Rakendustevahelised integratsioonipõhimõtted

Sünkroonne ja asünkroonne suhtlusviis

Rakendused saavad omavahel suhelda **sünkroonselt** või **asünkroonselt**:

Sünkroonne suhtlemine – teenuskutse tegija ootab seni kuni teenusepakkuja poolt on vastus saadaval. Samaaegselt võib toimuda üks toiming:

1. Üks laialdasemalt kasutatud suhtlusstiile;
2. Kontseptuaalne lihtsus võimaldab hõlpsalt rakendada;
3. Enamikes olukordades sobiv suhtlusstiil;
4. Tihedalt seotud HTTP protokolliga, kuid on ka teisi protokolle, nt RPC;
5. Sünkroonse integratsiooni puhul on eelistatud protokolliks HTTP ning andmeformaadiks JSON;
6. SOAP/XML lahendusi kasutada ainult nö olemasolevate süsteemidega liidestamisel, kus muid võimalusi ei ole;
7. Kasutada saab nii X-tee integratsiooni kui ka otse tehtavaid HTTP teenuseid;
8. Kõige levinum stiil HTTP teenuse loomisel on **REST**, kuid on võimalus ka kasutada **GraphQL**-i(ainult läbi eelneva kooskõlastuse)

Asünkroonne suhtlemine ehk sõnumivahetus – teenuskutse tegija ei oota teenusepakkuja poolt vastust. Samaaegselt võib toimuda mitu toimingut:

1. Sobib hästi hajussüsteemi, -arhitektuuri jaoks;
2. Teenused ei ole omavahel seotud, sest suhtlemiseks kasutatakse sõnumisiini, mis võimaldab rakenduste vahel sõnumeid vahetada.
3. Andmevahetusformaadiks võib tekstiliste andmete puhul samuti eelistada JSON-it
4. Andmevahetuseprotokollina on soovituslik kasutada AMQP protokolli

Allikas: Gupta, P. (05.06.2018). Patterns for Microservices – Sync vs. Async. DZone. https://dzone.com/articles/patterns-for-microservices-sync-vs-async

Integratsioon süsteemide vahel (nii välised kui sisemised)

Rakendusi omavahel integreerides peame peamiselt arvestama kolme aspekti. Kas kaks omavahel liidestavat komponenti on määratletud sama andmekogu alla, ehk toimub andmekogu sisene integratsioon või toimub andmekogu ülene integratsioon ning kas üks liidestavatest komponentidest asub välisperimeetris/õues (taotluskeskkonnad).

Andmekogu on see süsteem, mis on määrusega registreeritud, kui on segadus, konsulteeri oma tooteomanikuga.

1. Andmekogusiseseid integratsioone või liidestus lahendusega, mis ei kuulu ühegi andmekogu alla, võib teha nii sünkroonselt kui asünkroonselt ning ei pea kasutama X-TEEd;
2. Andmekogude vahelisi integratsioone tohib teha ainult üle X-TEE ning tulenevalt X-TEE piirangutest, siis täna ainult sünkroonselt (lähtuda tuleb X-TEE dokumentatsioonist);
3. Komponendid, mis asuvad välisperimeetris/õues (igasugu veebid, taotluskeskkonad jms), ei ole lubatud otse liidestuda sisemiste süsteemidega sh. X-TEE-ga (neil tuleb sisevõrku teha vaheadapter, mis päringud edastab);
4. Kui andmekogu soovib andmeid pärida lahenduselt, mis ei kuulu andmekogusse, võib seda teha ilma X-TEEd kasutamata.
5. Kui sisemine süsteem pärib andmekogu käest andmeid, tuleb kasutada X-TEEd.
6. Välist rakendust disainides, tuleks pöörata tähelepanu, et selle mõju sisemistele komponentidele oleks minimaalne (turva, käideldavus, terviklikkus). Eriti arvesse võtta olukordi, kus väline rakendus võib rünnaku alla sattuda.

Sõnumivahetuse nõuded (MQ)

Sõnumivahetuse mudelid

- Lubatud on kasutada vastavalt vajadusele nii publish/subscribe (queue - järjekorrad), kui point-2-point (topic - teemad) mudelit.

Sõnumid

- Kui ei ole määratud teisiti, on nõutud tekst tüüpi sõnumite kasutamine
- *Kui sõnumi sisu on JSON dokument, tuleb seda valideerida kasutades json-schema validaatorit (https://github.com/fge/json-schema-validator) või sellega samaväärset lahendust*
- Binaarkujul failide edastamiseks läbi sõnumite tuleb kasutada tekst tüüpi sõnumit, ning faili sisu lisada Base64 kodeeritud kujul
- Küsimus-vastus (request-response) tüüpi sõnumivahetuses tuleb küsimuse vastust identifitseerida korrelatsiooni identifikaatoriga (correlation-id)

Sõnumiserveri kanalid

- Juhul kui kanalite nimetused ning nende sisu ei ole eelnevalt kokku lepitud, tuleb kasutatavate kanalite arv ning nimetused kooskõlastada SMIT-iga
- Juhul kui sõnumite sisu on XML dokument, tuleb kanalite nimed koos tema sisuga defineerida XML-i kirjeldavas dokumendis (XSD-s)

Sõnumi edastamine

- Kanalite nimetuste nimekonventsioonina järgida üldjoontes malli: "[andmete omanik].[sõnumis olevad andmed]"
    - näiteks: "amr.ametijuhend"
- Küsimus-vastus (request-response) tüüpi sõnumivahetuse puhul lisada suffiksina kanalinimele vastavalt Request või Response - *näide manuses*
    - näiteks "amr.ametijuhendRequest" ja "amr.ametijuhendResponse"
- Sõnumiedastuse ühelt-mitmele (topic) puhul lisada suffiksina kanalinimele vastavalt Topic - *näide manuses.*
    - näiteks "amr.ametijuhendTopic"
- Sõnumiedastuse ühelt-ühele, ilma vastust ootamata (queue) tüüpi sõnumivahetuse puhul lisada suffiksina kanalinimele vastavalt Queue - *näide manuses*
    - näiteks "amr.ametijuhendQueue"

Ühenduskiht

- Juhul kui ole määratud teisiti, käib suhtlus sõnumiserveriga TCP tasemel (vm tase oleks kiire aga mõistlik on sõnumiserver hoida rakendusest väljaspool, et tagada käideldavus)

REST API ja HTTP/JSON API rakenduste teenuste arendamise nõuded

Kontekst

REST on arhiekturistiil, mis kirjeldab nõuded/piirangud rakenduste loomiseks, võttes aluseks veebiarhitektuuri (world-wide-web ehk www)

REST puhul on peamine informatsiooni abstraktsioon ressurss (ingl resource). Mistahes teave, mida saab nimetada, võib olla ressurss. (Fielding, 2000)

1. Ressurss varieerub ajas ning tähistab konkreetsel ajahetkel olemite kogumit või väärtusi (Fielding, 2000).
2. Väärtused võivad olla ressursi esitused (ingl resource representation) ja/või ressursi identifikaatorid (ingl resource identifier).
3. REST-komponendid, teisisõnu omavahel suhtlevad rakendused, teevad toiminguid ressurssidega, kasutades ressursi esitust praeguse või kavandatud oleku fikseerimiseks ja selle esituse ülekandmiseks rakenduste vahel. (Fielding, 2000)

Allikas: Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures. University of California, Irvine. https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

Üldised nõuded API-le

API-del peaks olema loodud dokumentatsioon, mis sisaldab detailset api kirjeldust. Soovituslik on läheneda API-FIRST, ehk api kirjeldatakse ära ja siis luuakse implementatsioon.

Võib kasutada näiteks: https://swagger.io/

API-i juures kasutatavatel andmekomplektidel (eeldatavasti siis JSON dokumendid) peaksid eksisteerima "schema" kirjeldused. Soovituslik on see teha API enda kirjelduse juures, kus saab API väljakutsed lihtsasti siduda "schema" objektidega.

Schemad on vajalikud nii kommunikeerimisel teiste osapooltega, kes API-t kasutavad, kui ka võimaldab luua paremini valideerimisloogikat lähtekoodis.

Ressurside URL-i kujud

1. **/X** - tagastab ressursi X massiivi.
    a. /persons
2. **/X/{id}** - tagastab konkreetse X ressursi id järgi
    a. /persons/1 - tagastab isiku 1
3. **/X/Y** - tagastab X listi, mida on filtreeritud Y järgi
    a. /persons/applications - üldiselt ei kasutata aga võiks tähendada: anna kõik isikud, kus on mõni taotlus küljes
4. **/X/{id}/Y** - tagastab ressursi Y listi, mis on X id-ga seotud
    a. /persons/1/applications - tagastab kõik isiku 1 taotlused
5. **/X/Y/{id}** - tagastab X ressursi listi, mida on filtreeritud Y id järgi
    a. /applications/persons/1 - anna kõik isiku 1 taotlused

{id} ehk id kui path variable-ga identifitseeritakse ressursi

Ressursside nimetamine

1. **Lihtsad ja arusaadavad ressursid vastavalt äriloogikale ja seostele**
    a. Nt /persons/1/applications
2. **Ressursi nimetus mitmuses väljendab, et ressurss on mitmuses**
3. **Ressursside nimetamisel ei kasutata käske (nt /getPersons või /createPerson ei ole õige)**
    a. Ressursi nimetamise asemel väljendavad HTTP käsud/meetodid ressurssidega tehtavaid toiminguid (vt Toimingud ressurssidega)

Toimingud ressurssidega

1. **Ressursidega toiminguteks tuleb kasutada HTTP käske. Nt /persons puhul**
    a. GET /persons - tagastab massiivi kõikidest isikutest
    b. POST /persons - loob uue isiku
    c. DELETE /persons - antud kontekstis vähetõenäoline kasutus, aga kustutab olemasolevad isikud ja nende seosed
        i. Sõltub ärivajadusest, kas rakenduses on otsustatud, et tegemist on loogilise või füüsilise kustutamisega
            1. Mitte kasutada DELETE-i loogiliseks JA füüsiliseks kustutamiseks ilma, et API ressurss väljendaks, millega täpselt tegemist
    d. PUT /persons - antud kontekstis vähetõenäoline kasutus, aga asendab kõik olemasolevad kirjed PUT teenuskutse sisendiga täies mahus
    e. PATCH /persons - antud kontekstis vähetõenäoline kasutus, aga asendab kõikidel olemasolevatel kirjetel PATCH teenuskutse sisendiga antud konkreetsete väljade väärtused

Otsingu tugi API ressurssides

1. **Kui tahta sorteerida, filtreerida, otsida, siis kasutada query parameetrit ehk päringu keerukus viia '?' taha**
    a. Nt GET /persons?name=X&age=Y
2. **Kui API võib tagastada pikki nimekirku, siis tuleb kindlasti kasutada osade kaupa küsimist (ingl *pagination/paging*)**
    a. Nt /persons?limit=25&offset=50
    b. vaikimisi limit=10 ja offset=0

API ressursside/teenuste versioneerimine

API versioonimine on loomulik API muutuste juhtimise osa

1. **Iga kasutuses oleva API muudatus peab olema tagasiühilduv**
2. **Uus versioon tuleb luua nt kui muudetakse API sisendi/väljundi struktuuri; eemaldatakse mõni väli**
3. **Uue versiooni loomisel peab eelmine versioon toetatud olema**
    a. Vanade versioonide eemaldamine eeldab tarbijatega kokkulepet ja vastavat protsessi/reeglistikku

Näide -

1. Versiooninimes kasutada v eesliidet ning väikeste versioonide asemel kasutada suuri versiooni numbreid, nt v1, v2, v3
    a. Versiooninumber tuleb hoida URL-i sees kõige vasakul pool: /v1/persons

Märkus: Kõige lihtsam versioonimine, aga ei ole REST arhitektuuristiiliga kooskõlas

API veahaldus

API rakenduses peab olema lahendatud, kuidas vigu hallatakse ning neid tarbijatele väljastatakse

1. **Luua korrektne veahaldus HTTP koodide põhiselt (seda mõistavad hästi ka teised süsteemid)**
   200 - OK
   400 - Bad Request from client
   500 - Internal Server Error
   304 - Not Modified
   404 – Not Found
   401 - Unauthorized
   403 - Forbidden
   Lisaks veakoodile lisada ka detailsem veateade
   {"status" : "401", "message":"Authentication Required","code": 20003}

API rakenduse kättesaadavus

1. **Kogu rakenduse API võiks kättesaadav olla eraldi domeeninime või kausta alt**
   a. Nt api.rakendus.smit või rakendus.smit/api/

Tuvastamine

Tuvastamist või isikustatud päringute tegemisel, peab identifitseerimist määratleb token (JWT) või vastav räsi minema HTTP päringu päisess (headers).

SOAP teenuste arendamise nõuded

- Liidesed luuakse *Contract-First* põhimõttel kasutades SOAP Document Literal sõnumivahetuse põhimõtet;
- XSD-sse peab saama sisse viia uuendusi, ilma et olemasolevate kasutamine oleks häiritud;

Nõuded XML schemale:

Complex Type definitsioonid:

- tohivad sisaldada ainult *sequence* tüüpi *model group* komponenti kordsustega minOccurs="1" ja maxOccurs="1".
- ei tohi sisaldada atribuutide deklaratsioone.
- liik (*variety*) peab olema *element-only*.

Sequence peab koosnema kas:

- ühest *element* deklaratsioonist kordsustega *minOccurs="0"* ja *maxOccurs="unbounded"*
- nullist või rohkemast *element* deklaratsioonist minimaalse kordsusega *minOccurs="0"* või *minOccurs="1"*, maksimaalse kordusega *maxOccurs="1"* ning lõppema *any* deklaratsiooniga atribuutidega processContents="lax" minOccurs="0" maxOccurs="unbounded" namespace="##any".

> ⚠ Struktuur kus viimane element on *minOccurs="0"* kordsusega ei ole XML Schema 1.0 nõuete alusel *any* deklaratsioon lubatud *Unique Particle Attribution* reegli tõttu. Vastavat reeglit on korrigeeritud XML Schema 1.1 versioonis. Juhul kui tööriistad ei toeta vastavat deklaratsiooni võib selle ära jätta eeldusel et on tagatud selle eesmärk - vastuvõtjale tundmatute elementide lisamine struktuuri lõppu ei põhjusta vastuvõtjas valideerimisviga.

Näide

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://ametnik.smit/services" xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormD
efault="qualified"
xmlns:veeb="http://ametnik.smit/services" xmlns:smit="http://ametnik.smit/smit">

<xs:annotation>
<xs:documentation xml:lang="et">
Näitedokumendi dokumentatsioon (seletav ja kirjeldav tekst)
</xs:documentation>
</xs:annotation>

<xs:complexType name="Ametijuhend">
<xs:sequence>
<xs:element name="isikukood" type="xs:string" />
<xs:element name="failiNimi" type="xs:string" />
<xs:element name="failiSuurus" type="xs:int" />
<xs:element name="failiLaiend" type="xs:string" />
<xs:element name="fail" type="xs:base64Binary" />
</xs:sequence>
</xs:complexType>
```

```
<xs:element name="AmetijuhendRequest">
<xs:annotation>
<xs:documentation xml:lang="et">
Ametijuhendi(te) pärimiseks vajaliku sõnumi element.
</xs:documentation>
<xs:appinfo>
<smit:queue>amr.ametijuhendRequest</smit:queue>
</xs:appinfo>
</xs:annotation>
<xs:complexType>
<xs:sequence>
<xs:element name="asutusId" type="xs:integer" />
<xs:element name="aktiivsedOnly" type="xs:boolean" />
<xs:element name="isikukood" type="xs:string" minOccurs="0" />
<xs:element name="yksusId" type="xs:integer" minOccurs="0" />
<xs:element name="muudetudAlates" type="xs:date" minOccurs="0" />
<xs:element name="muudetudKuni" type="xs:date" minOccurs="0" />
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="AmetijuhendResponse">
<xs:annotation>
<xs:documentation xml:lang="et">
Ametijuhendi(te) päringu vastusena tagstatav sõnumi element.
</xs:documentation>
<xs:appinfo>
<smit:queue>amr.ametijuhendResponse</smit:queue>
</xs:appinfo>
</xs:annotation>

<xs:complexType>
<xs:sequence>
<xs:element name="Ametijuhend" type="veeb:Ametijuhend" minOccurs="0"/>
</xs:sequence>
</xs:complexType>
</xs:element>

<xs:element name="AmetijuhendTopic">
<xs:annotation>
<xs:appinfo>
<smit:topic>amr.ametijuhendTopic</smit:topic>
</xs:appinfo>
<xs:documentation xml:lang="et">
Ametijuhendi(te) uuenemisel publitseeritava sõnumi sisu
</xs:documentation>
</xs:annotation>
<xs:complexType>
<xs:sequence>
<xs:element name="Ametijuhend" type="veeb:Ametijuhend" minOccurs="0"/>
</xs:sequence>
</xs:complexType>
</xs:element>


<xs:element name="AmetijuhendQueue">
<xs:annotation>
<xs:appinfo>
<smit:queue>amr.ametijuhendQueue</smit:queue>
</xs:appinfo>
<xs:documentation xml:lang="et">
Ametijuhendi(te) uuenemisel publitseeritava sõnumi sisu
</xs:documentation>
</xs:annotation>
<xs:complexType>
<xs:sequence>
<xs:element name="Ametijuhend" type="veeb:Ametijuhend" minOccurs="0"/>
</xs:sequence>
</xs:complexType>
</xs:element>

</xs:schema>
```

SOAP teenuste tarbimise head tavad

1. Nt Java puhul dünaamilise keele (nt Groovy) kasutamine SOAP päringute/vastuste konverteerimiseks selle asemel, et WSDL-ist klasse genereerida ning tekitada töömahtu ja pallastkoodi.
2. Näidis SOAP-ist andmete kättesaamiseks: https://stackoverflow.com/questions/39526932/groovy-parse-soap-response-xml-to-get-data
3. Näidis SOAP päringute koostamiseks: https://stackoverflow.com/questions/42290675/groovy-markupbuilder-how-to-create-markup-and-append-string