Detta är den svenska titeln

JULIUS RECEP COLLIANDER CELIK

Civilingenjör Informationsteknik
Datum: 17 april 2018
Handledare: Patric Dahlqvist
Examinator: Anders Västberg
Uppdragsgivare: LS Elektronik AB
Engelsk titel: This is the English translation of the title
Skolan för elektroteknik och datavetenskap

Sammanfattning

Svensk sammanfattning.

Abstract

English abstract.

Innehåll

Kapitel 1

Introduktion

Examensarbetet handlade om att undersöka möjligheten att skapa en modell för att beskriva och annotera redigerbar data, och sedan automatiskt generera ett användargränsnitt.

1.1 Bakgrund

Det här kapitlet beskriver bakgrunden till varför och i vilka ämnesområden arbetet utfördes, samt för att ge en förståelse för resten av Introduktionen. Kapitel 1.1.1 beskriver JSON och JSON Schema vilket är en stor del av teknikerna som arbetet undersökte. Kapitel 1.1.2 beskriver företaget arbetet utfördes hos, samt systemet som arbetet utvecklades mot.

1.1.1 JSON och JSON Schema

JavaScript Object Notation *(JSON)* är ett texbaserat dataformat för att utbyta data mellan webbtjänster. Till skillnad mot andra alternativ, som exempelvis XML, är det både läsbart för människor och datorer, samtidigt som det är väldigt kompakt, vilket är en anledning till att det är ett av de mest populäraste dataformaten för datautbyte mellan webbtjänster. [1] JSON utvecklades med inspiration till språket ECMAScript *(JavaScript)* men samtidigt programmeringsspråksoberoende, vilket lett till att implementationer för att generera och parsa JSON finns tillgängliga i många olika programmeringsspråk [2]. ECMAScript är ett språk som stöds av alla moderna webbläsare, och har därför blivit en kärnteknik för webben.

Trots att JSON är det populäraste dataformatet för datautbyte mellan webbtjänster saknas det ett väletablerat standariserat ramverk för metadata-definition [1]. En väldigt lovande formell standard är JSON Schema, vilket är ett ramverk som fortfarande utvecklas av Internet Engineering Task Force *(IETF)*. JSON Schema är ett ramverk för att beskriva och annotera JSON-data [3]. Kapitel 2 beskriver JSON och JSON Schema i mer detalj.

1.1.2 Mimer SoftRadio

Arbetet utfördes hos LS Elektronik AB *(LSE)*, som är ett tekniskt företag, som utvecklar och tillverkar elektroniska produkter [4]. LSE erbjuder bland annat ett radiosystem som heter Mimer SoftRadio vilket kan användas för att ansluta ett flertal annars inkompatibla radioenheter i ett och samma system, samt fjärstyra radioenheterna från en persondator med ett klientprogram. I resten av rapporten kan datorn med klientprogrammet kallas operatörsdator, där användaren kan kallas operatör.

Mimer SoftRadio är ett program med väldigt många möjliga inställningar. I många fall är dessa inställningar för komplexa för de vanliga operatörerna att redigera själva, så därför brukar vissa kunder låsa redigeringsmöjligheterna, och bara tillåta vissa administratörer att ställa in alla inställningar på rätt sätt. Det finns också kundfall där flera operatörer använder samma dator, vid olika tidpunkter. Ett förekommande kundfall är att en operatör jobbade dagtid med att leda och organisera dagsarbete, medan en annan operatör tar över nattskiftet för att övervaka många fler radioenheter.

För att förenkla dessa två kundfall påbörjade LSE utvecklingen av funktionalitet som skulle erbjuda användare att spara uppsättningar av inställningar i olika *profiler*. Det skulle gå att enkelt byta mellan flera förinställda konfigurationer av Mimer SoftRadio. För att konfigurera dessa profiler skapades ett administratörsprogram, som skulle kunna fjärkonfigurera profilerna hos operatörsdatorerna. Fjärstyrningen skulle underlätta administratörer att ställa in profiler på flera datorer samtidigt, som sannolikt skulle innehålla liknande inställningar.

1.2 Problemområde

Systemet för att konfigurera profiler, som beskrevs i kapitel 1.1.2, bygger på att alla operatörsdatorer exponerar ett API mot en server, över en TCP-port. Administratörsprogrammet skulle erbjuda ett användargränsnitt för att konfigurera profilinställningar, för att sedan kommunicera ändringarna till operatörsdatorerna. I resten av rapporten kan operatörsdatorerna och administratörsprogrammet kallas server respektive klient.

Kommunikationsprotokollet var ett egetskapat protokoll som byggde på att skicka JSON-Objekt via TCP. För en beskrivning av JSON, se kapitel 2.1. För en mer utförlig beskrivning av kommunikationsprotokollet se Appendix A. Problemet som LSE hade inför utvecklandet av användarprofilerna var skapandet av ett användargränsnitt på administratörsprogrammet. Olika operatörsdatorer, hos olika kunder, kunde ha olika versioner av Mimer, med olika funktionalitet tillgänglig, och därmed olika uppsättningar konfigurerbara inställningar.

Det vore orimligt kostsamt för LSE att skapa ett administratörsprogram för varje version av Mimer, då både Mimer ändrades med tiden, samt att olika kunder köpte till extra funktionalitet. Samtidigt behövde användargränssnittet på administratörsprogrammet anpassas så att det skulle vara tydligt vad en administratör kunde konfigurera. Helst skulle ett administratörsprogram fungera bra med framtida versioner av Mimer, utan några eller utan stora justeringar av programmet. LSE ville helt enkelt att servern kommunicerade tillgängligga inställningar, till klienten så att klienten sen skulle kunna anpassa sitt användargränsnitt, och det skulle ske på ett tillräckligt generellt sätt att administratörsprogrammet var framtidssäkert för framtida version av Mimer.

1.3 Problem

Vilka svårigheter finns det med att använda JSON Schema för att automatisk generera ett användargränsnitt, är det möjligt samt hur generella JSON Scheman går det att använda sig av?

1.4 Hypotes

En förutsatt hypotes sattes innan arbetet påbörjades. Hypotesen var att det borde vara möjligt att automatiskt skapa ett användargränsnitt från ett JSON Schema, men för att skapa ett användarvänligt användargränsnitt var det nödvändigt att klienten utvecklades för ett JSON Schema som delvis följde en förutbestämd struktur.

1.5 Syfte

Syftet med tesen är att systematiskt analysera problemen med att försöka skapa automatiskt genererade användargränsnitt utifrån olika JSON Scheman. Målet är att föreslå både en strukturell modell samt en metod för att lösa detta problem.

Syftet med arbetet är att med hjälp av JSON Scheman skapa ett dynamiskt användargränsnitt som anpassade sig efter syfte. Utan att uppdatera administratörsklienten ska samma administratörsklient kunna konfigurera inställningar hos olika datorer med olika versioner av Mimer SoftRadio, och därmed olika uppsättningar konfigurerbara inställningar. Det skulle inte bara innebära stark kompatibilitet utan också framtidssäkerhet hos LSEs produkter.

1.6 Mål

Målet med arbetet är att kunna skapa en grund för användare av Mimer SoftRadio att enkelt kunna konfigurera inställningar, oavsett version eller uppsättning extra funktionalitet. Det skulle kunna innebära att Mimer SoftRadio blir ett bättre verktyg för många potentiella kunder. Samtidigt finns det ett mål med att utforska samt att metodiskt utvärdera och beskriva hur användargränsnitt för att redigera data, automatiskt kan genereras.

1.6.1 Samhällsnytta, Etik och Hållbarhet

Ur ett samhällsnyttigt och etiskt perspektiv kan en implementation av fjärrstyrda användarprofiler i Mimer SoftRadio innebära effektivisering av samhällsnyttiga funktioner. Mimer SoftRadio används bland annat av polis, ambulans, brandkår, kollektivtrafik och internationella

flygplatser. Fjärrstyrda användarprofiler skulle hos befintliga kunder i många fall innebära effektivare arbete. Som följd av detta går det att argumentera för att det leder till effektivare kommunikation för organisationer som använder Mimer SoftRadio. Då dessa organisationer arbetar med säkerhet i samhället, räddandet av liv, och upprätthållandet av ett effektivt samhälle, lider hela samhället när dessa organisationer inte kan kommunicera ordentligt. Det är därför väldigt etiskt försvarbart att arbeta med att effektivisera arbetet hos dessa organisationer.

1.7 Risker

En ekonomisk risk är risken som alltid finns vid all hantering av data. Om någon datahantering skulle bli fel, och data skulle försvinna, skrivas över eller bli korrupt, så måste kunder tillägna tid åt att återskapa datan. Därför är det viktigt att implementera ett robust system som är delvis feltolerant. Ingen data som kommer hanteras kommer vara kritisk, och kommer vara relativt enkel att återskapa. Problemet blir att det skulle innebära en kostnad att behöva skapa profiler igen och ställa in inställningar igen, och utöver det så skulle korrupta filer till och med kunna innebära att Mimer SoftRadio inte går att använda alls.

1.8 Metodval

Arbetet som ska utföras är till viss del en fallstudie, men samtidigt ska den utforska något nytt och med det föreslå en ny modell. Designorienterad forskning *(Design science research)* är den metod som passar bäst för den här sortens arbete och därför har den metoden valts. Arbetet följde de följande stegen:

1. **Medvetenhet** En beskrivning av problemet som ska lösas med modellen.
2. **Förlsag** Förslag på lösning presenteras.
3. **Utveckling** Modellen utvecklas.
4. **Utvärdering** Modellen utvärderas. Lyckades modellen lösa problemet beskrivet i *Medvetenhet*?

5. **Sammanfattning** Dra slutsatser

1.9 Avgränsningar

En viktig avgränsning är att rapporten endast väldigt ytligt kommer undersöka olika användargränsnitt, och användbarheten hos dem. Arbetet handlar inte primärt om användbarhet, utan arbetet handlar i större grad om hur JSON Schema kan automatisera skapandet av användbara gränsnitt. Med hjälp av kunskapen som arbetet presenterar kan användbara användargränsnitt enklare skapas.

Ett annat ämne som också är viktigt är säkerhet av systemet som skapas. Säkerheten hos applikationen omfattas inte av arbetet, men det ignoreras samtidigt inte. Systemet som utvecklas för att utbyta JSON Scheman och JSON-data sker över en ssl-krypterad säker uppkoppling. Det här arbetet utvärderar inte säkerheten hos den uppkopplingen.

Att skapa ett användarvänligt användargränsnitt utifrån alla möjliga sorters JSON Scheman med samma verktyg omfattas inte av arbetet. Arbetet kommer utforska olika strategier och metoder för att arbeta med förutbestämda JSON Scheman.

Validering av data är något som webbtjänster ofta måste ta hänsyn till. En klient kan annars skicka otilåten data till en webbserver och därför måste webbservern alltid validera data när den tar emot data, innan data används eller lagras. JSON Scheman fungerar utmärkt för validering av data, men då datan valideras hos klienten, både klienten och servern omfattas av arbetet, samt att användarna inte anses ha uppsåt att förstöra eller falsifiera data, kommer JSON Scheman inte användas för att validera data hos servern.

1.10 Disposition

1.11 Frågeställning

Hur skulle man kunna använda en delmängd av JSON Schema för att dynamiskt anpassa ett användargränsnitt mot olika versioner av apier som erbjuder olika funkionalitet, på ett långsiktigt sätt?

Kapitel 2

Teoretisk Bakgrund OBS BYT UT!

2.1 JSON och webkommunikation baserat på JSON objekt

JSON erbjuder stöd för några enkla datastrukturer: textsträngar *(string)* (`"hej värld"`), tal *(number)* (`4`), ett tomt värde *(null)* (**`null`**), samt booleska värden *(booleans)* (**`false`**). JSON erbjuder dessutom stöd för två komplexa datatyper vilket är vektorer *(array)*, en ordnad lista av JSON-värden vilket visas i Figur 2.1, samt objekt *(object)*, vilket är en oordnad mängd av namn-värde-par, samt visas i Figur 2.2.
Med hjälp av att rekursivt använda *array* eller *object* går det att representera komplexa datastrukturer med hjälp av JSON. Det finns inga begränsningar i hur komplext datastrukturer kan representeras.

2.2 JSON i webbkomunikation

På grund av att JSON är kompakt, enkelt läsbart och har brett stöd hos många språk och implementationer, har JSON blivit väldigt utbrett bland webbtjänster. En hypotetisk förfrågan till en webbtjänst skulle kunna se ut som i Figur 2.3, där en klient förfrågar om de nuvarande väderförhållandena i Stockholm i Sverige. Svaret från webbservern

```
["hej värld", 4, null, false]
```

Figur 2.1: Exempel på JSON-array

```
{
  "firstName": "Erik",
  "lastName": "Andersson",
  "age": 30
}
```

Figur 2.2: Exempel på JSON-object

```
{
  "country": "Sweden",
  "city": "Stockholm"
}
```

Figur 2.3: Exempel på förfrågan till webbserver

skulle kunna se ut som i Figur 2.4 där webbservern svarar att temperaturen är minus tre grader celsius och att det snöar. Exemplet visar hur simpelt JSON som dataformat är att förstå, vilket skulle kunna vara en delvis förklaring för populariteten.

```
{
  "timestamp": "06/01/2018 10:45:08",
  "country": "Sweden",
  "city": "Stockholm",
  "weather": "Snowing",
  "temperature": -3
}
```

Figur 2.4: Exempel på svar på förfrågan från webbserver

2.3 JSON Schema

JSON Schema är ett ramverk för att förklara hur JSON värden kan se ut. JSON Schema specifiserar regler som kan användas för att antingen bestämma om befintliga JSON värden är giltiga, eller för att i förväg beskriva hur gilltiga värden får se ut.

Objektet i Figur 2.2 skulle kunna valideras enligt följande JSON Schema som visas i Figur 2.5. Användningsområden för detta är bland annat:

```
{
  "type": "object",
  "required": ["firstName", "age"],
  "properties": {
    "firstName": { "type": "string" },
    "lastName": { "type": "string" },
    "age": { "type": "integer" }
  }
}
```

Figur 2.5: Exempel på simpelt JSON Schema

1. Validering av data.
2. Annotering av data.
3. Beskrivning av REST APIer.
4. Automatisk generering av kompatibel kod, för att hantera JSON värden beskrivna med JSON Schema.
5. Automatisk generering av API-dokumentation.
6. Automatisk generering av användargränsnitt.

Att använda JSON Schema för användningsområdena 1-3 är trivialt. Det går att utveckla program som kan hantera alla oändligt möjliga permutationer av JSON Schema. Det som däremot inte är trivialt är hur användningsområdena 4-6 skulle kunna generaliseras så pass mycket att ett program eller algoritm skulle kunna hantera vilket giltigt JSON Schema som helst. Användningsområde fyra och fem omfattas inte av den här rapporten, och varför användningsområde sex inte är trivialt diskuteras mer i kapitel XXX.

Kapitel 3

Metodval

OBS! Detta kapitel kan ignoreras. Det är nästan bara egna anteckningar

Dela upp i "hur man brukar göra" och "hur jag faktiskt gör".
qualative data
textual analysis

1. Föreslå ett eget JSON Schema
2. Skapa en JSON Schema genererare. ????
3. Skapa en JSON Schema parser för Delphi. / Undersök befintliga JSON Schema parsers för Delphi.
4. Skapa en direkt mappning mellan JSON Schema och en eller flera datatyp(er) i Delphi
5. Create a dynamic interface based on parsed JSON Schemas.

3.1 Avgränsningar

JSON Schema kan användas till:

- validering av data
- annotering av data
- automatisk generering av kompatibel kod
- beskrivning av REST APIer

- automatisk generering av API-dokumentation

Det kanske går att använda för att automatisera tester?? Det skulle kunna gå att testa att datan ett api ger stämmer överens mot ett jsonSchema, eller att ett api klarar av att hantera all tillåten data.

Dessutom finns det exempel på JSON Schema som automatiskt genereras utifrån kod XXXXX.

Detta projekt intresserar sig enbart för att försöka använda JSON Schema för att annotera data som kan redigeras. Det vill säga beskriva vilken data som kan redigeras, samt hur den kan redigeras. Därför kommer inte all funktionalitet av en JSON Parser implementeras, då det är utanför intresseområdet av rapporten.

Utöver funktionalitet kommer JSON parsern bara förstå en förenklad delmängd av JSON Schema, då vissa egenskaper av JSON Schema inte är eftertraktade. Ett exempel på ej eftertraktade egenskaper nyckelordet `multipleOf` är att kunna specificera att en `number` eller `integer` ska vara en multiple av en annan siffra.

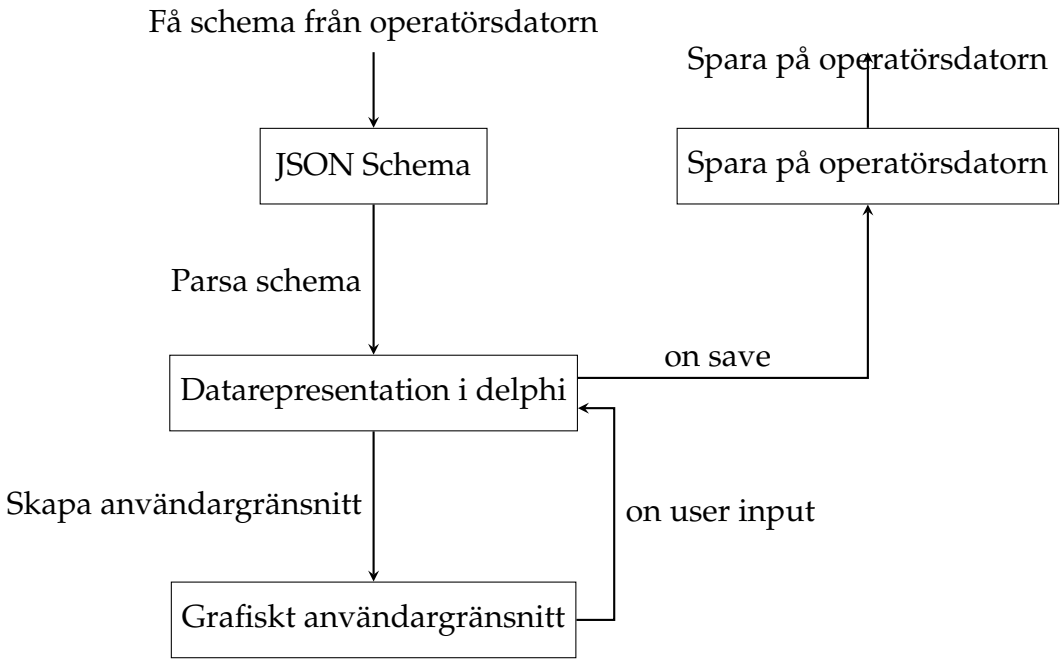
Modellen som rapporten föreslår för att annotera JSON data kommer inte nödvändigtvis vara en strikt delmängd av JSON Schema. Om ej implementerade egenskaper behövs, kan modellen utökas för att inkludera egenskaper som saknas i JSON Schema.

Kapitel 4

Arbetet

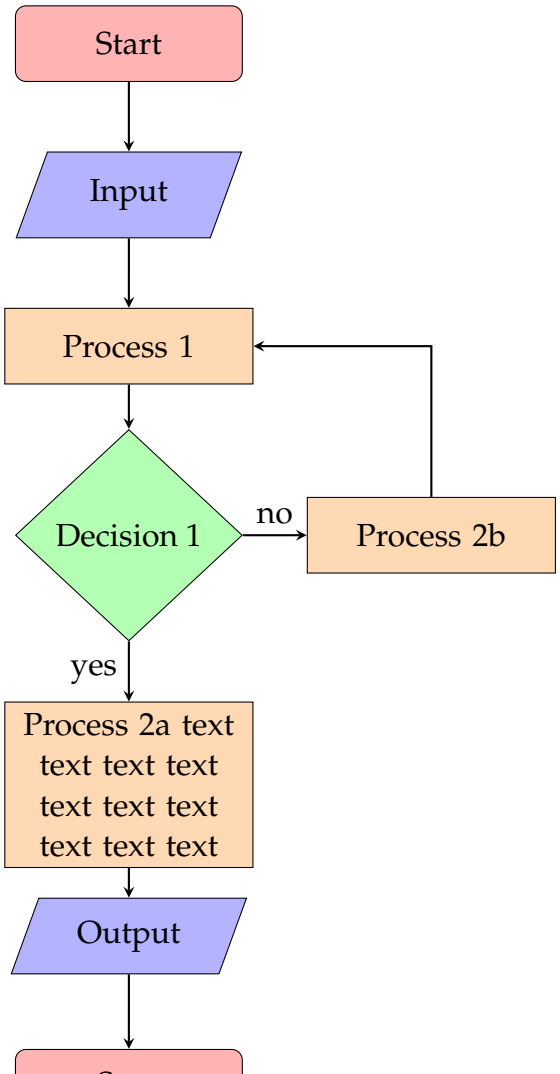
Stort problem är att json och json scheman inte skiljer på heltal och reela tal, medan många språk gör det. Specifikt Delphi som systemet utvecklades i.

Ingen har beskrivit parsning av JSON Schema!!!



4.0.1 Systemet för att översätta JSON Schema till användargränsnitt

4.0.2 Example diagram with TikZ



Kapitel 5

Resultat

5.1 Vad saknas i JSON Schema

Hur hanterar man olika valideringsfel?

Föreslå kanske att JSON Schemat som föreslogs i rapport X ska användas.

5.2 Hur ska schemat genereras?

5.3 Ett schema för att beskriva schemat

5.4 Datarepresentation i Delphi

Många språk skiljer på integer och double men det gör varken Javascript, JSON eller JSON Schema

```
{
  "$schema": "http://json-schema.org/draft-07/schema",
  "type": "object",
  "properties": {
    "TMimerMainSettings": {
      "title": "Mimer Main Settings",
      "description": "The main settings to set.",
      "type": "object",
      "properties": {
        "MyName": {
          "title": "My Name",
          "description":
            "This is the name that will be shown on a place",
          "type": "string",
          "default": ""
        }
      }
    }
  }
}
```

(a) Exempel på enkelt JSON Schema

Här ska det finnas en bild med resultatet

(b) Det resulterande användargränsnittet

Figur 5.1: Exempel på enkelt JSON Schema

5.4.1 JSON Pointer and $ref

5.5 Användargränsnittet

5.5.1 JSON Editor

5.5.2 Generaliserat användargränsnitt

5.5.3 Användargränsnitt utifrån specifika krav på schemat

Kapitel 6

Diskussion, slutsats och fortsatt arbete

Litteratur

[1] F. Pezoa, J. L. Reutter, F. Suarez, M. Ugarte och D. Vrgoč, "Foundations of JSON Schema", i *Proceedings of the 25th International Conference on World Wide Web - WWW '16*, New York, New York, USA: ACM Press, 2016, s. 263–273, ISBN: 9781450341431. DOI: 10.1145/2872427.2883029. URL: http://dl.acm.org/citation.cfm?doid=2872427.2883029.

[2] I. ECMA, "The JSON Data Interchange Format", *ECMA International*, årg. 1st Editio, nr October, s. 8, 2013, ISSN: 2070-1721. DOI: 10.17487/rfc7158. arXiv: arXiv:1011.1669v3. URL: http://www.ecma-international.org/publications/standards/Ecma-404.htmhttp://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf.

[3] H. Andrews, A. Wright och Internet Engineering Task Force, "JSON Schema: A Media Type for Describing JSON Documents", tekn. rapport, 2017. URL: json-schema.org/latest/json-schema-core.html.

[4] R. Ehne, *LS Elektronik About - LS Elektronik*. URL: http://www.lse.se/about/ (hämtad 2018-04-06).

Bilaga A

API-beskrivning

Bilaga B

Extra Material som Bilaga