UPPSALA
UNIVERSITET

Självständigt arbete i informationsteknologi
29 oktober 2023

Tjänst för skapande och ifyllande av juridiska avtal i ett interaktivt dialogsystem

Johan Boström
Daniel Jansson
Martin Larsson
Erik Rimskog

Civilingenjörsprogrammet i informationsteknologi

Master Programme in Computer and Information Engineering

UPPSALA
UNIVERSITET

Institutionen för informationsteknologi

Besöksadress:
ITC, Polacksbacken
Lägerhyddsvägen 2

Postadress:
Box 337
751 05 Uppsala

Hemsida:
http:/www.it.uu.se

Abstract

Tjänst för skapande och ifyllande av juridiska avtal i ett interaktivt dialogsystem

Johan Boström
Daniel Jansson
Martin Larsson
Erik Rimskog

Everyone will, sooner or later, have to sign a legally binding agreement. Two examples of such agreements are purchase agreements for a car and a consultant agreement for when someone is recruited as a consultant. Many of these agreements follow a standard template and rarely change.

It is perceived as cumbersome and costly for individuals to contact a lawyer for the assistance of standardised agreements and that legal terminology can be perceived as alien and frightening. Our thesis is that the handling of these standardised agreements can be streamlined and thus improve the work flow of a legal firm.

Our solution to this problem is a system that allows for a costumer to communicate with a chatbot. This chatbot can identify and fill in the agreement that is best suited for the users situation. The agreements available to the system are created by a lawyer through an administrative tool. Using this tool the lawyer can create new templates for agreements and also review the agreements that has been filled in by users.
The project resulted in a chat interface for the user and an administrative tool in which an administrator can create and manage templates for contracts. Based on short messages from the user, the system is capable of choosing a contract among those created in administrative tool and recommend it to the user.

Extern handledare: Jimmy Eriksson, Sisyfos AB
Handledare: Björn Victor och Mats Daniels
Examinator: Björn Victor

Sammanfattning

En privatperson kommer förr eller senare behöva teckna ett juridisk bindande avtal. Två exempel på sådana avtal är köpeavtal för en bil eller ett konsultavtal när någon anställts som konsult. Många av dessa avtal följer en standardiserad mall och ändras sällan. Det upplevs som omständligt och kostsamt för privatpersoner att kontakta en jurist för assistans med standardiserade avtal samt att juridisk terminologi kan upplevas som främmande och skrämmande. Vår tes är att hanteringen av dessa standardiserade avtal kan effektiviseras och därmed förbättra en juristfirmas arbetsflöde.

Vår lösning på problemet är ett system som ger en kund möjlighet att föra en dialog med en chattbott som identifierar och hjälper kunden att fylla i det avtal den behöver. De avtal som finns tillgängliga har en jurist tidigare har skapat med hjälp av vår administratörssida. På denna sida kan juristen skapa nya avtalsmallar samt se tidigare ifyllda avtal.

Projektet resulterade i ett chattgränssnitt för användaren och ett administratörsverktyg där en administratör kan skapa och hantera avtalsmallar. Systemet är kapabelt att rekommendera avtal baserat på korta meddelanden från användaren. De avtal som kan rekommenderas är de som skapats i administratörsverktyget.

Innehåll

1 Introduktion

I Sverige lever åtta av tio sambos utan samboavtal, vilket kan leda till dispyter vid separation, enligt en studie av Verahill [34]. Exempelvis, om två ogifta sambos har gemensamma barn och den ena i samboskapet går bort kan den andra tvingas lösa ut sina barns andelar från bostaden, då ogift sambo inte kan ärva av den andre utan avtal [36]. Vidare uppger Verahill att det totalt i Sverige står 1 042 miljarder kronor på spel vid separationer (sammanlagt för alla samboskap utan avtal). Som anledning till varför samboavtal inte tecknas anger många att de har en okunskap kring ämnet eller att det upplevs som invecklat. Alltså är okunskap och en benägenhet att spara energi anledningar till att många inte tecknar vissa avtal, trots att det skulle ge individen rättsligt skydd.

Detta projekt ämnade att utveckla ett system som agerar som en medlare mellan den som söker hjälp och den som erbjuder det. I många fall har kunden en viss uppfattning om vad den vill uppnå och vad dess behov är och vänder sig till en jurist för rådgivning. Men detta är för juristen mycket tidskrävande och begränsar hur många juristen kan ge råd till. Istället kan juristen använda detta system för att skapa mallar för de avtal som de kan tillhandahålla och sedan kan ett stort antal kunder få hjälp av systemet att fylla i dessa.

Vår lösning för att möjliggöra för jurister att kunna erbjuda mer hjälp bestod av flera delar. Den första delen var ett chattgränssnitt på en webbsida där en användare kan skriva och ställa frågor på svenska. Meddelanden skickas sedan till en server i systemet som använder utomstående verktyg för att tolka meddelandet och avgöra vilken avsikt användaren hade med det för att sedan kunna välja ett passande avtal. Om användaren exempelvis nämner ”varit anställd” och ”konsult” kan servern avgöra att ett konsultavtal passar avsikten. Sedan kommer frågor ställas för att få in den information som krävs för att fylla i avtalet. Systemet är kapabelt att tolka användarens avsikter mycket väl, och kan med bara en kort mening som indata avgöra vilket avtal som passar användaren bäst.

Vilka avtal som kan väljas bestäms av den sista delen av systemet, administratörsverktyget. Denna innefattar ett grafiskt verktyg där en administratör skapar avtalsmallar. Här anges vilka avsikter som passar vilka avtal, som att konsultavtal passar när användaren anger “varit anställd” och “konsult”. Administratören kan även lägga till de frågor som behöver ställas för att få den information som krävs för att fylla i avtalet.

Den datarepresentation som användes för avtalsmallar var tvungen att passa för två ändamål. Den skulle användas för att representera mallar för administratören, samt till analysen av användarens avsikter och behov. Resultatet blev att den passade båda ändamålen.

2 Bakgrund

För någon utan juridisk kunskap kan det vara oklart vad för avtal som behövs och hur de bör vara utformade i den givna situationen. Ett exempel kan vara när någon ska flytta ihop med sin partner. För den oinsatta kan det vara svårt att veta hur ett samboavtal ska vara utformat och vad det bör innehålla, eller i vilka fall det är just ett samboavtal som bör tecknas. Vidare kan bristen av insikt i juridisk terminologi försvåra processen ytterligare.

2.1 Effektivisering av arbeten

Människan har alltid strävat mot att effektivisera sitt arbete. Ett exempel på en väletablerad marknad som har blivit effektiviserad är jordbruket. År 1850 var 60% av USA:s arbetsförda befolkning aktiva inom jordbrukssektorn [20]. 120 år senare, år 1970 var det mindre än 5% aktiva inom jordbrukssektorn. Minskningen var inte en följd av att människan konsumerade mindre livsmedel 120 år senare. Istället hade arbetet inom jordbruk effektiviserats, nya metoder och tillvägagångssätt hade börjat användas.

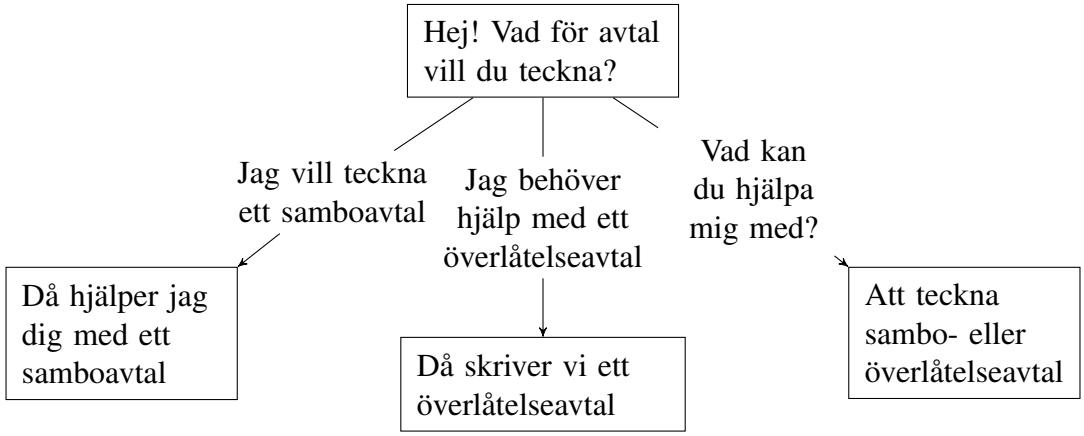
Det är inte bara jordbruket som kan effektiviseras, utan även andra arbetsområden. Ett sådant arbetsområde är juridik. Ett exempel inom effektivisering av juridiskt arbete, är en studie som visar att flera delar av en advokats arbete skulle, helt eller delvist, kunna automatiseras [23]. Mellan 13% och 23% av en advokats totala arbete kan automatiseras bort enligt studien. Automatiserbarheten i en arbetsuppgift avgjordes genom att undersöka mängden struktur och repetition i arbetsuppgiften, samt hur förutsägbara och hanterbara olika händelser kopplat till den arbetsuppgiften är. Desto mer struktur i en arbetsuppgift, desto mer automatiserbar ansågs den.

När ett avtal ska tecknas är det mycket repetitivt arbete som behöver göras av den som erbjuder juridisk rådgivning. Det kan vara sådant som att hjälpa någon att fylla i ett avtal, ta fram vilka avtal som kan vara nödvändiga att teckna i den givna situationen eller att tolka juridiska termer. Dessa repetitiva arbeten kan automatiseras och därmed effektivisera juristers arbete.

2.2 Chattbottar

En chattbott är ett system som simulerar mänsklig kommunikation mellan människa och maskin via tal eller skrift [21]. En av de första vetenskapligt erkända chattbottarna är ELIZA som uppfanns 1966 av Joseph Weizenbaum [38]. ELIZA använder sig

av språkteknologi (eng. natural language processing), när ELIZA skapades var begreppet inte etablerat och kallades för natural language conversation (Mer om detta under rubrik 2.3) [35]. De förekommer fem stycken huvudregler som ELIZA använder sig av [35], några av dessa lever kvar än idag. Kärnkoncept hos dessa regler är att identifiera nyckelord, avsikter och förprogrammerade svar [38].



Figur 1 Exempel av dialogträd. Kvadraterna motsvarar meddelande från systemet och meddelande i mitten på dialogpilarna motsvarar svarsalternativ.

Chattbottar kan använda sig av dialogträd. Dialogträd används för att sätta ramar och regler på hur en användare styrs genom en konversation. I figur 1 ges ett exempel på hur ett dialogträd skulle kunna vara utformat. I det här trädet är både svarsalternativ och svar fördefinierade, det går alltså inte att skriva in svar själv. Det här är en föråldrad och begränsad version av ett dialogträd.

Chattbottar är uppbyggda enligt två olika metoder: AI-baserade och regelbaserade. En AI-baserad chattbott identifierar språk, kontext, avsikt och därefter reagerar med ett svar. En regelbaserad chattbott agerar utifrån en serie av fördefinierade regler [27]. En regelbaserad chattbott är uppbyggd på ett regelsystem av villkorssatser där den traverserar framåt i ett dialogträd om ett förbestämt svarsalternativ uppfylls [28]. Exempel på detta är chattbotten Anna och programmeringspråket Artificial Intelligence Markup Language (AIML) [31, 28].

2.3 Språkteknologi

Språkteknologi är ett samlingsnamn för det tvärvetenskapliga forskningsområdet mellan språk- och datavetenskap. Syftet med språkteknologi är att förenkla kommunikation

mellan en dator och en människa. Ett exempel på användningsområden är automatiskt översättning mellan olika språk. Språkteknologi är det svenska samlingsnamnet för vad som i engelskan heter Natural Language Processing (NLP) och kommer refereras till som NLP i resterande delar av rapporten [25].

NLP anses vara en komponent inom paraplybegreppet artificiell intelligens [19]. Som Cambria och White beskriver uppnår ett NLP-system oftast en form av artificiell intelligens genom maskininlärning [5], där till exempel statistisk NLP har varit en populär metod. En viktig faktor att belysa är att många NLP system ger ett sken av faktiskt intelligent beteende, medan maskiner inte förstår vad de gör. I en analysprocess av naturligt språk i text upplevs uppgiften snarare som en matchning mellan ord och mönster.

3 Problemformulering

Här beskrivs projektets problemformulering, vad vi vill uppnå, vad vi vill åstadkomma och varför vi utför projektet. Dessa beskrivs som syfte, mål och motivation. Vi beskriver även vad vi inte hanterat i projektet under avgränsningar.

De globala hållbarhetsmålen togs i beaktande under projektet. Det är en agenda som antagits av FN:s utvecklingsprogram (UNDP) för att bekämpa problem som fattigdom, minska orättvisor och lösa klimatkrisen. De är uppdelade i 17 kategorier, där varje kategori har ett antal delmål [33].

3.1 Syfte

Projektet har två syften, att förenkla för jurister att tillhandahålla avtal för kunder samt att förenkla för kunden att fylla i dessa avtal. Projektet ska resultera i ett system som ger tillgång till juridisk rådgivning till många fler än vad som är möjligt idag. Personer som har låg utbildningsnivå eller som är underpriviligerade på andra sätt får det enklare att förskaffa sig kunskap om vad de kan få för rättsligt skydd och vilka rättigheter de har.

Systemet kräver inte några juridiska förkunskaper utan all kommunikation sker i talspråk (på svenska) för att göra det tydligare och enklare för användaren att förstå. Användaren kan få svar och mer kunskap genom att beskriva sin situation på ett mer vardagligt språk.

Projektet bidrar då till följande globala hållbarhetsmål: 16.3 (Säkerställ tillgång till rättvisa) och 16.10 (Säkerställa allmän tillgång till information och skydda de grundläggande friheterna) [32]. Målet 16.3 bidrags till genom att de avtal och den rådgivning

som vårt system erbjuder gör att fler får tillgång till avtal som kan skydda dem och därigenom erbjuda rättvisa. Genom att systemet hjälper användare, som inte nödvändigtvis använder korrekta juridiska termer, att få information och fylla i avtal bidrar systemet även till målet 16.10.

3.2 Mål

Det första målet med projektet är att leverera ett chattgränssnitt i en webbläsare där en kund kan få juridisk rådgivning. När systemet tar emot ett meddelande svarar det med ett rekommenderat avtal att fylla i utifrån det användaren skrivit. Systemet hjälper kunden fylla i avtalet genom att ställa frågor relaterade till avtalet och sparar sedan svaren.

Det andra målet med projektet är att skapa en webbapplikation där administratörer kan skapa avtalsmallar. Dessa mallar utgör de avtal som systemet kan erbjuda kunderna att fylla i. Webbapplikationen kommer också visa alla ifyllda avtal från kunderna.

3.3 Motivation

Möjligheten att teckna avtal i vissa sammanhang är inte alltid uppenbar. Brist på kunskap inom juridik kan leda till att individer hamnar i utsatta situationer. Även om det är uppenbart i en given situation att ett avtal är nödvändigt kan det vara omständligt och kostsamt att ta fram och teckna det. Det är även tidskrävande för den jurist som erbjuder hjälpen, vilket leder till att det skapas en flaskhals för hur många juristen har möjlighet att hjälpa, vilket i sin tur leder till att färre kan få den hjälp de behöver.

Om dessa flaskhalsar kunde elimineras, eller förminskas, skulle troligtvis fler vara benägna att söka hjälp. Fler skulle då vara bättre skyddade och mindre utsatta och de skulle troligtvis ha en bättre förståelse för de avtal de skriver under. Kapaciteten för juristen att bistå med hjälp skulle även öka, vilket skulle innebära att de kan hjälpa fler alternativt få mer tid åt mer komplicerade arbetsuppgifter.

I dagsläget har det tagits steg för att lösa problemet kring digitalt automatiserad avtalsifyllning men ingen har hittills löst det fullständigt, mer om detta under rubrik 5. Även om underskrift av avtal har underlättats genom digitalisering på olika sätt måste kunden just nu fortfarande nå ut till en jurist för att få rådgivning.

Som beskrivs i en rapport av Konkurrensverket upplevs marknaden för juridiska tjänster som en av de mest problematiska för konsumenter. Marknaden brister i flera aspekter, bland annat transparens, tillit, och möjlighet att göra medvetna och aktiva val [1]. Rap-

porten lyfter även att elektronisk dokumenthantering är en funktionalitet som underlättar och effektiviserar sökning och bearbetning av information. Projektet hanterar just sådan funktionalitet, alltså kan kunderna genom vårt system få en bättre förståelse för marknaden. Därmed känner sig kunden tryggare och nyttjandet av juridiska dokument ökar.

3.4 Avgränsningar

En avgränsning som gjordes för systemet är att signering av avtal inte hanteras. Det hade krävt integration med en identifieringstjänst som till exempel Mobilt BankID. Vidare om systemet hanterade signering skulle högre krav ställas på automatisk att verifiera och kontrollera svaren från användaren. Designen av systemet just nu kräver att efter ett avtal fyllts i av en användare ska en jurist manuellt granska och godkänna avtalet. Efter att ett avtal passerat och klarat en granskning är det upp till juristbyrån att följa upp med en signering.

Systemet är anpassat till avtal som kan anses standardiserade, vilket innebär att de är generella avtal som är applicerbara på flera kunder. Att skräddarsy avtal för att passa till en specifik kund stöds inte av systemet.

Inga anpassningar har gjorts i designen av systemet för personer med funktionsvariationer. Sådana kan vara färgblindhet eller variation i ålder och kan göra det svårare att använda systemet. Det är inte heller möjligt att föra en dialog på ett annat språk än svenska.

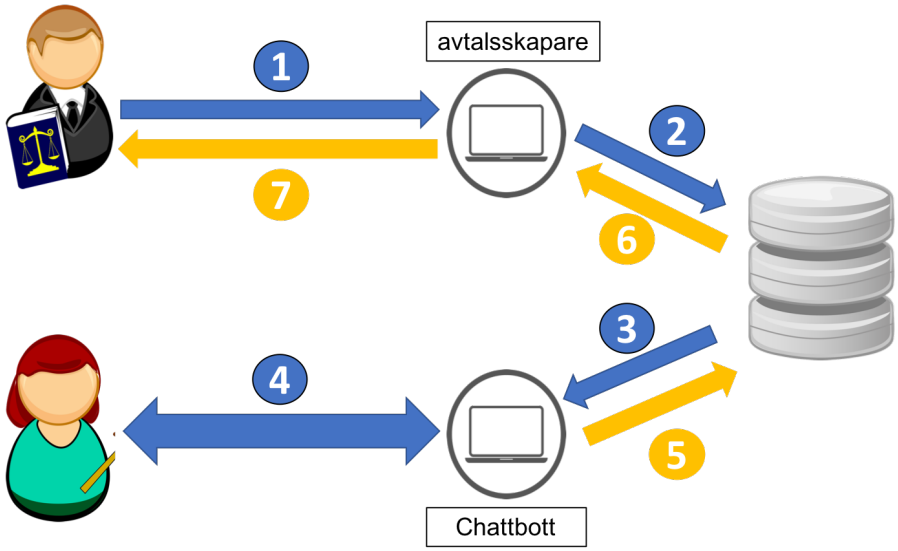
4 Systemöversikt

Här beskrivs hur systemet är tänkt att användas, vilka som ska använda det och hur dataflödet ser ut.

Systemet har två målgrupper: jurister och kunder. Juristerna är de med tillräcklig kunskap för att kunna skapa avtal och kunderna är de som vill teckna ett avtal eller få hjälp med att hitta rätt avtal att teckna, alltså privatpersoner.

Varje målgrupp har ett eget användargränssnitt till systemet för att kunna utföra den målgruppens roll. Juristerna ska använda gränssnittet som kallas för administratörsverktyget. I detta gränssnitt kan juristerna skapa mallar för olika avtal, som kunderna senare kan fylla i.

Den andra målgruppen, kunderna, har tillgång till ett annat gränssnitt som endast in-



Figur 2 Översiktligt diagram av relation mellan målgrupperna och systemet. (1) En jurist skapar via avtalsskaparen en avtalsmall, till exempel en mall för ett samboavtal. (2) Efter att mallen skapas sparas den i databasen. (3) Databasen gör den nya mallen tillgänglig för chattbotten. (4) Via chattdialog med användare kan en mall fyllas i. (5) När användare fyllt i ett helt avtal sparas dess svar i databasen. (6) När databasen erhållit de nya svaren för avtalet uppdateras en lista hos avtalsskaparen som innehåller alla ifyllda avtal. (7) En jurist kan nu granska avtalet och slutföra processen.

Bilder hämtade från *https://pixabay.com*

nehåller ett chattfönster. I detta fönster kan kunden föra en konversation med en chattbott för att först ta reda på vilket av juristens tidigare skapade avtal som ska fyllas i. Därefter kan kunden fylla i avtalet genom att svara på frågorna i avtalets avtalsmall.

5 Relaterat arbete

I projektet analyserades andra system som implementerat en chattbott och en avtalsskapare. Vi kommer diskutera och jämföra dels vilket djup de relaterade arbetens chattbott har i sin utformning och hur avtal hanteras. Som grund jämför vi med vårt system som består av komponenterna: en chattbott för kund och en avtalsskapare för administratör.

5.1 Liknande system

Automatisering av tidskrävande och standardiserade avtal är ingenting nytt, exempel på existerande system är Automio och DoNotPay [3, 10]. De nämnda systemen bistår, i olika utsträckning, användaren med behandling av juridiska dokument. Skillnader och detaljer rörande respektive systemen följer nedan.

5.1.1 Automio

Automio lanserades 2017 och syftar till att effektivisera repetitiva dokumentationstjänster. Automio är uppbyggt på en tjänst som skapar ett flöde av frågor kopplat till ett specifik avtal framtaget av en jurist. Vid användande av systemet lagras varje svar och när samtliga frågor är besvarade sammanställs ett avtal med alla svar inlagda [29].

Automios relation till detta projekt är dess behandling av automatisering och effektivisering av avtalshantering. Varje flöde av frågor är anknutet till ett specifikt avtal. Vidare är Automio utformat på ett sådant sätt att ifrån utgångsläget måste användaren explicit välja vad för avtal som ska fyllas i. Det krävs alltså att användaren på förhand är införstådd med vilket avtal denne vill teckna. Det här innebär en markant skillnad mellan Automio och vårt system. Vårt system genomför istället en kontinuerlig tolkning av kundens avsikt via chattdialogen för att därefter automatiskt ta fram korrekt avtal som kunden vill teckna. Detta beskrivs ytterligare under rubrik 6.1.2.

5.1.2 DoNotPay

DoNotPay lanserades 2017 [26]. Systemet har en enkel och väldigt specifik inriktning i dess grundutformning, den bestrider parkeringsböter. Med grundutformning menas dess originella funktionalitet som systemet hade 2017. 2018 påbörjades en stor och övergripande uppdatering av hela systemet för att möjliggöra för användaren att bestrida mer än bara parkeringsböter [14].

I grundutformingen, alltså bestridandet av parkeringsböter, hjälper systemet en användare att bestrida en parkeringsbot genom att en chattbott ställer ett antal frågor och registrerar svaren. Tjänsten kräver att användare initialt väljer typ av parkeringsbot och geografiskt område, liknande Automio. Systemet använder en chattbott vars beteende kan beskrivas som ett frågeformulär i form av ett chattgränssnitt.

Efter att DoNotPay erhållit initialvärden sker en dialog med chattbotten, som har ett antal förgenererade svar för de frågor som inte är kopplade till ett direkt bestridande.

Systemet registrerar de svar användaren anger, oavsett om de faktiskt innehåller den information som efterfrågas eller inte [18].

Ett exempel, om chattbotten frågar efter en adress och användaren svarar med ett klockslag kommer systemet att acceptera svaret. Likväl när systemet efterfrågar ID-numret på en parkeringsbott sker ingen kontroll att det är ett giltigt nummer.

DoNotPay har inte en dialog med användaren utan är ett mer utvecklat frågeformulär varifrån felaktig eller inkorrekt inmatning inte kan ändras utan processen måste börjas om. Vårt system fungera annorlunda jämfört med DoNotPay då vårt system frågar användaren via chattbotten vilket avtal som ska fyllas i, istället för att fylla i kryssrutor innan dialogen som DoNotPay gör. Vårt system kommer även att kolla om de inmatade svaren är på rätt format eller av rätt sort och kommer att fråga om frågan om de inte är det.

6 Metod

I det här avsnittet beskrivs, diskuteras och jämförs vilka metoder vi har använt. Sammanfattningsvist använde vi IBM Watson Assistant som tjänst för att tolka indata från användarna, React.js för att göra gränssnittet på webbsidorna, node.js för att driva servern och till sist MySQL som vår databas.

Efter en litteraturstudie där vi jämförde Watson med ett antal alternativ (mer om detta under rubrik 6.1.3) valde vi att använda Watson. Efter det initiala planeringsarbetet av projektet och en plan för de olika komponenterna som behövdes i projektet (se systemstruktur avsnitt 7) följde en designfas för chattgränssnitt och avtalsskaparen. När designfasen var avslutad och internt testad påbörjades en utvecklingsperiod med dagliga möten som hjälpte gruppen framåt och gjorde att vi kunde säkerställa att inget släpade efter.

6.1 Chattbottar

Under denna rubrik diskuteras två olika kategorier av chattbottar samt förklaringar av IBM Watson Assistant i detalj och kort om dess alternativ.

6.1.1 AI-chattbottar och regelbaserade chattbottar

I detta avsnitt kommer en chattbott som har NLP och AI-kapacitet att jämföras med en chattbott som är regelbaserad. För den NLP-baserade chattbotten kan svaret "Ja" tolkas som en avsikt att svara ja på en fråga, med möjlighet att även tolka synonymsvar som "absolut, yep, kör hårt, det godtar jag" som ett ja på grund av NLP:en. Medan ett regelbaserat system måste svaren vara explicit bestämda av programmeraren, om "yep" inte har programmerats som svar på frågan kommer systemet inte heller tolka det som ett ja. Det kan bli en skillnad om det regelbaserade systemet har en NLP. Med en NLP kommer det regelbaserade systemet ha en större felmarginal när den tolkar ett svar. Överlag är en NLP ett väldigt bra inslag i en chattbott då det kan bidra till känslan att användaren samtalar med en människa [4].

I projektet är det väsentligt att tolka kundens svar, utifrån perspektivet vad kunden har för avsikt snarare än att enbart reagera utifrån vad som skrivits. Som exempel, om systemet enbart låter en kund teckna samboavtal om ordet 'samboavtal' explicit skrivs i chattdialogen motverkas syftet med projektet. Tillåts systemet istället tolka texten 'jag vill flytta ihop med min respektive' som en avsikt att flytta ihop blir systemet inte bara effektivare utan också lättare att använda.

Som nämns i syftet under rubrik 3 kan juridiska termer ofta vara invecklade och svårförståeliga. En chattbott som kontinuerligt lär sig och utvecklas, vilket är möjligt med en AI uppbyggd NLP, och kan förstå vad kunden vill kan därmed blir effektivare, mer verklighetstrogen och kan bättre bistå en kund. Utifrån detta dras slutsatsen att vi vill använda en AI-baserad chattbott med NLP.

6.1.2 IBM Watson Assistant

Vi har valt att använda IBM Watson Assistant till detta projekt. Watson Assistant är en molntjänst, utvecklad av IBM [15]. Det är AI-baserad chattbott som kan lära sig att förstå och tolka bättre vad användarna skriver (se rubrik 2.3). Watsons förmåga att tolka meningar grundar sig i *avsikter* och *entiteter*.

- En **avsikt** är vad vi vill att Watson ska förstå. För att lära Watson innebörden av en sådan ges exempel på svar som en användare kan tänkas skriva. Exempelvis för att lära Watson hur den vet om en användare vill ha hjälp kan exempelsvaren vara: "jag behöver hjälp" och "jag kan inte själv".
- Ett **entitet** är ett objekt i en sats som en användare kan prata om. Till exempel kan Watson lära sig om bilmärken. Om en användare nämner ett bilmärke i kon-

Figur 3 Watson Assistant dialogträd

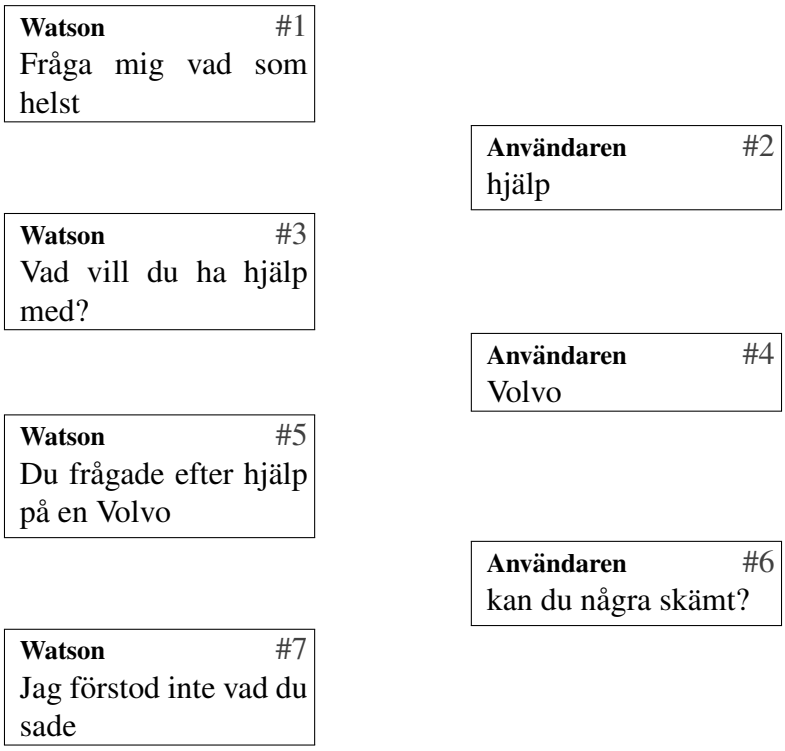
versationen kommer Watson att se det och extrahera ut och eventuellt spara det värdet.

Watson kommer sedan, med hjälp av dessa entiteter och avsikter, att bygga en maskininlärningsmodell som kan känna igen rätt entiteter och avsikter från andra liknande uttryck till de exempel som definierade dem.

För att få Watson att faktiskt ha en konversation behövs ett dialogträd. Ett exempel på en sådan visas i figur 3. Det är ett träd där varje nod kan ha flera barnnoder och varje nod innehåller vilken avsikt den ska svara emot och vilken respons den ska ge. Det finns speciella villkor som kan sättas på noder: *Welcome* som är den nod som körs en gång som initialt svar och *Everything else* som kör när ingenting annat passar.

En exempelkörning av trädet i figur 3 visas i figur 4. Där börjar konversationen i start med att Watson skriver ut det som står i welcome-noden (#1). Användaren skriver sedan "hjälp" (#2) varav Watson ser att det finns en nod som passar på avsikten för hjälp. Watson skriver ut svaret som står i den noden och väntar nu där (#3). Detta betyder att barnen till den noden kommer testas först till nästa svar som användaren skriver. Om inget där passar kommer Watson att gå tillbaka till start och kolla om någon nod passar där. Om inget passar där heller kommer inget svar att skrivas ut. I det här fallet finns en "Everything else" som alltid matchar, vilket betyder att någonting alltid kommer att skrivas ut.

Användaren fortsätter med att skriva bilmärket "Volvo" (#4), Watson ser att "volvo" är



Figur 4 Exempel på en konversation med figur 3.

ett bilmärke och skriver ut det som står i den matchande noden. Svaret på den noden är speciellt då den refererar mot den entitet som matchades. Detta betyder att Watson inkluderar svaret från användaren i #4 i sitt egna svar (#5).

Nu finns det inga fler barnnoder vilket betyder att Watson hoppar tillbaka till start och väntar på mer indata från användaren. Denna gång skriver användaren någonting som inte passar in i någon specifik nod (#6), vilket gör att fallback-noden matchas (#7).

Nu börjar allt om igen i start och konversationen kan fortsätta.

6.1.3 Alternativa chattbottsystem

På grund av det som diskuterades under rubrik 6.1.1 är det endast AI-chattbottar som är av intresse. Därav finns minst två intressanta alternativ till IBM Watson Assistant: wit.ai och Dialogflow.

Wit.ai Wit.ai är en AI-chattbott som är mer öppen än IBM Watson då den delar alla *avsikter* och *entiteter* publikt till alla [37]. Den gör detta för att lära sig från alla som använder tjänsten, alltså hjälper alla användare varandra. Den använder sig av *entiteter* och *avsikter* på precis samma sätt som IBM Watson. Wit.ai har dock inte något dialogträd, den analyserar bara text för att hitta *avsikter* och *entiteter*.

Dialogflow Dialogflow är en tjänst av Google som också är en AI-chattbott [9]. Precis som wit.ai och IBM Watson analyserar denna *avsikter* och *entiteter*. Dialogflow är som IBM Watson också en betaltjänst, men använder Agents istället för dialogträd. Dessa Agents är listor av *avsikter* och fungerar som dialogträd fast utan barnnoder.

Vi har valt IBM Watson framför dessa alternativ för att Watson har ett dialogträd. Eftersom att chattbotten i detta projekt kommer att ställa många frågor efter varandra är ett dialogträd att föredra då det enkelt kan skapas följdfrågor genom barnnoder (se exemplet i figur 4). Att använda dialogträd är ett passande sätt att strukturera upp chattbottens konversation, då dialogträdet som modell för en dialog stämmer väl överens med verkligheten.

6.2 Node.js

För att driva servern används Node.js, vilket är en asynkroniskt händelsestyrd exekveringsmiljö för Javascript [22]. Med detta menas att Node.js använder callback-funktioner istället för trådar och event-loopar[1]. En callback-funktion är en funktion som ska köras när en händelse sker. Exempel på händelser är när någon ansluter till node eller om en timer avfyras. I figur 5 visas ett exempel på en callback-funktion som körs varje gång ett HTTP-anrop tas emot.

Fördelen med att det är designat att bara ha callback-funktioner är att det blir enklare att hantera flera anslutningar samtidigt [22]. Istället för att göra en ny tråd för varje anslutning körs istället den callback som specificerats att köra på nya anslutningar. Det blir enklare för att den trådfria designen tar bort risken för deadlocks då behovet av lås försvinner [22, 11].

Den huvudsakliga anledningen till valet av Node.js är att programmeringsspråket den tolkar är javascript, vilket är samma språk som stöds och används i de flesta webbläsare

[1]Node.js egna interna event-loop är inte den som menas här, utan de som programmeraren måste göra själv

```
1 const server = http.createServer(function(req, res) {
2   res.statusCode = 200;
3   res.setHeader('Content-Type', 'text/plain');
4   res.end('Hej\n');
5 });
6 server.listen(3000, '127.0.0.1');
```

Figur 5 Ett exempel på en callback-funktion. Meddelandet "Hej" skickas tillbaka varje gång någon gör en HTTP-anrop till servern.

för att göra webbsidan interaktiv [13]. Fördelen med att ha samma språk både på klientsidan och serversidan är att bland annat datastrukturer inte behöver översättas för att kunna användas mellan två olika programmeringsspråk.

6.3 React.js

React är ett javascript-ramverk utvecklat av Facebook [24]. Detta ramverk är till för att bygga så kallade "single-page applications" vilket är webbsidor som laddar endast en HTML-fil och uppdaterar den dynamiskt [17]. Eftersom sidan uppdateras dynamiskt laddas inte sidan om då när vyn byts till en undersida, utan endast de relevanta bitarna laddas in.

En webbsida i React är uppbyggd av komponenter och varje komponent beskriver hur den själv ska renderas på webbsidan [24]. I figur 6 på rad 1 finner vi ett exempel på en komponent implementerat som en klass. Denna klass har en metod `render` som returnerar vad som ska visas. I det här fallet ska en `div` med innehållet av argumentet `name` visas i en `p`-tagg. Observera att en komponent endast kan bestå av en tagg. Den taggen kan innehålla hur många barntaggar den vill men render-metoden får bara returnera en tagg. För att rendera en komponent på webbsidan används funktionen `ReactDOM.render` (på rad 11), denna tar som argument vilken React-komponent som ska renderas och vart på HTML-sidan den ska renderas. Observera att det är här på rad 12 som name-argumentet till komponenten specificeras.

Att varje komponent beskriver hur den vill renderas istället för att göra det själv instruktion för instruktion kallas för deklarativ programmering [2]. Hur komponenten renderas och vilka som renderas ansvarar React för. Till exempel om `name` i figur 6 på rad 12 ändrades till `"Kalle"` skulle React jämföra den gamla HTML-strukturen med det nya för att komma fram till att endast `p`-taggen behöver ändras och låter `div`-taggen vara kvar som den är [24]. Den dynamiska uppdateringen är en av fördelarna och en av anledningarna till att vi valde React.

```
class HelloMessage extends React.Component {
  render() {
    return (
      <div>
        <p>Hello {this.props.name}</p>
      </div>
    );
  }
}

ReactDOM.render(
  <HelloMessage name="Taylor" />,
  document.getElementById('hello-example')
);
```

Figur 6 En enkel React-komponent

Valet att använda React föll på att det var ett av de populäraste ramverken vid rapportens skrivande och att det därför skulle vara bra att lära sig.

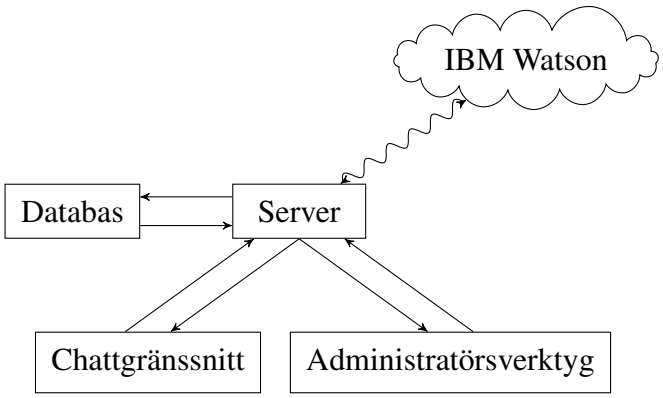
6.4 MySQL

MySQL är en relationsdatabashanterare där all data organiseras i tabeller. För att kommunicera med en MySQL-databasen används SQL (Structured Query Language) [7]. I SQL-tabeller måste alla element i varje kolumn (fält) vara av samma datatyp och antalet kolumner får inte ändras. Dessa två restriktioner gör att tabellerna inte är flexibla.

För att organisera SQL-tabeller kan man lägga till vissa fält som är speciella. En av dessa är **primary key** (PK), vilket är ett fält vars värden måste vara unika inom tabellen. Att hämta rätt rad ur tabellen underlättas om det finns ett värde som är garanterat att vara unikt för den raden. Ett annat speciellt fält som finns är **foreign key** (FK), vilket används för att koppla ihop tabeller. Att ett fält i en tabell är en FK mot ett fält i en annan tabell betyder att värdena i den första tabellen endast får bestå av värden som finns i den andra tabellen. Kopplingen uppnås genom att en tabell refererar till en PK i en annan tabell.

Eftersom vår data är strukturerad (inte flexibel) passar det bra att organisera den i tabeller, MySQL valdes av denna anledning.

7 Systemstruktur



Figur 7 Diagram över systemstrukturen

I figur 7 finner vi en teknisk överblick av systemet. Den består av fem moduler: ett chattgränssnitt, ett administratörsverktyg, en server, en databas, och en extern tjänst IBM Watson Assistant.

Chattgränssnitt är den modul där en kund söker juridisk rådgivning genom att skriva med chattbotten. Gränssnittet är en webbsida gjord med React.js och kommunicerar med servern via WebSockets och HTTP-anrop.

Modulen administratörsverktyget är den modul där juristen skapar avtalen som användaren kan fylla i. Denna modul är också gjord i React.js och kommunicerar med servern på samma sätt som användaren. Avtal som skapas här lagras i databasen tillsammans med svaren från ifyllda avtal.

Servern är utvecklad med Node.js och det är där den mesta logiken sker. Som går att utläsa ur figur 7 sammankopplar servern samtliga komponenter och koordinerar dem. För att använda IBM:s-molntjänster gör servern API-anrop till IBM med hjälp av HTTP-anrop, det som skickas i dessa anrop är JSON-objekt.

Ett typiskt dataflöde när en kund skriver med chattbotten är följande:

1. Kunden skriver ett meddelande i gränssnittet och meddelandet skickas till servern.
2. Servern skickar meddelandet till IBM Watson som tolkar det och skickar tillbaka ett svar.
3. Om det är ett avtal som fylls i avgör servern om detta avtal är ifyllt eller inte.

a) **Om ifyllt:** sparar servern de ifyllda svaren i databasen.

b) **Om inte:** fortsätter konversationen.

4. Servern skickar sedan svaret till användaren.

5. Börja om i punkt 1.

7.1 Designval

Här beskrivs de större val vi gjorde i designen av systemet och varför vi valt att utforma delar av systemet på ett visst sätt.

7.1.1 Behovet av en server

Mycket av logiken från servermodulen går att flytta till chattgränssnittet och nästan eliminera behovet av servern helt (webbsidan måste kommas åt på något sätt fortfarande). En nackdel med att låta chattgränssnittet ta över all logik är att det inte är lika tydligt vad för roll varje modul har i systemet. Chattgränssnittets roll är att representera ett gränssnitt för kunden och servermodulens roll är att hantera den bakomliggande logiken och kommunikationen mellan de olika modulerna. Genom att modularisera systemet kan ändringar göras på en modul utan att påverka, eller ha kunskap av, hur en annan modul fungerar. Kodseperation är gynnsamt då kod som inte har modifierats har mindre risk att sluta fungera än kod som har modifierats.

En annan nackdel med att flytta serverlogiken till chattgränssnittet är att information som bör hållas hemligt från kunden blir tillgängligt i klartext i en javascript-fil. Informationen kan vara sådant som API-nycklar och databaslösenord. Genom att separera servern och användaren är det möjligt att kontrollera vad för information som finns på chattgränssnittet. Samtidigt som den känsliga informationen som databaslösenord kan sparas på servern, där ingen utomstående har åtkomst till det. Om användaren behöver göra något med den känsliga informationen får användaren skicka ett anrop till servern för att få den att hantera det. Till exempel kan användaren skicka anropet “hämta alla avtal” till servern varefter den använder det hemliga databaslösenordet för att hämta alla avtal och skickar dem till chattgränssnittet.

7.1.2 Val av Dialogträd

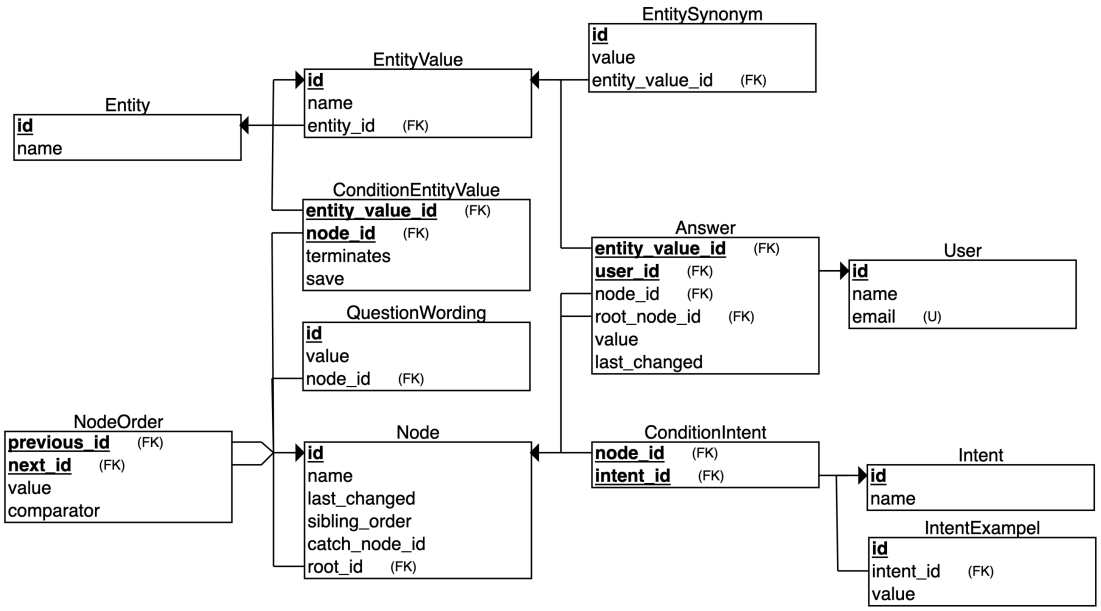
Antingen skulle dialogträdet hos Watson kunna nyttjas eller göra ett eget. Att använda det hos Watson betyder att servern blir en mellanhand som bara vidarebefordrar meddelandena mellan klienten och Watson. När ett nytt avtal skapas från avtalsskaparen måste servern utöver att lagra den i databasen också bygga upp dialogträdet på Watson. Detta kräver fler API-anrop, men i utbyte får systemet tillgång till ett redan färdigt dialogträd med mycket funktionalitet.

Att göra dialogträdet själv gör att det går att använda avtalen lagrade i databasen direkt utan att behöva skicka iväg dem till Watson. Men detta skulle kräva mer arbete vilket är anledningen till att vi bestämde oss för att inte göra dialogträdet själva.

8 Datastruktur för avtalsmallar

En datastruktur för avtalsmallar krävdes för att representera mallarna som dialogträd. Dialogträden användes sedan för att hjälpa chattbotten att leda en konversation med en användare. Konversationens mål är att hitta ett lämpligt avtal och ta fram den nödvändiga informationen som behövs för att kunna fylla i avtalet.

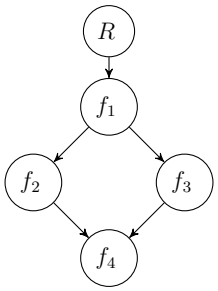
8.1 Implementation



Figur 8 Relationsdiagram över SQL-databasen. Rutorna representerar databasens tabeller och varje rad i dessa är namnet på en kolumn. De understrykna fältnamnen representerar att de är en **primary key** och pilarna representerar att de är **foreign keys** (se förklaringar under rubrik 6.4).

Kontrakt och frågor representeras som noder i dialogträd, och sparas i tabellen Node i figur 8. Frågor har föräldrar för att representera ordningen i trädet, om fråga B har A som förälder kommer A före B i trädet. Ordningen mellan frågorna sparas i tabellen NodeOrder där *previous_id* är en referens till föräldern och *next_id* är en referens till barnet. Skillnaden mellan kontrakt och frågor är att kontrakt inte har någon förälder, de är rotnoderna i alla träd. Rotnoderna används för att gruppera frågor som hör till det kontraktet.

För att systemet ska gå in i en specifik nod måste nodens villkor uppfyllas. Villkoret består av avsikter och entiteter och bestämmer vad för indata som förväntas av användaren. Exempelvis kan en nod fråga efter användarens adress och som villkor kräva entiteten adress. Då kommer systemet inte gå vidare till nästa fråga om användaren inte angett en adress, utan ställer om frågan igen tills dess att villkoret uppfyllts. Avsikter och entiteter sparas i tabellerna Intent respektive Entity och kopplas till noder genom tabellerna ConditionIntent och ConditionEntityValue.



Figur 9 Visualisering av nodernas ordning. Rotnoden R har ingen förälder och representerar därför ett kontrakt. Frågorna f_2 och f_3 har båda första frågan f_1 som förälder och representerar därför en förgrening. Frågar f_4 återsluter trädet genom att ha både f_2 och f_3 som föräldrar.

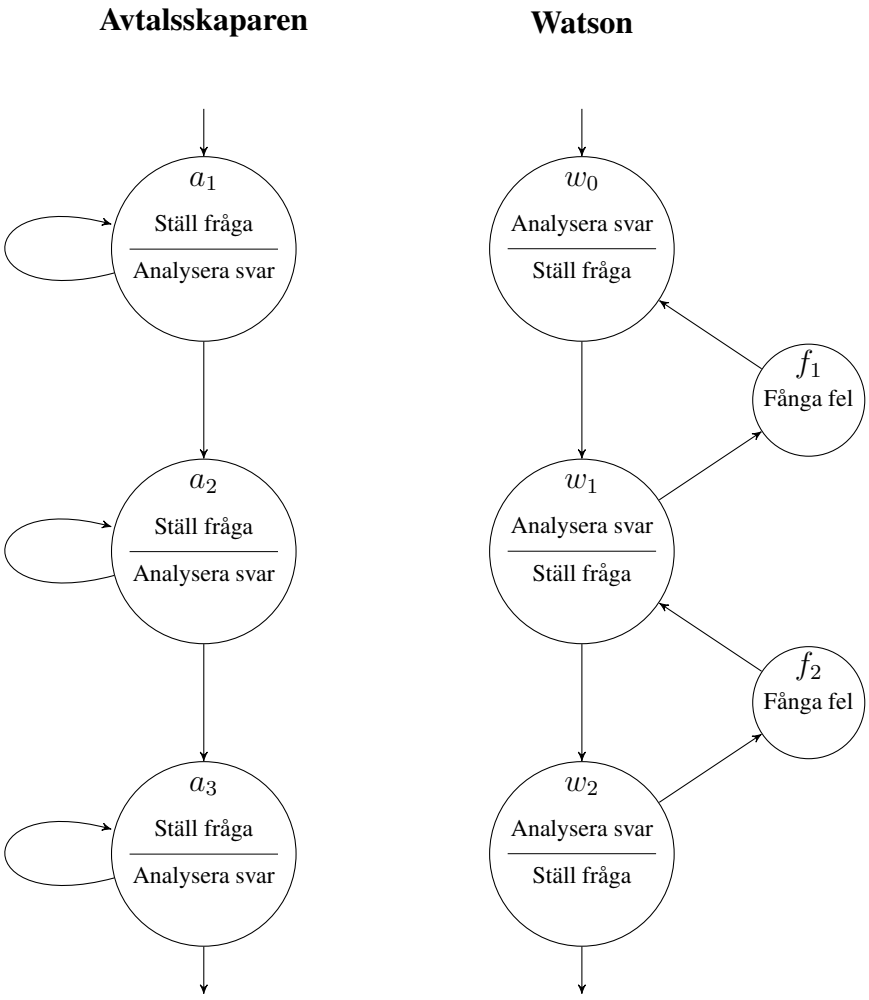
Frågor kan ha samma förälder, vilket innebär en förgrening i trädet (illustreras i figur 9). För att systemet då ska kunna avgöra vilken av frågorna den ska fortsätta till ges frågorna olika villkor. För att sedan återsluta förgreningen kan flera frågor leda vidare till samma fråga så att det bara finns en väg för systemet att gå.

8.2 Integration med IBM Watson

Eftersom avtalsmallen i avtalsskaparen endast består av frågor stannar systemet kvar i samma nod om svaret inte uppfyller nodens villkor. När systemet i figur 10 går in i nod a_1 ställs den fråga som tillhör noden. Exempelvis kan denna fråga vara “Hur många barn har du?”, och villkoret i noden är en siffra. Om användaren svarar med en färg, eller något annat som inte uppfyller villkoret kommer systemet stanna kvar i nod a_1. När användaren svarar med en siffra går systemet vidare till a_2.

Watson är mer generell än avtalsskaparen och vi var därför tvungna att ta fram lösningar för att den skulle passa vårt system. Som exempel kan noden w_1 representera samma fråga och villkor som a_1 (“Hur många barn har du?”, och ett nummer). Skillnaden mot avtalsskaparen är att frågan som är kopplad till w_1 sparas i noden w_0. Innan Watson går vidare till w_1 analyseras svaret som användaren ger till w_0 för att se om villkoret angivet i w_1 är uppfyllt.

Om ett svar inte uppfyller villkoret går Watson till fångnoden f_1, för att därefter gå tillbaka till w_0 och ställa om frågan. Fångnoder är inte inbyggt i Watson utan är vanliga noder som skapas och anpassas av oss. Fångnoder har inga villkor som måste uppfyllas och behöver inte heller ge ett svar till w_0, utan går direkt till att ställa frågan. Alla frågor måste ha en fångnod för att ge användaren möjlighet att ange fel svar men ändå vara



Figur 10 Skillnad i nodstruktur, mellan Watson och avtalsskaparen. Avtalsskaparen ställer en fråga och väntar på svar medan Watson läser ett svar och därefter ställer en fråga.

kvar i samma nod i dialogträdet. Utan en fångnod skulle Watson gå ur dialogträdet om användarens svar inte matchar ett villkor. För tabellen Node i databasen har varje fråga en referens till sin fångnod och denna sparas i kolumnen *catch_node_id* för att kunna redigera och radera denna på Watson.

9 Avtalsskapare

För administratörsverktyget så utgör avtalsskaparen den mest centrala delen. I det här avsnittet kommer vi beskriva dels den designprocess som föregick framtagandet av avtalsskaparen och hur avtalsskaparens dialogträd är visualiserad.

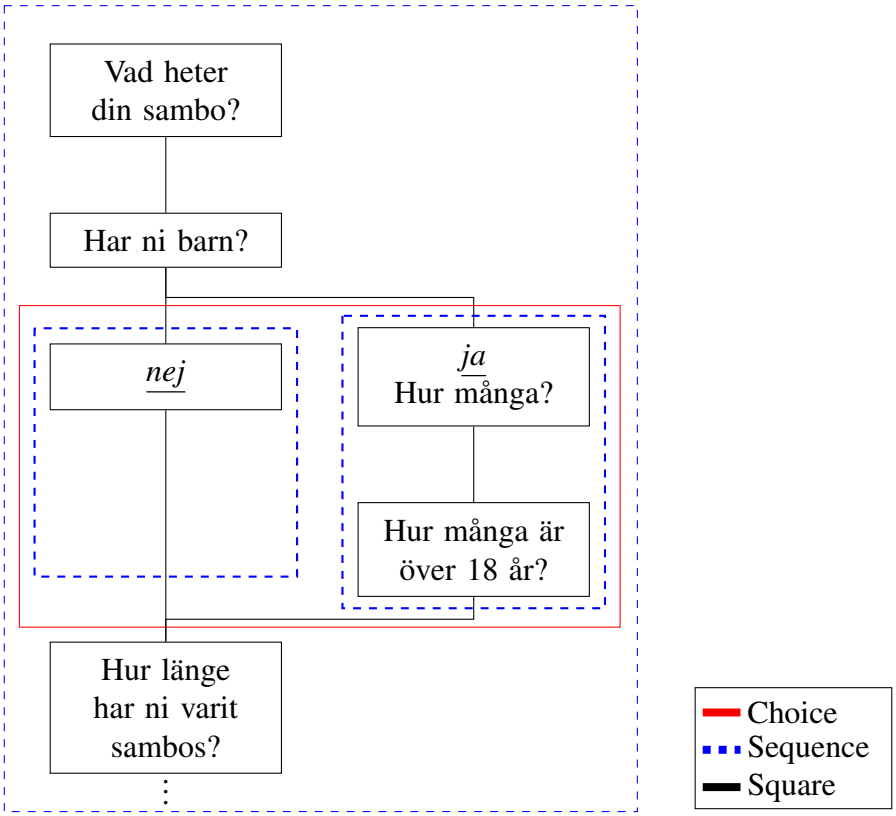
9.1 Framtagande av avtalsskaparen

För att uppnå syftet med att förenkla för jurister att tillhandahålla avtal måste mallarna för dessa avtal skapas. En utmaning med administratörsverktyget var avtalsskaparen, alltså modulen där mallar för avtal skapas. Utmaningen i avtalsskapren grundar sig i att avtalsmallarna måste byggas på ett lagringsbart sätt och som vi enkelt kan översätta till IBM Watson. Enkelt i den bemärkelsen att genom avtalsskaparen abstraherar vi bort behovet att den som nyttjar avtalsskaparen behöver skriva dialoger i IBM Watson. Mer om hur vi integrerar avtalsskaparen med IBM Watson går att läsa under rubrik 8.2.

Innan någon form av programmering för administratörsidan påbörjades var det viktigt att ta fram en ändamålsenlig design. Detta uppnådde vi genom en pappersprototyp som specificerade hur strukturen skulle se ut med tillhörande funktionalitet. I designen kom vi fram till ett antal viktiga koncept: En grafisk representation för relationen mellan frågorna inom en mall. Att de nödvändiga fälten för varje enskild fråga är: En frågetext, ett svar av en bestämd typ (till exempel ja,nej eller ett siffertal). Om det är en följdfråga krävdes även att ett villkor på föregående svar bestämdes.

Ett exempel, en fråga kan vara “har du barn” det förväntade svaret specificeras till “Ja” eller “Nej”. Därifrån kan följdfrågor bestämmas relaterade till svaret “Ja” respektive “Nej”.

Fullständiga pappersprototypen finns tillgänglig i appendix B



Figur 11 Exempel på en avtalsmall. De röda och blå rutorna syns inte på webbsidan utan är hjälprutor för att visa den underliggande datastrukturen.

9.2 Visualisering av avtalsmallar

Avtalsmallen är här en komponent i avtalsskaparen för att bestämma vilken ordning ett avtals frågor ska ställas i. Hur trädet kan se ut på webbsidan visas i figur 11. Varje nod innehåller vilken fråga som ska ställas och vilket sorts svar som accepteras, till exempel om bara ja eller nej accepteras.

En konversation med trädet i figur 11 börjar med första frågan “Vad heter din sambo?”. Efter att användaren har gett ett giltigt svar går trädet till nästa fråga som är “Har ni barn?”. Om svaret inte är giltigt ställs frågan om tills ett giltigt svar har givits. Om användaren svarar “nej” går trädet vidare till “Hur länge har ni varit sambos?”, men om användaren istället svarar “ja” kommer följdfrågorna “Hur många?” och “Hur många är över 18?” att ställas innan “Hur länge har ni varit sambos?” ställs.

Datastrukturen vi kom fram till för att representera en avtalsmall består av 3 delar:

- **Square** ■ är en ruta med en fråga som ska ställas och kriterium på giltiga svar till frågan.
- **Sequence** ■ är en vertikal lista som består av flera Square och/eller Choice. Frågorna i en Sequence kommer att ställas i ordning.
- **Choice** ■ är en horisontell lista som består av minst två Sequence. Varje Sequence i Choice representerar olika vägar att fortsätta på beroende på svaret på föregående fråga.

Trädet i figur 11 byggs upp genom att göra en Sequence på följande vis:

```
[Square(vad...), Square(hur...), Choice, Square(en...), ...]
```

där den första Square är frågan "Vad heter din sambo?", den andra "Har ni barn?". Det tredje elementet Choice består av en egen lista, nämligen:

```
[Sequence, Sequence]
```

där den högra Sequence består av:

```
[Square(Hur många?), Square(Hur många är...)]
```

och den vänstra är uppbyggd på ett liknande vis.

Datastrukturen är designad för att enkelt rendera till HTML. Det blir enklare för att varje del i datastrukturen har en egen korresponderande React-komponent. Varje Square kan omvandlas till en `<div>` med specificerad bredd och höjd och alla Sequence och Choice omvandlas också till en varsin `<div>`. De två senare `<div>`-arna behöver endast rada upp sina barnelement åt ett visst håll och behöver inte modifiera storlek eller annat på barnen. Varje ruta i figur 11 är alltså en React-komponent som alla är en `<div>`.

10 Krav och utvärderingsmetoder

Här förklaras vilka krav vi hade på systemet och hur vi testade att vi hade uppnått dem. Målen med projektet var att leverera en chattbott som underlättar avtalsfyllning samt en webbapplikation för jurister att skapa avtalsmallar i (se under rubrik 3).

10.1 Krav

Det första kravet är att chattbotten ska rekommendera korrekt avtal åtminstone hälften av fallen av vårt test som finns i appendix A. Testfallen är framtagna så att hälften rätt är acceptabelt, då fallen har en varierande språknivå och de svåraste fallen är mycket ovanliga.

Det andra kravet är att administratörsverktyget ska vara användbar. För att kravet ska vara uppfyllt ska alla användare, med varierande teknisk kunskap, klara av att skapa avtalsmallar. Detta ska klaras av inom en rimlig tidsram, det ska inte ta användaren mer än fem minuter att skapa en mall.

10.2 Utvärderingsmetoder

Kravet på chattbotten testades genom att vi skrev manuellt i chattgränssnittet för att få rätt rekommendationer bland tre tillgängliga avtal: sambo-, kompanjon- och konsultavtal. Om vi i vår kommunikation med chattbotten hade en avsikt som matchade något av avtalen och chattbotten föreslog detta avtal blev det ett lyckat testfall. Vi gjorde detta åtta gånger för varje avtal och räknade antalet lyckade testfall.

För att utvärdera det andra kravet avseende ett användarvänligt gränssnitt fick en testgrupp bestående av fyra personer (familj och bekanta) göra ett användartest. Den tekniska kunskapen hos personerna i testgruppen varierade, vilket är positivt då vi kan jämföra prestationen och förståelsen för systemet med hänsyn till deras kunskapsnivå. Användartestet utfördes genom att vi bad testpersonerna göra ett avtal medan vi observerade vad de gjorde. Testet gick ut på att de skulle återskapa avtalet i figur 11 från grunden genom att lägga till frågor, och efter det bad vi testpersonen ta bort en fråga. Genom att utföra denna sekvens av instruktioner behöver användarna interagera med all funktionalitet på webbsidan. Under testet iakttog vi:

- Vart de navigerade för att skapa en avtalsmall.
- Hur de försökte skapa en avtalsmall.
- Hur de försökte skapa frågor i avtalsmallen.
- Hur de försökte skapa förgreningar i avtalsmallen.
- Om de förstod hur en fråga togs bort.

11 Utvärderingsresultat

Resultatet från testet av rekommendationer av avtal var att systemet uppfyllde det första kravet vi ställde (se krav under avsnitt 10). Systemet rekommenderade rätt avtal för åtminstone hälften av indatan som tagits fram, där varje indata hade en passande avsikt för ett visst avtal (se appendix A för alla testfall). Sammanlagt för varje exempelavtal var resultatet:

- **samboavtal:** 50%
- **kompanjonsavtal:** 62.5%
- **konsultavtal:** 50%

Den indata som användes för att testa rekommendationer utformades för att innehålla rätt avsikt, men med olika nivåer av klarhet. Vi förväntade oss att systemet skulle rekommendera samboavtal till indatan ”Jag vill flytta in med min flickvän” men inte ”Har tjackat ny lya med kärringen”. Vi anser därför att testet är lyckat då systemet uppfyller kravet på åtminstone 50% rätt.

Utvärderingen av användarvänligheten av administratörsverktyget gick dåligt, se avsnitt 10 för utvärderingsmetoden. Samtliga testpersoner förstod var de skulle navigera i verktyget för att skapa avtalsmallar och hur en avtalsmallen skapades. Det var dock inte uppenbart hur avtalsmallen namngavs eller att den kunde namnges. Att skapa frågor i mallen klarade alla testpersoner, däremot var hälften av testpersonerna förvirrade över var frågan skulle skrivas in, men efter letande hittade testpersonerna var detta kunde göras. Vidare var det oklart hur förgreningar skapades. En testperson uppgav att den fick chansa sig fram men chansade rätt på första försöket och en annan testperson skapade först fler följdfrågor istället för en förgrening, men lyckades också till slut. Samtliga testpersoner lyckades utan några förhinder med att ta bort en fråga.

Sammanfattningsvis anser vi att användbarhetstestens resultat var under förväntan och inte godtagbart. Ytterligare steg vidtogs och en ny design togs fram, dock hann vi inte genomföra ett nytt användbarhetstest. I den nya designen förbättrades hur avtalsnamn specificerades och var frågan skulle skrivas in genom att förbättra det grafiska gränssnittet och tydligare förklarande rubriker till dem olika fälten.

12 Resultat och diskussion

Här tar vi upp resultaten av projektet och diskuterar det. Resultatet innefattar webbsidan med avtalsskapare och chattgränssnittet, systemets server med tillhörande databas och slutligen en etikdiskussion. Överlag är vi nöjda med resultat då våra krav som ställdes på projektet uppfylldes.

12.1 Webbsidan

Webbsidans grundstruktur är klar men det är mycket funktionalitet som saknas än. Undersidorna kommer att tas upp i mer detalj här.

Avtalsskaparen färdigställdes inte. Administratörer kan skapa nya avtalsmallar och redigera existerande avtalsmallar. Mallarna renderas korrekt med noder på rätt ställen och linjer mellan dem. Frågor går att lägga till och frågor kan tas bort. Det går att lägga till flera följdfrågor till varje fråga och det går att specificera vilket villkor som krävs för att en viss följdfråga ska ställas (till exempel ja eller nej på föregående fråga).

En tidig målsättning för avtalsskaparen var att göra den så enkel som möjligt för att en jurist på egen hand ska kunna bygga en avtalsmall utom bakomliggande kunskap eller förståelse av IBM Watson. Skapande av avsikter och entiteter (som berördes i rubrik 6.1.2) var dock mer krävande än så och för att skapa dessa krävs djupare kompetens i Watson. I rådande prototyp kan en administratör inte skapa egna avsikter och entiteter utan de är i nuläget fördefinerade.

Chatten blev helt klar. Det går att skriva i textfältet för att skicka meddelanden till vår tjänst.

12.2 Serversidan

I systemets databas representeras avtalsmallar på ett sätt som passar dialogträden i både IBM Watson Assistant och klientsidans avtalsskapare. Att kunna representera mallarna i båda dessa format (se skillnader under 8.2) var målet med databasens design, och därför anser vi att resultatet av databasen är bra.

I början av projektet gick en stor del av arbetet åt att komma fram med designen för databasen, eftersom det inte fanns direkt stöd i Watson för delar av den funktionalitet vi hade gett avtalsskaparen. Mycket tid lades alltså på att ta fram lösningar för kringgå Watsons begränsningar (se 8.2). Trots dessa begränsningar gjordes bara en kompromiss,

att frågor måste besvaras i kronologisk ordning. Om användaren anger mer information än vad frågan ber om sparas det inte, även om en följdfråga ber om den informationen.

Framtagningen av designen för databasen blev en flaskhals då utvecklingen av serverns API krävde en databas. Dock, tack vare den tiden vi lade på att göra en välgenomtänkt databasdesign fick vi en god uppfattning för hur resten av systemet på serversidan skulle struktureras. När designen av databasen var klar kunde modulerna till serverns API skrivas med relativ enkelhet. Vi hade inga prestandakrav på servern utan vi bedömmer resultatet av servern utifrån huruvida all behövlig funktionalitet är implementerad. All behövlig funktionalitet för avtalskapande existerar och därför anses resultatet av servern bra.

12.3 Etik

Vårt system kommunicerar via textdialog med användaren, vilket leder till att individer med vissa funktionsnedsättningar blir begränsade i användningen av vårt system. Exempelvis kan blinda individer inte använda vårt system alls, då de inte kan läsa vad systemet svarar. Detta kan lösas genom att lägga till stöd för text-till-tal och tal-till-text för att kunna prata med systemet istället [30]. Ett annat exempel på exkluderade av individer är de som har dyslexi då systemet kan ha svårt att förstå vad individen försöker yttra om det är många eller grova stavfel i texten.

Vårt system bidrar till att den digitala klyftan växer. Den digitala klyftan är ett begrepp som beskriver hur lösningar blir mer och mer tekniska medan vissa i samhället inte hänger med i utvecklingen [12]. Systemet bidrar till klyftan genom att vi har gjort en teknisk lösning av avtalsifyllning som annars i många fall har skett i fysisk form med en jurist. Om juristbyråer helt skulle övergå till vårt system skulle individer som är tekniskbegränsade få problem att få hjälp med den här sortens problem.

Vårt system samlar enbart in den information som en jurist explicit uppger krävs för att avtalet ska kunna tecknas. Informationen som lagras är enbart tänkt att användas i avtalen och inte till något annat. Det är alltså ingenting som säljs vidare eller lämnas ut till någon ytterligare part. Givet detta har inga särskilda åtgärder vidtagits för att garantera informationsäkerhet genom till exempel inloggningsystem eller kryptering av data. Eftersom vårt system med stor sannolikhet kommer behandla känslig data som personnummer och bankkontonummer bör denna data skyddas. Detta för att ingjuta tillit hos användaren att datan inte kommer hamna i fel händer.

12.4 Resultat gentemot relaterade system

Skillnaden gentemot de liknande systemen (se rubrik 5.1) och vårt system kommer främst beskrivas utifrån användareperspektivet av chattgränssnittet. Anledningen till att vi inte berör de liknande systemens motsvarighet till avtalsskaparen är för att de flesta är betaltjänster som vi inte har tillgång till inom ramen för det här projektet. Genom vårt system kringgår vi problematiken med att en användare inte är insatt i vad det korrekta namnet är för ett specifikt avtal är. Givet ett exempel i talspråk “att sälja en sak” är “överlåtelse av egendom” i juridiska termer, och då går vi inte in på att det även en är skillnad på lös och fast egendom. Sammanfattningvis finns det många termer som inte är uppenbara för någon som inte är påläst [6]. Att vårt system erbjuder möjligheten att tolka en avsikt kontra de liknande system där en användare explicit måste, med knappval, välja det avtal den vill teckna är en markant skillnad, då de i vårt system får hjälp av hitta rätt avtal. För en användare som är insatt och vet vad den vill, till exempel teckna ett samboavtal, blir det också enklare. I de liknande systemen måste rätt avtal hittas och klickas på medan i vårt system så kan användaren direkt skriva ‘Jag vill teckna ett samboavtal”.

En begränsning för systemet är Watsons kapacitet. Om en användare använder för otydliga, grammatiskt inkorrekta eller felstavade ord kommer det leda till att Watson inte kan hjälpa till och systemet sannolikt stannar i ett. Likväl såsom våra test visar förstår inte Watson i nuläget inte alltid grammatiskt korrekta meningar, vilket är ett tillkortakommande med grund i att Watson behöver tränas upp mer för att få en bredare kunskap och förståelse.

13 Slutsatser

Projektet resulterade i ett chattgränssnitt och ett administratörsverktyg. Chattgränssnittet kommunicerar med chattbotten för att identifiera vad för avtal kunden behöver och hjälper sedan till att fylla i det. För att göra avtal tillgängliga för chattbotten skapas mallar för avtalen i adminstratörsverktyget. Detta sker helt utan att administratören har kunskap om hur det bakomliggande systemet, IBM Watson, fungerar.

Vi har alltså förenklat för jurister att skapa specifika avtalsmallar i IBM Watson. Att skapa den här förenklingen har styrt hela projektet. En datarepresentation av en avtalsmall togs fram som både är intuitiv för användaren och kompatibel med IBM Watson. Skulle en vidareutveckling på detta projekt ske är den underliggande designidén en bra grund att utgå i från.

14 Framtida arbeten

Dialogträden med kontrakten och tillhörande frågor är fullt fungerande i nuvarande form. Frågorna ställs i turordning, när en fråga besvaras på ett sätt som uppfyller villkoret går systemet vidare till nästa fråga och upprepar processen . Det finns dock en svaghet i implementation, frågor måste besvaras i kronologisk ordning.

Exempelvis kan användaren ange ”Hej! Jag heter Arne Björk och jag ska flytta in med min flickvän på Gatuvägen 3 den 6 juni” till chattbotten. Systemet kommer då identifiera att användarens avsikt är att flytta in med en partner, och rekommendera att användaren tecknar ett samboavtal.

Däremot kommer resten av informationen, namnet och datumet, inte sparas, även om det är ett relevant svar på en senare fråga. För att komma runt detta bör svarets samtliga avsikter och entiteter analyseras för att kunna avgöra om det är information som kan användas senare.

Skulle systemet lanseras publikt skulle det även vara tvunget att följa de regler som anges av dataskyddsförordningen (GDPR) [8]. Därför är ett annat framtida arbete att implementera inloggningsfunktioner i systemet. Det skulle möjliggöra underskrift av avtal eftersom användaren då kan styrka sin identitet. Dessutom, I den nuvarande versionen av systemet kan vem som helst skapa, redigera och ta bort samtliga avtalsmallar och tillhörande frågor. För att undvika detta bör systemet säkerställa att endast användare med rätt behörighet kan göra anrop till serverns API. En lösning till detta problem är att implementera JSON Web Tokens som ger varje användare en unik nyckel när de loggar in som skickas med varje anrop för att verifiera behörighet [16].

Funktionalitet för att redigera avtalsmallar, som i att flytta på noder med “drag and drop”, är något som inte blev implementerat i avtalsskparen. Om denna funktionalitet implementerades skulle redigering av existerande avtal underlättas.

Datastrukturen för avtalsmallen på klientsidan som beskrivs under rubrik 9.2 kan bli effektivare. För hitta en nod (Choice, Sequence eller Square) i trädet, för att till exempel redigera, måste alla noder sökas igenom då de inte är organiserade på något sätt. Eftersom avtalen, och därmed träden, är relativt små märks inte detta av tidsmässigt. Men det är fortfarande onödigt många beräkningar som körs och det kan potentiellt bli för långsamt i framtiden.

I en slutgiltig version av systemet så kommer vi även vilja att både skapade och ifyllda avtalsmallar ska kunna granskas av administratör respektive användare. De svar som användaren har angett ska sparas och presenteras på ett tydligt sätt, i olika format beroende på situationen. När administratören granskar dokument kan det ske direkt på

webbsidan medan användaren kan få en .pdf-fil för att spara sina svar.

Referenser

[1] J. Adamsson and B. Genfors, “Juridiska tjänster och konsumenter,” *Konkurrensverkets rapportserie*, Juni 2017. [Online]. Tillgänglig: http://www.konkurrensverket.se/globalassets/publikationer/rapporter/rapport_2017-10.pdf

[2] K. Apt. Logic programming. Hämtad 2019-05-22. [Online]. Tillgänglig: https://ir.cwi.nl/pub/13580/13580A.pdf

[3] Automio. Automio’s homepage. Hämtad 2019-04-17. [Online]. Tillgänglig: https://autom.io

[4] P. Boutin, “Does a bot need natural language processing?” *chatbots magazine*, April 2017, hämtad 2019-04-18. [Online]. Tillgänglig: https://chatbotsmagazine.com/does-a-bot-need-natural-language-processing-c2f76ab7ef11

[5] E. Cambria and B. White, “Jumping nlp curves: A review of natural language processing research?” *IEEE ComputatIonal Intelligence magazine*, Maj 2014. [Online]. Tillgänglig: http://www.krchowdhary.com/ai/ai14/lects/nlp-research-com-intlg-ieee.pdf

[6] Consector. Consector förklarar. Hämtad 2019-05-21. [Online]. Tillgänglig: https://www.consector.se/ordlista/overlatelseavtal/

[7] O. Corporation. What is mysql? Hämtad 2019-05-03. [Online]. Tillgänglig: https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html

[8] Datainspektionen. Dataskyddsförordningen (gdpr). Hämtad 2019-05-21. [Online]. Tillgänglig: https://www.datainspektionen.se/lagar--regler/dataskyddsforordningen/

[9] Dialogflow. Dialogflow homepage. Hämtad 2019-04-19. [Online]. Tillgänglig: https://dialogflow.com/

[10] DoNotPay. Donotpay’s homepage. Hämtad 2019-04-17. [Online]. Tillgänglig: https://donotpay.com

[11] J. E. G. COFFMAN, M. J. ELPHICK, and A. SHOSHANI, “System deadlock,” *Computer surveys*, Juni 1971, hämtad 2019-05-22. [Online]. Tillgänglig: https://carlstrom.com/stanford/quals/mirror/www-db.stanford.edu/~manku/quals/zpapers/71-deadlocks.pdf.gz

[12] O. Findahl. Svenskarna och internet 2003. Hämtad 2019-05-17. [Online]. Tillgänglig: https://internetstiftelsen.se/docs/SOI2003.pdf

[13] D. Flanagan, *JavaScript: The Definitive Guide*, 5th ed. 1005, Gravenstein Highway North, Sebastopol: O'Reilly Media, 2006.

[14] C. Haskins, "New app lets you 'sue anyone by pressing a button'," *Motherboard*, Oktober 2018, hämtad 2019-04-08. [Online]. Tillgänglig: https://motherboard.vice.com/en_us/article/bj43y8/donotpay-app-lets-you-sue-anyone-by-pressing-a-button

[15] IBM. Watson Assistant. Hämtad 2019-04-07. [Online]. Tillgänglig: https://www.ibm.com/cloud/watson-assistant/

[16] Internet Engineering Task Force. 7 Web Token (JWT). Hämtad 2019-05-13. [Online]. Tillgänglig: https://tools.ietf.org/html/rfc7519

[17] M. A. Jadhav, B. R. Sawant, and A. Deshmukh. Single page application using angularjs. Hämtad 2019-05-22. [Online]. Tillgänglig: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.736.4771&rep=rep1&type=pdf

[18] C. Kohler, "Meet the college student who built a chatbot to fight parking tickets," *The Penny Hoarder*, Juni 2018, hämtad 2019-04-08. [Online]. Tillgänglig: https://www.thepennyhoarder.com/life/cars-transportation/how-to-fight-a-parking-ticket-with-donotpay/

[19] K. Lindström. Fem olika sätt att prata om ai. Hämtad 2019-05-06. [Online]. Tillgänglig: https://computersweden.idg.se/2.2683/1.652787/ai-fem-satt

[20] J. Manyika, S. Lund, M. Chui, J. Bughin, J. Woetzel, P. Batra, R. Ko, and S. Sanghvi. Jobs lost, jobs gained: Workforce transitions in a time of automation. Hämtad 2019-04-18. [Online]. Tillgänglig: https://www.mckinsey.com/~/media/mckinsey/featured%20insights/Future%20of%20Organizations/What%20the%20future%20of%20work%20will%20mean%20for%20jobs%20skills%20and%20wages/MGI-Jobs-Lost-Jobs-Gained-Report-December-6-2017.ashx

[21] M. L. Mauldin. Chatterbots,tinymuds,and the turing test entering the loebner prize competition. Hämtad 2019-04-28. [Online]. Tillgänglig: http://new.aaai.org/Papers/AAAI/1994/AAAI94-003.pdf

[22] Nodejs. Node.js homepage. Hämtad 2019-04-07. [Online]. Tillgänglig: https://nodejs.org/

[23] P. Nussey. How much of what lawyers do can be automated? a look at new research. Hämtad 2019-04-18. [Online]. Tillgänglig: https://www.linkedin.com/pulse/how-much-what-lawyers-do-can-automated-look-new-research-peter-nussey?trk=portfolio_article-card_title

[24] React. React homepage. Hämtad 2019-04-28. [Online]. Tillgänglig: https://reactjs.org/

[25] redaktionen. Välkommen till språkteknologi.se! Hämtad 2019-05-06. [Online]. Tillgänglig: http://sprakteknologi.se

[26] J. Robinson, "The 18-year-old student who has saved british drivers £2million in just four months: Teenager sets up free appeals website in his bedroom which has overturned 30,000 parking fines already," *Daily Mail*, December 2015, hämtad 2019-04-08. [Online]. Tillgänglig: https://www.dailymail.co.uk/news/article-3341645/The-18-year-old-student-saved-British-drivers-2MILLION-just-four-months-Teenager-sets-free-appeals-website-bedroom-overturned-30-000-parking-fines-already.html

[27] N. Shah, "Which is best for you: Rule-based bots or ai bots?" *chatbots magazine*, Februari 2017, hämtad 2019-04-18. [Online]. Tillgänglig: https://chatbotsmagazine.com/which-is-best-for-you-rule-based-bots-or-ai-bots-298b9106c81d

[28] K. Shridhar. Rule based bots vs ai bots. Hämtad 2019-04-18. [Online]. Tillgänglig: https://medium.com/botsupply/rule-based-bots-vs-ai-bots-b60cdb786ffa

[29] L. society. Automio. Hämtad 2019-04-08. [Online]. Tillgänglig: https://www.lawsociety.org.nz/practice-resources/the-business-of-law/technology-recent-developments/automio

[30] Synskadades Riksförbund. It, datorer och synskadade. Hämtad 2019-05-13. [Online]. Tillgänglig: http://www.srf.nu/globalassets/tillganglig-text/it.pdf

[31] L. Tiptree. Anna an aiml chatbot. Hämtad 2019-05-13. [Online]. Tillgänglig: https://pandorabots.com/pandora/talk?botid=fd5c287f1e357db6

[32] UNDP. Mål 16: Fredliga och inkluderande samhällen. Hämtad 2019-04-05. [Online]. Tillgänglig: https://www.globalamalen.se/om-globala-malen/mal-16-fredliga-och-inkluderande-samhallen/

[33] UNDP. Om oss – undp och globala målen. Hämtad 2019-06-09. [Online]. Tillgänglig: https://www.globalamalen.se/om-undp/

[34] Verahill. Larmrapport: Miljarder regleras inte av samboavtal – stor risk för extra tråkig separation. Hämtad 2019-04-17. [Online]. Tillgänglig: https://www.verahill.se/om-verahill/pressmeddelanden/#/pressreleases/larmrapport-miljarder-regleras-inte-av-samboavtal-stor-risk-foer-extra-traakig-separation-2835831

[35] J. Weizenbaum, "Computational linguistics," *Communications of the ACM*, Januari 1966, hämtad 2019-04-18. [Online]. Tillgänglig: https://web.stanford.edu/class/linguist238/p36-weizenabaum.pdf

[36] M. Wernström. Vad händer om min sambo dör? Hämtad 2019-05-21. [Online]. Tillgänglig: https://www.inter.se/vad-hander-om-min-sambo-dor/

[37] wit.ai. wit.ai homepage. Hämtad 2019-04-19. [Online]. Tillgänglig: https://wit.ai/

[38] Wizu. A visual history of chatbots. Hämtad 2019-04-18. [Online]. Tillgänglig: https://chatbotsmagazine.com/a-visual-history-of-chatbots-8bf3b31dbfb2

A Rekommendationstest

Följande termer matades in manuellt i chattbotten för att se om det matchar mot rätt avtal. De tre avtalen som fanns var samboavtal, kompanjonsavtal och konsultavtal.

Resultatet från försök att matcha mot samboavtal:

Indata	**Matchade**
Jag vill flytta in med min flickvän	Ja
Min pojkvän och jag ska flytta ihop	Ja
Jag och min partner ska köpa lägenhet	Ja
Nästa vecka ska jag flytta in hos min respektive	Nej
Min kille kommer flytta in här nästa vecka	Ja
Tjejen flyttar in	Nej
Gumman och jag ska flytta ihop	Nej
Har tjackat ny lya med kärringen	Nej

Resultatet från försök att matcha mot kompanjonsavtal:

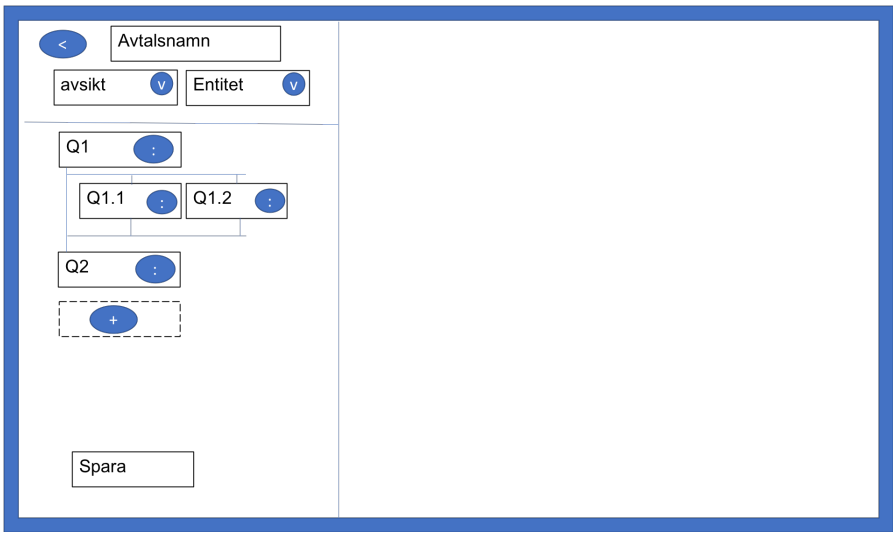
Indata	**Matchade**
Jag ska starta nytt företag med en kollega	Ja
Jag ska starta nytt med en partner	Ja
Jag och Gurra ska starta ny firma	Nej
Har startat firma med en gammal kollega	Ja
Öppnade ny affär med en kompis	Ja
Har slått ihop min firma med en ny partners firma	Ja
Kommer starta nytt med min gymnasieklasskompis	Nej
Har startat bilmek med kumpan	Nej

Resultatet från försök att matcha mot konsultavtal:

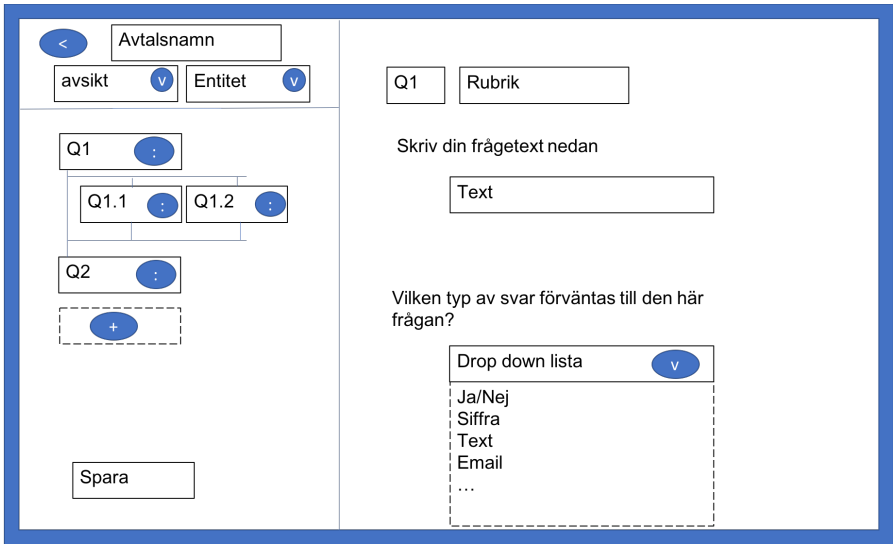
Indata	**Matchade**
Jag har anställt en konsult	Ja
Jag har blivit anställt som konsult	Ja
Vi ska ta in en konsult	Nej
En firma kontaktade mig om en konsulttjänst	Nej
Vi ska hyra in en rådgivare	Nej
Jag erbjöd rådgivning	Nej
Vi tog in en expert	Ja
Firman anlitade en konsult	Ja

B Designprototyp

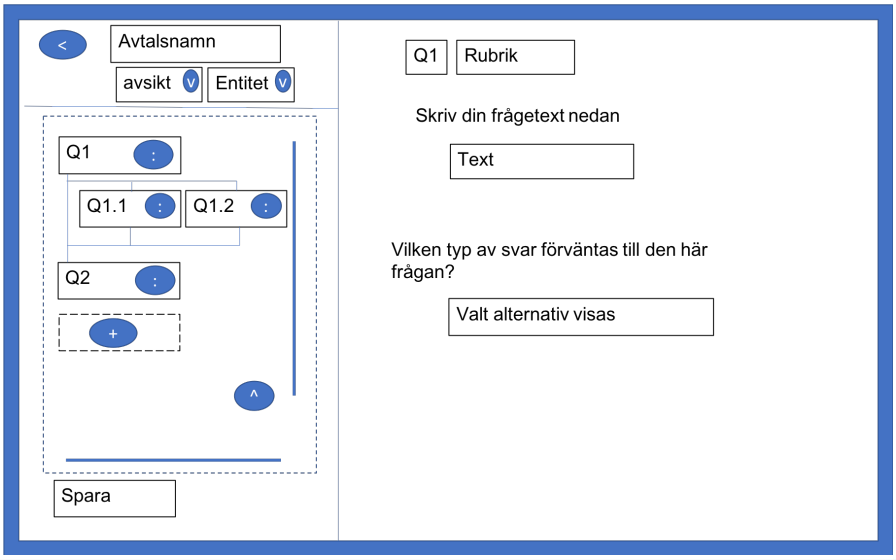
I figur 12 följer den designprototyp som togs fram för avtalsskaparen. I en första prototyp vägde en funktionell layout tyngre än en snyggare grafisk design. Detta är även en design som innehåller viss funktionalitet som inte implementerats i nuvarande version, men som beskrivs under framtida arbeten.



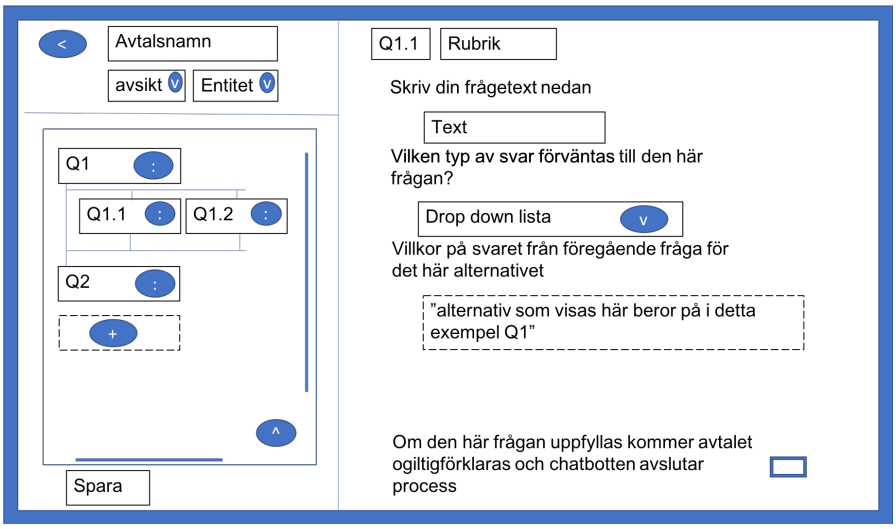
Figur 12a Initial vy för avtalsskaparen med ett exempel dialogträd för en avtalsmall



Figur 12b Fönster som kommer fram när en av noderna i dialogträdet klickas på



Figur 12c Efter att typ av förväntat svar för den här frågan har valts så visas det valda alternativet



Figur 12d Vi är nu i en följdfråga, tillkommet till denna vy är att en administratör måste definiera vilket svarsalternativ från föregående fråga som leder till den här frågan, även möjligheten att avsluta processen om svaret på en följdfråga skulle leda till det.