CHALMERS

Chattapplikation för effektiv vårdkommunikation

En chatbot med maskininlärning

Examensarbete inom Högskoleingenjörsprogrammet i Datateknik

OSKAR LIGNELL
ROBIN PUNELL

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2017

EXAMENSARBETE

Chattapplikation för effektiv vårdkommunikation

En chatbot med maskininlärning

OSKAR LIGNELL
ROBIN PUNELL

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET

Göteborg 2017

Chattapplikation för effektiv vårdkommunikation
En chatbot med maskininlärning
OSKAR LIGNELL
ROBIN PUNELL

Examinator: Peter Lundin

Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola / Göteborgs Universitet
412 96 Göteborg
Telefon: 031-772 1000

Institutionen för Data- och Informationsteknik
Göteborg 2017

SAMMANFATTNING

Inom vården sker idag kommunikation med gammalmodig teknik. Vårdtagare behöver ta kontakt via telefon för att få svar på generella frågor. Denna ineffektiva metod kan orsaka långa väntetider för enkla saker. Universitetssjukhuset i Linköping, som gett företaget ÅF uppdraget, ser ett behov av att förenkla och effektivisera den här kommunikationen för vårdtagare. Denna rapport avser att ge svar på hur detta kan effektiviseras med hjälp av en chattapplikation som använder en chatbot. En undersökning görs av vilka olika chatbotar som finns tillgängliga och vad för funktioner dessa tillhandahåller, samt vilka funktioner som krävs för att använda en sådan tjänst inom vården. Den prototyp som utvecklats är en chattapplikation implementerad i en webbläsare. Chattapplikationen kan besvara frågor som vårdtagaren tidigare ofta ställt via telefon. Svaren presenteras på olika sätt beroende på vad det är för typ av fråga, exempelvis med en bild om frågan handlar om var en avdelning finns eller ett formulär som fylls i av vårdtagaren om det är en tid som skall bokas. I prototypen lagras ingen personlig information då inget annat än en demonstration av hur chattapplikationen kan fungera har gjorts.

Nyckelord: chatbot, chattapplikation, vårdkommunikation

ABSTRACT

Within the healthcare today communication is used with old fashion technology. Patients must make a phone call to get an answer on a general question. This could cause long idle times and is an inefficient method for simple things. The University hospital in Linköping, which has given the task to the company ÅF, is in need to simplify and make this communication more efficient. This report intends to give answers on how to make this communication more efficient with the help of a chatbot and a chat application. An investigation of different chatbots and their functionalities is also made. The prototype that has been developed is a chat application embedded in a website. The application can answer questions previously asked by patients over the phone. Depending on what type of question that was asked, the answer is presented in different ways. If the question is about where a department is located the answer will contain a picture of a map, or with a form if the patient want to book an appointment. The prototype does not store any personal information since nothing more than a demonstration of how a chat application can work has been made.

Keywords: chatbot, chat application, health communication

FÖRORD

Denna rapport avser ge läsaren en inblick i hur en chatbot fungerar och hur denna kan nyttjas inom vården. Rapporten poängterar även hur viktigt det är att diskutera andra parametrar, såsom etik, än ekonomiska när den här typen av lösningar skall implementeras inom vården.

Denna rapport har skrivits av två studenter på högskoleingenjörsprogrammet datateknik på Chalmers Tekniska Högskola.

Ett stort tack till ÅF och Universitetssjukhuset i Linköping som gjort detta examensarbete möjligt. Ett särskilt tack till Fredrik Hofflander på ÅF som sett till så vi fått det material vi behöver, kommit med tips i mjukvaruutvecklingen och hjälpt oss med det vi haft behov av. Han har även gett oss en värdefull inblick i företagsvärlden.

Tack till vår handledare på Chalmers, Uno Holmer, som kommit med förslag och bidragit med erfarenhet gällande skrivandet av denna rapport.

INNEHÅLLSFÖRTECKNING

FÖRKORTNINGAR OCH BEGREPP

AI	Artificiell Intelligens
API	Application Programming Interface
Chatbot	En tjänst man interagerar med via ett chattfönster, tjänsten styrs av regler och ibland även artificiell intelligens
Emulator	Hårdvara eller mjukvara avsedd för att efterlikna annan hårdvara/mjukvara
Entity	Relevant information för en intent. Exempelvis ett datum vid bokning av en tid.
IDE	Integrated Development Environment
Intent	Representerar en handling användaren vill göra. Chatboten hanterar dessa intents.
Open Source	Källkod för datorprogram som är öppen för alla att läsa och ändra
REST	Representational State Transfer
SDK	Software Development Kit
Utterance	Sättet en användare förmedlar och formulerar sin intent. Exempelvis ”Jag vill boka en tid.”

1. INLEDNING
I detta kapitel introduceras bakgrunden till uppgiften och uppdragsgivaren, projektets syfte samt dess frågeställningar som skall besvaras.

1.1 Bakgrund
På företaget ÅF finns avdelningen Division Technology som är ett av Sveriges ledande konsultföretag inom IT och försvarsteknologi [1]. Deras fokus är digitaliseringen av en värld som blir mer och mer uppkopplad. De har fått i uppdrag av Universitetssjukhuset i Linköping (US) att ta fram ett effektivare sätt för vårdtagaren att få svar på frågor och exempelvis boka en tid.

US använder idag gammalmodig teknik där vårdtagaren behöver kontakta dem via telefon för saker som tidsbokning och generella frågor. US ser därför ett behov av att förenkla och effektivisera den här kommunikationen för vårdtagare då nuvarande process orsakar längre vårdtider och mer administrativt arbete för alla parter.

1.2 Syfte
Syftet är att ta fram en prototyp för en chattapplikation där vårdtagaren kan få svar på frågor samt boka eller ändra en redan bokad tid. Chatten skall använda sig av en chatbot och är tänkt att ersätta det gamla sättet med kommunikation via brev, pappersformulär och telefon som är dagens standard inom vården. I arbetet kommer det även utvärderas vad som krävs och om det är möjligt att vidareutveckla prototypen till en användbar produkt. Prototypen kommer att implementera en chatbot samt ett system för tidsbokning, detta är funktioner som uppdragsgivaren vill skall finnas med vid demonstrering av prototyp.

1.3 Frågeställning
- Vilka egenskaper bör en chatbot för vården ha?
- Vilka potentiella chatbotar finns på marknaden?
- Vad behöver tas hänsyn till för att kunna använda chatbotar inom vården?

1.4 Avgränsningar
Det kommer endast vara mjukvaruutveckling och inget annat än en prototyp av chattapplikationen kommer göras. AI och chatbot API tillhandahålls av externt företag via molntjänst som inte tar hänsyn till vård- och patientuppgiftslagar.

1.5 Läsanvisningar
Denna rapport går att läsa på två olika sätt. Om en mer översiktlig bild av systemets funktionalitet önskas behöver endast inledningen i avsnitt 4.2 läsas. För att få en tydligare bild av vad som krävs för att kunna utveckla med en chatbot läses hela avsnitt 4.2.

2. TEKNISK BAKGRUND

I detta kapitel redogörs den förståelse som behövs för de tekniska system och tjänster som används i arbetet.

2.1 Chatbot

En chatbot är en applikation som via text eller tal kommunicerar med en användare. Kommunikationen är tänkt att simulera en mänsklig konversation där chatboten exempelvis kan tillhandahålla information som efterfrågats av användaren.

2.2 C#

C# (uttalas ”see sharp”) är ett modernt, objektorienterat programmeringsspråk. Det kommer omedelbart se bekant ut för personer som använt sig av programmeringsspråk som C++ och Java. C# har flera hjälpmedel för att kunna konstruera stabila applikationer, exempelvis skräpsamling som automatiskt återtar minne från onåbara objekt, undantagshantering och type-safe design som gör det omöjligt att läsa från variabler som inte initialiserats [2].

2.3 Visual Studio

Visual Studio är ett komplett utvecklingsverktyg för att skapa applikationer som bygger på ASP.NET, XML och mobilapplikationer. Visual Basic, C# och C++ använder samma IDE vilket förenklar lösningar med blandade språk. Dessa använder även funktionalitet från .NET ramverket vilket ger åtkomst till speciella funktioner som förenklar utvecklingen [3].

2.4 IaaS

Infrastructure as a service, IaaS är en molntjänst som fungerar som en fjärrserver för valfri programvara och operativsystem. Tjänsteleverantören tillhandahåller en digital infrastruktur för kunden att köra en egen applikation över internet. Leverantören säljer servertid och serverkraft där kunden använder sina egna programvaror och operativsystem. Kunden behöver på så sätt inte administrera egna servrar men får samma funktionalitet som hos en egen server [4].

2.5 PaaS

Platform as a service, PaaS, är en vidareutveckling av en IaaS. Funktionerna från en IaaS finns kvar men operativsystem etc behövs inte administreras manuellt av kunden. Istället tillhandahålls allt som behövs för att köra applikationen av tjänsteleverantören [5].

2.6 Microsoft Bot Framework

Microsoft Bot Framework är ett ramverk som används för att bygga chatbotar. Det består av Bot Builder SDK, Bot Connector, Developer Portal samt Bot Directory. Det finns även en emulator som används för att testa den utvecklade chatboten. Ramverket tillhandahåller två olika SDK, en för .NET och en för Node.js [6].

2.6.1 Bot Builder SDK

Bot Builder SDK är Open Source gjord av Microsoft som finns på GitHub. Det ger utvecklare verktyg för att skapa dialoger och formulär [7].

2.6.2 Bot Connector REST API

Bot Connector API gör det möjligt för chatboten att skicka och ta emot meddelanden på kanaler som konfigurerats i Developer Portal. Bot Connector använder industristandardiserad REST och JSON över HTTPS [8].

2.6.3 Developer Portal
Med hjälp av Developer Portal är det möjligt att ansluta, redigera och publicera en chatbot på olika kanaler. Det går att ansluta till exempelvis Skype, Facebook Messenger och som en inbyggd chatt i en webbapplikation.

2.7 Microsoft Azure
Microsoft Azure är en molntjänst som används av utvecklare för att distribuera program via Microsofts datacenter [9]. Microsoft Azure använder PaaS och IaaS och kan alltså fungera som server över internet. Det fungerar som en Hub/samlingsplats för Microsofts tjänster och är utformat i moduler, endast de delar som behövs används.

2.8 Microsoft Cognitive Services
Microsoft Cognitive Services är en samling av olika APIs och SDKs som erbjuds till utvecklare för att göra deras applikationer smartare. Dessa APIs tillhör Microsofts utveckling av maskininlärning och består av exempelvis ansiktsigenkänning. Målet är att skapa en mer personlig datorupplevelse och en förbättrad produktivitet med hjälp av system som kan lyssna, se och prata [10].

2.8.1 Language Understanding Intelligent Services (LUIS)
LUIS är en lösning för att hjälpa datorn förstå vad en person vill i människa-dator interaktion. Det är designat för att förstå mänskliga språk och agera utefter vad som efterfrågades. Detta görs med intents (det användaren vill göra) och entities (nyckelord relevanta för intentionen). LUIS har en webbsida, den fungerar som en kontrollpanel där det är möjligt att skapa intents och entities relevanta för applikationen som utvecklas. När det inkommer trafik till applikationen använder LUIS en process kallad aktiv inlärning för att förstå mer. Under processen identifieras utterances (meningar) som LUIS har svårt att förstå, och därmed osäker på vilken intent som skall användas. Dessa meningar sparas på webbsidan där en administratör väljer vilken intent meningen är bäst lämpad för. Detta maximerar systemets inlärningsprocess [11].

2.8.2 Translator
Microsoft Translator API kan integreras i applikationer och andra verktyg för att möjliggöra en användarupplevelse på flera olika språk. Det använder sig av industri-standard och kan därför appliceras på godtycklig plattform med valfritt operativsystem. Translatorn kan då utföra flertalet olika språkrelaterade operationer såsom språkdetektering och översättning, text-to-speech och speech-to-speech [12].

Tekniken Statistical Machine Translation (SMT) har använts sedan API:t påbörjades. Denna teknik har dock nått sin platå när det kommer till förbättring av kvaliteten på översättningar. SMT håller därför på att bytas ut mot en ny AI-baserad teknik som bygger på Deep Neural Networks (DNN). DNN skapar mer naturliga översättningar då den använder hela meningen som kontext för att översätta ord. SMT tar bara ett par ord före och efter det ordet som skall översättas [12].

2.8.3 QnAMaker
Microsoft QnAMaker (Questions and Answers), som också är en del av Cognitive Services, är ett API för att ge chatboten möjligheten att svara på frågor som redan är svarade på i en FAQ (frequently asked questions). QnA-makern kan använda FAQ:n till att förstå användarens fråga och svarar med det givna svaret [13].

3. METOD
I detta kapitel beskrivs de verktyg och metoder som används i projektet.

3.1 Planering och organisation
Arbetet kommer att ske agilt för att på ett flexibelt sätt kunna planera och fatta beslut vid lösning av olika problem. Den agila metoden som kommer användas är iterativ och inkrementell utveckling, vilken är en av grundpelarna för agila metoder [14]. Detta för att alltid ha en fungerande prototyp som utvecklas och förbättras i omgångar efter utvärdering av tidigare gjorda implementationer. Utvärdering sker kontinuerligt och görs så fort något nytt implementerats. Nya funktionen testas och förbättring av funktionen påbörjas direkt. Utöver detta används verktyget Kanban Boards via Visual Studio Team Services [15]. Kanban visualiserar arbetsflödet för de funktioner som skall implementeras. Det skapar en tydlig översiktsbild över vilket arbete som pågår och vad som är färdigutvecklat. Detta görs med post-it lappar som sätts under olika rubriker på en tavla [16].

Versionshantering av källkoden kommer göras via Visual Studio Team Services som använder git [17]. Detta finns integrerat i Visual Studio vilket ger ett grafiskt gränssnitt vid exempelvis sammanslagning av olika grenar.

3.2 Utveckling av prototyp
Utvecklingen av applikationen sker i utvecklingsmiljön Visual Studio med tanke på att programmeringsspråket C# som används också är utvecklat av Microsoft [18]. Vid design av applikationen kommer Microsofts rekommenderade praxis avseende utveckling av chatbotar tas i beaktande [19]. Utöver en design med bästa praxis i åtanke kommer egna kombinationer av möjliga svar göras för att skapa en bra användarupplevelse med så människolik konversation som möjligt.

Då chatboten skall vara lättillgänglig för vårdtagaren från valfri plats kommer en enkel inbäddning i webbläsare göras för applikationen. Detta görs för att ge uppdragsgivaren en fingervisning för hur det kan se ut när chatboten implementerats på deras hemsida.

3.3 Testning och verifikation
chatboten kommer att testas manuellt med hjälp av emulatorn som finns i Microsoft Bot Framework. Efter en implementation gjorts i webbläsare kommer tester ske med olika datorer och webbläsare. Manuella tester har valts då en överblick på chatbotens svar vid ställda frågor fås enkelt och snabbt.

4. GENOMFÖRANDE

Vårdtagare skall kunna få svar på allmänna frågor och vid behov kunna boka en tid med hjälp av en chatbot. Detta görs på valfri plats av vårdtagaren genom att denne navigerar till chatboten via vårdgivarens hemsida, därefter upprättas en privat kommunikationsplattform. Vid tidsbokning fyller vårdtagaren i ett formulär som presenteras av chatboten. Formuläret skiljer sig åt beroende på om vårdtagaren vill boka om en redan befintlig tid eller boka en helt ny. När konversationen är färdig och tiden är bokad syns detta hos vårdgivare.

En prototyp utvecklades av den beskrivna tjänsten i syfte att skapa en demo-produkt som visas upp för uppdragsgivaren. Uppdragsgivaren skall kunna fortsätta arbetet för att införa denna typ av tjänst inom vården. Prototypen lagrar inte någon patientinformation, därför sker ingen säkerhetsanalys med kryptering av data.

4.1 Undersökning av tillgängliga chatbotar

För att kunna välja en chatbot till applikationen behövs en undersökning göras av vilka olika chatbotar som finns tillgängliga på marknaden. Den som väljs behöver bland annat ha stöd för implementation i webbläsare. Kostnader och vilka extratjänster som behöver läggas till hos respektive bolag för att få en användbar chatbot är andra aspekter att ta hänsyn till. Se bilaga 1 för en mer utförlig kravspecifikation.

4.1.1 Amazon Lex

Chatboten Amazon Lex är skapad av Amazon och finns tillgänglig genom Amazon Web Services (AWS) [20]. AWS är en molntjänst där det är möjligt att köpa exempelvis lagringsutrymme eller processorkraft för att köra kraftfulla program externt [21]. Med Natural Language Understanding (NLU) identifierar Amazon Lex intentionen i texten. Amazon Lex använder samma teknologi som Alexa [22], vilken är Amazons motsvarighet till den mer allmänt kända personliga assistenten Apple Siri [23]. Amazon Lex lär sig olika sätt som användaren kan uttrycka sin intent genom att mjukvaruutvecklaren lägger in ett par exempel på utterances [22].

För att kunna använda Amazon Lex behövs inloggningsuppgifter till AWS. Kontot som dessa uppgifter hör till måste ha behörigheter för användning av de resurser som finns på AWS som tillhör Lex. Det finns olika typer av konton som har tillgång till detta, ett konto med maximal access är av typen AWS account root user [24](s. 128).

Det finns olika kanaler där det är möjligt att implementera Amazon Lex vilket möjliggör ett användande över flera samtidigt. De alternativ som finns för implementation är Facebook Messenger, Slack, Twilio SMS samt en mobilapplikation som utvecklas via AWS Mobile Hub [22]

Med Amazon Lex sker betalning utefter användning. Kostnaden beräknas månadsvis på antal förfrågningar som skickats. En textförfrågan kostar $0.00075, vilket ger en kostnad av $0.75 per 1000 förfrågningar [25]. Det finns en testperiod där det är möjligt att använda Amazon Lex i ett år. I testperioden ingår 10 000 textförfrågningar. Varje input som görs räknas som en förfrågan [25].

All data skickas över HTTPS protokoll, vilket är det enda som stödjs. Det som skickas krypteras med en AWS Secret Access Key, denna används som autentisiering vid API förfrågan [26]. Av den data som skickas till Amazon Lex så kommer alla utterances att sparas av Amazon. Detta görs, enligt Amazon, för att kunna förbättra produkter relaterade till

maskininlärning. Samt att datan är viktig för vidare utveckling av Amazon Lex som i slutändan skall ge en förbättrad användarupplevelse [26].

4.1.2 IBM Watson – Conversation service

Conversation service är en chatbot-tjänst som är utvecklad av IBM. Tjänsten kombinerar en enkel chatbot med funktioner för språkförståelse och maskininlärning. Conversation service är tänkt att implementeras i en applikation och hjälper användaren att kommunicera med applikationen genom olika interface. Beroende på applikationens utformning kan till exempel ett chatfönster på en hemsida, i en mobilapplikation eller genom SMS användas [27]. Applikationen använder sig av molntjänsten Bluemix, som är IBM:s plattform för PaaS och IaaS, för att för att använda Conversation service [28]. I Bluemix är det möjligt att ansluta fler tjänster för att analysera datan från användaren samt träna chatboten till att fungera på önskat sätt. Efter att datan analyserats får applikationen reda på den troligaste begäran från användaren och man kan utforma applikationen för att svara på denna [27].

Conversation service finns i en gratisversion där man får tillgång till 1000 förfrågningar/månad, upp till 3 arbetsytor och maximalt 25 intents. För standardversionen betalar man per förfrågan, priset för detta är $0.0025 per förfrågan, man får då även tillgång till 20 arbetsytor och möjlighet att använda 2000 intents [29]. I Bluemix finns även tillgång till översättning via Watson Language Translator, kostnaden för detta $0.02 per 1000 tecken. De första 250 000 tecknen är gratis [30].

4.1.3 Micosoft Bot Framework

Bot Framework är Microsofts variant på ett paket för att bygga chatbotar. Ramverket är gratis att hämta från Microsoft. Det går även att utveckla via Azure Bot Service som finns tillgängligt på Microsoft Azure, vilket är Microsofts molntjänst. Azure Bot Service erbjuder en utvecklingsmiljö i molnet, där man alltså använder sig av en webbläsare så ingen separat IDE är nödvändig [31].

För att få en smart chatbot med Bot Framework behöver Microsoft Cognitive Services implementeras. Det behövs ett konto på Microsoft och Azure för tillgång till detta. Tjänsterna som finns har både en gratis- och betalversion [32]. Med hänsyn till kravspecifikationen i bilaga 1 krävs två tjänster, Language Understanding Intelligent Service (LUIS) samt Translator. Gratisversionen för LUIS ger 10 000 förfrågningar per månad, medan betaltjänsten kostar $0,00075 per förfrågan [33]. När det gäller Translatorn har gratisversionen ett tak på två miljoner tecken per månad, medan betaltjänsten är $10 per en miljon tecken [34].

Bot Framework kan anslutas till flera olika kanaler. Efter att chatboten blivit registrerad och upplagd på Azure är det möjligt att konfigurera den för bland annat Skype, Bing, Facebook Messenger, Slack och en webapplikation [35].

4.1.4 Val av chatbot

Efter undersökningen skulle en chatbot väljas. Eftersom Amazon Lex både saknade stöd för implementation i webbläsare och översättningsfunktion vid undersökningstillfället var denna inget alternativ. Microsoft och IBM har likvärdig funktionalitet, båda är väletablerade företag med tillgång till molntjänst. Tabell 4.1 visar en sammanfattning över de olika tjänsternas funktionalitet. Efter diskussion med uppdragsgivare om vad som finns tillgängligt beslutades att Microsofts Bot Framework skulle användas. Bland annat på grund av möjligheten att använda Microsoft Azure som molntjänst.

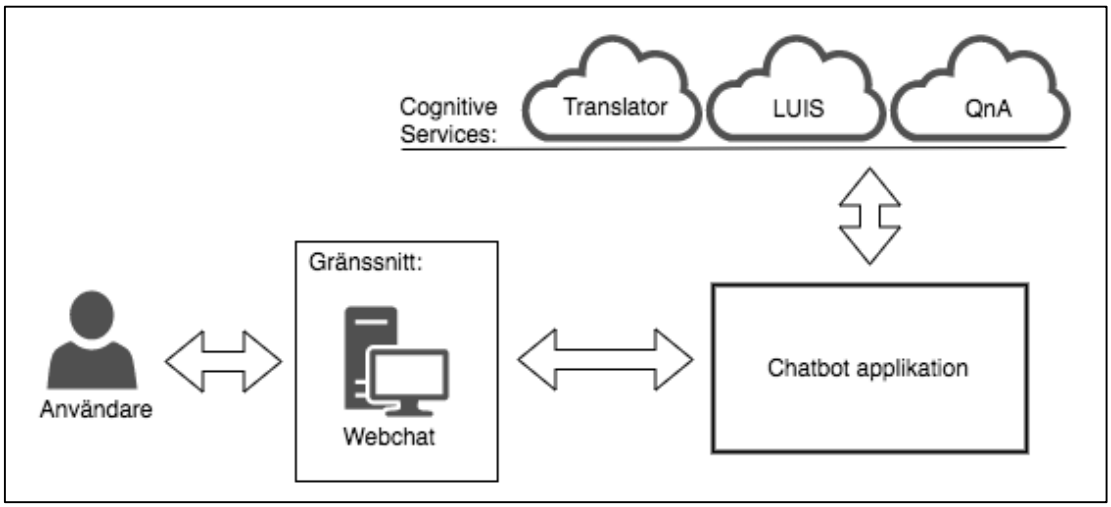
Tabell 4.1 Sammanfattning av priser och viktiga funktioner för de olika chatbotar som undersökts

	Microsoft Bot Framework	IBM Watson Conversation	Amazon Lex
Implementera på websida	Ja	Ja	Nej
Översättningsfunktion	Ja	Ja	Nej
Text to speech	Ja, med Bing speech	Ja	Implementerad med Polly
Pris	Gratis med LUIS: 10 000 förfrågningar/månad Standard: $0,00075/förfrågan	Gratis: 1000 förfrågningar/månad 25 intents Standard: $0,0025/förfrågan 2000 intents	Gratis: 1år med 10 000 text förfrågningar/månad Standard: $0,00075/förfrågan
Förstår "natural languages"	Ja, med LUIS	Ja	Ja
Språkstöd	English, French, Italian, German, Spanish, Brazilian Portuguese, Japanese, Korean and Chinese	Brazilian Portuguese, English, French, Italian, Spanish, German, Traditional Chinese, Simplified Chinese, and Dutch. Arabic (endast genom API)	US English

4.2 Konfiguration inför utveckling

För att kunna använda Microsoft Bot Framework krävs viss förkonfigurering. Det behöver skapas konton för användning av vissa tjänster, program skall installeras och ställas in med korrekta värden för att skapa en anslutning mellan tjänster. Slutligen skall chatboten registreras för att bli publik och möjlig att använda.

Innan en djupare beskrivning av konfiguration påbörjas visas en enklare översiktsbild av det färdigkonfigurerade systemet i figur 4.1 nedan. Detta för att underlätta förståelsen av vad det är som registreras i de olika stegen fortsättningsvis. En mer detaljerad bild finns i figur 4.4.



Figur 4.1 Förenklad översiktsbild av färdigkonfigurerat system

4.2.1 Utvecklingsverktyg

Först installeras utvecklingsmiljön Visual Studio för en optimal integrering med resterande Microsoft-tjänster som används. I programmet kommer sedan ett nytt projekt skapas med hjälp av en mall för chatbotar som lagts in. Mallen laddas ned från en direktlänk i dokumentationen för chatbotar och läggs in i Visual Studios mapp för mallar [36]. Detta görs för att mallen skall bli synlig i Visual Studio och möjlig att använda då det inte är en av originalmallarna för utveckling i C#.

Det behövs ett paket för Bot Framework som installeras via NuGet Package Manager som finns i Visual Studio. NuGet är Microsofts pakethanterare för utvecklingsplattformar såsom .NET [37]. För att installera paketet skrivs ”Install-Package Microsoft.Bot.Builder” in i Package Manager Console. Paketet hämtar hem och installerar de klasser etcetera som behövs från ramverket för att kunna påbörja utveckling av en egen chatbot.

4.2.2 Microsoft Azure

För att kunna ha en chatbot tillgänglig på internet krävs en server eller molntjänst med PaaS-funktionalitet. Microsoft Azure tillhandahåller dessa tjänster utan behovet av egen server. Azure fungerar även som en samlingsplats för Microsofts olika tjänster. Under sitt huvudkonto skapas och registreras alla moduler som behöver användas för applikationen. Dessa moduler registreras med unika applikations-ID som används för att integrera modulen med programvaran.

Först skapas huvudkontot på Azure. Under detta konto samlas alla övriga konton och registreringar. För att hålla koll på kostnader samt ha möjligheten att bestämma vilka fysiska resurser som skall användas på Microsoft Azures servrar skapas en Service Plan i Azure. Service Plan används för att organisera och lättare få en helhetsbild av de fysiska resurser som används för att driva applikationen [38]. Genom att skapa en Service Plan är det möjligt att välja hur mycket fysiska resurser som skall finnas tillgängliga på Microsofts servrar. Beroende på hur mycket kraft som behövs så kan olika priskategorier/instanser väljas. För applikationens syfte räcker den kostnadsfria instansen F1 men vid en realisering av projektet går det lätt att ändra prenumerationstyp efter det utökade resurskravet [39]. I tabell 4.2 finns en nedkortad prislista med information.

Tabell 4.2 Nedkortad prislista med de fysiska begränsningar och de tjänster som ingår i Microsoft Azures Service Plan instanser. Månadskostnaderna är uppskattade värden baserade på användningen av chatbot-applikationen i projektet [40]. För komplett prislista se bilaga.

Priskategori:	Premium	Standard	Basic	Free
Antal kärnor:	1–4	1–4	1–4	Delad infrastruktur
RAM-minne:	1,75 - 7GB	1,75 - 7GB	1,75 - 7GB	-
Lagring:	250GB	50GB	10GB	1GB
Övrigt:	Backup:50ggr/dag 20 instanser	Backup 1 ggr/dag 10 instanser	3 instanser	-
Uppskattad månadskostnad:	1700 – 7000kr/mån	600 – 2300kr/mån	400 – 1700kr/mån	0kr/mån

För att kunna börja använda kontot och den skapade mallen för en chatbot krävs att projektet med kod från Visual Studio publiceras till Azures servrar. Detta görs i Visual Studio och genom att använda samma kontouppgifter som används i Azure kan projektet publiceras direkt.

4.2.3 Developer Portal

Chatboten måste registreras vid Microsoft Application Portal för att kunna läggas upp på Microsoft Azure [41]. När ett användarkonto skapats kan applikationen registreras, applikationen erhåller ett unikt applikations-ID tillsammans med en unik nyckel. chatboten måste sedan registreras på Microsoft Bot Frameworks developer portal [42], en samlingssida där det går att samla alla skapade chatbotar. Här registreras applikationen som en chatbot genom att använda applikations-ID och nyckel som fåtts från Microsoft Application Portal.

När dessa registreringar gjorts behöver koden till chatboten kompletteras med ID och nyckel som kan ses i figur 4.2. Denna konfiguration görs i XML-filen web.config som skapas automatiskt av Bot Framework-mallen.

```
<appSettings>
  <add key="BotId" value="Bot_Application_Name" />
  <add key="MicrosoftAppId" value="Application_ID" />
  <add key="MicrosoftAppPassword" value="Application_Key" />
</appSettings>
```

Figur 4.2 Insättning av applikations-ID och nyckel för chatbot

4.2.4 Integrering mellan Cognitive Services och Azure

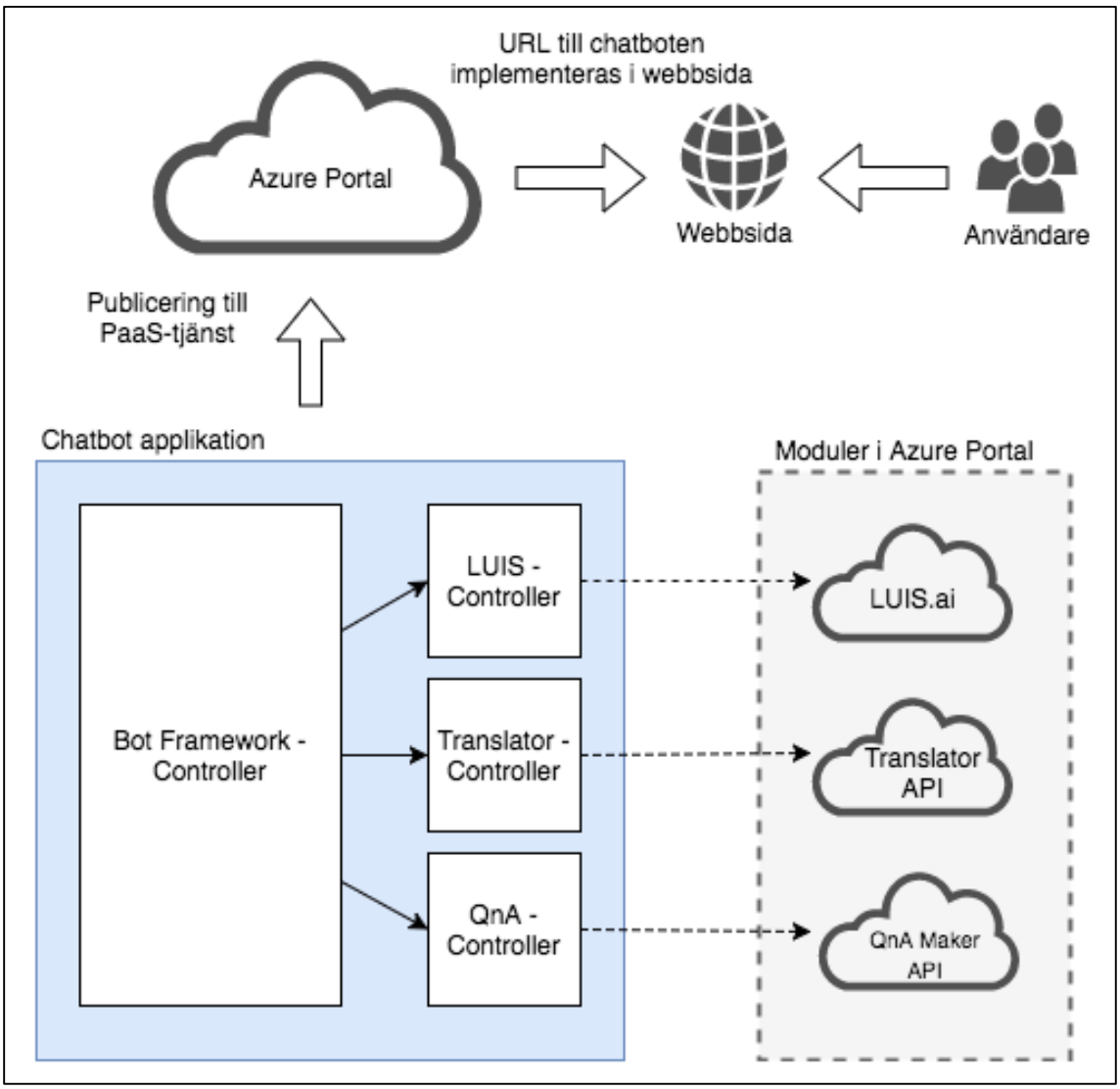
För att kunna implementera LUIS och Translator skapas registreringar för dessa tjänster genom Cognitive Services i Azure. Genom registreringen erhålls applikations-ID och nycklar för de båda tjänsterna. LUIS-registrering behöver sedan kopplas samman med en kontrollpanel för LUIS. Denna kontrollpanel sköts via www.luis.ai där ett konto skapas och kopplas samman med registreringen på Microsoft Azure med hjälp av applikations-ID från Azure-registreringen. Detta gör att de båda kontona syftar till samma LUIS-applikation. För att koppla samman den egenskrivna koden till att använda sig av den skapade LUIS-applikationen infogas applikations-ID och applikationsnyckel under ”LUIS_Application_ID” samt ”LUIS_Subscription_Key” enligt figuren nedan.

```
namespace Bot_Application.Dialogs
{
    [LuisModel("LUIS_Application_ID", "LUIS_Subscription_Key")]
    [Serializable]
    public class LuisDialog1 ...
}
```

Figur 4.3 Insättning av applikations-ID och nyckel för LUIS

Translatorn kräver inte någon hopkoppling mot en kontrollpanel utan använder sig endast av API-anrop. Dessa anrop görs och godkänns genom att inkludera den givna applikationsnyckeln. Den översatta texten returneras efter att nyckeln godkänts.

När alla moduler i Microsoft Azure konfigurerats med rätt applikations-ID från de olika registreringarna och allt kopplats ihop med chatboten fås ett system motsvarande det i figur 4.4.



Figur 4.4. Översiktsbild av färdig konfigurerat chatbot-system

4.3 Utveckling av chatbot

Chatboten som utvecklats är ett substitut för vårdtagaren vid kontakt med vårdgivare. Istället för att behöva vänta i telefonkö eller åka in till sjukhuset kan vårdtagaren få svar direkt genom att ställa sin fråga i chatten på hemsidan. Chatboten svarar på många av de frågor som personalen tidigare gjort. Detta möjliggör att vårdgivare får tid till andra uppgifter som leder till ökad effektivitet.

4.3.1 Grundläggande funktionalitet

Vid utvecklingen har applikationen utformats i hög grad enligt Microsofts rekommenderade bästa praxis. Bästa praxis är Microsofts manual för hur en chatbot bör bete sig. En chatbot som utvecklats med denna praxis i beaktande har ett bra välkomstmeddelande som ger användaren information om vad som går att få svar på, samt uppmanar till konversation. Det är också viktigt att låta användaren ha kontroll över chatten [16]. Ett exempel på hur detta kan se ut visas i figur 4.5 nedan.

Figur 4.5 Välkomstmeddelande

Varje gång en användare ansluter till chatten startas en aktivitet med konversationen och chatboten inväntar ett meddelande från användaren. När aktiviteten tas emot innehållandes ett meddelande hämtas detta. Meddelandet hanteras sedan på olika sätt beroende på vad det innehåller. När meddelandet hanterats skickas ett relevant svar till användaren. Chatboten går sedan tillbaka i vänteläge och inväntar nästa inkommande aktivitet för att upprepa meddelandeproceduren. Figur 4.6 visar hur ett välkomstmeddelande skickas till användaren när den inkommande aktiviteten signalerar att en användare anslutit.

```
else if (message.Type == ActivityTypes.ConversationUpdate)
{
    // Handle conversation state changes, like members being added and removed
    // Use Activity.MembersAdded and Activity.MembersRemoved and Activity.Action for info
    // Not available in all channels
    IConversationUpdateActivity update = message;
    var client = new ConnectorClient(new Uri(message.ServiceUrl), new MicrosoftAppCredentials());
    foreach (ChannelAccount member in message.MembersAdded)
    {
        if(member.Name.Equals("BotApplication") || member.Name.Equals("Bot"))
        {
            var welcome = message.CreateReply();
            welcome.Text = $"Välkommen till Röntgen-info. Här kan du få svar på allmäna frågor, " +
                           $"till exempel information om olika behandlingar och avdelningens öppettider.{Environment.NewLine}" +
                           $"Om du vill se lite mer i detalj vad du kan få svar på kan du fråga om hjälp";
            await client.Conversations.ReplyToActivityAsync(welcome);
            welcome.Text = $"Ingen data kommer att sparas och jag kan inte hantera ärenden som kräver personnummer.";
            await client.Conversations.ReplyToActivityAsync(welcome);
        }
    }
}
```

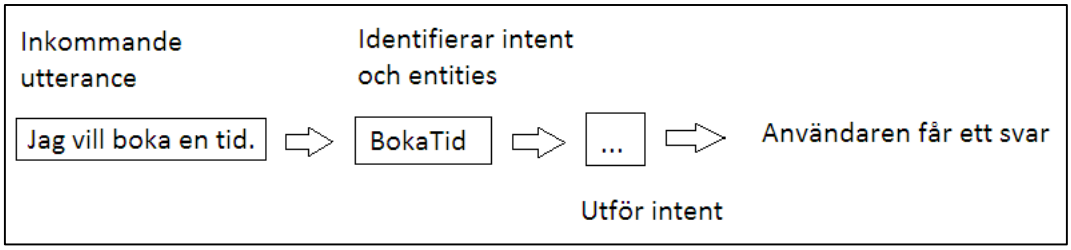
Figur 4.6 Kodexempel välkomstmeddelande när användare ansluter

4.3.2 LUIS

För att göra chatboten mer intelligent och ha en större förståelse för meddelanden som tas emot implementeras LUIS som använder sig av Natural Language Understanding (NLU) [43]. NLU möjliggör en mer mänsklig konversation. Det bygger på att applikationen, chatboten i detta fall, skall kunna lösa ut kontexten ur meningar [44]. Genom att använda LUIS kan entities och intents skapas som hjälpmedel för att få utterances att tolkas rätt. När användaren sedan skickar ett meddelande kommer rätt svar att skickas tillbaka tack vare kategoriseringen som gjorts i LUIS. Nedan följer en kort punktlista med förklaringar till de begrepp som används vid implementeringen av LUIS.

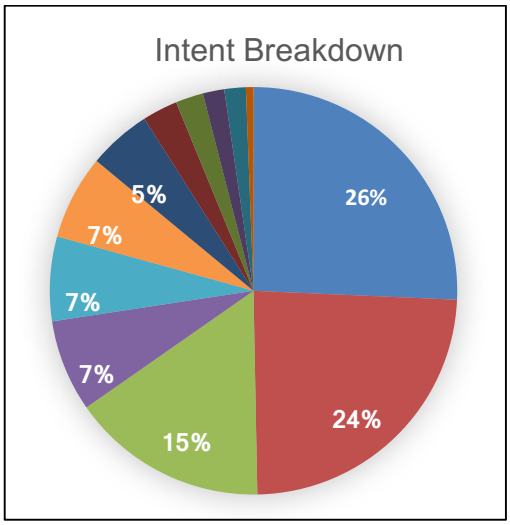
- **Utterance** är input till LUIS, alltså den text som användaren har skrivit. Det är denna utterance som chatboten skall tolka. Det kan vara en vanlig mening, exempelvis ”Jag vill boka en tid”, eller bara delar från en fulländad mening, exempelvis ”boka tid”. En utterance är inte alltid välformulerad och det finns många olika sätt att uttrycka samma sak.
- **Intent** representerar handlingen som användaren vill göra, likt ett verb. Intenten är det användaren vill göra uttryckt i text som blir en utterance. I utterancen ”Jag vill boka en tid” är verbet boka, användaren har alltså för avsikt att boka en tid i det här fallet. Applikationen kan därmed ha en intent som hanterar denna handling.
- **Entity** är relevant information för en intent. Så om intent är ett verb kan entity vara ett substantiv. För utterancen ”Jag vill boka en tid” är alltså intenten att boka, och entityn kan då vara substantivet tid. Som exempel kan man säga att det finns en klass kallad Tid, med subklasserna veckodag, datum och klockslag. Med en utterance som innehåller ett datum så är datumet en entity av klassen Tid.

I figuren nedan ges en överblick av hur LUIS fungerar som komplement till beskrivningarna ovan.

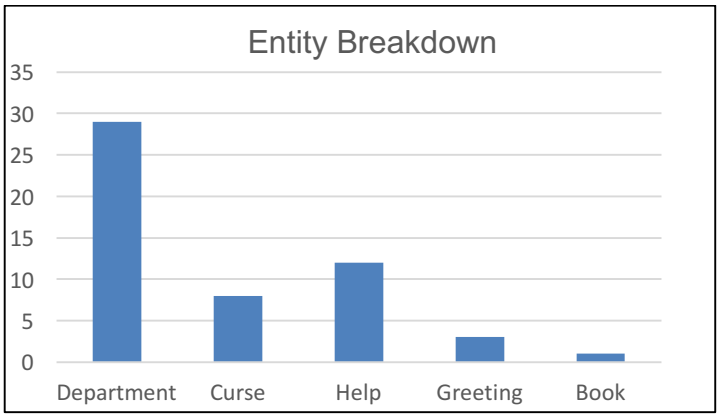


Figur 4.7 Visuell beskrivning LUIS implementerad i chatbot

Alla inställningar görs i LUIS kontrollpanel för en tydlig översikt med ett grafiskt gränssnitt. Det finns en meny för navigering, som bland annat innehåller instrumentpanel, konfigurering av intents och konfigurering av entities. Ett exempel på hur instrumentpanelsvyn med statistik kan se ut finns i figur 4.8 och 4.9. Varje gång något nytt konfigurerats och ändringar gjorts krävs att LUIS ”tränas”. Detta för att uppdatera och synkronisera med resterande funktionalitet så det nya implementeras korrekt.



Figur 4.8 Exempeldiagram över den procentuella användningen av de olika intents som finns i LUIS.



Figur 4.9 Stapeldiagram över hur många ord som refererats till olika entities

För att kunna utnyttja LUIS skapas först en intent. Den kopplas sedan till koden som skall exekveras vid den handlingen. För varje handling som skall kunna hanteras behöver det skapas en intent. Det finns en förinställd intent kallad ”none” där alla utterances som anses otydliga för chatboten bör lagras. Det kan vara irrelevanta meddelanden, eller något som behöver förtydligas för att hamna i lämplig intent. I figur 4.10 visas det som sker när LUIS hanterar en aktivitet med ett meddelande som hamnar under none-intenten. Strängen som skapas skickas med i metoden Send. I denna metod bearbetar översättaren (se avsnitt 4.3.3 för mer information) meddelandet för att sedan skicka svaret till användaren.

```
// "None"-intent
[LuisIntent("")]
public async Task none(IDialogContext context, LuisResult result)
{
    string reply = "Can you please be more specific? I do not understand completely.";
    await Send(reply, context, result);
}
```

Figur 4.10 Kodexempel för none-intent

När intenten skapats och olika utterances lagts in kan relevant data i den inkommande texten markeras som en entity. Entities gör så chatboten kan upptäcka och spara den datan som behövs för motsvarande intent. Detta gör att om användaren uppger den datan som behövs i ett tidigt skede märker chatboten detta och hoppar vidare till ett senare skede i processen, istället för att be användaren om information som redan uppgetts. Ett exempel på hur det här fungerar finns i figur 4.11 och figur 4.12.

Jag vill boka en tid på onsdag 17:e maj.	(1)
Jag vill boka en tid på {$datetime}	(2)

Figur 4.11 Exempel på användning av entities

Meningen markerad som (1) i figur 4.11 är inlagd under en intent kallad BookTime. När det sedan finns en entity kallad datetime kan dagar/datum/tider markeras såsom (2) visar. När LUIS får resultatet från aktiviteten kan ord från det skickade meddelandet som matchas mot entityn sparas. Detta gör så formuläret BookForm, som skapas med hjälp av en FormDialog (se avsnitt 4.3.4 för mer information) i figur 4.12, justeras med inkommande värden.

```
[LuisIntent("BookTime")]
public async Task BokaTid(IDialogContext context, LuisResult result)
{
    await context.PostAsync("Hej! Det kan vi ordna, det kommer komma några alternativ nu där du får välja det som passar bäst.");

    var form = new FormDialog<BookForm>(new BookForm(), BookForm.BuildForm, FormOptions.PromptInStart, null);
    context.Call(form, BookFormComplete);
}
```

Figur 4.12 Kod för intenten BookTime som skapar ett formulär

4.3.3 Translator API

En viktig del för att göra chatboten mer användarvänlig och för att kunna rikta sig till en bredare målgrupp var att implementera översättningsfunktionalitet. Översättningen gör så att vårdtagare kan kommunicera med chatboten på valfritt språk och få svaren översatta till det valda språket. På så sätt minimeras information som inte är tillgänglig men det ställer också höga krav på översättningens kvalitet.

Enligt en studie genomförd av Västra Götalandsregionen står asylsökande endast för en mycket liten del av den totala hälso- och sjukvårdskonsumtionen. Detta har enligt rapportförfattarna delvis att göra med stor brist på tolkar och svårigheter att ta del av information. Den språkliga barriären utgör även ett hinder för att boka planerad vård [45]. För att minska denna barriär var en implementering av översättningsfunktionen nödvändig, eftersom chatboten skall underlätta arbetet för vårdpersonal, samtidigt som vårdtagare lättare skall hitta rätt information.

Ett val som tidigt behövde göras var att bestämma om översättningen skulle bestämmas av användaren, där man genom en meny väljer vilket språk man talar och vill ha svaren på, eller om applikationen automatiskt ska känna av vilket språk som användaren skriver på och sedan

svara på det språket. En kort lista över positiva och negativa saker gällande de olika alternativen finns i tabell 4.3. Efter input från uppdragsgivaren valdes den automatiska översättningen. För prototypens målsättning stämde detta val bäst med den önskade slutprodukten.

Tabell 4.3 Lista över de olika alternativen för översättning

	Eget val av språk	**Automatiskt val av språk**
Positivt	Blir alltid rätt språk, Lättare att implementera, Låg risk för felöversättning	Få alternativ för användare, Snyggare, Låga krav på användare
Negativt	Många alternativ på skärmen, Krav på datorvana ökar	Krav på att översättaren känner av rätt språk

För att minimera felöversättningar sparas alla informationsmeddelanden på engelska. Dessa översätts sedan till användarens talade språk. Detta gäller även information som skall ges på svenska. Att all information översätts från engelska till användarens språk ställer höga krav på att översättningen fungerar som den är tänkt och att kvalitén uppnår de utsatta kraven. Skulle kvalitén på översättning från svenska till övriga icke-engelska språk visa sig vara hög går det att använda sig av svenska som huvudspråk för informationsmeddelanden etcetera, tester för detta gjordes ej. För att se om den valda översättningstjänsten höll hög standard gjordes en enklare kravspecifikation samt tester. Kravspecifikationen kan ses nedan och resultatet från översättningarna kan läsas i bilaga 3.

Kravspecifikation för översättningstjänst:

- Stödja översättning åt båda håll
- Mycket bra översättningsförmåga för svenska och engelska
- Ha stöd för översättning av de vanligaste språken i världen
- Snabb
- Utbyggnadsbar
- Hög kvalitet på översättningen

Vid en realisering och vidareutveckling av prototypen krävs en mer utbredd undersökning av kvalitén på översättning mellan olika språk. Här har endast kvalitén på översättningen mellan svenska och engelska granskats. Resterande språk har testats för att se om det översätts men ej granskats eller kontrollerats.

I nuläget har inte LUIS stöd för svenska [46]. För fullt utnyttjande av språkförståelsen i LUIS behöver alltså användarens input skrivas på engelska. Till detta används översättaren och den översatta inputen bearbetas av LUIS som ger svar på engelska. Figuren nedan visar hur inputen hanteras och översätts innan den returneras till användaren.

Figur 4.13 Översiktsbild på hur input hanteras

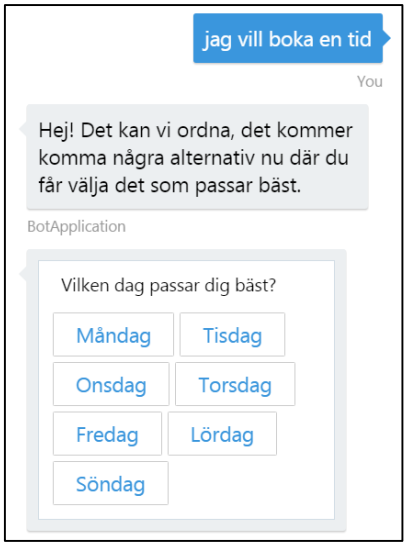
4.3.4 FormFlow

I konversationerna som sker med chatboten händer det ibland att användaren behöver guidas igenom en process. Genom att implementera det Microsoft kallar FormFlow, som finns i SDK:n, är det möjligt att skapa färdiga formulär. FormFlow genererar automatiskt det som krävs för att utföra dialogen. Formuläret utformas utefter viss indata som utvecklaren skrivit [47].

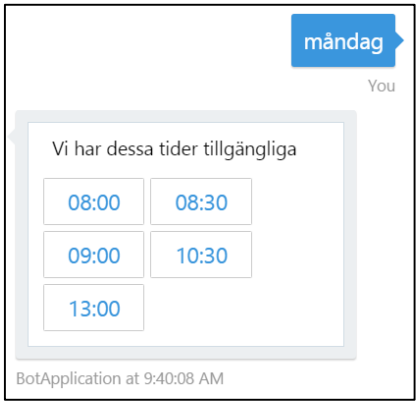
För att få ett fungerande formulär behövs bland annat valbara alternativ. Detta skapas med enum för varje fråga. I figur 4.14 visas TimeOption, vilket är när användaren behöver välja en tid som passar. Ett exempel på hur det kan se ut när FormFlow skapar dialogen med hjälp av indatan visas i figur 4.15 och 4.16.

```
public enum TimeOption
{
    [Describe("08:00"), Terms("08:00"), Description("08:00")]AM0800,
    [Describe("08:30"), Terms("08:30"), Description("08:30")]AM0830,
    [Describe("09:00"), Terms("09:00"), Description("09:00")]AM0900,
    [Describe("10:30"), Terms("10:30"), Description("10:30")]AM1030,
    [Describe("13:00"), Terms("13:00"), Description("13:00")]AM1300
};
```

Figur 4.14 Kodexempel för svarsalternativ i formulär



Figur 4.15 Start av FormFlow



Figur 4.16 Fortsättning av FormFlow

4.4 Implementering i webbläsare

För att vårdtagare skall ha möjlighet att konversera med chatboten behöver det vara möjligt att implementera denna i en hemsida. Inför demonstration av prototyp skapades en simpel hemsida där konversation är möjligt och för att visa uppdragsgivaren hur det kan se ut i ett senare skede. chatboten har en endpoint URL som används för att nå boten, denna URL kapslades in i en enkel iframe som lades inuti hemsidan enligt figur 4.17.

```
<div class="jumbotron">
    <h1>Bot-Bengt</h1>
    <iframe style="height:480px; width:300px" src='https://webchat.botframework.com/embed/AfBotApplication'></iframe>

</div>
```

Figur 4.17 iframe med förkortad endpoint-url till chatboten.

För att simulera prototypen på ett så bra sätt som möjligt skapades även en kopia av Universitetssjukhuset i Linköpings hemsida där chatboten implementerades. Detta för att få en klarare bild över hur en realisering av prototypen skulle se ut.

4.5 Attachments

Vid vissa konversationer är det önskvärt att presentera svaret mer visuellt genom att förslagsvis visa en bild med tillhörande text eller video. För att kunna göra detta i Bot Framework används attachments (Bilagor). Beroende på hur svaret skall presenteras väljs ett presentationskort. Det finns i dagsläget åtta typer av kort som presenterar data för användaren på olika sätt [48]. Bland dessa finns Hero Card, som visar en bild samt en text och/eller en knapp, Video Card som kan presentera en video inuti chatfönstret samt Audio Card som kan spela upp en ljudfil för användaren. I figuren nedan visas hur presentationskortet Hero Card används i prototypen.

Figur 4.18 Exempel på hur ett Hero Card används

5. RESULTAT

I detta kapitel redovisas de resultat som åstadkoms med chatbot respektive bokningssystem, sammansättning av delar till demo-produkt och hur läget är på marknaden.

5.1 Tidsbokning

Systemet för tidsbokning blev ej färdigställt och implementerades aldrig. Orsaker och tankar kring detta finns under kapitlet Diskussion.

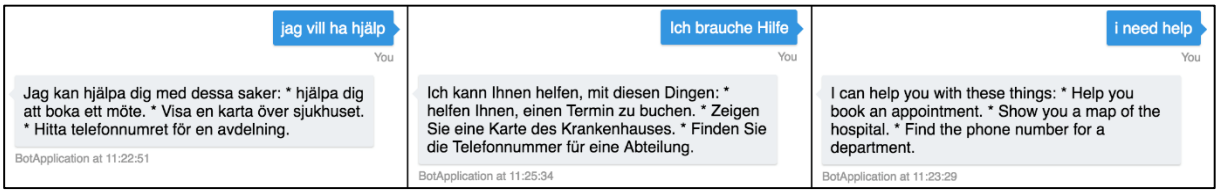
5.2 Prototyp till demonstration

Under projektet har en chattapplikation utvecklats som använder sig av en chatbot. Denna är implementerad i en webbapplikation för att visa uppdragsgivaren hur den slutgiltiga produkten skulle kunna fungera. Målet var att utveckla en prototyp för en lösning som erbjuder vårdtagaren en tjänst som är effektivare vid kontakt med vårdgivare. Kontakten består av att vårdtagaren skall få svar på frågor som inte hanterar personnummer. Genom att uppfylla detta är det möjligt att demonstrera ett system som i ett senare skede kan expandera och ta över fler uppgifter.

Prototypen som skapats är en molnbaserad chatapplikation som använder sig utav flera av Microsofts tjänster för skapande av chatbotar, språkförståelse och språköversättning. Applikationen kan föra en enkel konversation med en användare och svara på vanligt ställda frågor. Frågorna behöver inte ställas på ett specifikt sätt, applikationen använder avancerad språkförståelse för att uppfatta innebörden av meningen och svara därefter med det bäst lämpade svaret.

Figur 5.1. Exempel på hur chatboten använder språkförståelse för att avgöra användarens avsikt.

Applikationen klarar dessutom av att svara på frågor på en mängd olika språk. Användaren kan skriva på en stor mängd språk, chatboten känner av vilket språk som använts och svarar därefter på samma språk. I exemplet nedan skrivs samma mening på svenska, tyska och engelska för att demonstrera hur svaret kan se ut på olika språk.



Figur 5.2. Exempel på hur chatboten känner av användarens språk och översätter svaret till samma språk.

Applikationen kan starta en bokningsförfrågan och fylla i formulär utifrån ett antal givna tider. Denna funktion är endast för demonstrationssyfte och inget riktigt bokningssystem har implementerat. För att implementera ett fungerande bokningssystem behövs fortsatt utveckling.

Figur 5.3. Exempel på hur bokningsformuläret ser ut.

5.3 Chattapplikation

Huvuduppgiften har varit att utveckla en chattapplikation som använder sig av en chatbot och implementera funktioner enligt uppdragsgivarens önskemål. Den har utvecklats med Microsofts rekommenderade bästa praxis för chatbotar i åtanke för att skapa en bra användarupplevelse. Applikationen ger vårdtagare en möjlighet att få svar på frågor som annars hade krävt ett telefonsamtal eller att fysiskt ta sig till vårdgivaren med risk för väntetider. Resultatet blir att både vårdtagaren och vårdgivaren får tid till annat och därmed ett effektivare system.

5.4 Chatbotar på marknaden

Undersökningen som gjordes av vilka chatbotar som finns tillgängliga visade att de flesta från väletablerade företag håller likvärdig nivå. Det finns liknande typer av funktioner som erbjuds hos de olika aktörerna och prissättningen på tjänsterna är även den likvärdig. Alla är även väldigt lika när det kommer till syntax då de använder sig av vad som kan liknas vid en standard på vilka uttryck som används för olika saker, exempelvis utterance och intent.

6. DISKUSSION

I detta kapitel presenteras egna tankar kring resultatet och de tjänster som använts för att ta fram en fungerande prototyp. Det förs även en diskussion om det etiska dilemmat och säkerhetsaspekten med den här typen av teknik i samhället. Avslutningsvis ges förslag till fortsatt arbete.

6.1 Resultatet

Prototypen som togs fram uppnår de flesta fastställda krav i kravspecifikationen. Det centrala delarna; skapande av boten, språkförståelse och översättare gjordes först och mycket tid lades ner för att detta skulle fungera bra. Dessa delar är fungerar som en stomme för hela applikationen men är byggda för att lätt kunna anpassas till nya funktioner, moduler eller delar. När QnA-makern, senare i projektet, lades till så behövde inte andra delar av koden modifieras nämnvärt. QnA-funktionen ligger som en lös modul och går att modifiera och/eller tas bort utan att resterande kod påverkas. Detta är något vi är nöjda med och anser är en av applikationens styrkor.

Vi är även nöjda med mängden funktioner som är implementerade och enkelheten att fylla på med fler frågor och svar. För prototypens syfte används en viss typ av frågor och språk men applikationen kan genom förhållandevis små modifikationer få ett helt nytt beteende. Funktionerna är desamma men genom att byta språk, LUIS-databas samt frågor/svar så skulle applikationen kunna fungera på en mängd andra typer av webbplatser och plattformar.

Något som vi blev positivt överraskade av var hur väl LUIS fungerade. Det var ett krav för applikationen att förstå meningar som inte var korrekt skrivna samt förstå innebörden av en mening utan att behöva hårdkoda alla möjliga alternativa sätt att ställa en viss typ av fråga. Genom att inte hårdkoda frågor blir det mycket lättare att utvidga applikationen med fler funktioner och frågor/svar. Detta gick mycket smidigt och även om vissa meningar/ord var extra svårtydda, och behövdes läggas in i LUIS manuellt, så klarade den av de flesta meningar.

Några saker med applikationen som vi är mindre nöjda med är bl.a. möjligheten att demonstrera applikationen på önskat sätt. Eftersom det bara är en prototyp och inte en full implementering så är fråge/svar-databasen väldigt tunn. Detta göra att chatboten endast kan svara på några få frågor och framstår därför inte så kompetent.

6.1.1 Vad som kunde gjorts annorlunda

Något som skulle kunna ha gjorts annorlunda var utvecklingen av tidsbokningssystemet. Systemet stoppades under utvecklingsperioden. Detta då uppdragsgivaren valde att ge frågedelen av prototypen en högre prioritet efter ett möte. För egen vinning och för rapportens syfte hade det varit bättre att utvecklingen av tidsbokningssystemet fortsatt. Detta hade resulterat i en applikation med fler funktioner och ett mer avancerat system när chatbot satts ihop med ett system för tidsbokning.

Något som också borde fått större fokus är driftsäkerheten och stabiliteten av chatboten. Eftersom prototypen endast skapades i demonstrationssyfte och för att nya tjänster användes, var det svårt att implementera ordentliga automatiska tester. chatbotens stabilitet kunde varit bättre, anslutningen mellan chatbot och användare tappas ibland. Detta gör så meddelanden behöver skickas om. Det kan vara så att detta sker på grund av att vissa tjänster ligger långt ifrån Sverige vilket leder till hög fördröjning och i slutändan en timeout, men det kan också

bero på att vissa tjänster inte används optimalt. Det är svårt att avgöra orsaken till dessa problem utan automatisk testning eftersom problemen uppkommer till synes slumpmässigt.

Vi har under användningen av chatboten konstaterat att den automatiska översättningen bör förbättras. I nuläget översätts alla svar från chatboten till användarens språk. Detta fungerar bra i många fall men vid korta meningar från användaren har chatboten svårt att avgöra vilket språk som använts. Om t.ex. användaren skriver in ett personnummer, så skall boten svara med att den inte behandlar personnummer. Problemen uppstår när användaren endast skriver ett personnummer, utan att skriva några ord för chatboten att analysera. I det fallet så svarar chatboten på engelska, eftersom alla informationsmeddelanden är skrivna på engelska, vilket kanske inte alls stämmer överens med resten av konversationen. Detta är något som behöver undersökas och utvecklas för att få en bra slutprodukt. Några exempel på hur detta skulle kunna ordnas är att spara användarens tidigare använda språk, ha svenska som standardöversättning eller att implementera någon form av knappval för valt standardspråk.

6.2 Allmänna tankar

Eftersom chatbotar i allmänhet är ett relativt nytt område har inte tjänsterna nått en stabil nivå. Det är svårt att hitta information om exempelvis bästa praxis och det finns ingen satt industristandard för hur programkoden bör skrivas. Det ger å andra sidan utvecklaren fria tyglar i utvecklingsprocessen. Dock tar utvecklingen längre tid än vanligt eftersom dokumentationen gällande klasser och metoder är ytterst begränsad.

Många av de tjänster som använts har varit i ett utvecklingsstadie så allting uppdateras, ungefär som en betaversion. Under arbetets gång har registreringsprocesser förändrats och dokumentation för hur vissa tjänster fungerar har försvunnit eller flyttat till andra hemsidor. När informationen bytt plats har den ibland även förändrats, så informationen om ett tillvägagångssätt kan plötsligt se annorlunda ut.

6.3 Valet av chatbot

Under projektet har tre olika chatbotar som finns tillgängliga undersökts, IBM, Microsoft och Amazon. De har alla liknande upplägg med priser och möjliga extrafunktioner. Vid undersökningstillfället hade dock Amazon brist på funktioner i de områdena som behövdes. Microsoft används i prototypen då tjänsten fungerar bra tillsammans med molntjänsten Azure. Om projektet utförts utan uppdragsgivare hade troligen IBM använts. Detta då det upplevdes stora svårigheter att komma igång med Microsoft på grund av programvaruproblem och en utdragen process med konfigurationen.

6.4 Miljö och hållbar utveckling

Ett införande av en mer utvecklad prototyp inom vården kan påverka miljön och samhället positivt. När vårdtagaren kan prata med chatboten på vårdgivarens hemsida istället för att åka in till sjukhuset för vissa frågor och svar minskar antalet resor. Det kan även bli kortare kötider i samband med att fler vårdtagare får hjälp via chatboten, sjukvårdspersonal får minskad arbetsbelastning vilket kan leda till färre sjukskrivningar som leder till bättre arbetsresultat. När vårdtagaren får svar på frågor kan antalet sjukvårdsbesök minska vilket i sin tur minskar sjukvårdskostnader.

6.5 Etik inom vårdkommunikation

Verbal kommunikation är endast en liten del av kommunikationen eftersom kroppsspråk, ansiktsuttryck, mimik, tonläge etcetera kan förändra mycket i en konversation. När den

verbala delen även uteblir och konversationen endast görs i textform ökar risken för missförstånd ännu mer när vårdtagaren behöver hjälp.

Som människa kanske man inte känner sig som just en mänsklig unik varelse när man kommunicerar med en robot eftersom den reagerar lika på all text oavsett vem som skrivit den. Detta kan även tas ett steg längre där många meningar kan ha olika innebörd för olika människor vilket är svårt för en bot att förstå. Om någon som har svårt att uttrycka sig och chatboten inte förstår eller tolkar meddelandet fel så kan vårdtagaren känna sig otillräcklig.

Utöver missförstånd så kan det var svårt att sätta en klar gräns för vilka besked och uppgifter som bör ges ut på det här sättet. Vilken typ av besked är okej att skickas online och vad behöver en vårdgivare lämna över och meddela personligen? Innan modern teknik likt chatbotar börjar implementeras inom vården behöver den här typen av frågor diskuteras, även säkerhetsaspekten som tas upp i kommande kapitel bör tas hänsyn till.

6.6 Säkerhetsaspekten

Om en applikation av den här typen skall realiseras inom vården behöver inte bara den etiska biten funderas över, men även ur ett säkerhetsperspektiv. Först behöver vårdtagaren identifiera sig, med tanke på andra tjänsters inloggningar görs detta smidigt med mobilt bankID. I det här skedet måste allt stämma med personuppgiftslagen och inom en snar framtid EU:s nya regelverk för behandling av personuppgifter [49] [50].

Chatten behöver även ha någon form av kryptering så konversationen mellan vårdtagare och chatbot är privat. Om känslig data som provsvar eller liknande skickas finns risken att utomstående fiskar upp datan. Den här typen av information kan leda till utpressningssituationer och andra otrevligheter för vårdtagaren. Någon begränsning för vilken typ av opersonlig data som chatboten skall lämna ut bör också diskuteras.

6.7 Förslag till fortsatt arbete

För att kunna ta prototypen till nästa steg och bli mer användbar måste en implementation i webbläsaren göras mer utförligt, detta kan göras med Microsoft Direct Line [51]. Detta ger möjligheten att modifiera utseendet på boten. Utan denna implementering är det inte möjligt att ändra färg, typsnitt eller utformning av chatboten. Ett annat förslag till fortsatt utveckling är en hopkoppling med Vårdguiden, 1177.se, och där använda Mobilt bankID, detta möjliggör en användning av det som finns tillgängligt för vårdtagaren på 1177, och skicka detta i chatten. Applikationen måste då även följa alla lagar inom Sverige och EU.

Ett införande där nya intents och utterances skapas automatiskt via anrop till LUIS API istället för att gå via kontrollpanelen är en möjlig vidareutveckling. Detta skulle minska arbetsbelastningen på personal som då inte behöver administrera chatboten i lika hög utsträckning med kontrollpanelen. Vidare kan text-to-speech implementeras för att ge en förbättrad användarupplevelse. Denna funktion finns tillgänglig som en modul via Microsoft Cognitive Services och bör vara enkel att implementera. Eftersom det inom sjukhuset finns mycket fackspråk och ord som kan vara svåra att översätta kan det vara bra att skapa en egen databas med översättningar, denna databas går sedan att sammanfoga med Translator via Microsoft Translator Hub [52]. På så sätt kan man minimera felöversättningar och missförstånd.

I dagsläget finns inget skydd för att skicka många meddelanden på kort tid till chatboten. Om en användare gör detta kan det leda till att chatbotens svar blir fördröjda eller felaktiga men

inga större problem skulle uppstå. Om en attack med hundratals meddelanden, eller mer, skickas så kan dock kostnaderna för boten stiga. Det krävs väldigt många meddelanden och det är möjligt att Bot Framework har en inbyggd funktion för att motverka detta, men detta är något som bör tittas närmare på.

KÄLLFÖRTECKNING

[1] ÅF, ”Fokus på digitalisering och den uppkopplade världen,” [Online]. Available: http://www.afconsult.com/sv/investor-relations/bolagsstyrning/organisation/division-technology/. [Använd 4 April 2017].

[2] Microsoft, ”A Tour of the C# Language,” 10 August 2016. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/articles/csharp/tour-of-csharp/. [Använd 4 April 2017].

[3] Microsoft, ”Introducing Visual Studio,” [Online]. Available: https://msdn.microsoft.com/en-us/library/fx6bk1f4(v=vs.100).aspx. [Använd 4 April 2017].

[4] C. S. IDG, ”Infrastructure as a service,” [Online]. Available: http://it-ord.idg.se/ord/infrastructure-as-a-service/. [Använd 10 May 2017].

[5] C. S. IDG, ”Platform as a service,” [Online]. Available: http://it-ord.idg.se/sprakwebben/ord.asp/?ord=platform-as-a-service. [Använd 10 May 2017].

[6] Microsoft, ”Bot Framework Overview,” [Online]. Available: https://docs.botframework.com/en-us/ . [Använd 6 April 2017].

[7] Microsoft, ”Bot Builder SDK,” [Online]. Available: https://github.com/Microsoft/botbuilder. [Använd 6 April 2017].

[8] Microsoft, ”Bot Connector,” [Online]. Available: https://docs.botframework.com/en-us/restapi/connector/. [Använd 6 April 2017].

[9] Microsoft, ”Vad är Azure?,” [Online]. Available: https://azure.microsoft.com/sv-se/overview/what-is-azure/. [Använd 11 April 2017].

[10] Microsoft, ”Cognitive Services Overview,” [Online]. Available: https://www.microsoft.com/cognitive-services/en-us/documentation. [Använd 11 April 2017].

[11] Microsoft, ”Language Understanding Intelligent Service,” [Online]. Available: https://docs.microsoft.com/en-us/azure/cognitive-services/LUIS/Home. [Använd 13 April 2017].

[12] Microsoft, ”Translator Overview,” [Online]. Available: https://www.microsoft.com/cognitive-services/en-us/translator-api/documentation/TranslatorInfo/overview. [Använd 13 April 2017].

[13] Microsoft, ”QnA Maker Overview,” [Online]. Available: https://qnamaker.ai/Documentation. [Använd 19 May 2017].

[14] C. Larman och V. R. Basili, ”Iterative and Incremental Development: A Brief History,” June 2003. [Online]. Available: http://www.craiglarman.com/wiki/downloads/misc/history-of-iterative-larman-and-basili-ieee-computer.pdf. [Använd 11 April 2017].

[15] Microsoft, ”Agile Tools,” [Online]. Available: https://www.visualstudio.com/team-services/agile-tools. [Använd 11 April 2017].

[16] Leankit, ”What is kanban?,” [Online]. Available: https://leankit.com/learn/kanban/what-is-kanban. [Använd 11 April 2017].

[17] Microsoft, ”Git for individuals, teams and enterprises,” [Online]. Available: https://www.visualstudio.com/team-services/git. [Använd 11 April 2017].

[18] Microsoft, ”C# and .NET Programming,” [Online]. Available: https://msdn.microsoft.com/en-us/library/orm-9780596521066-01-01.aspx. [Använd 11 April 2017].

[19] Microsoft, ”Best practice,” [Online]. Available: https://docs.botframework.com/en-us/directory/best-practices/#navtitle. [Använd 23 April 2017].

[20] S. Perez, ”Amazon Lex, the technology behind Alexa, opens up to developers,” Tech Crunch, 20 April 2017. [Online]. Available: https://techcrunch.com/2017/04/20/amazon-lex-the-technology-behind-alexa-opens-up-to-developers/. [Använd 25 April 2017].

[21] A. Hern, ”Amazon Web Services: the secret to the online retailer's future success,” The Guardian, 2 February 2017. [Online]. Available: https://www.theguardian.com/technology/2017/feb/02/amazon-web-services-the-secret-to-the-online-retailers-future-success. [Använd 25 April 2017].

[22] Amazon, ”Amazon Lex Product Details,” [Online]. Available: https://aws.amazon.com/lex/details/. [Använd 26 April 2017].

[23] L. Goode, ”Hey Siri, who's better: you or Alexa?,” The Verge, 28 April 2016. [Online]. Available: http://www.theverge.com/2016/4/28/11463336/apple-siri-vs-amazon-alexa-which-is-better-versus. [Använd 26 April 2017].

[24] Amazon, ”Amazon Lex: Developer Guide,” 2017. [Online]. Available: http://docs.aws.amazon.com/lex/latest/dg/lex-dg.pdf. [Använd 27 April 2017].

[25] Amazon, ”Amazon Lex Pricing,” [Online]. Available: https://aws.amazon.com/lex/pricing/. [Använd 3 May 2017].

[26] Amazon, ”Data and Security,” [Online]. Available: https://aws.amazon.com/lex/faqs/#data-security. [Använd 3 May 2017].

[27] IBM, ”Overview of the IBM Watson Conversation service,” [Online]. Available: https://console.ng.bluemix.net/docs/overview/whatisbluemix.html#bluemixoverview. [Använd 27 Apr 2017].

[28] IBM, ”What is Bluemix?,” [Online]. Available: https://console.ng.bluemix.net/docs/overview/whatisbluemix.html#bluemixoverview. [Använd 27 Apr 2017].

[29] IBM, ”Conversation - Pricing,” [Online]. Available: https://www.ibm.com/watson/developercloud/conversation.html#pricing-block. [Använd 22 May 2017].

[30] IBM, ”Language Translator,” [Online]. Available: https://console.ng.bluemix.net/catalog/services/language-translator. [Använd 22 May 2017].

[31] Microsoft, ”How the Bot Framework works,” 10 May 2017. [Online]. Available: https://docs.microsoft.com/en-us/bot-framework/overview-how-bot-framework-works. [Använd 16 May 2017].

[32] Microsoft, ”What is Cognitive Services?,” 2 January 2017. [Online]. Available: https://docs.microsoft.com/en-us/azure/cognitive-services/welcome. [Använd 16 May 2017].

[33] Microsoft, ”Cognitive Services Pricing - Language Understanding Intelligent Services,” [Online]. Available: https://azure.microsoft.com/en-us/pricing/details/cognitive-services/language-understanding-intelligent-services/. [Använd 16 May 2017].

[34] Microsoft, ”Cognitive Services Pricing - Translator Text API,” [Online]. Available: https://azure.microsoft.com/en-us/pricing/details/cognitive-services/translator-text-api/. [Använd 16 May 2017].

[35] Microsoft, ”Connect a bot to channels,” 11 May 2017. [Online]. Available: https://docs.microsoft.com/en-us/bot-framework/portal-configure-channels. [Använd 16 May 2017].

[36] Microsoft, ”Getting Started,” [Online]. Available: https://docs.botframework.com/en-us/csharp/builder/sdkreference/gettingstarted.html. [Använd 8 May 2017].

[37] Microsoft, ”NuGet Documentation,” [Online]. Available: https://docs.microsoft.com/en-us/nuget/. [Använd 8 May 2017].

[38] Microsoft, ”Azure App Service plans in-depth overview,” [Online]. Available: https://docs.microsoft.com/en-us/azure/app-service/azure-web-sites-web-hosting-plans-in-depth-overview. [Använd 8 May 2017].

[39] Microsoft, ”App Service Priser,” [Online]. Available: https://azure.microsoft.com/sv-se/pricing/details/app-service/. [Använd 8 May 2017].

[40] Microsoft, ”Microsoft Azure Portal,” [Online]. Available: https://portal.azure.com. [Använd 9 May 2017].

[41] Microsoft, ”Microsoft Application Registration Portal,” [Online]. Available: https://apps.dev.microsoft.com. [Använd 9 May 2017].

[42] Microsoft, ”Microsoft Bot Framework,” [Online]. Available: https://dev.botframework.com/. [Använd 9 May 2017].

[43] Microsoft, ”Understanding Natural Language,” [Online]. Available: https://docs.botframework.com/en-us/node/builder/guides/understanding-natural-language/. [Använd 9 May 2017].

[44] E. Bastianelli, G. Castellucci, D. Croce, R. Basili och D. Nardi, ”Effective and Robust Natural Language Understanding for Human Robot Interaction,” i *European Conference on Artificial Intelligence*, Prague, 2014.

[45] A. Kjellström, B. Palaszewski, Z. Cherigui, M. Telemo Taube, G. Henning och H. Ascher, ”Asylsökandes vårdkonsumtion i Västra Götaland 2011-2016,” Västra Götalandsregionen, 2017.

[46] Microsoft, ”Localization support in LUIS apps,” [Online]. Available: https://docs.microsoft.com/en-us/azure/cognitive-services/luis/luis-concept-language-support. [Använd 11 May 2017].

[47] Micirosoft, ”Basic features of FormFlow,” 10 May 2017. [Online]. Available: https://docs.microsoft.com/en-us/bot-framework/dotnet/bot-builder-dotnet-formflow. [Använd 15 May 2017].

[48] Microsoft, ”Add rich card attachments to messages,” [Online]. Available: https://docs.microsoft.com/en-us/bot-framework/dotnet/bot-builder-dotnet-add-rich-card-attachments. [Använd 18 May 2017].

[49] Sveriges Riksdag, ”Personuppgiftslag (1998:204),” [Online]. Available: http://www.riksdagen.se/sv/dokument-lagar/dokument/svensk-forfattningssamling/personuppgiftslag-1998204_sfs-1998-204. [Använd 19 May 2017].

[50] Datainspektionen, ”Dataskyddsreformen,” [Online]. Available: http://www.datainspektionen.se/dataskyddsreformen/. [Använd 19 May 2017].

[51] Microsoft, ”Bot Connector - Direct Line API - v3.0,” [Online]. Available: https://docs.botframework.com/en-us/restapi/directline3/. [Använd 22 May 2017].

[52] Microsoft, ”Microsoft Translator Hub,” [Online]. Available: https://www.microsoft.com/en-us/translator/hub.aspx. [Använd 24 May 2017].

[53] Microsoft, ”App Service plans,” [Online]. Available: https://azure.microsoft.com/en-us/pricing/details/app-service/plans/. [Använd 24 May 2017].

BILAGOR

Kravspecifikationer

Kravspecifikation för val av chatbot-tjänst

- Tjänsten skall ha funktionalitet för implementation på en websida
- Tjänsten skall ha en översättningsfunktion
- Tjänsten skall använda språkförståelse för att tolka användares input
- Tjänsten skall ha möjligheten att skapa formulär för användaren
- Tjänsten skall vara möjlig att köras från tillhandahållen molntjänst
- Tjänsten skall tillhandahållas av väletablerat företag för att upprätthålla hög driftsäkerhet
- Tjänsten skall vara kostnadseffektiv och skalbar vid utökning till att hantera större mängder data

Kravspecifikation för demonstrationsmodell

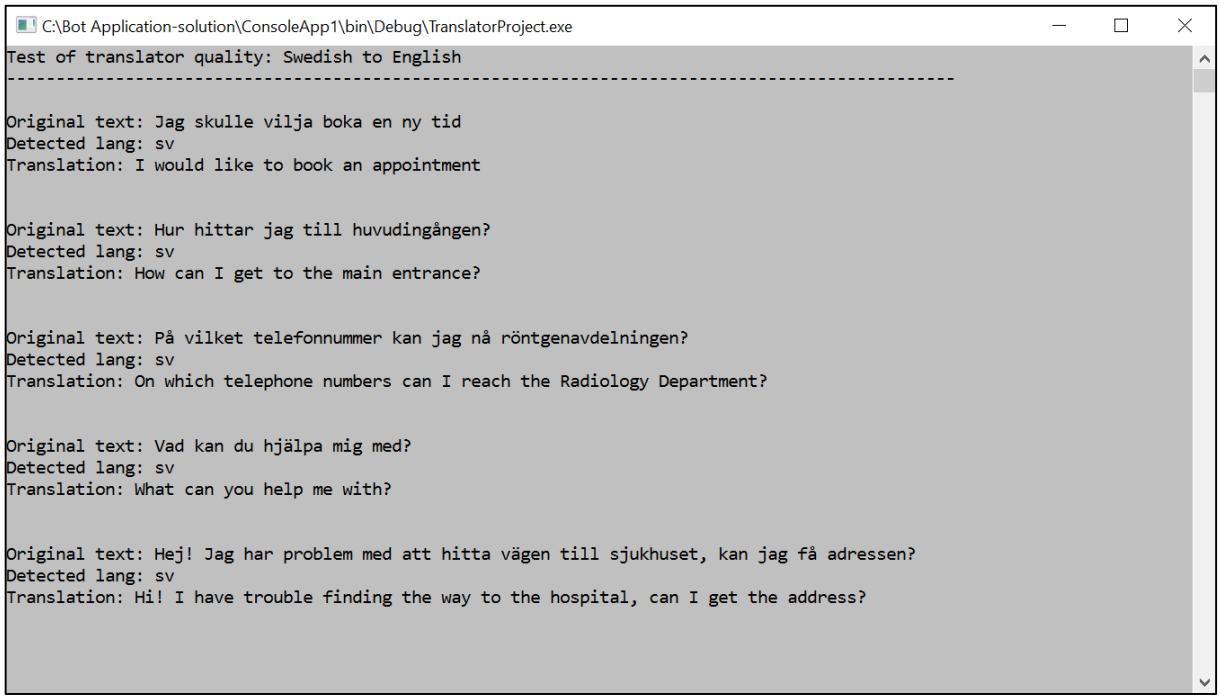
- Chatboten skall finnas tillgänglig på en hemsida
- Chatboten skall kunna svara på vanligt ställda frågor från vårdtagare
- Chatboten skall kunna erbjuda vårdtagaren möjlighet att boka en tid
- Chatboten skall förstå vad vårdtagare menar utan korrekt syntax och stavning

Prislista med tjänster för Microsoft App Service Plans

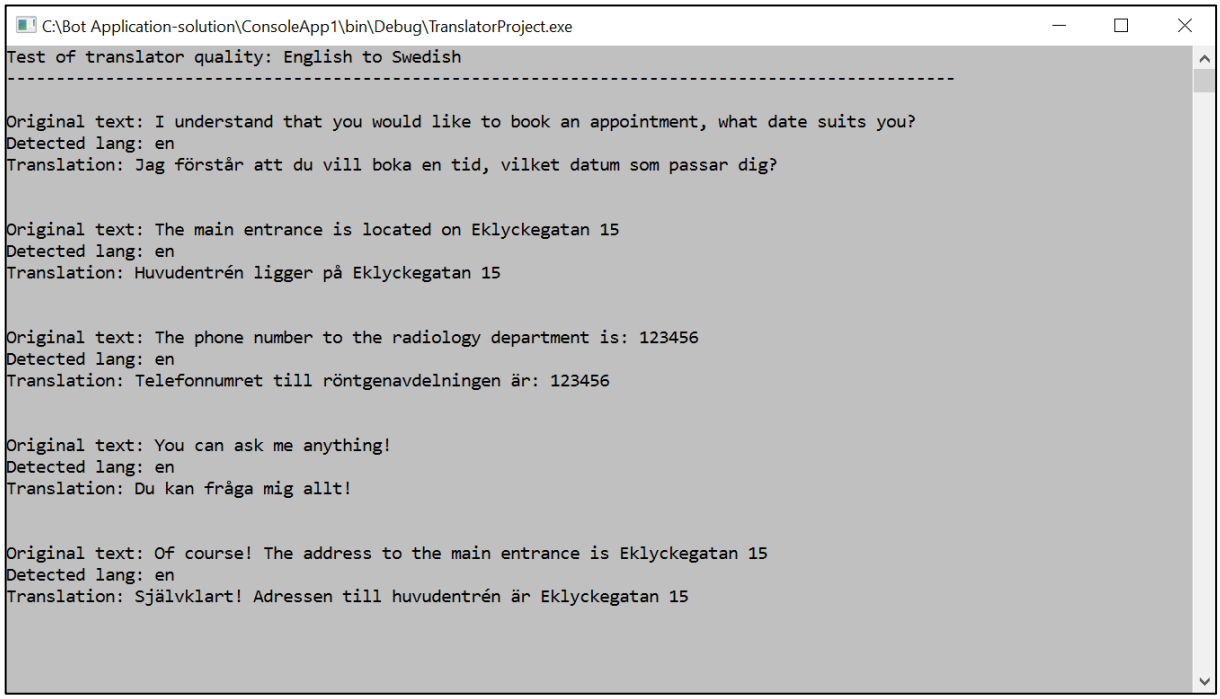
Service Plan	FREE	BASIC	STANDARD	PREMIUM
Web, mobile, or API apps	10	Unlimited	Unlimited	Unlimited
Disk space	1 GB	10 GB	50 GB	250 GB
Logic App Actions (per day)	200	200	10 000	50 000
Maximum instances	–	Up to 3	Up to 10	Up to 50
SLA	–	99.95%	99.95%	99.95%
Auto-Scale	–	–	Supported	Supported
Geo-distributed deployment	–	–	Supported	Supported
VPN hybrid connectivity	–	–	Supported	Supported
Staging environments	–	–	5	20
Custom domain	–	Supported	Supported	Supported
SSL certificates	–	Unltd. SNI SSL certs	Unltd. SNI SSL certs + 1 IP SSL included.	Unltd. SNI SSL certs + 1 IP SSL included.
Automated Backups (/day)	–	–	2	50
Active mobile devices	500 / day	Unlimited	Unlimited	Unlimited
Offline Sync	500 calls / day	1000 calls / day	Unlimited	Unlimited
Logic Apps Definitions	10	10	25	100
Logic App data storage cap	1 Day	1 Day	7 Days	30 Days
Cores:	Shared Infrastructure	1 to 4	1 to 4	1 to 8
Memory	–	1.75 - 7Gb	1.75 - 7Gb	1.75 - 14 Gb
Estimated monthly prices	0 SEK / Month	400 – 1700 SEK / Month	600 – 2300 SEK / Month	1700 – 16000 SEK / Month

Prislista med de fysiska begränsningar och de tjänster som ingår i Microsoft Azures Service Plan instanser. Månadskostnaderna är uppskattade värden baserade på användningen av chatbot-applikationen i projektet [53] [39].

Test av översättningskvalitet



Test av översättning från svenska till engelska.



Test av översättning från engelska till svenska.

Figurerna i denna bilaga visar delar av de enkla test som utförts för att se översättningskvalitén för Microsoft Translator API. Texten som står efter ”Original text” är text som skrivits in manuellt. ”Detected lang” är de av översättaren detekterade språket. ”Translation” är den automatiska översättningen av originaltexten från det detekterade språket till det valda språket.

Externa konton

Externa konton som skapats för projektet, och som behövs för applikationens funktionalitet, samt webbplatserna de nås på.

Microsoft Azure
https://portal.azure.com

LUIS
https://www.luis.ai

QnA-maker
https://qnamaker.ai

Bot Framework
https://dev.botframework.com/bots

Application registration portal
https://apps.dev.microsoft.com