lagra information om vilket sheet som skulle sökas igenom. Denna information skulle vara lagrat i en lokal databas hos Telavox som pilarna mellan Telavox och webbtjänsten i Google Cloud visar. Detta skulle innebära att *sheetId* skulle finnas med som en url-parameter i det anrop från Telavox *talsvar*-system till webbtjänsten. Webbtjänsten skulle sedan söka igenom *anknytnings-sheet* efter *identifikationsnummer* och *anknytning* som pilen mellan Telavox Cloud-konto och en *kunds* Cloud-konto visar. På så vis skulle Googles fördelar vad gäller tillgänglighet, användbarhet och mångsidighet utnyttjas utan att tumma på allt för onödig tidsåtgång. *Kunder* hade behövt ge Telavox tillgång till sitt *sheet* för att tillåta webbtjänsten att söka igenom det.



Figur 11: *Id-sheet* lokalt hos Telavox och *anknytnings*-sheet på *kunds* Cloud-konto

4.3. Begränsningar i Sheets

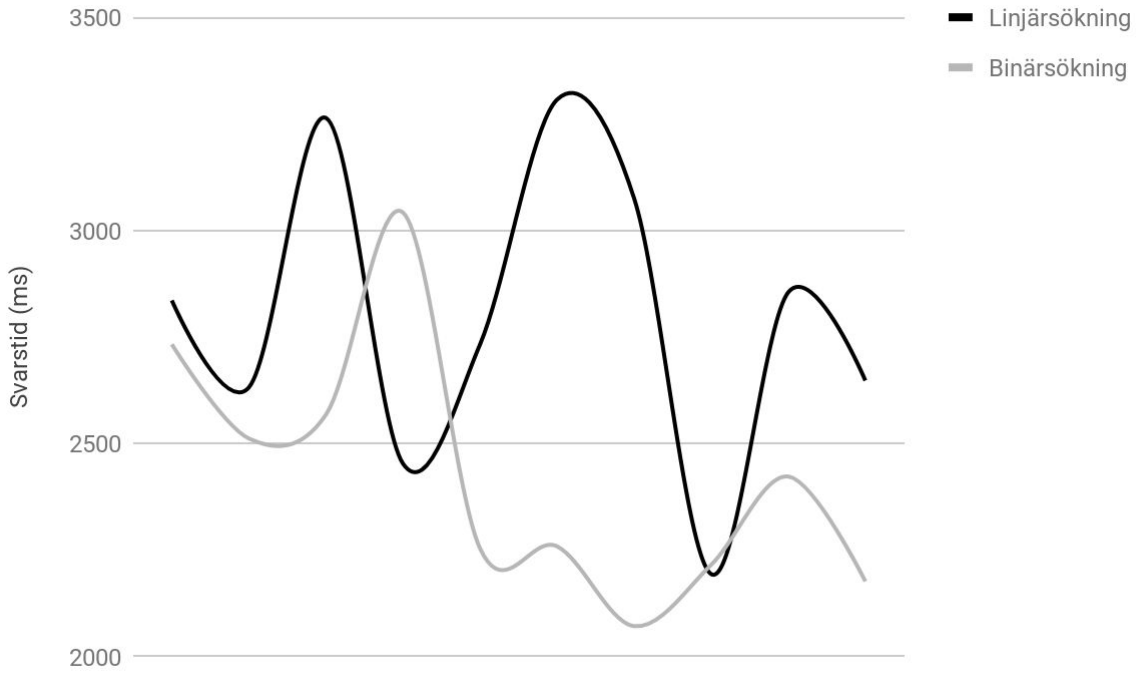
Vid skrivandet av denna rapport är begränsningen i Google Sheets är 2 000 000 celler. Genom att endast behålla två kolumner kan 1 000 000 *identifikationsnummer* få plats och lika många *anknytningar* hos Telavox *kunder* [11].

4.4. Tid för genomsökning av ett sheet

Även om webbtjänsten främst skulle rikta sig till mindre företag gjordes mätningar av genomsökningar i ett *sheet* med 100 000 rader med unik data. Detta för att simulera ett *anknytnings-sheet* som har 100 000 *slutkunder* registrerade i ett *sheet*. Syftet med mätningarna var att jämföra binär- och linjärsökning för att uppskatta eventuella skillnader vid uppskalning.

4.4.1. Sökning efter slumpvärden

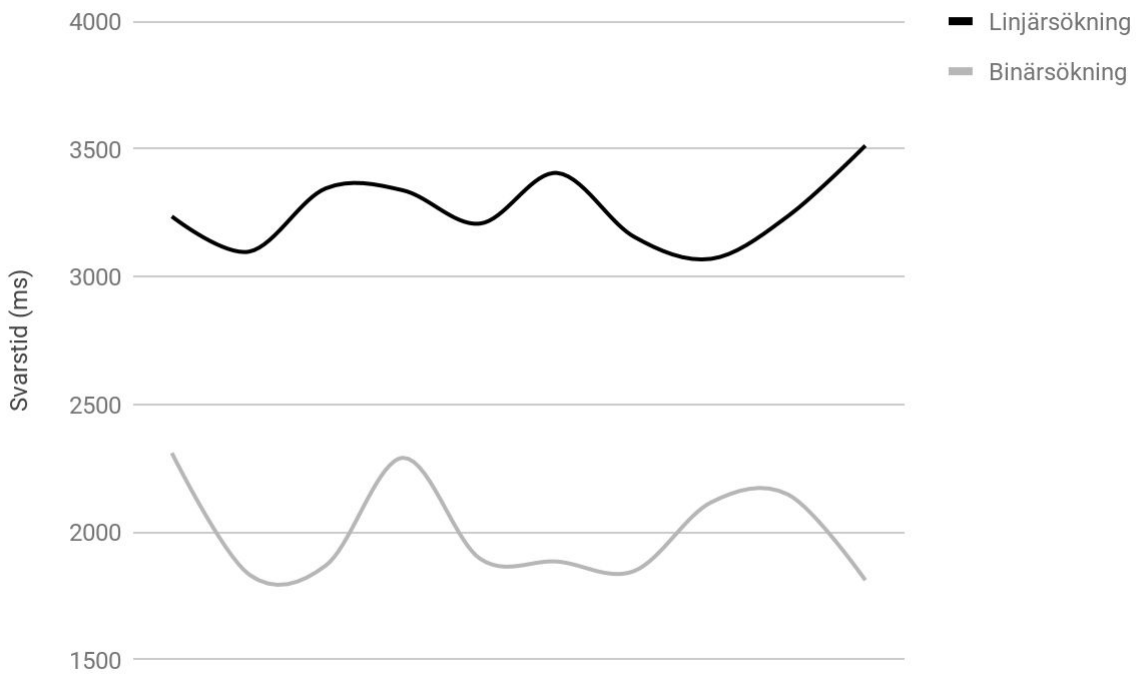
Tio värden slumpades och söktes efter. I figur 12 visas tiderna för linjär-och binärsökning för samma slumpade värden. Medeltiden för linjärsökningen var 2800 ms och 2426 ms för binärsökningen. I medeltal var linjärsökningen 15 % långsammare med en skillnad i medeltid på 374 ms.



Figur 12: Medeltid för slumpvärde

4.4.2. Sökning efter det sista värdet

För att mäta ett extremfall gjordes sökningar efter det sista värdet, det vill säga det värde som låg på sista raden. Samma data eftersöktes tio gånger med några enstaka sekunders intervall. Binärsökningens tid gav ett medelvärde på 2002 ms medan linjärsökningen gav ett medelvärde på 3261. Linjärsökningen var således 63% långsammare än binärsökningen med en skillnad på 1260 ms (se figur 13).

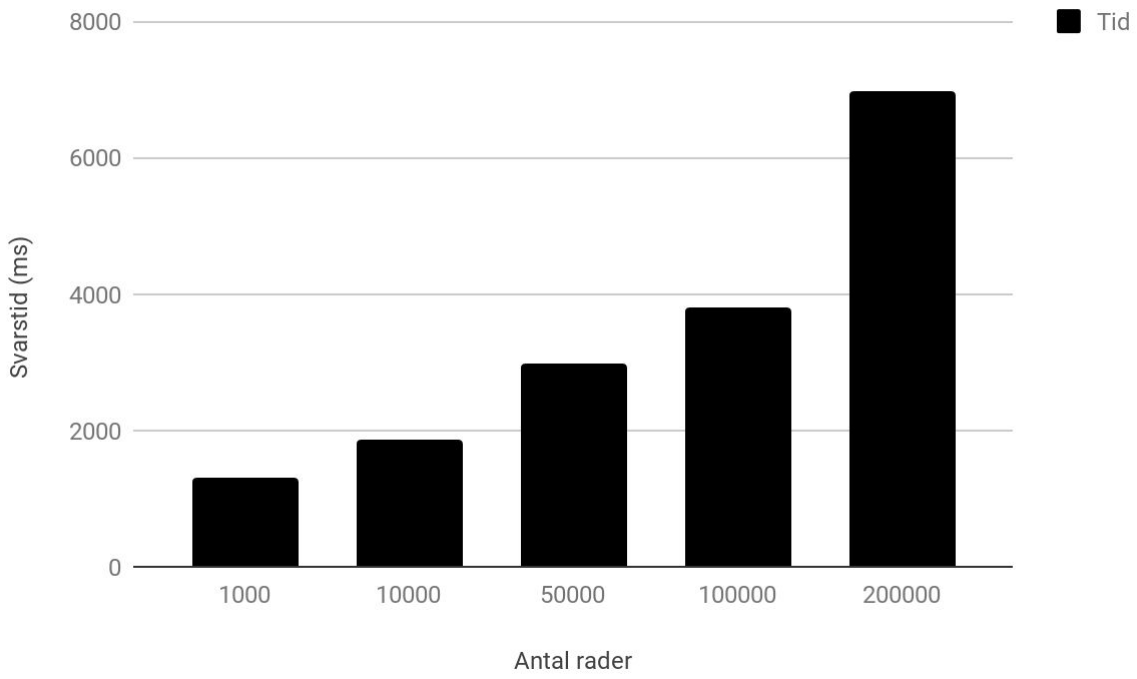


Figur 13: Medeltid för sistavärde

4.5. Responstid och uppskalning

För att kontrollera hur webbtjänstens tid för respons påverkades vid uppskalning gjordes mätningar via Postman för ett *sheet* innehållandes 1 000, 10 000, 100 000 och 200 000 rader. Sökningen gjordes efter ett värde som inte fanns i det *sheet* som var aktuellt. Detta tvingade webbtjänsten att söka igenom samtliga rader. I figur 14 syns en graf över svarstid för det antal rader som genomsöktes. I de avstämningsmöten som hölls med Telavox skulle det finnas utrymme för anrop och respons så att det skulle upplevas som naturligt för en inringare. Om det skulle bli tyst för länge fanns risk att en inringare skulle misstänka att något var på tok och avbryta samtalet. För 100 000 rader var medeltiden 4081 ms vid anrop strax efter varandra.

Vid uppskalning visade sig att ett *sheet* får längre svarstider vid hantering av data vid ett radantal på runt 200 000 vilket gör det något besvärligare att arbeta i. Trots att maxgränsen är på 2 000 000 celler skulle ett *sheet* bli svårt att använda på det sätt som är tänkt för webbtjänsten då det tar tid att läsa in data och uppdatera celler vilket gör aktuellt *sheet* svårarbetat. Om en *kund* har fler än 100 000 *slutkunder* skulle aktuellt *sheet* behövas delas upp i flera.



Figur 14: Uppskalning

4.6. Mellanlagring

Initialt användes en inbyggd funktion i Google Sheets vid namn “Vlookup” för att genomsöka ett *sheet*. Det innebar att resultatet av sökningen mellanlagrades i ett *sheet* innan det returnerades. Detta visade sig vara riskfyllt då datan skulle kunna skrivas över vid flera samtida anrop. För att minimera denna risk skrevs webbtjänsten om så att resultatet av sökningen lagrades lokalt i webbtjänsten istället för i en cell i aktuellt *sheet*.

4.7. Dataformat

Ett problem med att använda Google *Sheets* för att lagra den data som webbtjänsten skulle söka igenom var standardformateringen i Google *Sheets*. *Sheets* tog bort inledande nollor i de telefonnummer som matades in då det som standard vid inmatning av siffror tolkade telefonnumret som ett tal istället för en sträng. Utöver detta krävdes det från Telavox sida att telefonnumren var formaterade likadant med komplett landsnummer utan ‘+’ och ‘-’. För att underlätta uppladdning av data till ett sheet gjordes det därför en *Sheets*-mall som på förhand var formaterad för text och innehöll även ett skript som kontrollerade så att värdena i *sheetet* lagrades i korrekt format.

4.8. Dataintegritet och konfidentialitet

Två sätt att identifiera inringare var aktuella för examensarbetet; inringande nummer och möjlighet för *slutkunder* att knappa in ett nummer. Detta nummer skulle exempelvis kunna vara ett personnummer eller organisationsnummer som författarna till denna rapport valde att

kalla *identifikationsnummer*. Oavsett vad som skulle bli aktuellt skulle Telavox att behöva få tillgång till *kundens sheet*.

4.8.1. Inringande nummer som identifikationsnummer

Om inringande telefonnummer skulle användas som *identifikationsnummer* skulle den data som webbtjänsten söker igenom vara data som Telavox sedan tidigare haft tillgång till. Det som skulle vara ny information för Telavox var således vilken anknytning det inringande numret skulle kopplas till.

4.8.2. Kund- eller personnummer som identifikationsnummer

Baserat på om kund- eller personnummer skulle användas hade Telavox fått tillgång till de kundnummer eller personnummer som var kopplade till deras *kunder.* Detta eftersom kunddata skulle lagras i ett *sheet* som Telavox har tillgång till.

4.9. Utökad funktionalitet, UX eller produktionssättning

Under de veckovisa möten som hölls mellan författarna till denna rapport och handledaren på Telavox diskuterades åt vilket håll arbetet skulle ta sin riktning när *POC* bevisats. Utav olika alternativ föll valet på att förbereda webbtjänsten inför en produktionssättning. Detta innebar att analysera vilken funktionalitet som behövdes finnas inför produktionssättning. Detta inkluderande en fallstudie, UX-arbete, design av en mellanliggande service och testning.

4.9.1. UX

Gällande UX stod valet mellan att ta fram en prototyp som en sida i *Flow Admin* eller lägga till funktionalitet direkt i *växeltjänster*. Att ta fram en sida till *Flow Admin* gjorde att den kunde designas från grunden utan att ta hänsyn till tidigare funktionalitet som om det skulle tas fram i *växeltjänster.* Valet föll därför på att göra wireframes utifrån en ny sida i *Flow Admin.* Det skulle göra det möjligt att i ett tidigare stadie kunna testa produkten ut mot en *kund* utan att påverka den befintliga funktionalitet som redan fanns i *växeltjänster.* Om Telavox senare skulle lansera webbtjänsten mot samtliga *kunder* skulle implementering blivit mer enhetlig genom att integrera webbtjänsten i *växeltjänster*.

4.9.2. Produktionssättning

Författarna av denna rapport hade för avsikt att styra arbetet mot att lyckas produktionssätta webbtjänsten för att sedan kunna testa den mot en av Telavox *kunder*. Då *talsvaret* ej var förberett på agera på någon respons var det ett arbete som Telavox åtog sig att lösa. På grund av tidsramen för detta examensarbete förblev identifikationsprocessen som i den ursprungliga prototypen, det vill säga att ett samtal identifieras med hjälp av ett manuellt inknappat *identifikationsnummer* och inte ett telefonnummer. När det fanns en testmiljö att testa i gjordes analys kring vilka tester som skulle göras.

4.10. Fallstudie

I detta kapitel analyseras möjligheten för den *kund* som uttryckt intresse för webbtjänsten att köra den i produktion. Genom att på ett teoretiskt plan gå igenom de punkter som för *kunden* är centrala i deras problemformulering och kontrollera om webbtjänsten skulle kunna lösa dessa.

4.10.1. Bakgrund

Det företag som undersöktes i denna fallstudie var ett större företag som tillhandahöll försäljning av bilar samt verkstäder i flera län. Om en *slutkund* skulle lämna in sin bil för service eller reparation fick *slutkunden* en personlig servicetekniker tilldelad. Genom detta system fick *slutkunden* en personlig kontakt med kännedom om bilens historik och kunde på så vis upprätthålla och vårda sina kundrelationer även efter en lyckad bilförsäljning. Företaget har vid författandet av denna rapport ca 23 000 *slutkunder* med registrerad personlig servicetekniker.

4.10.2. Problemformulering

Det fanns två huvudsakliga problem som företaget önskade lösningar på gällande deras telefoni.

- Deras *slutkunder* visste sällan vem som var deras personliga servicetekniker.

- Eftersom företagets anläggningar låg utspridda hände det att *slutkunder* ringde fel anläggning för verkstadsbokning. Detta genererade ett tidsödande arbete i att lokalisera och sedan koppla *slutkunden* till rätt anläggning och till rätt personlig servicetekniker.

4.10.2. Efterfrågad lösning

Kund hade som önskemål att *slutkunder* som ringde in och hade en en personlig servicetekniker skulle automatiskt bli kopplad direkt till rätt tekniker oavsett vilken regions huvudnummer *slutkunden* ringde till. Identifieringen av *slutkund* skulle ske genom det telefonnummer *slutkunden* ringde ifrån.

4.10.3. Webbtjänstens möjligheter

Baserat på det antal *slutkunder* det rörde sig om samt den efterfrågade lösning som *kunden* uppgav skulle webbtjänsten kunna fungera i produktion då den kunde behandla de problem och de önskemål *kunden* angav.

Webbtjänsten hade vid skrivandet av denna rapport en kapacitet som med marginal kunde hantera det antal *slutkunder kunden* uppgivit. Antalet skulle dessutom kunna fyrfaldigas utan att *kunden* skulle behöva dela upp i flera *sheet*.

Som prototypen i *Flow Admin* utformades skulle det gå att ansluta webbtjänsten på samtliga telefonnummer hos *kunden* som är kopplade till en verkstad och där en personlig servicetekniker. Då *kunden* hade som önskemål att samtliga *slutkunder* skulle bli rätt kopplade oavsett vilken region de ringde skulle samtliga *slutkunders* telefonnummer kunna finnas i ett och samma *sheet.* I prototypen skulle sedan samma *sheetId* anges på samtliga av de huvudnummer där webbtjänsten skulle vara aktiverad.

4.10.4. Slutsats av fallstudien

Utifrån den analys som gjorts av *kundens* problemformulering och önskemål skulle webbtjänsten kunna användas för att tillgodose dessa. Detta bygger dock på att identifiering av inringande *slutkund* sker med hjälp av *slutkundens* telefonnummer.

5. Resultat

Resultatet av examensarbetet blev den webbtjänst som utvecklats samt wireframes av det förslag som togs fram till *Flow Admin.*

5.1. Webbtjänsten

Applikationen skrevs i Javascript med hjälp av Google *Apps Script.* Vid exekvering kunde webbtjänsten söka i ett givet *sheet* efter ett givet *identifikationsnummer.* Vid exekvering anropade applikationen den URL som var angiven i den specifikation som syns i figur 8. För att det skulle gå att göra en sökning med hjälp av applikationen krävdes det att applikationen skickade med ett giltigt *sheetI*d som URL-parameter. Ett giltigt *sheetI*d i detta fall innebar att att både *sheetet* är delat med det *Google-konto* som körde applikationen och att *sheetId* ledde till ett unikt *sheet.* Responsen från ett lyckat anrop till applikationen gavs i JSON-format, och beskrivs i specifikationen i figur 8.

5.1.1. Källkod

Källkoden[Appendix 9.2] till webbtjänsten består av tre metoder:
`doGet(e)`, `searchSheet(sheetId, value)` och `buildResponse(result)`.

- `doGet(e)` är den metod som först exekveras vid anrop till applikationen, och den kontrollerar så att givet *sheetId* är giltigt. Vid lyckat anrop till `searchSheet(sheetId, value)` så anropas sedan `buildResponse(result)` för att generera respons i JSON-format.
  Med hjälp av den inbyggda klassen ContentService [8] skapas sedan den text-output som returneras till anropande applikation.

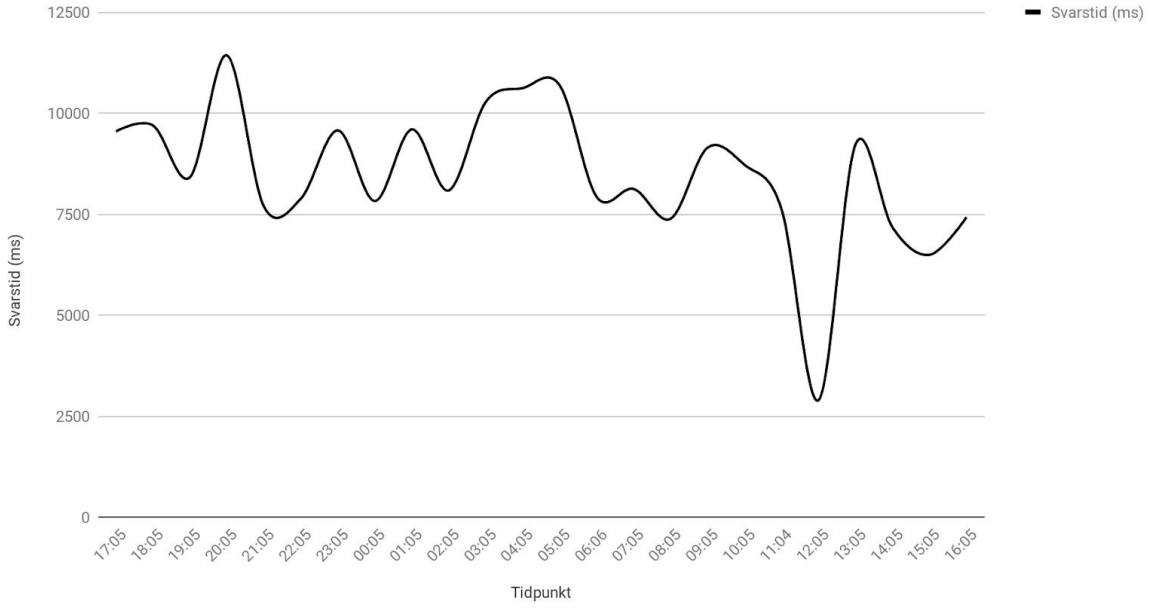
- `searchSheet(sheetId, value)` är den metod som genomför sökningen utifrån de givna parametrarna. Den använder sig av de inbyggda klasserna SpreadSheetApp [4], SpreadSheet [5], Sheet [6], Range [7] för att få åtkomst till den data som finns i *sheetet*, för att sedan med hjälp av linjärsökning söka igenom den första kolumnen i det givna *sheetet* efter det givna värdet. Beroende på resultatet av denna sökning så returneras antingen en tom sträng eller det matchande värdet.

- `buildResponse(result)` genererar JSON-respons beroende på ett givet resultat, som sedan `doGet(e)` returnerar till anropande applikation.

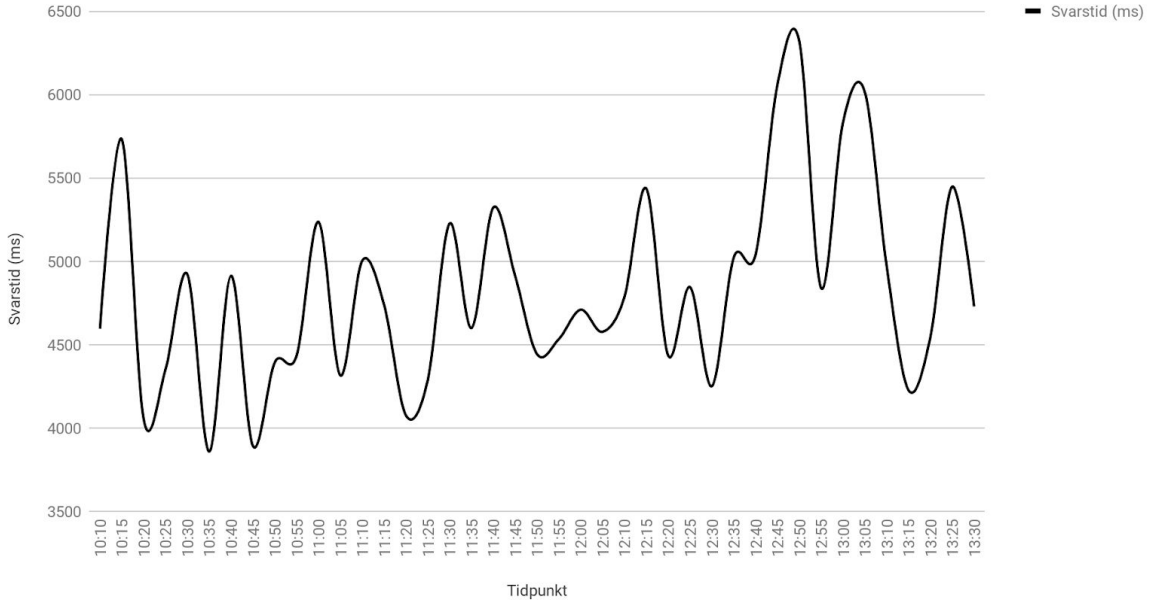
5.1.2. Responstid vid url-anrop

Genom att använda Postman för att simulera anrop kunde den totala tid det tog från anrop till respons mätas. Mätningar gjordes var 5:e minut samt var 60:e minut i *sheet* som innehöll 100 000 rader. Sökningar gjordes på sista elementet i aktuellt *sheet*. Initialt gjordes enstaka

manuella anrop och då verkade det som om ett första anrop tog ca 7-10 sekunder medan anrop någon minut senare minskas till ca 3 sekunder. Efter att inte rört det aktuella *sheet* som arbetades i under en timme gick tiden upp till 7-10 sekunder igen. Detta på grund av Googles cache-funktionalitet. I figur 15 syns en tydlig minskning vid 12:05 och det är ett manuellt anrop gjordes ca klockan 11:50, på så vis låg aktuellt *sheet* i Googles cache. Vid anrop var 60:e minut var medelvärdet 8478 ms (se figur 15). Var 5:e minut resulterade i ett medelvärde om 4827 ms (se figur 16).



Figur 15: Anrop var 60:e minut



Figur 16: Anrop var 5:e minut

5.1.3. API-specifikation

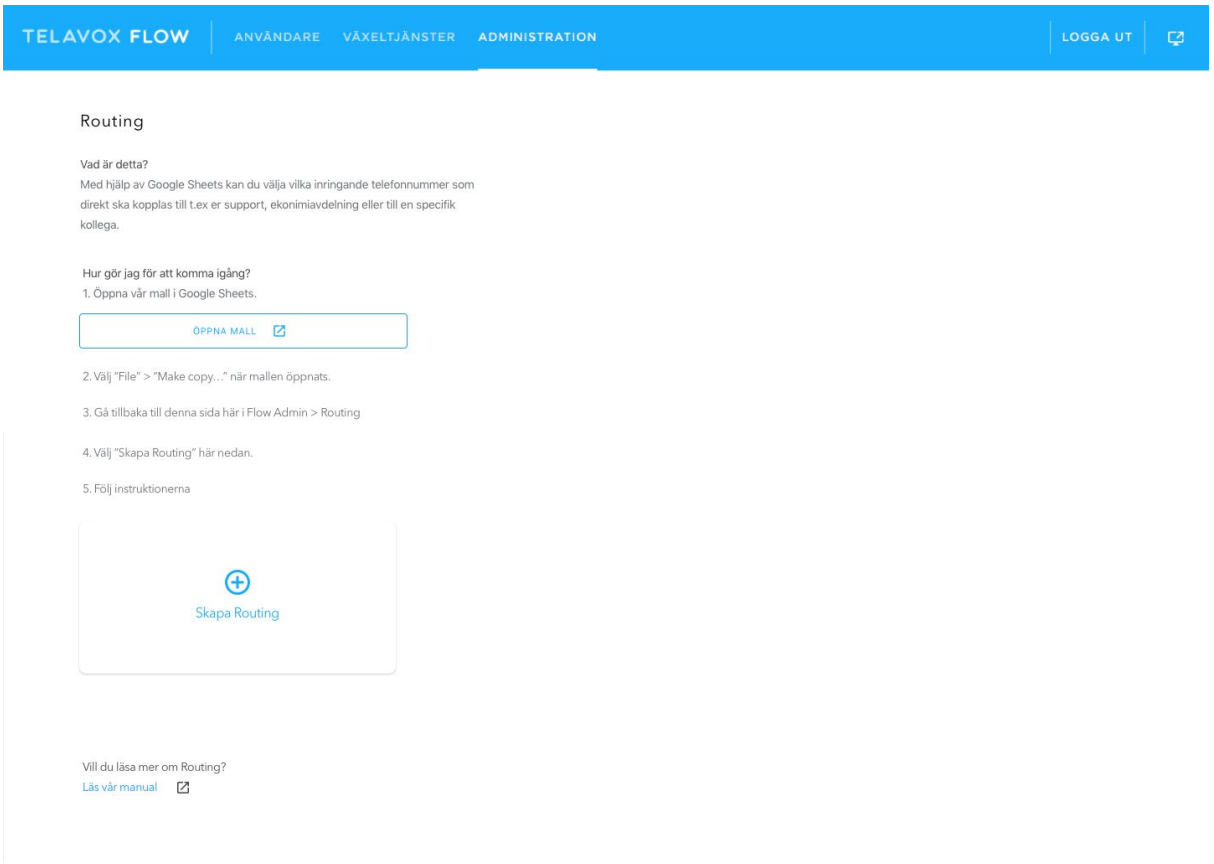
API-specifikationen [appendix 9.3] specificerar hur anrop ska göras, vilka parametrar som ska användas och hur responsen ser ut för webbtjänsten.

5.2. Prototyp av sidan för "Routing" till *Flow Admin*

I detta kapitel redovisas resultaten av webbtjänsten till *Flow Admin* och användarmanualen som kan användas vid anslutning.

5.2.1. *Flow Admin*

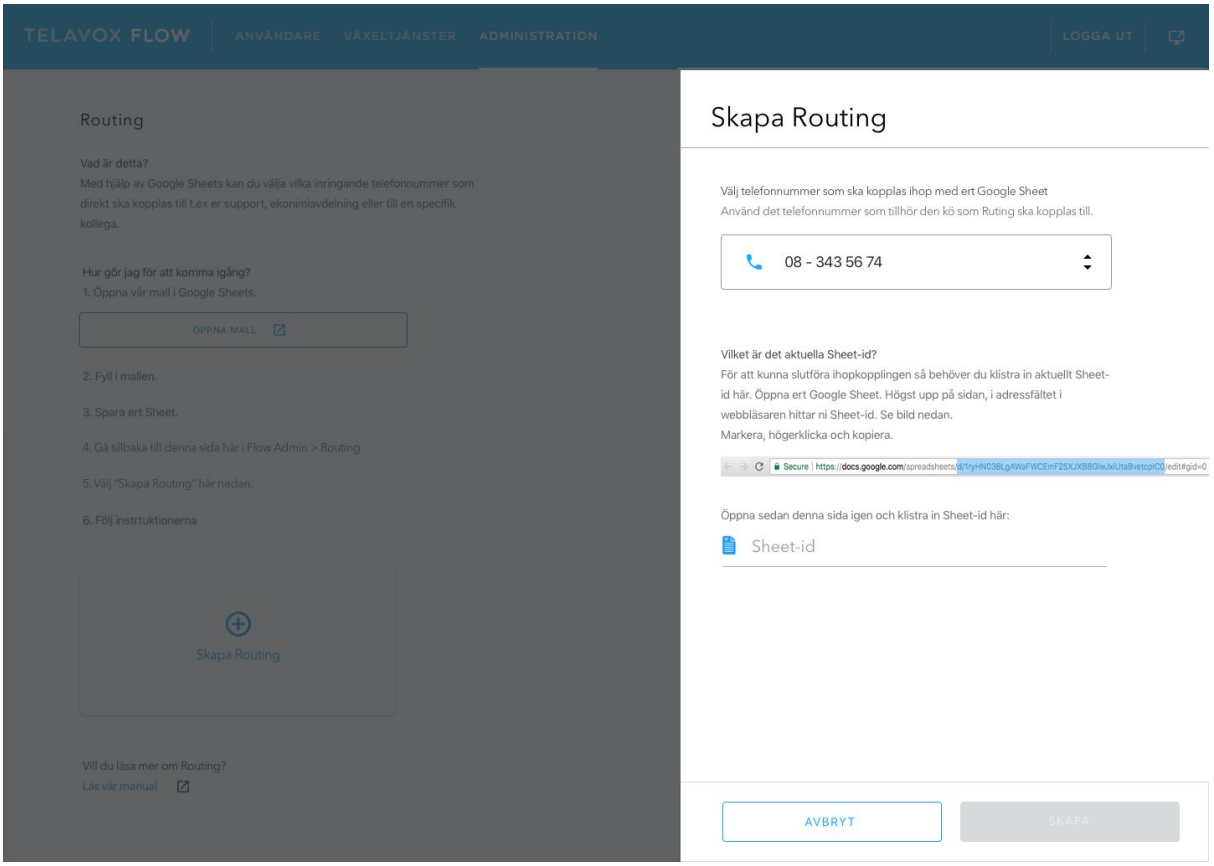
Den klickbara prototypen till produktsidan på *Flow Admin* består i grunden av en startsida som innehåller information hur man kommer igång och hur man anslutar sig till Routing (se figur 17).



Figur 17: Wireframe av startsida

Vid klick på "Skapa Routing" ska en meny öppnas från vänster med instruktioner och fält för att koppla ihop *kundens sheet* med webbtjänsten samt välja det telefonnummer webbtjänsten ska vara aktiv på. Telefonnummer väljs från en rullist där de telefonnummer kopplade till den aktuella *kunden* ska dyka upp. Fältet för sheetId kopieras in från webbläsaren enligt instruktionerna. I figur 18 visas denna ruta. Först när all data matats in ges möjlighet att

klicka på “Skapa”. På samma sätt fungerar sidan när man är inne och ska redigera. Först när data ändrats ska möjlighet att “Spara” ges. Resterande av prototypens sidor finns i appendix 9.1.



Figur 18: Wireframe av sidan för “Skapa Routing”

Resterande sidor bygger på samma grund som de två nämnda ovan. Skillnaden är befintliga anslutningar visas på startsidan om webbtjänsten finns aktiverad på något nummer sedan tidigare.

Sidan “Skapa Routing” finns med i olika utföranden då möjlighet att redigera *sheetId* och kopplat telefonnummer.

5.2.2. Användarmanual

Användarmanualen [appendix 9.4] är ett komplement till den information en *kund* får vid anslutning via *Flow Admin.* I prototypen till *Flow Admin* finns manualen länkad till användare.

6. Slutsats

Gällande *molnplattform* valdes Google då den uppnådde det som efterfrågats av Telavox gällande användbarhet och tillgänglighet. Vid författandet av rapporten fanns inte funktionalitet för motsvarande i Excel Online. Att utveckla motsvarande via Amazon AWS är troligtvis möjligt, men det skulle vara mer komplext och behövas mer utveckling från grunden än vad som var tanken med detta examensarbete.

Genom Google *Apps Script* kunde webbtjänsten anropas från Telavox befintliga Java-applikation och skicka med ett *identifikationsnummer*. Med hjälp av detta *identifikationsnummer* sökte webbtjänsten igenom ett *sheet* efter en träff och sedan returnera denna på JSON-format. På så vis kan Telavox telefonisystem koppla samtalet till det returnerande numret. Så genom möjligheten för Telavox att anropa en URL körs sedan kod via Google *Apps Script* på Googles *molnplattform* vilket besvarar problemet kring hur Telavox telefonisystem ska kunna exekvera kod på Googles *molnplattform*.

Det var svårt att definiera en tid som skulle upplevas som naturlig för en slutkund att vänta när de ringer in till en kund. Beroende på hur kundens telefonväxel är konstruerad och hur långa de ljudmeddelanden som spelas upp är kan tiden som känns naturlig att vänta skilja sig avsevärt. På grund av detta var det svårt att sätta en exakt gräns på hur lång tid det fick ta att göra anrop till webbtjänsten. Man hade kunnat spara tid vid anrop genom att göra sökningar binärt istället för att söka linjärt. Detta hade dock ställt krav på hur datan skall vara sorterad i en *kunds sheet*. Då själva genomsökningen av *sheetet* stod för en relativt liten del av totala anropstiden bestämdes det att linjärsökning skulle användas för att undvika att ställa krav på sorteringen av datan.

Vilka personuppgifter som får lagras på en molnplattform är i grunden en juridisk fråga som inte fanns inom ramen för detta examensarbete. Däremot kan nämnas att förutsatt att webbtjänsten använder ett inringande telefonnummer som *identifikationsnummer* får Telavox inte tillgång till något de tidigare inte känt till. Om däremot ett personnummer används ligger det på *kunden* att se till att avtala om detta med sina *slutkunder* huruvida de får tillåtelse att lagra och samla in deras personuppgifter hos Google.

För att besvara frågan kring hur ett samtal ska kopplas till rätt *anknytning* beroende på *identifikationsnummer* gjordes tester för att bekräfta att *grundläggande funktionalitet* uppnåtts. Källkoden[appendix 9.2] visar hur man med hjälp av Googles *Apps Script* kan arbeta med data i Googles olika typer av tjänster utan att behöva mellanlagra det i ett befintligt dokument. Det kan således lagras som lokala variabler som returneras till den anropande applikationen.

När denna rapport skrivs finns en fungerande prototyp av webbtjänsten vilket gör möjligt att uppnå examensarbetets syfte. Det finns *kunder* till Telavox som är redo att använda denna webbtjänst för att på så vis förenkla och effektivisera sin kommunikation. Det är även Telavox ambition att nu sikta på att driftsätta prototypen.

6.1. Reflektion över etiska aspekter

Detta kapitel behandlar de etiska aspekter som författarna av denna rapport identifierat under arbetets gång.

6.1.1. Sekretess av personuppgifter

Webbtjänsten är en helt öppen applikation som vem som helst med rätt länk och parametrar kan anropa. Detta skulle i teorin kunna göra att någon som fått tillgång till en *kunds sheetId* och känner till något av de *identifikationsnummer* som finns i detta skulle kunna ta reda på vilket telefonnummer hos *kunden* som är kopplat till *identifikationsnumret*. Det är ej möjligt att tvärt om ta reda på *identifikationsnumret*. Det som således går att ta reda på om man har *sheetId* och *identifikationsnummer* är om *slutkunden* bakom *identifikationsnumret* har någon koppling till företaget bakom aktuellt *sheetId.*

6.1.2. Avtal mellan parter samt datalagring och insamling

Om Telavox skulle erbjuda webbtjänsten till sina *kunder* är de tre parter inblandade; Telavox, *kunden* och Google. Det Telavox måste ha tillgång till för att webbtjänsten ska fungera är tillåtelse att söka i en *kunds sheet*. Om detta sedan är ett telefonnummer eller annat *identifikationsnummer* är en överenskommelse mellan *kunden* och Telavox. Telavox är en telefonoperatör och för att kunna erbjuda viss funktionalitet kräver det tillgång till relaterad data. Om personnummer används som *identifikationsnummer* bör avtal mellan Telavox och *kund* upprättas för att reglera vad Telavox får göra med den data som finns tillgänglig så det endast används för webbtjänstens ändamål.

Mellan en *kund* och deras *slutkunder* bör *kunden* vara uppmärksam på att webbtjänsten använder sig av en tredjepartsleverantör som i detta fall är Google. Det är något en *kund* bör avtala med sina *slutkunder* om så att vetskap om att denna datainsamling och tredjepartslagring existerar men att den endast har för avsikt att användas i webbtjänstens syfte. Likaså bör *kunden* ta del av de avtal som finns mellan *kunden* och Google innan de registrerar sig för webbtjänsten för att vara införstådd med vilken data som Google får tillgång till och vad de får göra med denna.

6.1.3. Samhällsnytta

Att kunna kommunicera effektivt borde ligga i alla företags intresse. Tyvärr är det svårt för ett mindre företag med begränsade resurser att få tillgång till den teknik som hade gjort kommunikationen mer effektiv. Denna teknik har varit reserverad för större företag med

resurser att kunna utveckla detta själva. Genom detta examensarbete kan det bli möjligt för mindre företag att redan i ett tidigt stadie i deras utvecklingsprocess kunna få tillgång till teknik som gör att de kan optimera sin kommunikation och på så vis kunna använda sin redan i många fall begränsade tid till annat som främjar företagens utveckling. På så vis tror författarna av detta examensarbete att det medför samhällsnytta.

6.2. Framtida utvecklingsmöjligheter

Detta kapitel beskriver de framtida utvecklingsmöjligheter som identifierats under arbetets gång.

6.2.1. Excel Online

När olika *molnplattformar* analyserades föll Microsoft Excel bort då stöd för att köra makron i online-versionen saknades. Skulle denna möjlighet läggas till skulle vidare undersökningar kunna göras om Excel skulle kunna vara en värdig kandidat för att utveckla en liknande webbservice.

6.2.2. Google API

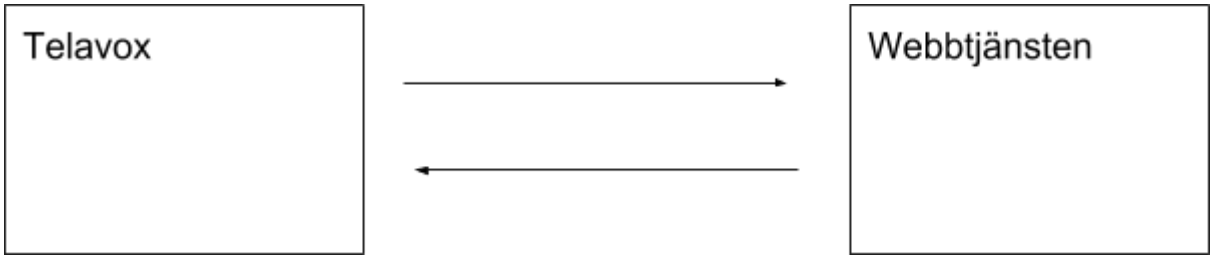
För att kunna använda sig av *Apps Scripts* funktion att driftsätta som "körbart API" krävdes att den kallande applikationen delade samma Cloud-konto som API:et vilket begränsade webbtjänsten att kunna anropa API-tjänsten externt. Om möjlighet ges att kunna anropa detta externt finns möjlighet att utveckla webbtjänsten på detta sätt också för att undersöka hur hantering av API-nycklar görs och vilken inverkan det har.

6.2.3. Proxy-server

Vid skrivande av denna rapport sker kommunikation mellan Telavox och webbtjänsten direkt, som figur 19 visar. Detta innebär att webbtjänsten har ansvar för all respons som ges. Att webbtjänsten har ansvar för responsen innebär att ingen respons ges om Googles servrar inte svarar. Det medför även risk att tidskrävande anrop görs till webbtjänsten med felaktiga parametrar.

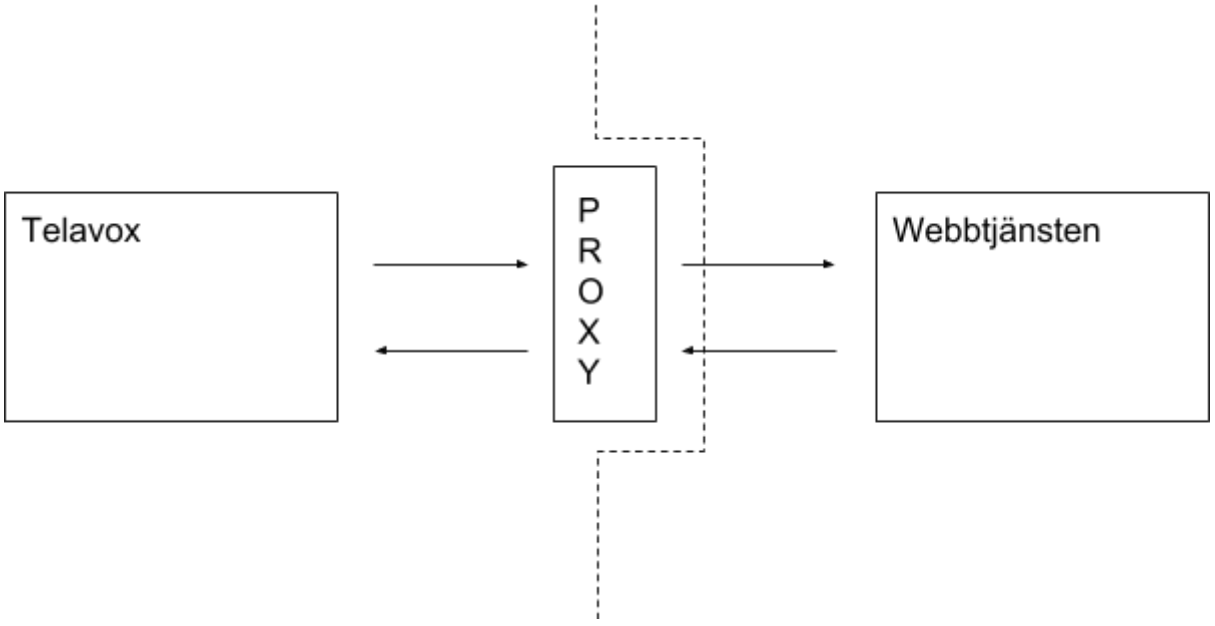
Genom att implementera en proxy-server som ett mellanlager mellan webbtjänsten och Telavox, som figur 20 visar, så hade det varit möjligt att effektivisera anrop till webbtjänsten. Proxy-servern skulle kunna hantera timeouts om webbtjänsten inte ger någon respons, och då kunna ge standardiserad respons tillbaka till Telavox. Den hade även kunnat kontrollera format på de parametrar som skickas in för att undvika onödiga anrop till webbtjänsten. Genom att Telavox själva ansvarar för drift och övervakning av denna proxy-service hade man även kunnat få samma statistik på tillgänglighet som för Telavox övriga tjänster.

Före:



Figur 19: Utan proxy

Efter:

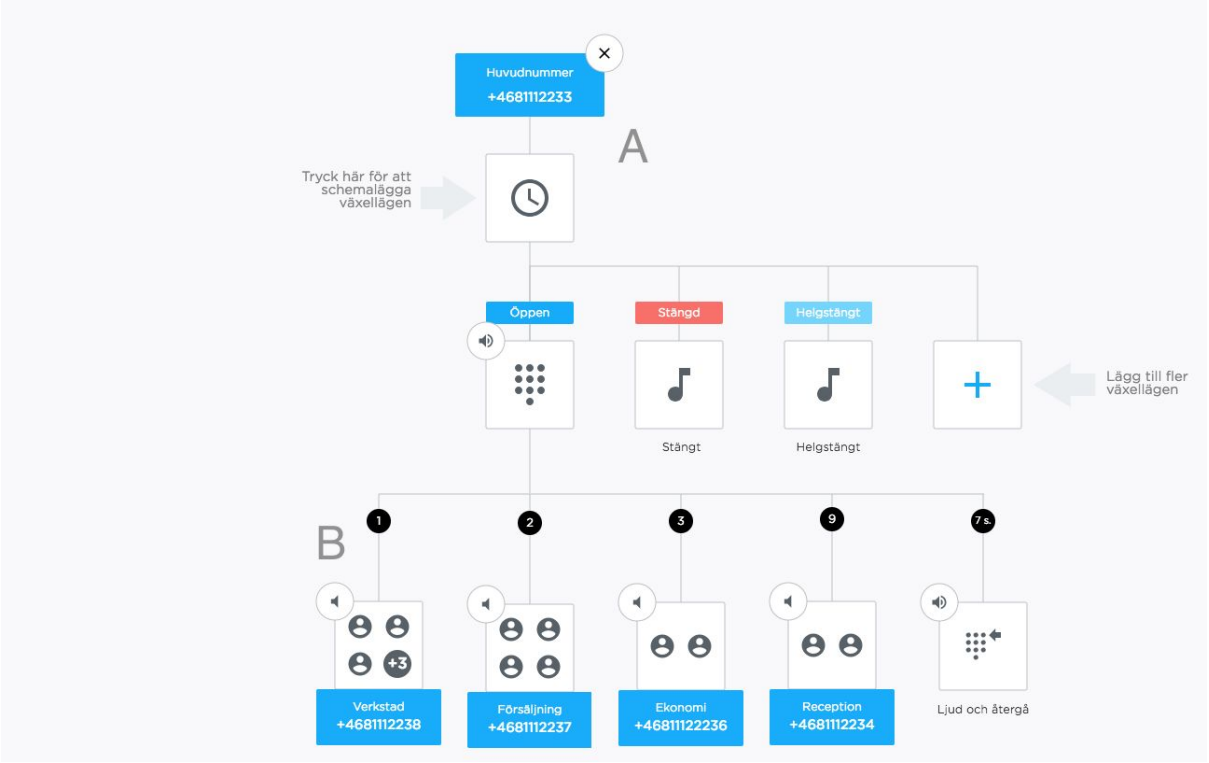


Figur 20: Med proxy

6.2.4. Implementation i *växeltjänster*

För åstadkomma en mer enhetlig implementering av webbtjänsten skulle ett framtida utvecklingsområde kunna vara att bygga in funktionalitet för webbtjänsten i *växeltjänster*. I figur 21 finns två bokstäver A och B. Vid bokstaven A, det vill säga direkt när en *slutkund* ringer in till *kunden,* skulle genomsökning med hjälp av webbtjänsten kunna göras. Oavsett vilket knappval *slutkunden* sedan väljer kan webbtjänsten arbeta i bakgrunden under tiden

välkomstmeddelandet spelas upp. Om *slutkunden* väljer ett knappval där webbtjänsten finns aktiverad är sökningen redan gjord och samtalet kan då direkt kopplas till rätt *anknytning*. I figur 21 är webbtjänsten aktiverad vid bokstaven B, vilket innebär att samtalet skulle kopplas vidare utifrån webbtjänstens respons istället för till verkstadsnumret. Detta är ett exempel på de möjligheter som skulle vara möjliga vid implementation i *växeltjänster*. Fokus skulle kunna ligga på att utöka funktionaliteten i växeltjänster samt hur överblicken i användargränssnittet kan göras tydlig för *kund* i samband med detta.



Figur 21: Framtida funktionalitet i *växeltjänster*

7. Terminologi

Talsvaret - Är den Java-applikation som hanterar inhämtande av knapptryckningar från inringare i Telavox växeltjänster

Grundläggande funktionalitet - Funktionalitet för att anropa en länk, söka igenom data och sedan returnera data.

Kund - Kund till Telavox

Slutkund - Kund till Telavox kunder

Anknytning - Ett telefonnummer hos en *kund*

Molnplattform - Google Cloud, Microsoft Azure och Amazon Web Services är molnplattformar som erbjuder en uppsättning av olika molntjänster, bland annat olika typer av lagring och analys av stora mängder data.

Operatör - Anställd hos *kund* som hanterar inkommande telefoni

Identifikationsnummer - Telefonnummer/personnummer som identifierar en *slutkund*

sheetId - Nummer som identifierar en *kunds* Google Sheet där identifikationsnummer och anknytningar lagras.

Proof Of Concept (POC) - Är en demonstration för att verifiera att vissa koncept eller teorier har den potential för att kunna fungera i verkligheten. *POC* är således en prototyp som är designad för att demonstrera de möjligheter som finns hos prototypen. [16]

Minimum Viable Product (MVP) - Är en utvecklingsteknik där en ny produkt utvecklas med tillräcklig funktionalitet för att tillgodose “early adopters”. Den slutliga funktionalitet läggs sedan till efter man fått feedback från användare. [17]

Flow - Telavox molnbaserade kommunikationsplattform.

Flow Admin - Administrationssystem för *Flow.*

Växeltjänster - Verktyg för *kunder* att konfigurera sina upplägg i telefonin

JSON - Dataformat som ofta används vid utbyte av data mellan server och applikation.

sheet - Ett kalkylblad i Google Sheets.

Google Apps Script - Apps Script är en utvecklingsplattform för kod och applikationer som finns inbyggt i Googles olika applikationer och gör det möjligt att lägga till funktionalitet till bland annat Sheets och Docs.

Standalone Script - Ett fristående skript utvecklat i *Google Apps Script* som ej är direkt länkat till något dokument(t.ex. Ett *sheet*)

Google Web App - En webbapplikation via *Google Apps Script*

Monitor - En funktion i Postman som genererar automatiskt anrop och mäter dess tider.

8. Källförteckning

Ferreira, James. *Google Apps Script*. Second Edition. Sebastopol: O'Reilly Media Inc, 2014. [1]

Google Apps Script. Guides. Types of Scripts. *Standalone Scripts.* 2018. https://developers.google.com/apps-script/guides/standalone (Hämtad 2018-04-09) [2]

Google Apps Scripts. Guides. Types of Scripts. *Web Apps.* 2018. https://developers.google.com/apps-script/guides/web (Hämtad 2018-04-09) [3]

Google Apps Script. Reference. G Suite Services. Spreadsheet. *SpreadsheetApp.* 2018 https://developers.google.com/apps-script/reference/spreadsheet/spreadsheet-app (Hämtad 2018-04-09) [4]

Google Apps Script. Reference. G Suite Services. Spreadsheet. *Spreadsheet.* 2018 https://developers.google.com/apps-script/reference/spreadsheet/spreadsheet (Hämtad 2018-04-09) [5]

Google Apps Script. Reference. G Suite Services. Spreadsheet. *Sheet.* 2018 https://developers.google.com/apps-script/reference/spreadsheet/sheet (Hämtad 2018-04-09) [6]

Google Apps Script. Reference. G Suite Services. Spreadsheet. *Range.* 2018. https://developers.google.com/apps-script/reference/spreadsheet/range (Hämtad 2018-04-09) [7]

Google Apps Script. Reference. G Suite Services. Content. *ContentService.* 2018 https://developers.google.com/apps-script/reference/content/content-service (Hämtad 2018-04-30) [8]

Sketch. Updates. *Prototyping, Libraries on Sketch Cloud and an official iOS UI kit in Sketch 49*. 2018. https://blog.sketchapp.com/prototyping-libraries-on-sketch-cloud-and-an-official-ios-ui-kit-in-sketch-49-bf090c70796c (Hämtad 2018-04-26) [9]

Microsoft. Office. *Work with macros in Excel Online*. 2018. https://support.office.com/en-us/article/work-with-macros-in-excel-online-98784ad0-898c-43aa-a1da-4f0fb5014343 (Hämtad 2018-04-03) [10]

Google. Google Drive Help. *Files you can store in Google Drive*. 2018.

https://support.google.com/drive/answer/37603?hl=en (Hämtad 2018-04-03) [11]

Google Apps Script. Rest API. *Executing Functions using the Apps Script API*. 2018. https://developers.google.com/apps-script/api/how-tos/execute (Hämtad 2018-04-03) [12]

BlazeMater. Testing Hacks. *How to Use Postman to Manage and Execute Your APIs.* 2016. https://www.blazemeter.com/blog/how-use-postman-manage-and-execute-your-apis (Hämtad 2018-04-04) [13]

Postman. Sending API requests. *Requests*. 2018. https://www.getpostman.com/docs/v6/postman/sending_api_requests/requests (Hämtad 2018-04-04) [14]

Postman. Monitors. *Setting up a monitor.* 2018. https://www.getpostman.com/docs/v6/postman/monitors/setting_up_monitor (Hämtad 2018-04-04) [15]

Techopedia. Dictionary. Tags. Software. *Minimum Viable Product (MVP)*. 2018. https://www.techopedia.com/definition/27809/minimum-viable-product-mvp (Hämtad 2018-04-30) [16]

Techopedia. Dictionary. Tags. Enterprise. *Proof of Concept (POC)*. 2018. https://www.techopedia.com/definition/4066/proof-of-concept-poc (Hämtad 2018-04-30) [17]

Telavox. Startsida. *Effektiv kommunikation.* 2018. https://www.telavox.com/sv/ (Hämtad 2018-04-30) [18]

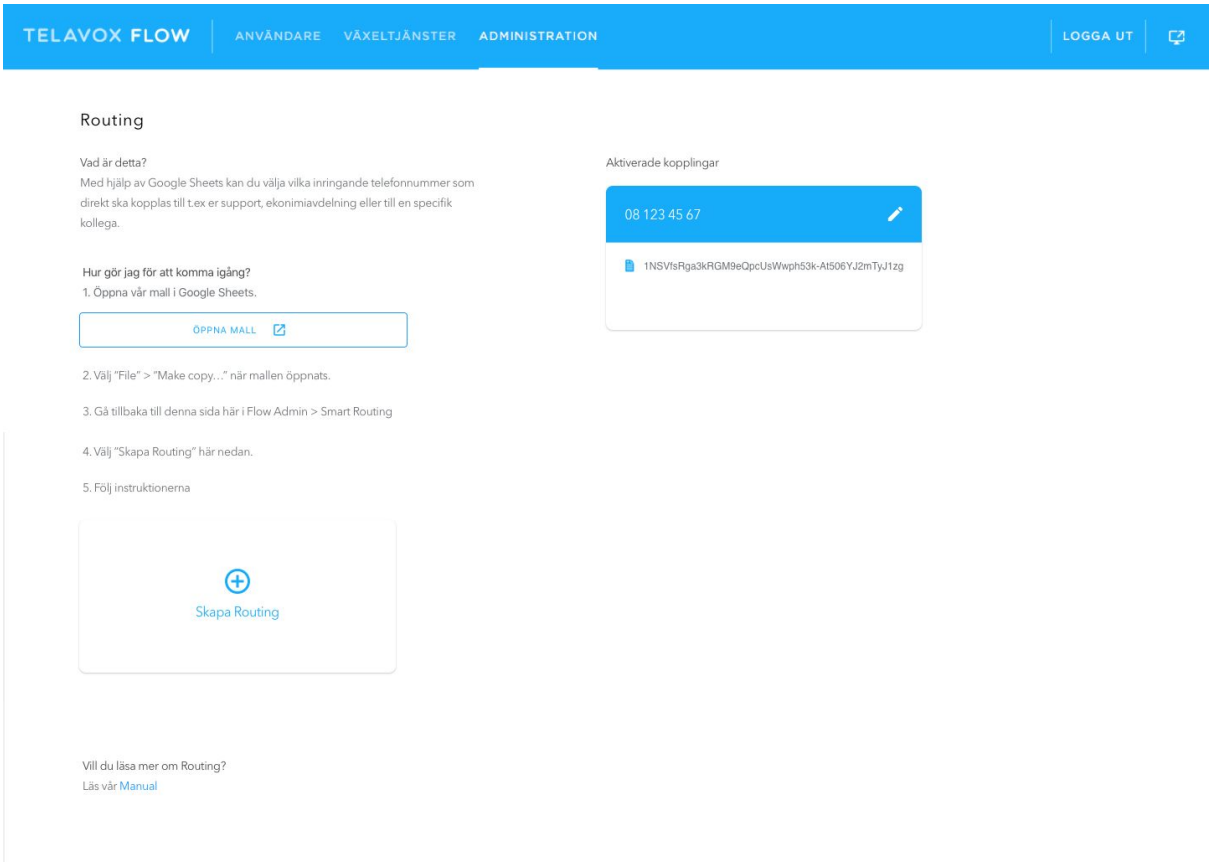
9. Appendix

I detta kapitel finns alla appendix listade.

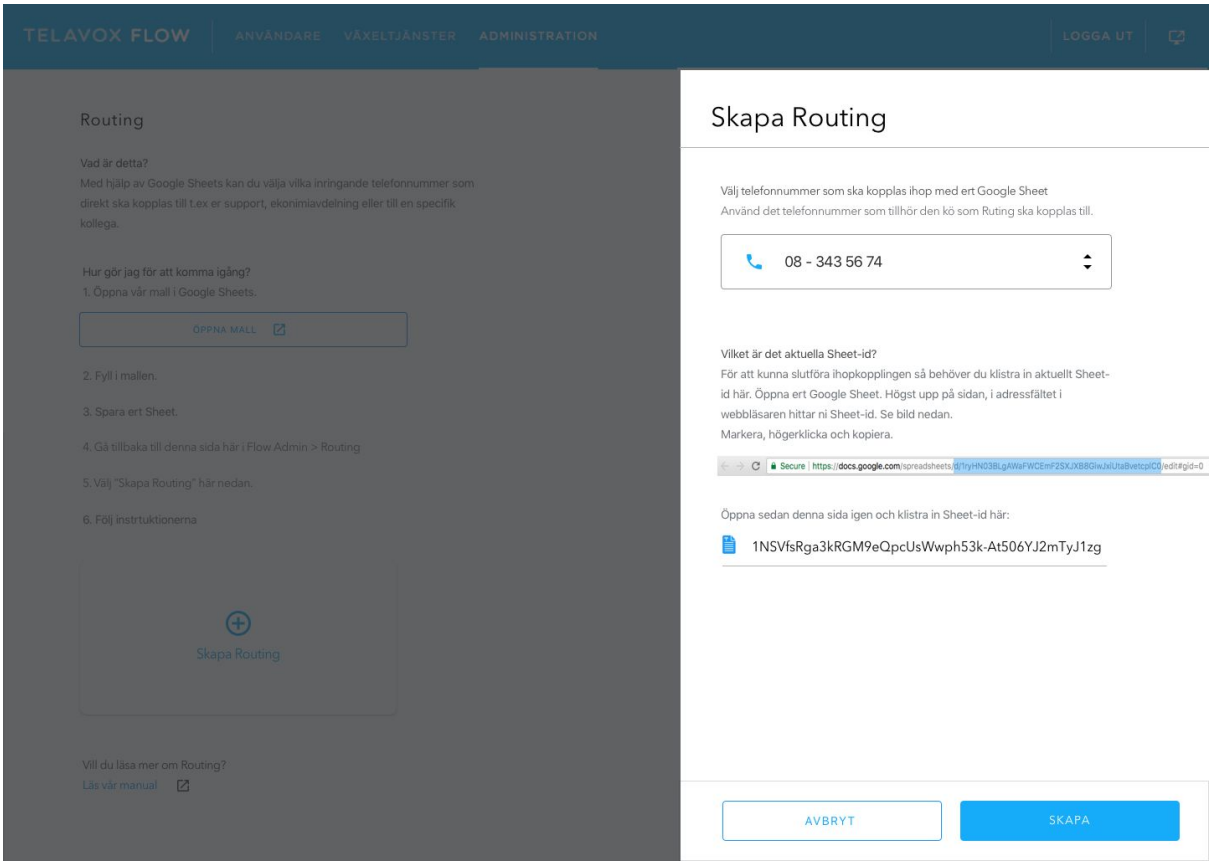
9.1. Skärmdumpar från användargränssnitt i Flow Admin

Skärmdumparna nedan visar exempel på ett användargränssnitt som skulle kunna implementeras i Flow Admin vid en produktionssättning.

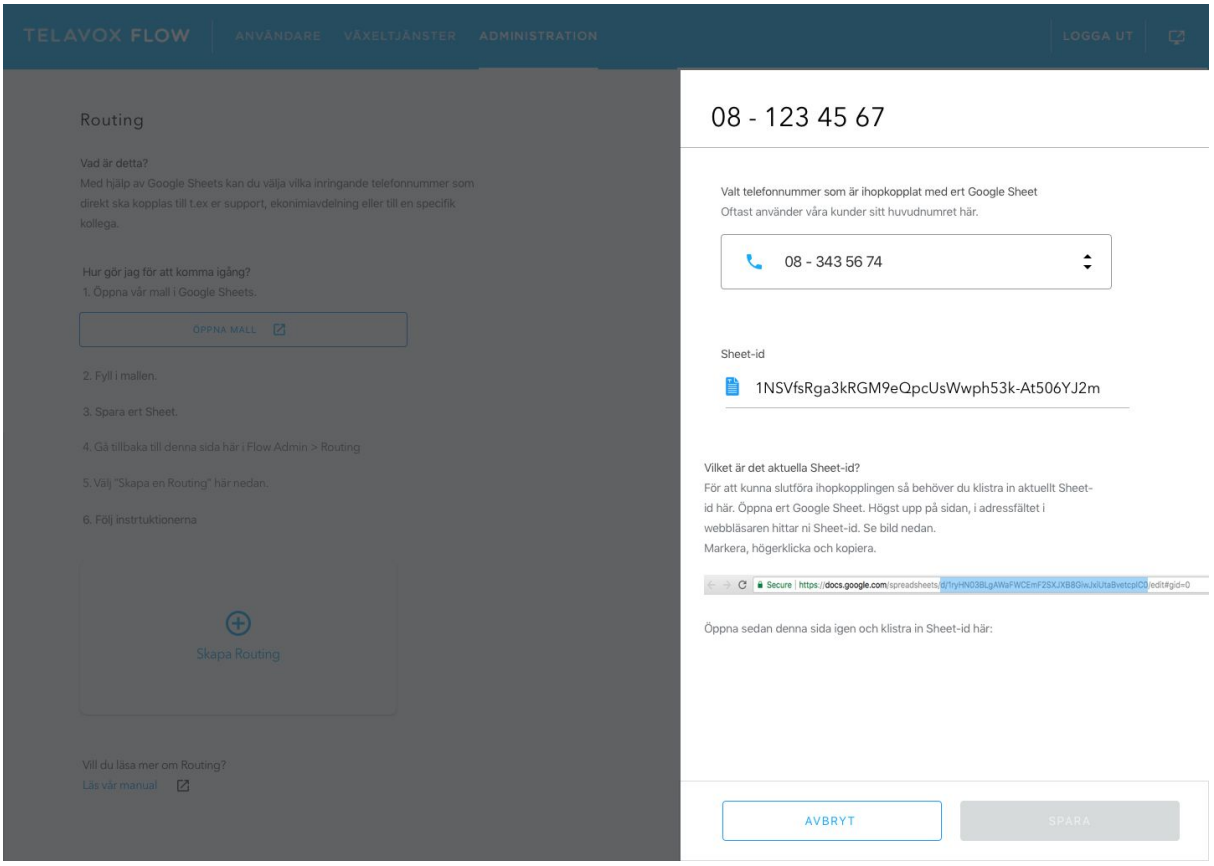
9.1.1. Startsida vid anslutning till Routing, ansluten sedan tidigare.



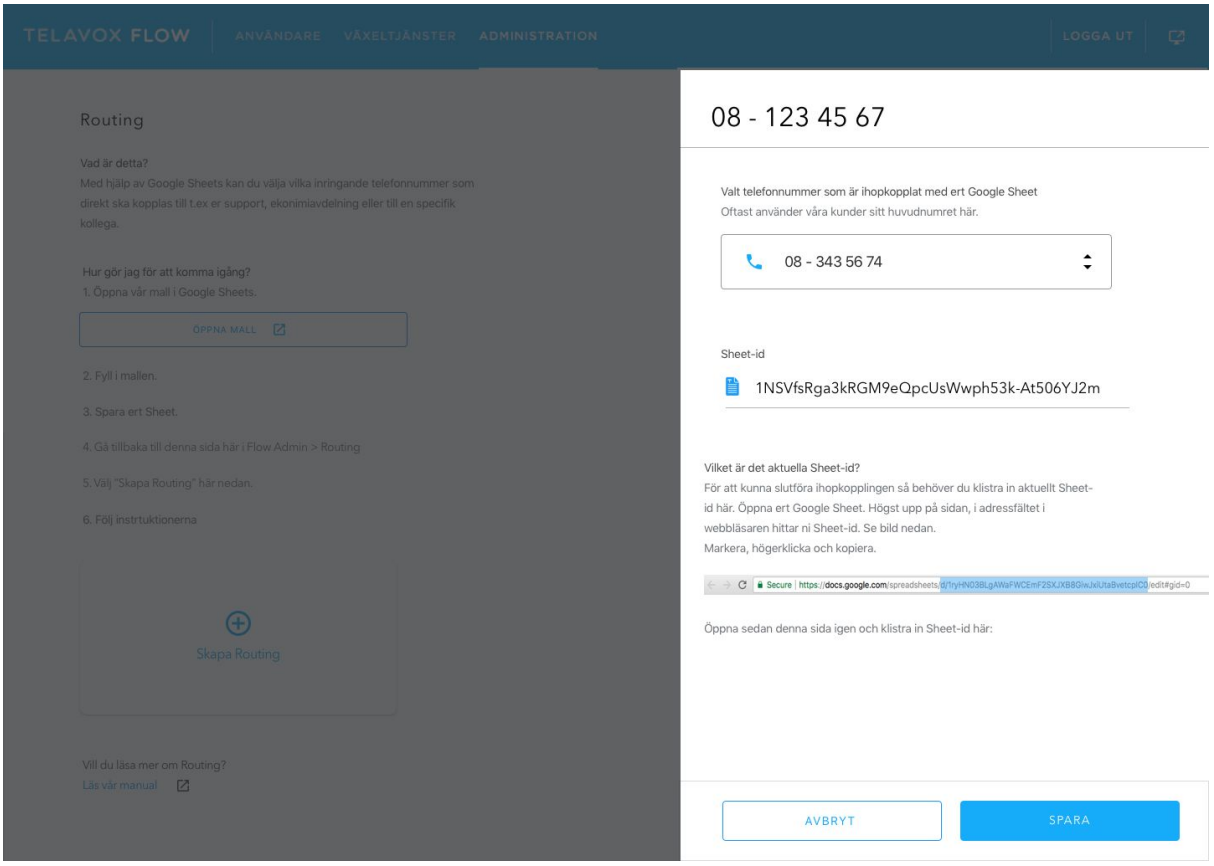
9.1.2. Skapa anslutning, komplett ifyllt



9.1.3. Redigera befintlig anslutning, ingen data ändrad



9.1.4. Redigera befintlig anslutning, data korrekt ändrad.



9.2. Källkod, Google Apps Script

```
function doGet(e) {
  var callerId = e.parameter.callerId;
  var sheetId = e.parameter.sheetId;

  try{
    var result = searchSheet(sheetId, callerId);
    return ContentService.createTextOutput(JSON.stringify(buildResponse(result))).setMimeType(ContentService.MimeType.JSON);
  } catch(err) {
    var errorResponse = {
      "status": "error",
      "message": "Invalid sheetId"
    };
  }
  return ContentService.createTextOutput(JSON.stringify(errorResponse)).setMimeType(ContentService.MimeType.JSON);

}

function searchSheet(sheetId, value){
  var connectionSpreadSheet = SpreadsheetApp.openById(sheetId);
  var sheet = connectionSpreadSheet.getSheets()[0];
  var dataRange = sheet.getDataRange();
  var lastRow = dataRange.getLastRow();
  var searchRange = sheet.getRange(1, 1, lastRow, 1);
  var rangeValues = searchRange.getValues();
  var foundValue = "none";
  var counter = 0;
  while(foundValue == "none" && counter < lastRow){
    if(rangeValues[counter] == value){
      foundValue = dataRange.getCell(counter+1, 2).getValue();
    }
    counter++;
  }
  return foundValue;
}

function buildResponse(result){
  var response, status, number;

  if(result != "none"){
    status = "success";
    number = result;
  } else {
    status = "fail";
    number = "";
  }

  response = {
    "status": status,
    "number": number
  };

  return response;
}

```

9.3. API-Specifikation: Webbtjänsten

API Specification

Version	Date	Author	Description
1.0	2018-04-24	peter.hesslow@telavox.com	Initial draft

Search sheet

Searches through a Google Sheet specified by sheetId for *digits_pressed*.

Request

Method	URL
`GET`	`https://script.google.com/macros/s/AKfycbwylJ6bxfq_SgiwcDh14SZ4ewMJtzG` `7L4FTJkoeNPp79Q6lUQ/exec?sheetId=<sheetId>&callerId=<digits_pressed>`

Type	Params	Values
URL_PARAM	*<sheetId>*	string
URL_PARAM	*<digits_pressed>*	string

`sheetId`
Id of the sheet that you want to search through.

`digits_pressed`
Id of the caller you want to search the sheet for. Could for example be phone number or customer number.

Response

All responses to successful requests are in JSON format

Status	Response

200	**An array containing status of the request and a message/result**
	
	Response is one of the following:
	
	{
	"status": “success”
	"number": *result*
	}
	
	{
	"status": “fail”
	"number": “”
	}
	
	{
	"status": “error”
	"message": “invalid sheetId”*
	}

*Indicates either that the sheetId itself is invalid/missing or that access to the sheet is not granted.

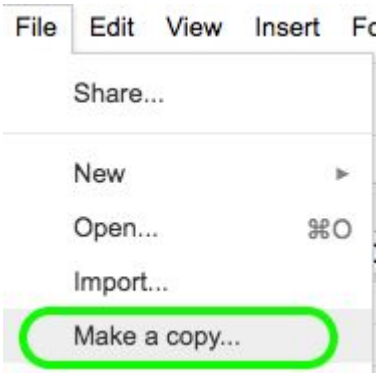
9.4. Användarmanual

Innan du fortsätter behöver du följande:

- Ett Google-konto. För att registrera dig för detta går du in på www.drive.google.com

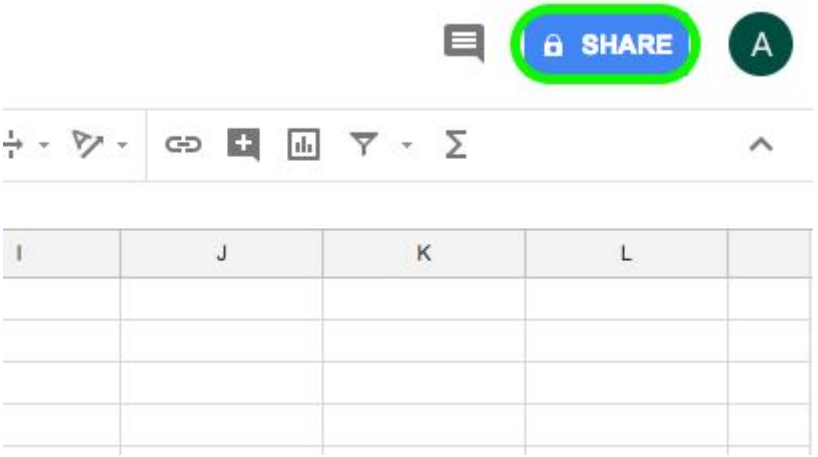
1. Det finns en mall som ni kan kopiera direkt till ert Google-konto. Klicka här för att komma till mallen.

2. När mallen öppnats klicka på “File” och “Make copy...”.

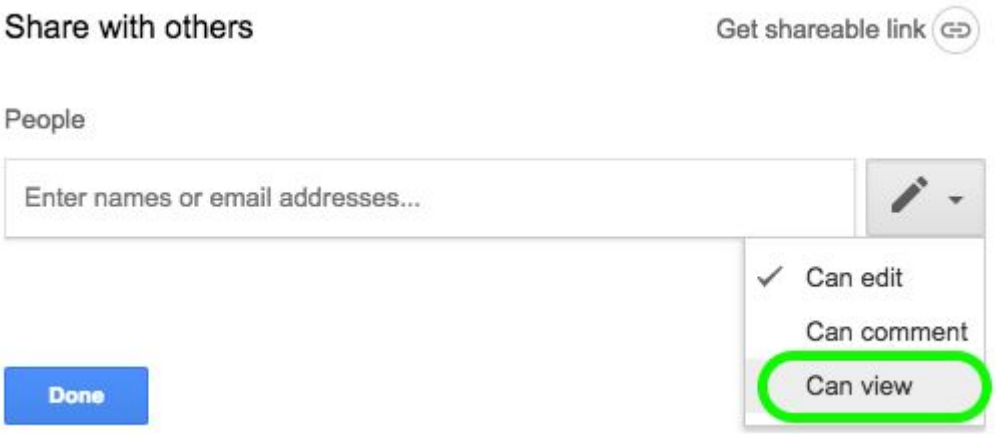


3. När kopian är skapad loggar ni in på www.drive.google.com och öppnar mallen som ligger på er drive.

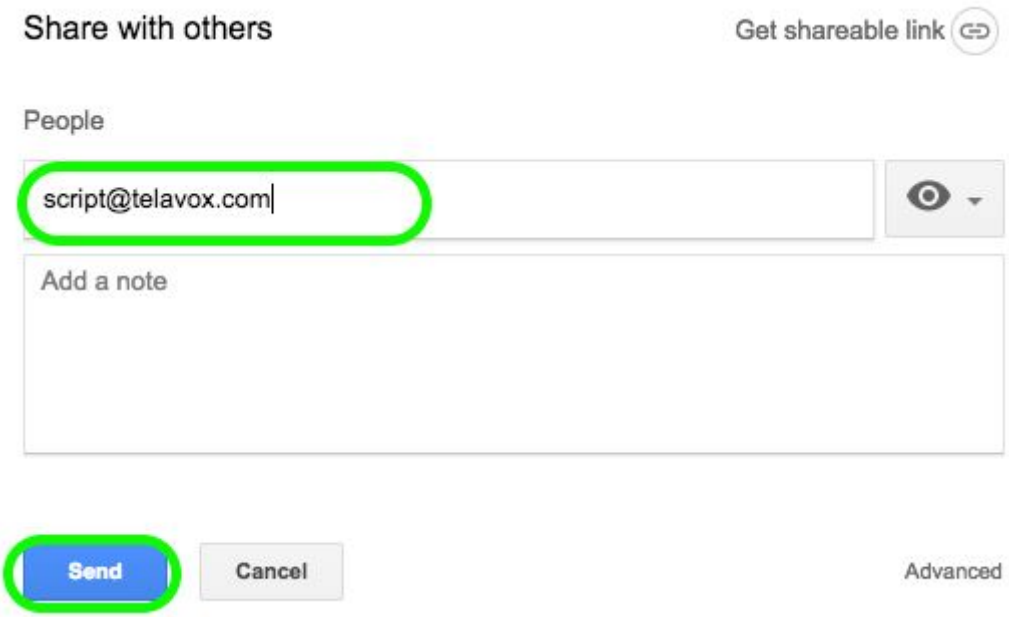
4. Klicka på “Share” upp i det högra hörnet.



5. Klicka på pennan i rutan som dyker upp och välj “Can view”.



6. Skriv sedan in “script@telavox.com” utan “ i rutan för “Enter names or email addresses...”. Klicka sedan på “Send”.



7. Logga sedan in på Flow Admin och välj “Routing”. Klicka därefter på “Skapa Routing” och följ instruktionerna.

9.5. Funktionstest av webbtjänsten

Test	Förväntat utfall	Resultat/kommentar
Anrop med ogiltigt *sheetId*	{ "status": “error” "message": “invalid sheetId” }	Utfall var som förväntat.
Anrop med *sheetId* som ej är delat med webbtjänsten	{ "status": “error” "message": “invalid sheetId” }	Utfall var som förväntat.
Anrop med *identifikationsnummer* som ej finns i aktuellt *sheet*	{ "status": “fail” "number": “” }	Utfall var som förväntat.
Anrop med *identifikationsnummer* som finns i aktuellt *sheet*	{ "status": “success” "number": *result* }	Utfall var som förväntat.
Anrop med *identifikationsnummer* som finns duplicerat i aktuellt *sheet*	{ "status": “success” "number": *result* }	Utfall var som förväntat, det vill säga att det första av de duplicerade värdena returnerades.