UPPSALA
UNIVERSITET

IT 12 052
Examensarbete 15 hp
December 2012

Webbokning i TeleQ 5

Rasmus Middendorff

Institutionen för informationsteknologi
Department of Information Technology

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Web booking in TeleQ 5

Rasmus Middendorff

The Uppsala based company Aurora Innovation AB develops a call center solution called TeleQ with customer callback scheduling that decreases customer waiting time and increases agent productivity. There is also a possibility for the customer to book a callback time on the web. During the development of the fifth (and latest) version of TeleQ the need for a redesigned web booking module has emerged. This project aims to develop a future proof concept for how web booking should be handled in TeleQ 5.

Handledare: Kristina Engdahl
Ämnesgranskare: Lars Oestreicher
Examinator: Olle Gällmo
IT 12 052
Tryckt av: Reprocentralen ITC

Innehållsförteckning

1 Introduktion

Aurora Innovation (AIN) tillhandahåller telefonitjänster för hantering av stora volymer av inkommande samtal, vilka används av t.ex. landsting runtom i landet i kontakten med allmänheten. Kunder som ringer in till en organisation som använder samtalshanterings-systemet TeleQ kan ställas inför olika val, beroende på vilka tjänster som finns aktiverade i det aktuella systemet.

De fyra huvudtjänster som marknadsförs inom TeleQ är:

- **Callback**
  - Återuppringning
- **Picker**
  - Traditionell telefonkö
- **Duo**
  - Telefonkö med valbar återuppringning, en kombination av Picker och Callback
- **Push**
  - Telefonkö med automatisk framkoppling till ledig handläggare

I sin enklaste utformning, *Picker*, agerar TeleQ som en traditionell telefonkö där kunden väntar i telefonen med ett pågående samtal på att en handläggare ska bli ledig att ta emot samtalet (se illustration 1).[1] Med *Callback*-funktionen kan kunden välja bland föreslagna tider för återuppringning och kan sedan efter sin bokning lägga på luren istället för att vänta I en telefonkö.[2] En handläggare ringer sedan upp kunden så nära den bokade tiden som möjligt. En viktig del i TeleQ är bokningsalgoritmen som schemalägger utgående samtal beroende på tillgängligt antal handläggare och deras samtalskapacitet.

Systemen *Picker* och *Callback* kan även samköras i samma telefonkö och kallas då *Duo*. Kunden placeras då först i en *Picker*-kö men får i samband med den regelbundna informationen ("Du har plats (X) i kön..") även ett erbjudande om att trycka stjärna för att istället bli återuppringd senare.

Med *Push* som ett komplement till *Picker* kan handläggare välja att vara inloggade i systemet enbart via sin telefon. TeleQ kopplar då automatiskt fram inkommande samtal till en ledig handläggare. Skillnaden blir att handläggare inte kan se någon information om

1 TeleQ Picker, http://ain.se/sv/products/teleq/picker 2012-09-20
2 TeleQ Callback, http://ain.se/sv/products/teleq/callback 2012-09-20

samtalet innan de väljer att svara, men lösningen är praktisk på verksamheter där handläggare inte har möjlighet att aktivt arbeta vid en dator.

Med Choice tillkommer en meny där kunden kan välja på ett antal olika alternativ, som om så önskas kan kopplas till olika moduler (om kunden önskar att bli återuppringd eller hellre vill vänta, exempelvis) eller bara användas för att märka upp ett samtal så att en handläggare vet i vilket ärende kunden som samtalet rör har hört av sig. I anslutning till Callback finns ett enkelt webbgränssnitt för att boka tider online. Gränssnittet har även stöd för Choice och visar en platt meny som motsvarar den Choice-meny som finns definerad för telefondialogen.

I samband med att Aurora Innovation utvecklar en helt ny version av TeleQ, tänkt att nå marknaden under första halvan av 2012, behöver även ett nytt webbgränssnitt utvecklas för att bättre motsvara det nya bakomliggande systemet. Examensarbetet handlar om att ta fram en kravspecifikation, utreda tekniska lösningar för, samt i mån av tid implementera en ny webbapplikation för bokning av telefontider. Gränssnittet över vilket den nya webbapplikationen ska kommunicera med TeleQ-servern ska sedan specificeras efter att designen av webbokningen tagit form. Det finns även tidigare önskemål från kunder till Aurora Innovation på olika sätt att integrera webbokningen på befintliga webbplatser. Dessa önskemål ska sammanställas och vilka möjligheter som finns att tillgodose dem ska utredas.

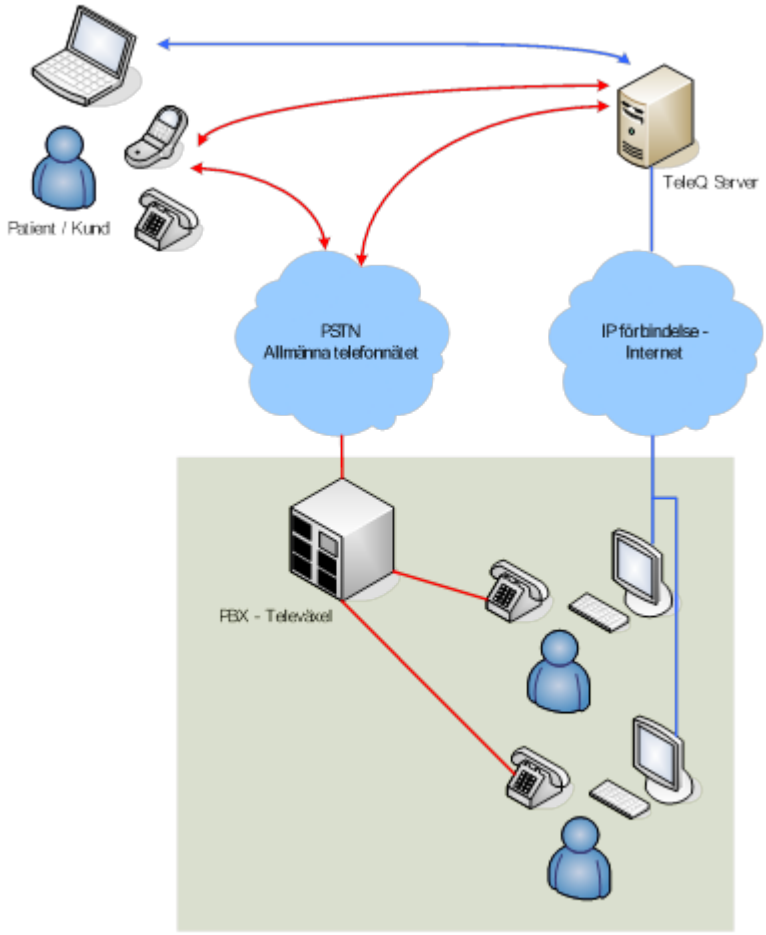

Illustration 1: Bilden visar hur TeleQ fungerar. TeleQ kan köras både som fristående server hos en kund och som en molntjänst.

Källa: Aurora Innovation AB

1.1 Problemformulering

På samma sätt som den nya versionen av TeleQ utvecklas för att bli mer kompetent, användbar och flexibel än sin föregångare måste den nya webbokningen utvecklas i samma riktning. Min uppgift är att ge förslag på hur man kan åstadkomma detta och hur en specifikation för ett sådant system skall göras.

1.2 Avgränsningar

Utvecklingen av den nya webbokningen lämpar sig väl som ett examensarbete då den är relativt fristående från TeleQ och kan till stor del bedrivas som ett fristående projekt. Med anledning av detta faller det sig naturligt att avgränsa projektet där modulen möter det egentliga TeleQ. Ingen utveckling ska ske direkt i TeleQ. Istället ska ett API defineras vilket webbokningen kan använda sig av för att kommunicera med TeleQ men implementationen av detta ligger utanför projektets ramar.

I det ständiga arbetet att förbättra produkten och anpassa den till kundernas behov är den viktigaste källan för idéer och förslag den dagliga erfarenhet som handläggarna gör i användandet av TeleQ. Detta gäller även webbokningen och därför består många av önskemålen på den nya webbokningen av tankar och synpunkter på den gamla som uttalats av kunder till Aurora Innovation. Att sammanställa dessa och formulera krav är en del av uppgiften men att uppsöka personer utanför Aurora Innovation i syfte att inhämta önskemål ligger utanför projektets omfattning.

2 Metod

2.1 Projektmetoden

I valet av metod för produktutvecklingen är det viktigt att ta hänsyn till två saker; dels den typ av produkt som ska utvecklas, dels i vilket slags projekt detta ska ske.

Att produktens huvuduppgift är att tillgodose ett behov av en enkel och lättbegriplig användarupplevelse fick mig att inrikta mig på en metod kallas användarcentrerad design (eng. User-centered design) som kortfattat bygger på att användarupplevelsen i alla lägen används som utgångspunkt för utformningen av resten av produkten (Gulliksen & Göransson, 2002).

2.2 Önskemålen

Då en stor del av projektet var att utifrån en grov kravspecifikation och önskemål om olika sorters funktionalitet utveckla en produkt som på ett tillfredsställande sätt tillgodoser dessa förväntningar föll det sig naturligt att tillämpa ett *agilt* (Cockburn, 2002) arbetssätt med relativt korta iterationer, mellan vilka avstämningar med produktägaren (min handledare, Kristina Engdahl) sker där resultatet av arbetet dittills och den fortsatta riktningen av detsamma diskuteras. Då hela utvecklingsgruppen endast består av mig själv, och min initiala kunskap om de större sammanhang, som inte nödvändigtvis framgår av en allmänt hållen kravspecifikation men som ändå bör påverka produktens utformning, är det av största vikt att iterationerna är korta och avstämningarna många.

Under arbetets gång har processen varit öppen för insyn från kollegorna på AIN och deras önskemål, synpunkter och funderingar har till stor del bidragit till att göra applikationen rustad för att möta de krav och förväntningar som användare och kunder har.

2.3 Utvecklingen

För utvecklingen av själva produkten har jag valt att arbeta med en bred prototyp, med s.k. *evolutionär prototyping* (Crinnion, 1991), där prototypen av slutprodukten stegvis utvecklas till något bättre, men alltid befinner sig i ett användbart stadium. Användarupplevelsen som helhet har varit central och det första som som tagits fram av varje del av applikationen har varit det visuella gränssnittet mot användaren.

3 Genomförande

I detta kapitel följer en beskrivning av hur genomförandet av arbetet skett. De olika stegen följer i kronologisk ordning, men flera iterationer av flödet har genomförts då prototyper visat på brister och förbättringsmöjligheter som gjort att kraven och önskemålen modifierats.

3.1 Kravspecifikation

Arbetet med kravspecifikationen har varit en permanent pågående process där upptäckta hinder, nya idéer och lösningar under projektets gång har lett till revideringar av densamma.

3.1.1 Önskemål

Under tiden som den befintliga webbokningen varit i drift har Aurora Innovation samlat in en omfattande kunskap om vad användare och kunder önskar sig i form av utökad funktionalitet och förbättringar. I inledningsskedet sammanställde jag en lista över dessa önskemål att ha som en av utgångspunkterna för kravspecifikationen.

- Möjlighet att hantera registreringsnummer för bilar. För fordonsbranschen finns ett intresse av att kunna låta användarna ange vilket fordon ett ärende gäller. Då detta inte går att lösa enkelt vid kommunikation över enbart telefon är det ett viktigt komplement i webbapplikationen.
- Avbokning av möten och dylikt med textmeddelande
- Avbokning av callback, eventuellt med kod vid bokning alternativt endast telefonnummer
- Jag vill bli uppringd och få en ny bokningstid
- Jag vill inte ha en ny tid och inte bli uppringd
- Begränsning av andelen samtalspositioner som kan bokas via webben (i motsats till via telefon) exempelvis för att förhindra en de facto förtur för mer tekniskt kunniga (yngre) personer
- Olika öppettider för allmänhet respektive interna användare.
- Bilduppladdning
- Direkt chatt med handläggare

3.1.2 Liknande lösningar/Inspirationskällor

I valet av telefonilösning för en organisation finns det många olika alternativ. En förutsättning för att en lösning ska vara liknande är att den innehåller en återuppringningsfunktion samt en till denna tillhörande webbokningsfunktion för bokning av återuppringningstider. Andra typer av telefonisystem, även om de erbjuder konkurrens på marknaden, är inte relevanta för en jämförelse med det här projektet.

Datatal

Datatal har i sin produkt *Flexi Tid* samma princip som *TeleQ Callback*.[1] En inringande kund kan i en telefondialog boka en tid för återuppringning. De väntande kundärendena presenteras sedan för en handläggare i ett webbgränssnitt och kan därifrån hanteras. Till detta bokningssystem finns även en webbokning där kunder i en platt meny kan välja mellan ett antal olika kategorier beroende på vad samtalet rör och där de sedan kan stega sig fram till en ungefärlig återuppringningstid som passar dem. Det framgår inte om föreslagna tider är reserverade i och med att kunden får se dessa eller inte, eller om det som visas för kunden endast är de tider som fanns tillgängliga när kunden efterfrågade informationen. Applikationen har även ett översiktsläge finns där kunden kan se beläggningen under olika intervall på dagen och kan välja att bli föreslagen en tid ur ett visst intervall.[2]

För att avboka en återuppringningstid ringer kunden in och anger det telefonnummer som återuppringningen registrerats med.

nicmar media

nicmar media driver *timecenter.com* som är en molnbaserad tjänst för bokning av tider i en kalender.[3]

Systemet saknar koppling till telefoni men är intressant som en studie i hur tidsbokning som sådan kan se ut på webben. En kund som önskar boka en tid får ett överskådligt veckoschema där alla bokningsbara tidsblock finns utmärkta. Om ett block inte redan är bokat så innehåller det en länk för att boka tiden.

1 Datatal Flexi Tid http://www.datatal.se/index.php/page/product/flexi_tid/ 2012-09-20
2 Datatal demo http://www.datatal.se/index.php/page/tidweb/ 2012-09-20
3 Timecenter demo http://www.timecenter.com/demo-sv/ 2012-09-20

3.2 Flödesstruktur

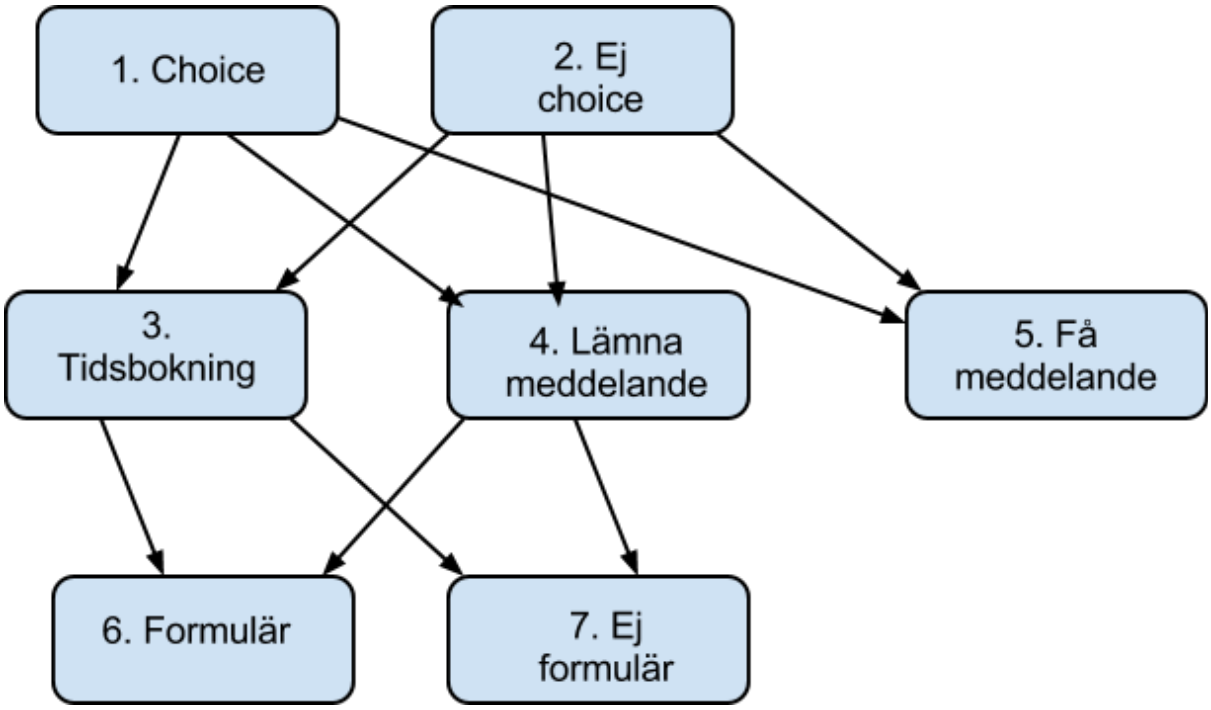

Illustration 2: En första skiss över ett möjligt flödesschema för applikationen

En viktig begränsande faktor i mitt arbete har varit att den gamla applikationen har ett visst flöde som inte kan förändras. I telefonmenyn läses ett antal alternativ upp för Kunden som sedan kan välja ett alternativ genom att trycka på en av telefonens knappar. Denna meny motsvaras av en platt lista på alternativ i applikationen. För att kunna spegla allt det som går att göra via telefon så behövs en mer flexibel applikation som klarar av flera olika uppgifter beroende på de val som kunden gör för sitt system. Efter att ha undersökt vilka förväntningar och möjligheter som fanns gällande slutresultatet av en kunds interaktion kristalliserade sig de tre möjligheterna benämnda (3), (4) och (5) i bild 2 ovan. Då hela syftet med applikationen är att tillhandahålla kommunikation mellan två parter, närmare bestämt Kunden samt Handläggaren, finns det exakt tre möjliga fall hur denna kan ske:

1. Genom en dialog. I alternativ (3) ovan bokar Kunden en tid för en dialog (telefonsamtal)

2. Från Kund till Handläggare. I alternativ (4) ovan lämnar Kunden ett meddelande till Handläggaren.

3. Från Handläggare till Kund. I alternativ (5) ovan får Kunden ett meddelande från Handläggaren.

Detta förfarande kan även föregås av en meny där Kunden specificerar sitt ärende, eller i det enklaste fallet, av ingen meny alls eftersom tjänsten då bara har ett möjligt val. Detta illustreras av alternativ (1) och (2) som anger om en meny (kallad “Choice”) används eller inte. Till sist kommer Kunden i fallen med Tidsbokning samt Meddelandelämning till ett formulär för att ange kompletterande uppgifter, så som eventuellt meddelande som skall lämnas, telefonnummer och/eller e-mail för kontakt, personnummer och/eller namn för identifiering. Dessa är endast exempel på vad ett formulär kan innehålla men beskriver ett normalfall av en implementation.

Efter diskussion kring det här förslaget beslutades det med flexibiliteten i åtanke att modulerna skulle vara helt frikopplade från någon form av fördefinerat flöde och således kunna komma i vilken ordning och antal som helst. Applikationens flöde blir då cirkulärt och potentiellt oändligt:

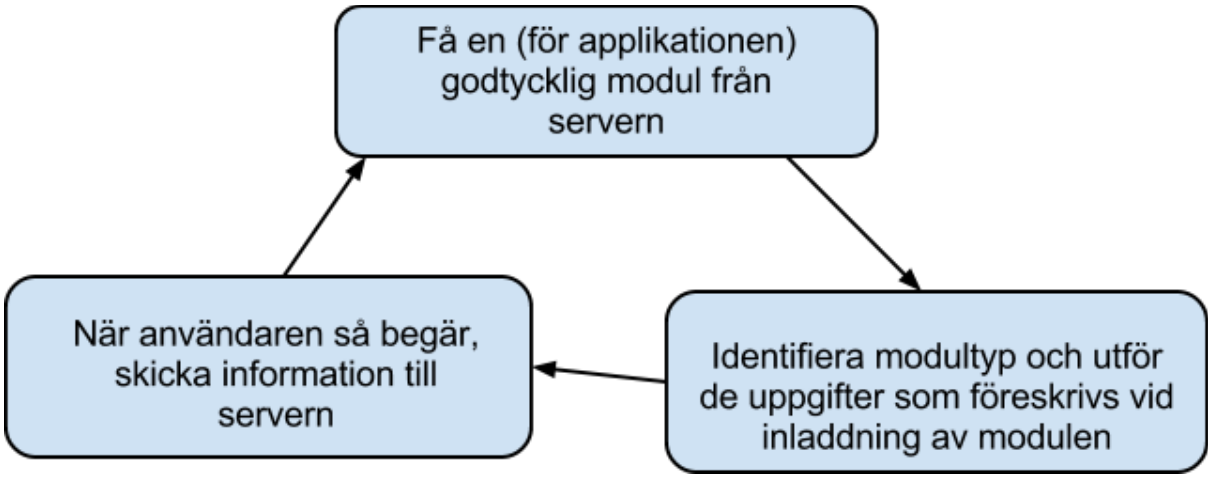

Illustration 3: Idé om flöde

Finessen är att oavsett vilken operation applikationen använder sig av mot servern så kommer svaret i form av en modul. Detta innebär att vid design av flödet för en given implementation kan det helt fritt defineras hur olika händelser ska länkas samman. Det applikationen behöver ha kännedom om är hur den ska reagera på varje given modultyp, vilket inkluderar:

- Visuell representation mot Användaren
- Skriptade händelser knutna till element i representationen
- URI(er), till vilken information ska skickas när Användaren begär någon form av serverinteraktion

Detta och inget annat är vad applikationen utför. Genom att separera all flödeslogik från applikationen undviker man i största möjliga mån att behöva ändra i den vid implementation av nya flöden samt att samma applikation oförändrad kan användas för en mängd olika implementationer.

3.3 Plattform

TeleQ är skrivet i java och använder sig av den javabaserade webbplattformen ZK för alla visuella gränssnitt mot användare. Detta ledde till att jag valde att sätta mig in i för och nackdelar med ZK för att utreda om det vore enklast att implementera webbokningen i samma miljö.

3.3.1 ZK

ZK är ett opensource-ramverk för webbutveckling som ägs och utvecklas av Potix Corporation. Ramverket ger utvecklare som saknar kunskap om traditionell webbutveckling möjlighet att skapa kraftfulla webbplatser med samma tillvägagångssätt som används vid utveckling av klassiska desktop-applikationer.[1]

Ramverket abstraherar kommunikationen mellan användaren och servern till en event-hantering där egenskaper hos element i gränssnittet är kopplade till java-metoder. Något felaktigt marknadsförs ZK därför som “Ajax[2] without JavaScript”, syftandes på det faktum att utvecklaren aldrig behöver se en enda rad JavaScript eller fundera på hur kommunikationen mellan gränssnittet och de metoder som hanterar händelser fungerar. Likafullt så använder sig ZK av JavaScript för kommunikationen mellan användaren och servern.

Licenser

ZK finns i tre olika versioner med olika licensieringsmöjligheter.[3]

- Grundversion, begränsad funktionalitet men open source. Licensierad enbart under LGPL (GNU Lesser General Public License).
- Stegvis mer kompetenta versioner som båda har två licensieringsmöjligheter.
- Kommersiell licens för alla kommersiella implementationer
- ZOK (ZK Open Source License) som är en gratislicens för projekt som är open source eller kompatibla med GPL (GNU General Public License).

3.3.2 jQuery

jQuery är ett JavaScript-bibliotek utvecklat för att förenkla utvecklingen av skript att köras hos klienten (i webbläsaren). Bibliotekets första version utvecklades 2006 av John Reisig som ett svar på ett ökande behov av ett enkelt och kraftfullt verktyg för att hantera den ökande efterfrågan av avancerade skript på klientsidan.[4] I dagsläget används jQuery på

1 ZK Framework http://www.zkoss.org/product/zk 2012-09-20
2 **A**synchronous **J**avaScript **a**nd **X**ML, se avsnitt **4.2.3**
3 ZK Licensing http://www.zkoss.org/license/ 2012-09-20
4 jQuery history http://jquery.org/history/ 2012-09-20

drygt 50% av alla webbplatser och är med en marknadsandel på över 88% det i särklass populäraste biblioteket för JavaScript.[1]

Grunden i jQuery är selektorerna. Det effektiva sätt på vilket man enkelt kan välja ut vilka element i DOM:en[2] man vill arbeta med. De kan filtreras fram utifrån id, klass, element-typ, sin relation till andra element eller baserat på ett eller flera attribut de besitter. Med exakthet kan man så när som helst enkelt påverka vilken del av en sida som helst. Till detta kommer event-hanteringen med en lång rad fördefinerade event-typer som kan bindas till selektorer så att exempelvis en viss funktion körs vilken länk man än klickar på, eller en annan funktion om man håller musen över ett element som har en viss klass.

Till dessa gränssnittsmanipulerande verktyg följer en uppsättning funktioner som förenklar ajax samt synkrona anrop oerhört (jämfört med vanilla JavaScript).

Licenser

Det finns två typer av licenser: MIT eller GPL, beroende på användningsområdet.[3]

GPL har en väldigt stark så kallad *copy-left,* vilket innebär att om programvara licensierad under GPL används på något sätt i en produkt så får produkten inte licensieras under en mer restriktiv licens än GPL. Bland annat så måste alla produkter vara open source, vilket ofta blir problematiskt med kommersiella produkter.

MIT är en tillåtande licens som ger användaren rätten att sälja, modifiera och på annat vis använda den distribuerade programvaran hur som helst så länge licensen inkluderas i alla kopior av programvaran eller verk där en betydande del av programvaran ingår.

3.3.3 Utredning plattformsval

ZK är ett ramverk som lämpar sig väl för större applikationer men för en så pass liten och specifik tillämpning som webbokningen lämpar det sig bättre med en plattformsneutral lösning i JavaScript/jQuery. Med jQuery krävs ingen webbserver med ZK konfigurerat till att hantera webbapplikationen parallellt med API:et utan det räcker med en metod i API:et som levererar filen och hela lösningen kan lätt installeras fristående. En annan fördel är att jag sedan tidigare är bekant med jQuery och detta besparar projektet mycket tid i form av studier av ZK då inlärningströskeln är relativt hög.

3.4 CeHis

CeHis (Center för eHälsa i samverkan) är namnet på en organisation som drivs av alla landsting och regioner i Sverige gemensamt och som har till uppgift att förbättra kommunikationen och samarbetet dessa emellan.[4]

1 jQuery market share http://w3techs.com/technologies/overview/javascript_library/all 2012-09-20
2 **D**ocument **O**bject **M**odel: Plattform/språk-oberoende gränssnitt för att manipulera XML-filers innehåll
3 jQuery licensing http://jquery.org/license/ 2011-10-24
4 CeHis bakgrund och uppdrag http://www.cehis.se/om_cehis/ 2011-11-02

Inom CeHis uppdrag ligger att underlätta patienters kommunikation med vårdgivare och bidra till en ökade möjligheter för patienter att aktivt påverka sin vård. Detta sker genom att bygga gemensamma plattformar för kommunikation mellan olika vårdenheter samt fler och enklare möjligheter för patienter att få en överblick över sin egen vårdsituation samt att komma i kontakt med rätt enhet så snabbt som möjligt vid behov.

Jag har tittat på vilka möjligheter som finns för AIN att anpassa TeleQ till CeHis standarder för att underlätta för kunderna inom vårdsektorn att enkelt integrera TeleQ i sina befintliga plattformar.

3.4.1 Tjänsteplattformen

CeHis tjänsteplattform är en samling virtuella tjänster definerade i tjänstekontrakt som stipulerar hur olika typer av kommunikation mellan enheter kan ske. För att kunna ta del av eller erbjuda en viss sorts tjänst implementerar en vårdenhet tjänsten enligt specifikationen angiven i tjänsteplattformen.[1]

3.4.2 Mina vårdkontakter

För kontakten mellan patienter och vårdgivare har ett antal så kallade Invånartjänster utvecklats, bland andra 1177, som är en webbplats för sjukvårdsupplysning, och Mina vårdkontakter som är en webbportal där invånare med e-legitimation kan autentisera sig varpå de direkt får tillgång till en rad tjänster.[2][3] Exempelvis

- Beställning av kopia av sin egen journal
- Bokning av tider vid vårdenheter
- Receptförnyelse
- Råd
- Byte av vårdcentral

3.4.3 Möjligheter för TeleQ

Integration i tjänsteplattformen är ett omfattande företag oavsett vilken funktionalitet som önskas. Att ge möjlighet för patienter att använda TeleQs webbokning i Mina vårdkontakter är däremot ett relativt enkelt sätt att integrera med de snart rikstäckande lösningarna för e-hälsa. Via Mina vårdkontakter kan en patients identitet (personnummer) säkerställas och en för syftet specialutformad webbokningsimplementation tillhandahållas som erbjuder återuppringningstider hos patientens vårdcentral.

1 CeHis Tjänsteplattform http://www.cehis.se/infrastruktur/tjansteplattform/ 2011-11-02
2 1177 http://www.1177.se 2011-11-02
3 Mina vårdkontakter http://www.minavardkontakter.se 2011-11-02

3.5 Validering

Validering av data som användaren direkt eller indirekt matat in till systemet är av största vikt för att säkerställa att inga säkerhetsluckor uppstår i applikationen samt för att förhindra ett oönskat beteende av densamma. I en situation som den här då applikationen är uppdelad i en klientdel och en serverdel mellan vilka kommunikationen rör sig över internet blir validering av data mer komplext då möjligheten finns att data som skickas mellan klienten och servern manipuleras på vägen. Här följer mina tankar kring valideringsbehovet i mitt specifika fall.

Att validera i klienten skulle kräva mindre prestanda för TeleQ och är enklare att implementera då jag direkt kan ge respons på det användaren matar in istället för att behöva skicka detta till servern och sedan hantera svaret, valideringen skulle i det fallet ske flödesmässigt nära själva inmatningen.

Att däremot validera i serverdelen ger en stor säkerhetsmässig fördel, då datat kan ha manipulerats på vägen. Klientens kommunikation med servern kan, i de fall då klienten befinner sig i användarens våld, påverkas av användaren (eller en tredje part) och på vis kan felaktig data, eller i värsta fall data som utgör en säkerhetsrisk för systemet, injiceras i flödet. Om applikationen i detta fall har förlitat sig på en validering i klientdelen är all validering effektivt satt ur spel och säkerheten i TeleQ satt på spel.

3.5.1 I praktiken

Då projektets applikation består av ett skript som kommunicerar över internet med en server är möjligheterna att injicera oönskat data oändliga, vilket leder till att servern ofrånkomligen måste validera allting den tar emot för att garantera säkerheten.

Om klienten dessutom först validerar all inmatad data innan den skickas till servern kan vi begränsa fallen då servervalideringen borde förväntas hitta ett fel till de fall då någon försökt skicka korrupt data, rimligtvis med avsikt. Felhanteringen i servern behöver därför inte vara särskilt pedagogisk utan kan begränsas till ett generiskt felmeddelande som skickas till klienten att visas för användaren.

3.6 Stilmallar

Då den vanligaste implementationsmetoden för den befintliga webbokningen är i form av en komponent som integreras på kundens webbplats finns ett behov av att kunna anpassa utseendet på denna för att överensstämma med kundens grafiska profil. Målet är en komponent som är flexibel och skalbar i sin funktion och som samtidigt heller inte förutsätter någonting vad det gäller layouten. För att uppnå detta måste alla element som används ha CSS[1]-klasser som dels kan identifiera typen av element (för generella layoutregler som

1 Cascading Style Sheets: Språk för att styra hur innehållet i ett dokument skrivet i ett märkspråk (exempelvis XML) visas för läsaren

exempelvis hur kanten på alla textboxar ska se ut) samt för det sammanhang i vilket dessa används, i det här fallet i vilken modul de ingår.

4 Applikationen

Den färdiga applikationen består av en javascriptbaserad parser för fördefinerade moduler som tillhandahålls i JSON[1][2]-format över ett http-API. I det här kapitlet beskrivs de i skrivande stund definerande modultyperna och deras respektive API-anrop.

4.1 Moduler

I dagsläget finns det 7 olika modultyper definerade. Två av dessa, Contact och ContactCancel är identiskt implementerade men med egna CSS-klasser och namn för att underlätta implementation där olika utseende önskas på det ordinarie kontaktformuläret och det som används för avbokning.

Choice

Choice är ett riktat träd där varje löv motsvarar ett valbart alternativ för Kunden. Trädet visas för Kunden som en trådad meny med kryssrutor iför varje nod. När Kunden kryssar för en ruta visas de noder som finns under den förkryssade noden. Detta fortgår tills dess att Kunden når ett löv i trädet, modulen anropar då API:et och meddelar vad Kunden valde.

Illustration 4: Exempel på användning av Choice-komponent

I bilden syns två nivåer där den över raden motsvarar de två noderna på översta nivån i trädet och den övre raden de två noder som ligger under noden “Tidbokning”, som är förkryssad.

Time

Med hjälp av Time kan Kunden boka en tid, främst för callback (återuppringning från Handläggare) men i teorin för precis vad som helst. Modulen Time kan enkelt utökas med nya versioner om man önskar ett annorlunda beteende.

1 **J**ava**S**cript **O**bject **N**otation: Format för att, på ett för människor läsbart sätt, representera JavaScript-objekt i textform

2 RFC 4627 - http://tools.ietf.org/html/rfc4627

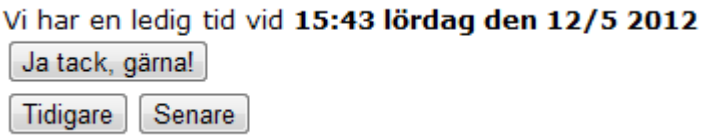

Illustration 5: Exempel på ett tidsförslag med Time-komponenten

Contact

För nästan alla (undantag förklaras nedan) formulärdata som Kunden ska kunna skicka till Handläggaren används modulen Contact. Modulen innehåller en lista på de formulärfält som Kunden ska få tillgång till. För varje fält finns ett antal egenskaper med vilka man kan styra funktionen hos fältet.

Gemensamt för alla fält är de har:

- Ett Id för identifiering mot servern
- Ett felmeddelande som visas brevid fältet
- Ett standardvärde som visas i fältet

De två sistnämnda är valfria och kan med fördel användas i samband med servervalidering då felmeddelanden behöver visas samt att inmatad data ofta ska vara kvar när modulen laddas om.

Illustration 6: Exempel på hur ett formulär kan utformas med Contact-komponenten

Alert

För att visa visa ett popup-meddelande för användaren i form av en JavaScript-alert-dialog med önskat innehåll används modulen Alert. Då det inte finns någon möjlighet för Kunden att interagera med modulen förutom för att stänga dialogrutan bör en tidigare modul ligga kvar i gränssnittet om inte flödet ska stanna.

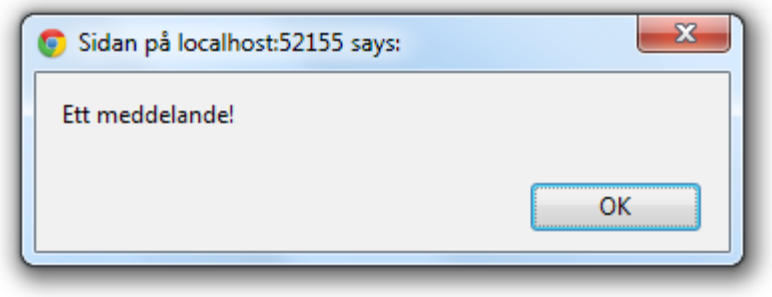

Illustration 7: En alertpopup som visar ett meddelande för användaren

Message

Modulen Message saknar, i likhet med Alert, interaktionsmöjligheter för Kunden och är tänkt att främst användas som sista modul i ett flöde, för att visa upp ett godtyckligt meddelande

för Kunden. Message stöder xhtml och lägger till meddelandeinnehållet i oförändrad form i DOM:en.

ContactCancel

ContactCancel är en modul identisk med Contact men med egna CSS-klasser samt ajax-funktioner för att underlätta implementeringen av en avbokning som är skild från bokningen.

CancelList

Vid avbokningsförfarandet används modulen CancelList för att för Kunden visa en lista på de tider som denne har bokade i systemet. För varje tid finns möjligheten för Kunden att välja att avboka tiden

4.2 Uppdelning

För att underlätta översikten under utvecklingen och för att enkelt kunna lyfta in olika implementationer av de olika delarna i applikationen har jag valt att dela in den i tre olika javascriptfiler baserat på de tre huvudsteg som finns i applikationens cirkulära flöde.

4.2.1 Events

Events är enbart kopplade till CSS-klasser. För att koppla ett element till ett visst event räcker det med att applicera den aktuella klassen på elementet. På detta vis kan ett godtyckligt antal element läggas till och tas bort ur DOM:en och helt dynamiskt göras att använda vissa event som finns fördefinerade.

4.2.2 Parsers

När en ny modul hämtats anropas alltid en och samma funktion som har till uppgift att avgöra vilket slags modul det är samt utifrån detta anropa rätt funktioner för hantering av modulen. Oavsett vilken modul som läses in utförs några handlingar innan bearbetningen av modulen påbörjas.

I container-objektet som modulen kommer i befinner sig två listor med namn på CSS-klasser. I samband med skapandet av den div-tag som ska innehålla alla element som modulen använder sig av tas först alla div-taggar i DOM:en som har klasser representerade i den första listan bort. Därefter appliceras alla klasser i den andra listan på den nyskapade div-taggen. På så vis kan moduler som laddas in sorteras i ett godtyckligt antal grupper för att sedan rensas bort när så önskas.

4.2.3 Ajax

Ajax är en förkortning för det väldigt beskrivande begreppet "Asynchronous JavaScript and XML" och myntades år 2005 av Jesse James Garrett[1] för att beskriva de tekniker hans

1 http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications

utvecklingsgrupp använt sig av för att dynamiskt ladda innehåll till webbläsare. Ajax bygger på XMLHttpRequest-API:et som finns tillgängligt i JavaScript (och stöds av alla aktuella webbläsare på marknaden) som möjliggör att sända HTTP/HTTPS-anrop till webbservern i bakgrunden (asynkront) och att genom JavaScript tolka svaret. På så vis kan innehåll sändas och tas emot när det behagar och utan att användaren behöver ladda en ny sida för att gränssnittet skall kunna uppdateras.

All kommunikation mellan applikationen och API:et sker via Ajax. Alla funktioner som används för att förmedla modulernas kommunikation har följande egenskaper:

- Använder HTTP POST
- Bifogar nödvändig information som url-kodade key-value-pairs
- Förväntar sig ett svar i form av en modul i JSON
- Vid lyckat anrop läses modulen som mottagits in i applikationen

Förutom dessa finns även specialfunktioner som använder sig av GET och bara används till att starta en ny session. Även dessa tar emot en modul enligt samma mönster som ovan. Att alla metoder som hämtar ny information till applikationen endast accepterar moduler och samtidigt accepterar vilken modul som helst är styrkan som gör applikationen enkel att underhålla, översiktig och lätt att felsöka.

5 API

I detta kapitel har jag valt att beskriva API:et för tillämpningen, för att ge en bättre förståelse för funktion och design så att en programmerare lättare kan få en förståelse för hur delar är tänkta att samverka. De API-metoder som beskrivs nedan är de som behövs för att hantera alla funktioner hos de i skrivande stund implementerade modulerna. Om en tillämpning inte är tänkt att använda alla moduler kan givetvis de anrop som hör till dessa oanvända moduler helt ignoreras.

5.1 Datastrukturer

Då applikationen är en “dum” bearbetare av data innehåller datastrukturerna som används i kommunikationen all information som visas för användaren. Datastrukturernas design reflekterar detta och varje modul har sin egen uppsättning datastrukturer för att facilitera de operationer som är möjliga. API:et levererar alla objekt beskrivna nedan på formen JSON.

5.1.1 Container

Varje svar från API:et kommer i form av ett objekt av typen Container som innehåller information relevant för varje svar samt den modul som ska laddas in av applikationen.

```
{
“Type”: string,
“Data”: object
“SessionId”: string
}
```

Type är en sträng som beskriver vilken modultyp som medföljer och är tvungen att ha ett värde som applikationen känner igen, d.v.s. beskriva en redan definerad modultyp. Värden för olika moduler följer vid beskrivningen för respektive modul.

Data innehåller modulobjektet som medföljer.

SessionId sparas i applikationen vid det första anropet och används sedan för att identifiera klienten i alla kommande anrop. Om applikationen redan har initierats med ett SessionId behöver moduler som laddas in senare inte ha något angivet då applikationen ändå kommer att ignorera dessa.

5.1.2 Generellt för moduler

Varje modul innehåller ett antal obligatoriska fält som används i de operationer som är gemensamma för alla modultyper.

```
{
"Type": string,
"OwnClasses": string array,
"RemoveClasses": string array
}
```

Type innehåller samma värde som fältet med samma namn i det övergripande Container-objektet.

OwnClasses innehåller en lista på CSS-klasser som ska läggas till den div-tag (html) som kommer att innehålla alla element som är relaterade till modulen när denna visas för användaren.

RemoveClasses är en lista på de klasser för vilka alla element som har dem tas bort när modulen laddas in. Detta sker innan modulen läggs till vilket innebär att det inte är något hinder att ta bort element med en viss klass för att samtidigt lägga till den klassen på den aktuella modulen. Modulen kommer att visas ändå.

5.1.3 ChoiceNode

Choice är modulen för trädformade menyer. I praktiken består en meny av en samling rekursiva element jag valt att kalla ChoiceNode. Dessa representerar var och en ett menyval och innehåller 0 till oändligt många undermenyval i form av andra ChoiceNode-objekt. Överst i hierarkin finns ett objekt som agerar placeholder för den översta menynivån (som är undernoder till denna) och som aldrig visas för användaren.

```
{
      "Id": string,
      "Title": string,
      "Children": array ChoiceNode
}
```

Id identifierar den enskilda noden och är det som skickas till API:et i det fallet att noden i fråga är en slutnod som väljs av användaren.

Title är den text som visas i menyn för användaren.

Children är en array av de noder som ligger logiskt under den aktuella noden.

5.1.4 Generellt för Time

Objektet Time innehåller en tidväljarmodul samt en eller flera bokningsbara tidpunkter. Då förfarandet vid bokning av en specifik tid kan se ut på många olika sätt är applikationen anpassad för att kunna expanderas med ett godtyckligt antal olika varianter av tidväljarmoduler i framtiden. I skrivande stund är bara en trivial variant implementerad men applikationen förutsätter ändå att typen anges i objektet för att hanteringen ska fungera.

```
{
"TimeChooserType": integer,
"TimeFormat": string,
"Times": array TimeField
"Localization": localization object
}
```

TimeChooserType beskriver vilket slags tidväljarmodul som ska användas och måste korrespondera till en typ som finns definerad i applikationen. I skrivande stund finns bara stöd för typ 1.

TimeFormat är en formateringssträng där formateringen för datumvisningen sker. De variabler som kan användas är:

- “HH” för timme
- “MM” för minut
- “hh” för timme, 12-timmarsklocka
- “dd” för datum
- “mm” för månad
- “yy” för år med två siffor, ex. “11”
- “yyyy” för år med fyra siffror, ex. “2011”
- “weekday” för namnet på veckodagen.

Times innehåller en array av tidpunkter som var och en när de ska visas passerar genom formateringssträngen som definerats ovan.

Localization varierar i utförande beroende på vilken typ av tidväljarmodul som används. I det här objektet finns de variabler som den aktuella tidväljarmodulen behöver för att kunna fungera.

5.1.5 TimeField

En specifik tidpunkt defineras i ett TimeField-objekt som sedan skickas med Time-objektet till applikationen.

```
{
“Hour”: integer,
“Minute”: integer,
“Year”: integer,
“Month”: integer,
“Day”: integer,
“Id”: string,
“Weekday”: string
}
```

Hour är timmen angiven med 24-timmarsklocka.

Minute är minuten.

Year är året angivet med fyra siffror.

Month är månaden angiven med siffror.

Day är dagen i månaden angiven med siffror.

Id är ett unikt id för att identifiera den här bokningsbara tiden. Detta värde skickas till API:et för det fallet att användaren väljer den här tiden.

Weekday är en sträng innehållandes namnet på den aktuella veckodagen (på vilken form och vilket språk man vill!)

5.1.6 Localization

All data som är specifik och relevant för den aktuella typen av tidväljarmodul finns i respektive Localization-objekt. Utseendet som beskrivs nedan är endast ett exempel på ett antal fält som kan finnas med och korresponderar till tidväljarmodul av typ 1.

```
{
“Text1”: string,
“Text2”: string,
“AcceptButton”: string,
```

```
"NextButton": string,

"PrevButton": string,

}
```

I det enkla fallet med typ ett så behövs bara två textsträngar för den beskrivande texten samt texten som ska vara på de olika knapparna.

5.1.7 TimeRequest

När applikationen vill begära en eller flera nya föreslagna bokningstider skickas en Ajax-förfrågan till API:et, innehållande ett TimeRequest-objekt. Detta har stöd för (hittills) två sätt att specificera önskad tid. I exemplet nedan visas först generella fält, därefter fält för respektive sätt att specificera tid.

```
{
    "Type": string,
    //**** TYPE 1**********
    // YYYYMMDDHHmm
    "TimeStart": string,
    "TimeEnd": string,
    "AllowOutsideInterval": bool,
    //[1..]
    "MaxEntries": integer,
    //******** TYPE 2 *************
    // YYYYMMDDHHmm
    "ReferenceTime": string,
    // "after" or "before" or "around"
   "Direction": string
}
```

Type är det enda fältet som alltid måste finnas med i en TimeRequest. Vilka andra fält som följer med beror på värdet av Type.

För typen Type1 finns följande parametrar:

TimeStart anger intervallets startpunkt.

TimeEnd anger intervallets slutpunkt.

AllowOutsideInterval sätts till true om resultat utanför det angivna intervallet önskas i det fall att inga tider inom intervallet hittas.

MaxEntries är det maximala antalet önskade tider.

För typen Type2 finns följande parametrar:

ReferenceTime som anger en referenstidpunkt att söka utifrån.

Direction som anger sökriktningen (framåt eller bakåt i tiden)

5.1.8 Contact

Contact är containerobjektet för Contact-modulen och innehåller den övergripande information som behövs för ett kontaktformulär, samt en lista med de kontaktfält som ingår i formuläret.

```
{
"TopText": string,
"BottomText": string,
"AcceptButton": string,
"MaskDefinitions": keyvaluelist(string, string),
"Fields": array ContactNode
}
```

TopText är den eventuella text som ska visas ovanför formuläret.

BottomText är motsvarande för text som visas under formuläret.

AcceptButton blir texten på knappen som används för att skicka formuläret.

MaskDefinitions innehåller en key-value-lista (array) på de eventuella extra definitioner som behöver skickas till jquery.mask-pluginet.

Fields är en array av ContactNode-objekt.

5.1.9 ContactNode

Varje separat inmatningsfält motsvaras av ett ContactNode-objekt som innehåller information om vilken typ av fält det är och hur det ska gestaltas.

```
{
"Id": string,
"Label": string,
```

```
"ValidationMask": string,
"ValidationMessage": string,
"FieldType": integer,
"Default": string
}
```

Id används för identifiering av fältet när det skickas till API:et.

Label är den beskrivande text som visas i anslutning till fältet.

ValidationMask är bara relevant för input text och textarea-fälten och beskriver hur innehållet i fältet får se ut. Utmärkt att se till att en användare bara matar in korrekta registreringsnummer för bilar och liknande.

ValidationError innehåller om så önskas ett meddelande som visas brevid fältet.

FieldType är en integer som anger vilken sorts fält det är. För tillfället stöds input text, textarea, input checkbox, input password och select.

Default är som namnet antyder ett förvalt värde som visas i fältet. För checkbox innebär detta att om värdet på Default är "true" så är den iklickad från början.

5.1.10 Message

Message är en modul för att visa upp ett textmeddelande för användaren.

```
{
"Text": string
}
```

Text är det meddelande man vill ska visas för användaren. XHTML kan användas och visas upp i oförändrad form.

5.1.11 Alert

Med Alert kan man visa upp ett popupmeddelande för användaren. Användningsområden är alla felaktiga val som användaren gjort och som servervalideringen reagerat på. Istället för att skicka nästa relevanta modul till applikationen så skickas ett popupmeddelande.

```
{
"Text": string
}
```

Text är den text man vill ska visas i popuprutan.

5.1.12 ContactCancel

Funktionsmässigt identisk med Contact är ContactCancel-modulen separerad av den anledning att den representerar ett formulär som används för att identifiera en användare som vill avboka en av sina tider i systemet. Då en separat modul används kan separata API-metoder för hantering av avbokning specificeras. Givetvis finns även möjligheten att använda samma metoder som för Contact och att genom sessionens id identifiera datat som rörande en avbokning. Precis som med Contact kan ett godtyckligt antal inmatningsfält av olika slag användas.

5.1.13 CancelList

För att en användare ska kunna avboka en av sina bokade tider används modulen CancelList som visar en lista över bokade tider tillsammans med en möjlighet att avboka varje tid separat.

Objektet CancelList är containerobjekt till listan.

```
{
      "ButtonText": string,
      "ConfirmText": string,
      "Fields": array CancelListNode
}
```

ButtonText anger den text som visas på de knappar som används för att avboka en given tid.

ConfirmText visas i en OK/Avbryt-popup i vilken användaren måste välja “OK” för att bekräfta att en given tid ska avbokas.

Fields är en array med tider i form av CancelListNode-objekt.

5.1.14 CancelListNode

CancelListNode motsvarar en bokad tid i systemet. Objektet innehåller information om hur tiden ska visas för användaren vid avbokningsförfarandet samt ett id för att kunna kommunicera till API:et vilken tid som ska avbokas.

```
{
      "Id": string,
      "Label": string
}
```

Id identifierar tiden som ska avbokas när API:et anropas

Label är den visuella representationen av en visst tid för användaren.

5.2 Metoder

Vilka metoder som finns tillgänliga beror på vilka moduler som finns definerade i applikation och API. I skrivande stund finns de som listas nedan definerade..

NewSession

NewSession accepterar GET utan parametrar. Vid anrop returneras den första modulen i flödet. Viktigt här är att SessionId är angivet i modulen så att applikationen kan identifiera sig vid nästa anrop.

ChooseCategory

Det enda anropet från Choice-modulen är ett POST-anrop till ChooseCategory, som sker när användaren valt ett alternativ. Bifogar alternativets id som argument.

AcceptTime

Time-modulernas gemensamma sätt att välja en föreslagen tid är metoden AcceptTime. Den accepterar den föreslagna tidens id som argument i ett POST-anrop, ett förfarande som kan ändras till att inte kräva id i de fall endast en tid åt gången är föreslagen och systemet baserat på SessionId kan avgöra vilken tid som föreslagits.

ChangeTime

ChangeTime accepterar POST med ett TimeRequest-objekt. Returnerar vanligtvis en Time-modul som representerar en ny tid, men kan givetvis returnera vad som helst.

SubmitForm

SubmitForm tar emot ett Contact-formulär via POST och hanterar innehållet. Vid eventuella felaktigheter så är det normala beteendet att samma Contact-formulär returneras med defaultvärden satta till det som användaren angivit samt med felmeddelanden motsvarande det som blivit fel.

SubmitCancelForm

SubmitCancelForm är identisk med SubmitForm. Så länge inget annat beteende önskas för avbokningen jämfört med det vanliga formuläret kan samma ingång i API:et anropas från applikationen. Skillnaden är återigen endast för att förenkla en åtskillnad mellan bokning och avbokning.

SubmitCancel

När en användare som vill avboka en tid har identifierats och de tider som användaren har bokat har hämtats ur systemet och visats för användaren kan denne välja att avboka en av dem åt gången. Vid en sådan avbokning anropas SubmitCancel med en POST innehållandes information om vilken tid som ska avbokas.

6 Slutsatser och Framtid

Hela styrkan ligger i flexibiliteten.

I början av utredningen försökte jag upprepade gånger resonera mig fram till ett visst flöde eller en viss modell för applikationen. Men i takt med att funderingarna kring de möjligheter som fanns växte till att inkludera fler och fler möjliga tillämpningar så reviderades modellen gång på gång i riktning mot en mer generell modell. När jag slutligen insåg att de framtida önskemålen för tillämpningar inte skulle bli överskådliga för mig under projektets gång valde jag att inrikta mig på att göra applikationen så modulär som möjligt för att i största möjliga mån underlätta vidareutveckling vid framtida behov. Samtidigt valde jag att helt släppa alla förutfattade meningar om hur de initiala moduler som jag utvecklat i det här projektet ska komma att användas tillsammans och lät detta vara helt upp till varje tillämpning att enkelt välja.

Den största utmaningen har legat i att undvika att måla in mig i ett hörn. Att undvika att begränsa möjligheterna för ett oförutsett framtida användningsområde för applikationen. Hur väl jag lyckats med den föresatsen får framtiden utvisa.

Mitt misstag i början var att tänka för mycket i termer av att replikera de funktioner som en telefon kunde tillhandahålla (med vissa enklare utökningar) istället för att med ett blankt papper sätta mig från början för att ta reda på vilka möjligheter webben erbjuder.

7 Källor

Majoriteten av mina skriftliga källor för projektet finns på webben och förändras i de flesta fallen med tiden då den kunskap jag behövt hämta utifrån till största delen består av teknisk dokumentation som ständigt aktualiseras. Angivet datum nedan anger när jag första gången besökte en webbplats i syfte att söka information rörande det här projektet.

7.1 Litteratur

Cockburn, A. (2002). *Agile software development*. Boston: Addison-Wesley

Crinnion, J. (1991). *Evolutionary systems development: a practical guide to the use of prototyping within a structured systems methodology*. London: Pitman

Gulliksen, J. & Göransson, B. (2002). *Användarcentrerad systemdesign*. Lund: Studentlitteratur

Rutter, J. (2010). *Smashing jQuery*. Hoboken: John Wiley & Sons, Ltd.

7.2 Webb

Datatal

Datatal är företaget bakom Flexi Tid

http://www.datatal.se/ 2011-10-15

Timecenter

Timecenter är nicmar medias tidsbokningssystem

http://www.timecenter.com/ 2011-10-15

ZK

Javaramverk för ajaxdrivna webbplatser

http://www.zkoss.org/ 2011-10-25

jQuery

Javascriptramverk avsett att förenkla utveckling på klientsidan i webbmiljö

http://jquery.org/ 2011-10-24

jQuery docs

Teknisk dokumentation av jQuery

http://docs.jquery.com/Main_Page 2011-10-24

W3Techs

Statistik över användandet av olika javascriptramverk

http://w3techs.com/technologies/overview/javascript_library/all 2011-10-20

TeleQ

Information om TeleQ på Aurora Innovations hemsida

http://www.ain.se/sv/products/teleq 2011-10-10

CeHis - Center för eHälsa i samverkan

Sveriges regioner och landstings samarbetsorgan för eHälsa

http://www.cehis.se/ 2011-11-02