Opinnäytetyö (AMK)

Tieto- ja viestintätekniikka

2020

Ville Tuominen

POTILASTIETOJÄRJESTELMÄN RAJAPINTOJEN TESTIAUTOMAATIOPROSESSIN LUOMINEN

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tieto- ja viestintätekniikka

2020 | 27 sivua, 1 liitesivu

Ville Tuominen

POTILASTIETOJÄRJESTELMÄN RAJAPINTOJEN TESTIAUTOMAATIOPROSESSIN LUOMINEN

Tämän opinnäytetyön tarkoituksena oli toteuttaa CGI Suomi Oy:lle potilastietojärjestelmän rajapintojen testiautomaatioprosessi. Testiautomaation avulla pystytään nopeasti sekä toistuvasti testaamaan testattavia kohteita. Rajapintatestauksen automatisoinnin tarkoituksena on laajentaa nykyisiä testauksia sekä tukea nykyistä potilastietojärjestelmän testiautomaatiota.

Työssä hyödynnettiin SoapUI -rajapintatestaustyökalua, TortoiseSVN -versionhallintaa sekä Jenkins -automaatiotyökalua. Pääsääntöisesti työssä testattiin HL7 FHIR -rajapintoja, jotka perustuvat REST-arkkitehtuurityyliin. Versionhallintaan luotiin kansiorakenne ympäristöjen sekä testattavien rajapintojen perusteella. Lisäksi luotiin testiautomaatio ja tulosten raportointi Jenkinsiin.

Työn tuloksena saatiin luotua toimiva prosessi rajapintojen testiautomaatiolle, jota pystytään laajentamaan muihin potilastietojärjestelmän rajapintoihin.

ASIASANAT:

testiautomaatio, tietokoneohjelmat, rajapinta, API, SoapUI, TortoiseSVN, Jenkins

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information and communications technology

2020 | 27 pages, 1 page in appendices

Ville Tuominen

CREATING PATIENT INFORMATION SYSTEM APPLICATION PROGRAMMING INTERFACE TEST AUTOMATION PROCESS

The purpose of this theses was to implement patient information system's application programming interfaces test automation process for CGI Suomi Oy. With test automation test cases can be tested much faster and repeatedly. The purpose of automating API testings is to extend existing testings and support current patient information system's test automation.

The used tools in this work were SoapUI API testing tool, TortoiseSVN version control tool and Jenkins automation server. HL7 FHIR are based on REST-arcitecture and they were mainly tested in this theseus. A folder structure was created for version control base on the enviroments and application programming interfaces. Test cases were automated and reporting was created at Jenkins.

As the result a workable process for automating application programming interface testings which can be extended to patient information system's other APIs.

KEYWORDS:

test automation, interfaces, application programming interface, API, SoapUI, TortoiseSVN, Jenkins

SISÄLTÖ

LIITTEET

KUVAT

KÄYTETYT LYHENTEET

FHIR	Fast Healtcare Interoperability Resources, Health Level Seven yhdistyksen kehittämä standardi terveydenhuollon tietojen siirtoon
HL7	Health Level Seven, yhdysvaltalainen terveydenhuollon standardeja kehittävä yhdistys
HTTP	Hypertext Transfer Protocol, Hypertekstin siirtoprotokolla
JSON	JavaScript Object Notation, tiedostomuoto tiedonvälitykseen
JWT	JSON Web Token, JSON-pohjainen autentikointi avain
REST	Representational State Transfer, HTTP-protokollaan perustuva arkkitehtuurimalli rajapintoihin.
SOAP	Simple Object Access Protocol, Tietoliikenneptorokolla rajapintoihin
SVN	Subversion, versionhallintajärjestelmä
XML	Extensible Markup Language, tiedostomuoto

1 JOHDANTO

Testiautomaation avulla pystytään nopeasti sekä toistuvasti testaamaan testattavia kohteita. Sen tarkoituksena ei ole poistaa kokonaan manuaalitestausta vaan tukea sitä. Kattavilla testauksilla saadaan tutkittua, ettei ohjelmasta löydy virheitä ennen kuin se esimerkiksi toimitetaan asiakkaalle tai otetaan omaan käyttöön.

Työskentelin CGI:llä Future Talent harjoitteluohjelmassa, jonka aikana keskustelimme opinnäytetyön aiheesta toimeksiantajan kanssa. Aiheeksi lopulta muodostui potilastietojärjestelmän rajapintojen testiautomaation luominen. Aihetta rajattiin kyseisen prosessin luomiseksi, koska potilastietojärjestelmän rajapintoja on todella paljon, eikä aika riittänyt kaikkien rajapintojen tutkimiseen ja testaamiseen.

Opinnäytetyön tavoitteena on luoda potilastietojärjestelmän rajapintatestaukseen testiautomaatioprosessi. Aiheen tarkoituksena on laajentaa nykyisiä testauksia rajapintatestauksiin. Työssä luotiin muutama testitapaus rajapinnoille SoapUI -rajapintatestaussovelluksella, suunniteltiin kansiorakenne TortoiseSVN -versiohallintaan sekä testien automatisointi ja tuloksien raportointi Jenkinsillä.

Opinnäytetyön alussa käydään läpi keskeisiä käsitteitä. Tämän jälkeen käydään läpi testiautomaatiota, käytettyjä työkaluja sekä toteutuksen rajapintatestausta, versionhallintaa ja testitapausten automatisointia sekä raportointia. Opinnäytetyössä esitetään yleisesti, mitä tapoja rajapintatestauksessa hyödynnettiin potilastietojärjestelmän rajapintojen testauksessa.

Työni toimeksiantajana toimi IT-alan konsultointi- ja ulkoistuspalveluita Suomessa tuottava CGI Suomi Oy. CGI on maailmanlaajuisesti toimiva yritys, jonka on perustanut Serge Godin vuonna 1976 Kanadassa (CGI 2019.). Opinnäytetyössä keskityttiin luomaan pohja potilastietojärjestelmän rajapintatestauksen automatisoinnille. Työssä esitetään olennaista tietoa keskeisistä käsitteistä ja tarvittavista työkaluista sekä tapoja rajapintatestaukseen ja automatisointiin.

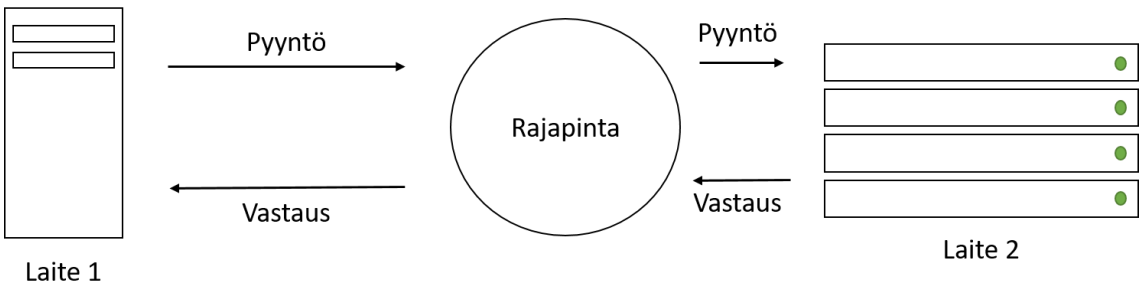
2 KESKEISET KÄSITTEET

2.1 Rajapinta

Rajapinta on ohjelma, joka jakaa ohjelmiston tietoja tai palveluita muille laitteille tai ohjelmille. Rajapintoja on yleisesti kahdenlaisia, pelkkää tietoa jakavia datarajapintoja tai toiminnallisuuksia tarjoavia rajapintoja. (Avoin rajapinta 2019.)

Rajapinnat voidaan jakaa myös ulkoisiin ja sisäisiin rajapintoihin. Ulkoiset rajapinnat ovat kaikkien käyttäjien tai sovellusten käytettävissä, kun sisäiset rajapinnat ovat rajattu ohjelmiston tai kehittäjän käyttöön. Ulkoiset rajapinnat käyttävät yleisesti SOAP- tai REST-tyylisiä www-palveluita. (CGI 2018.)

Rajapinta toimii tiedonvälittäjänä ohjelmiston ja muiden laitteiden tai ohjelmien välillä. Esimerkkinä datarajapinnasta on, kun haet videota Youtubesta hakupalkilla. Rajapintaa pyydetään tietyllä hakusanalla, jolla se hakee tietokannasta ne videot, kanavat ja toistolistat, jotka liittyvät hakusanaan ja palauttaa saadun vastauksen ohjelmistolle tai käyttäjälle. Toiminnallisen rajapinnan esimerkkinä on taas, kun rajapinnan avulla pystytään luomaan, muokkaamaan tai poistamaan tietoja. Kuvassa 1 on esimerkki rajapinnantoiminnasta. Käyttäjä tai ohjelmisto tekee pyynnön rajapinnalle, joka välittää pyynnön toiselle laitteelle, josta saa vastauksen ja välittää sen käyttäjälle tai ohjelmistolle.



Kuva 1. Rajapinnan toimintaperiaate (The Marketing Technologist 2017.) muokattu.

2.2 REST -arkkitehtuurityyli

REST eli Representational State Transfer on arkkitehtuurityyli, joka hyödyntää HTTP-protokollaa ja sen standardimetodeja. Suurin osa www-rajapinnoista perustuu REST-arkkitehtuurityyliin sen suorituskyvyn, keveyden ja helppokäyttöisyyden vuoksi. REST voi käyttää useampaa esitysmuotoa kuten XML, HTML ja JSON, mutta tiedonsiirtotapa on sidottu HTTP:hen. (CGI 2018.)

REST hyödyntää toiminnoissaan HTTP-prokotollan toimintoja, joista käytetyimpiä ovat GET, POST, PUT, DELETE ja PATCH. GET -toiminnolla saadaan tietoa järjestelmästä, kun taas POST lisää tietoa järjestelmään. DELETE poistaa tietoa järjestelmästä ja PATCH päivittää tietoa järjestelmästä. Lisäksi PUT -toiminnolla pystytään myös luomaan uutta tietoa tai korvaamaan tietoa. (Digia 2019.)

2.3 Health Level Seven -standardointiorganisaatio

HL7 eli Health Level Seven International on yhdysvaltalainen terveydenhuollon standardointiorganisaatio. HL7 tarjoaa kattavia viitekehyksiä ja standardeja terveydenhuollon sähköisten tietojen siirtämiseen, integroimiseen, jakamiseen ja hakemiseen. Lisäksi HL7 tarjoaa viitekehyksiä ja standardeja terveyspalveluiden hallintaan, palveluiden toimittamiseen ja arviointiin. HL7:llä on useampia terveydenhuollon tietotekniikan standardeja, joista keskeisimpiä ovat HL7 V2, HL7 V3, CDA ja FHIR. (HL7 2019b.)

Suomessa toimii paikallisyhdistys HL7 Finland ry. Yhdistyksen keskeisenä tavoitteena on edistää tietojärjestelmien kehittämistyötä. Tämän lisäksi se edistää terveydenhuollon tietojärjestelmästandardien käyttöä Suomessa. Yhdistyksessä on mukana joukko tietojärjestelmien käyttäjäorganisaatioita sekä tietojärjestelmien toimittajia, mukaan lukien CGI Suomi Oy. (HL7 2019c.)

Yhdistys tarjoaa käyttäjille rajapintakartan. Sen tarkoituksena antaa käyttäjille tukea sosiaali- ja terveydenhuollon tietojärjestelmien standardeista ja avoimista rajapintamäärittelyistä, joita Suomen alueella on tarjolla. (HL7 2019e.)

2.4 Health Level Seven Fast Healtcare Interoperability Resource -standardi

FHIR eli Fast Healthcare Interoperability Resource on tarkoitettu yksinkertaistamaan ja nopeuttamaan integraatioiden toteuttamista verrattuna aiempiin standardeihin. Standardi on vapaasti verkosta saatavilla ja esimerkit ovat osana standardia. HL7 FHIR:iä käytetään terveydenhuollon tietojen vaihtamisessa. Siinä hyödynnetään sen aiempia standardeja ja yleistä www-teknologiaa. Tähän kuuluvat esimerkiksi HTTP-pohjainen REST-arkkitehtuurityyli rajapinnassa, HTML ja CSS:n käyttö käyttöliittymän integroinnissa sekä tiedon esittämisessä voidaan hyödyntää JSON, XML tai RDF:ä. Lisäksi Atomia käytetään tuloksien esittämiseen. (HL7 2019d.)

2.5 Versionhallinta

Versionhallintajärjestelmä on työkalu, jolla pystytään seuraamaan ja hallitsemaan tiedostojen muutoksia. Versiohallinta auttaa kehittäjiä jakamaan, yhdistämään sekä vertailemaan tiedostoja. Muutoshistoriaan jää jokaisesta muutoksesta merkintä, mitä ollaan muokattu ja kuka on muokannut. Muutoshistorialla pystytään vertaamaan muutoksia tiedostoissa, palauttamaan tiedostoja vanhoista versioista tai jopa palaamaan vanhaan versioon. Muutoshistoria antaa kehittäjille siis hyvät mahdollisuudet kokeilla isompiakin muutoksia. (Atlassian 2019.)

Versionhallintajärjestelmiä on useampia tarjolla, joista yleisempiä ovat Git ja SVN. Git on hajautettu versionhallintajärjestelmä, jolloin käyttäjällä on oma kopio Git:ssä olevasta projektista, sisältäen historian kaikista muokkauksista. Git:n muutokset voivat tapahtua myös paikallisesti, jolloin käyttäjän ei tarvitse aina olla yhteydessä verkkoon. Subversion on keskitetty versionhallinta, jolloin käyttäjällä on ne tiedostot projektista, mitä hän työstää. Käyttäjän halutessa tarkistaa tiedostot ja lisätä muutokset serverille, täytyy hänen olla yhteydessä verkkoon. (Peforce 2017.; Peforce 2018.)

3 TESTIAUTOMAATIO

Testiautomaatio tarkoittaa testien suorittamista automaatiolla. Automaatio käynnistää tarvittavat sovellukset, ajaa testitapauksen ja vertaa annettua odotettua tulosta testitapauksesta saatuun tulokseen. Testiautomaatio ajaa testit ilman ulkopuolista käyttäjää itsenäisesti. (Smartbear 2019.)

Testiautomaatiolla testataan niitä testejä, jotka ovat yksinkertaisia, toistuvia ja aikaa syöviä. Sen avulla testaajalle jää enemmän aikaa testitapauksiin, joita ei testiautomaatiolla pystytä testaamaan. Testaajalle voi sattua testauksen aikana virhe tai hän testaa testattavaa ohjelmaa väärin, jonka takia testi vaikuttaa onnistuneelta tai virheelliseltä. Huolella tehty testiautomaatio testaa yhdenmukaisesti, jolloin inhimillisiä virheitä ei satu ja testejä pystytään suorittamaan ajasta riippumatta. Tämä mahdollistaa yrityksen resurssien tehokkaan käytön sekä varmistaa tuotteen korkean laadun. (Guru99 2019.)

3.1 Testiautomaation edut

Automaatiolla vähennetään aikaa vieviä testitapauksia manuaalitestaajalta sekä varmistaa kriittisemmät testitapaukset. Kriittisillä testitapauksilla tarkoitetaan niitä olennaisia osia ohjelmasta, jotka on hyvä testata aina, kun ohjelmaa päivitetään. Testauksista saadaan laajempia, kun yhdistetään automaatiolla tehdyt testaukset sekä manuaalitestauksella tehdyt testaukset. (Guru99 2019.)

Testiautomaatiolla saadaan nopeasti palautetta ohjelmiston toimivuudesta. Palautteella saadaan tieto, toimiiko ohjelma oletetusti vai onko jokin mennyt pieleen. Nopea palaute saadaan vain yksikkö- ja rajapintatestauksesta. UI- ja systeemitasolla palautteessa kestää kauemmin, mutta yleensä testien suorittamiseen menee vähemmän aikaa, kuin manuaalisesti testattuna. (DevQA 2019.)

Testauksista saadaan laajempia, kun yhdistetään testiautomaatio ja manuaalitestaus. Regressiotestauksessa vaikeammat testitapaukset hoitaa testaaja ja helpommat testiautomaatio. Automaatiolla saadaan myös laajemmin testattua yksinkertaiset testitapaukset Data-Driven testauksella eli ajetaan sama testitapaus useasti eri syötteillä. Näillä varmistetaan, että ohjelma on testattu kattavasti. (SearchSoftwareQuality 2019.)

3.2 Testiautomaation haasteet

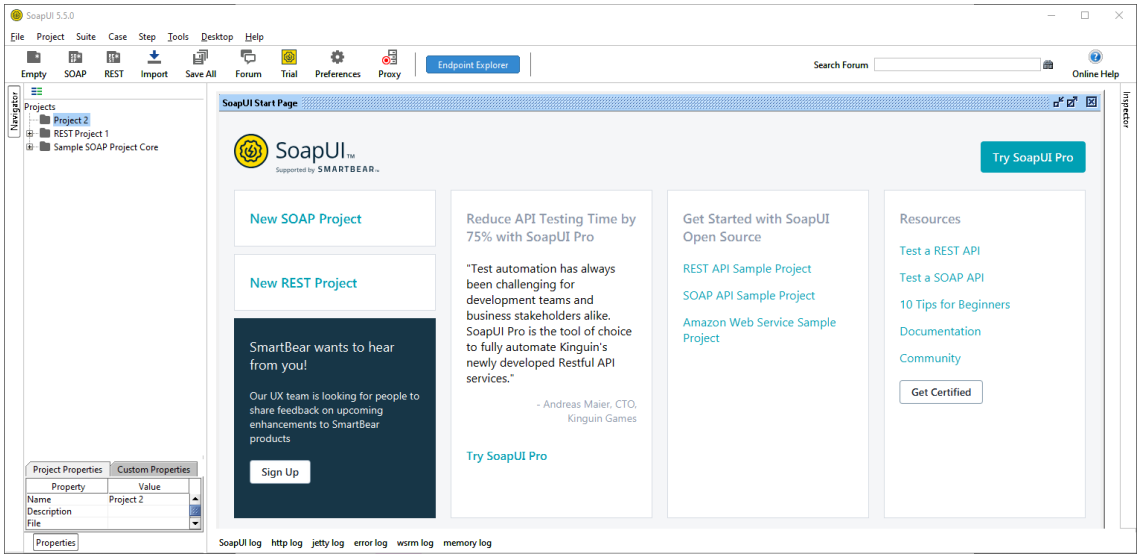
Testiautomaatiolla on omat haasteensa. Se ei osaa testata laajasti eli ohjelma testaa vain, miten se on ohjelmoitu testaamaan. Pienikin muutos testattavassa ohjelmassa voi rikkoa testiautomaation. Testiautomaatiolla ei myöskään pysty kokonaan korvaamaan manuaalitestausta, koska kaikkea ei pysty automaatiolla testaamaan. Helposti ajatellaan, että testiautomaatiolla säästetään paljon aikaa, kun automaatio testaa testitapaukset. Kuitenkin automaatioon pitää luoda testitapaukset ja ylläpitää niitä, jotka vievät aikaa. Testitapausten luominen ei myöskään ole aina nopeaa, sillä automaatio voi joutua useampaan kertaa ajamaan ja todentamaan, että se toimii oikein. Testiautomaatio voi vaatia myös useamman testitapauksen, jotta testattava kohde testataan laajasti. Manuaalitestauksessa testaaja voi tutkivalla menetelmällä testata laajasti ja huomata virheitä. (DevQA 2019.)

3.3 Rajapintojen testiautomaatioon valitut sovellukset

Rajapintojen automaatiotestaamiseen käytetään tässä työssä rajapintatestauksen sovellusta, versiohallintaa sekä automaatioserveriä. Rajapintatestauksen sovelluksena käytetään SoapUI:ta. SoapUI tarjoaa hyvän käyttöliittymän testitapausten luomiseen. Sovelluksella pystyy testaamaan SOAP- ja REST-rajapintoja sekä testitapauksiin pystyy lisäämään väitteitä, joilla tarkistetaan saatu vastaus. Automaatioserverinä käytetään Jenkinsiä ja versiohallintaan TortoiseSVN:ää, joita ollaan jo hyödynnetty CGI:n Welfaren Turun yksikössä testiautomaatiossa.

3.3.1 SoapUI -rajapintatestaustyökalu

SoapUI on avoimen lähdekoodin ilmainen työkalu rajapintojen, kuten REST:n ja SOAP:n sekä HTTP-palvelujen testaamiseen. Sovelluksesta on tarjolla SoapUI Pro -versio, joka tarjoaa ilmaiseen verrattuna enemmän ominaisuuksia. Kuvassa 2 näkyy perusnäkymä SoapUI:sta. (SoapUI 2019b.)



Kuva 2. SoapUI:n perusnäkymä.

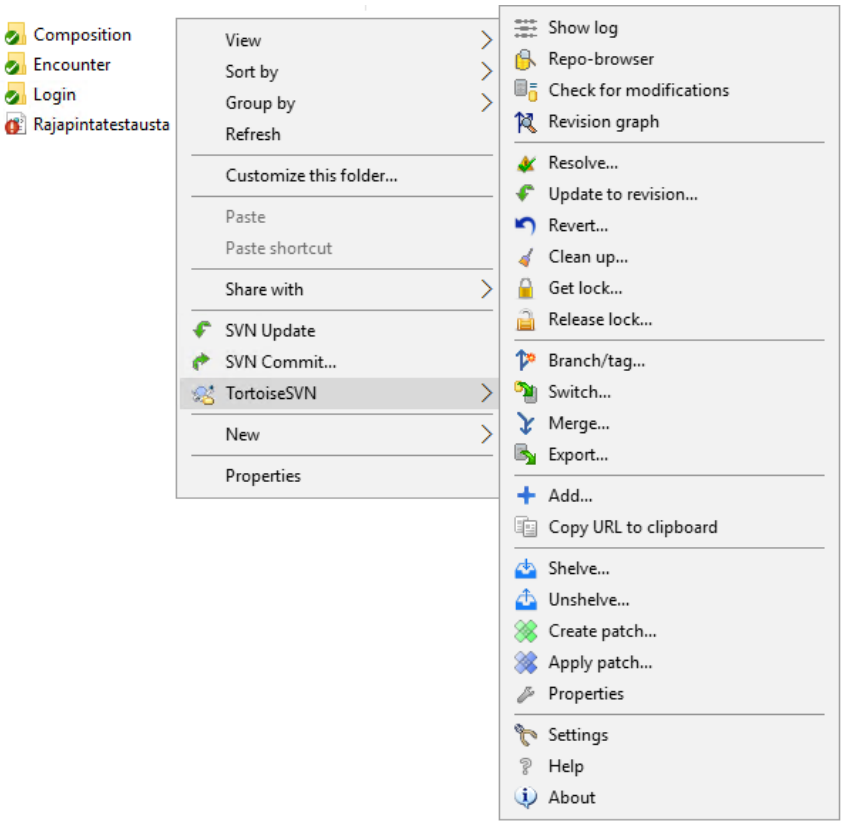
Sovelluksen käyttöliittymä on tehty mahdollisimman helppokäyttöiseksi. Sen käyttö vaatii pientä opettelua, jonka jälkeen käyttö on suhteellisen helppoa. SoapUI:ssa luodaan projekti, johon määritetään www-sovelluspalvelun osoite ja testiohjelmat. Tämän jälkeen www-sovelluspalveluun pystyy lisäämään siihen liittyvät rajapinnat ja testiohjelmaan testitapauksia, johon taas lisätään testivaiheita. SoapUI:ssa pystyy myös lisäämään www-sovelluspalvelun ja siihen liittyvät rajapinnat joko WSDL:lla (Web Service Description Language) tai WADL:lla (Web Application Description Language). WSDL ja WADL sisältävät www-sovelluspalvelun tiedot ja siihen liittyvistä rajapinnoista, joten niiden avulla saadaan nopeasti luotua perusrakenne SoapUI:hin testattavasta www-sovelluspalvelusta. (SoapUI 2019c.)

SoapUI:ssa rajapinnoille lähetetään ennalta määritetty pyyntö, jonka jälkeen se odottaa palvelusta vastausta. Vastauksen saavuttua sitä verrataan ennalta määritettyihin väitteisiin, jolloin tulos on joko hyväksytty tai hylätty. Ennalta määritettyjä väitteitä voi olla maksimaalinen aika pyynnön lähettämisestä ja vastauksen välillä, HTTP-tilakoodit tai määritetty teksti. Samanaikaisesti käytössä voi olla useampia väitteitä. Testitapaukseen voi myös lisätä suorituskyky- ja turvallisuustestejä. (SoapUI 2019a.)

3.3.2 TortoiseSVN -versionhallintajärjestelmä

TortoiseSVN on versionhallintajärjestelmä, joka perustuu Apache Subversion versionhallintaan. Siihen kuuluu myös muutoshistoria eli sovellus pitää kirjaa tiedostoihin ja hakemistoihin tehdyistä muutoksista. Tiedostot on tallennettu serverille, josta käyttäjät voivat tuoda tiedostoja omaan hakemistoon. Muutoshistorian avulla sovelluksessa pystyy palauttamaan vanhoja versioita tiedostoista ja tutkia tiedostojen historiaa. Historiasta näkee kuka käyttäjä on muokannut tiedostoa tai hakemistoa ja millä tavalla. Lisäksi siinä näkyy myös milloin sitä on muokattu. (TortoiseSVN 2019b.)

TortoiseSVN toimii Windowsin resurssienhallinnassa. Käyttäjä klikkaa hiiren oikealla tiedostoon tai hakemistoon, jotta saa valikon. Valikossa näkyy vain tarpeelliset komennot, joita käyttäjä pystyy käyttämään. Resurssienhallinnassa käyttäjän hakemistossa näkyy tiedostojen ja hakemistojen tila. Vihreä hyväksytty merkki kertoo, että tiedostot tai hakemistot ovat ajantasalla, kun taas punainen ruksi tarkoittaa, että tiedostot tai hakemistot ovat muuttuneet tai ne ovat poistettu serveriltä. Kuvassa 3 näkee resurssienhallinnan valikon sekä hakemiston tiedostojen tilan. (TortoiseSVN 2019a.)



Kuva 3. TortoiseSVN -valikko sekä hakemiston tiedostojen tila.

3.3.3 Jenkins -automaatiotyökalu

Jenkins on avoimeen lähdekoodiin perustuva ilmainen automaatiotyökalu. Se on jatkuvaan integrointiin ja kehitykseen tarkoitettu www-pohjainen palvelinsovellus. Sitä voidaan käyttää automatisoimaan kääntämiseen sekä testaamiseen liittyviä tehtäviä. Lisäksi Jenkinsillä pystytään automatisoimaan toimittamiseen sekä käyttöön liittyviä tehtäviä. (Jenkins 2019a.; Edureka 2019a.)

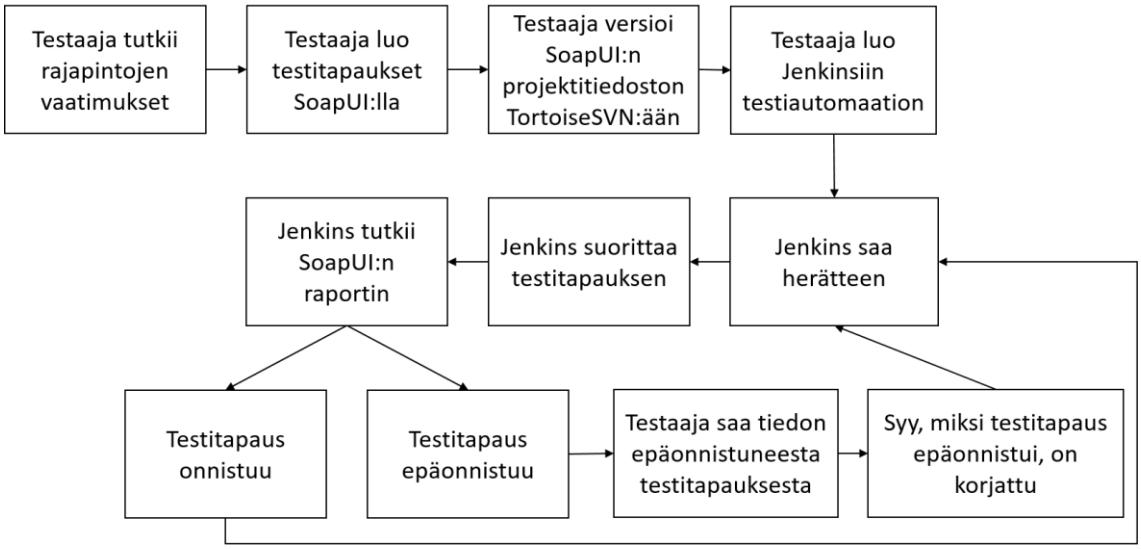
Koko prosessin automatisointiin Jenkins tarjoaa Pipeline:n, jolla pystytään automatisoimaan edellä mainittuja tehtäviä laajennuksien ja Jenkinsfile tekstitiedoston avulla. Jenkinsfile luodaan käyttämällä joko deklaratiivista tai skiprattua syntaksitapaa. Deklaratiivinen Pipeline on uudempi syntaksitapa, joka tarjoaa paremmat syntaksi ominaisuudet sekä se on suunniteltu helpottamaan käyttäjää kirjoittamaan ja lukemaan Pipeline koodia. Skriptattua Pipeline koodia kirjoitetaan Apache Groovy ohjelmointikieleltä käyttäen. (Jenkins 2019b.; Edureka 2019b.)

Jenkinsillä pystytään ajoittamaan tehtävien suoritusta. Tehtäviä pystytään esimerkiksi ajoittamaan silloin, kun tietokoneilla on vähiten käyttöä sekä jakamaan niitä tietokoneiden välillä, jolloin saadaan suurempi määrä tehtäviä nopeammin suoritettua. Tehtävien käynnistämiseen voidaan hyödyntää myös muitakin tekijöitä, esimerkiksi versionhallintaan tulee uusia muutoksia, josta lähtee käsky Jenkinsille suorittaa määritetyt tehtävät.

Jenkinsin tehtävään pystyy määrittelemään jälkitehtäviä. Jälkitehtäviin kuuluu esimerkiksi raportin julkaiseminen. Raportin julkaisemisella Jenkins tallentaa raportin serverille ja muodostaa siitä kaavion onnistuneista sekä epäonnistuineista tehtävistä. Raportista se pystyy myös kertomaan, mitkä tehtävät ovat epäonnistuneet ja miksi. (JavaTpoint 2019.)

4 TOTEUTUSVAIHE

Tässä luvussa käydään läpi prosessia, miten potilastietojärjestelmän rajapintoja testattiin yleisellä tasolla ja luotiin versionhallintaan kansiorakenne. Lisäksi tarkastellaan testitapausten automatisointia ja raportointia. Kuvassa 4 on esitetty testiautomaation prosessin luomisesta.



Kuva 4. Prosessi testiautomaation luomisessa.

Rajapintojen testaamisessa olennaista on tuntea rajapinnan vaatimukset. Vaatimusten perusteella luodaan testitapaukset, joilla varmistetaan rajanpinnan toimivuus. Näitä vaatimuksia voivat olla esimerkiksi tietyssä ajassa vastauksen saaminen tai muita toiminnallisia vaatimuksia.

4.1 Rajapintojen testaus

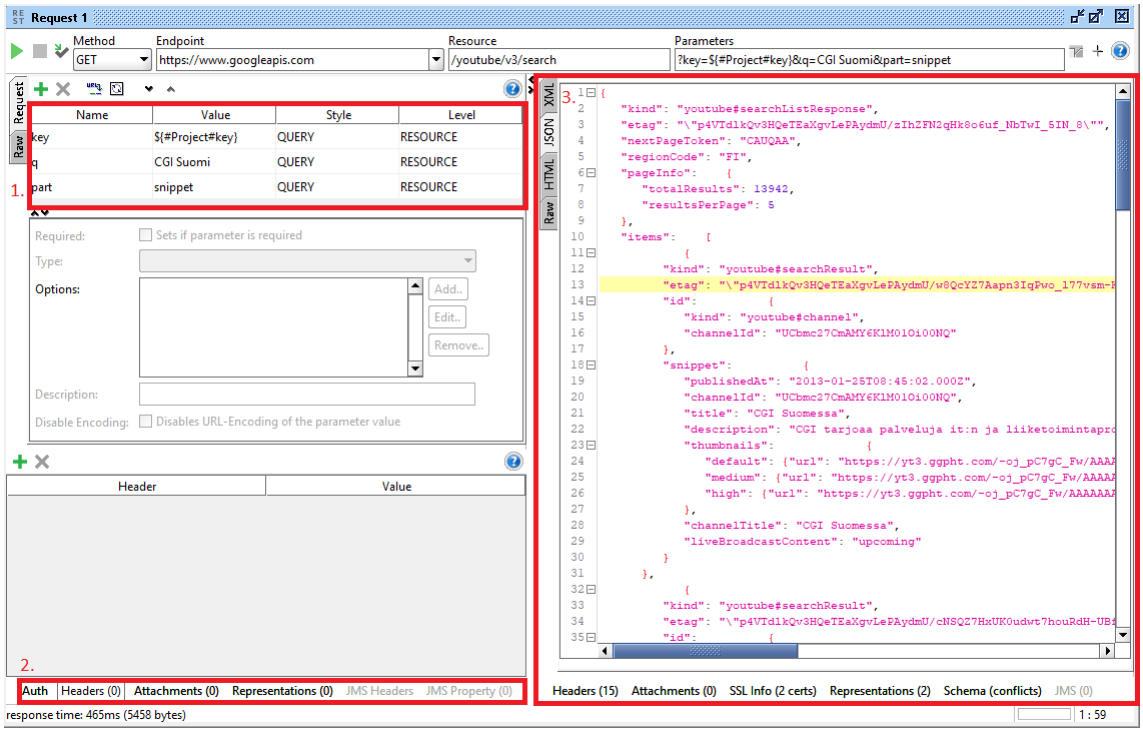
SoapUI:lla aloitetaan testaus luomalla projekti, johon määritetään www-sovelluspalvelimen osoite tai tuodaan WSDL tai WADL-tiedosto, joka määrittää www-sovelluspalvelimen. Pelkän www-sovelluspalvelimen määrittämisellä täytyy itse lisätä sovelluspalvelimen rajapintojen sijainnit SoapUI:hin kun taas WSDL tai WADL voivat sisältää sovelluspalvelimen rajapintojen sijainnit. SoapUI:ssa luodaan testiohjelma, joka voi sisältää useamman testitapauksen. Testitapauksiin luodaan vaiheita, joita ovat esimerkiksi pyynnön lähettäminen rajapinnalle, tiedonsiirto vastauksesta seuraavalle pyynnölle, Groovy Scripti, viivästys tai manuaalinen testivaihe. Näin ollen testiohjelma voisi olla kirjautumisen testaaminen ja testitapaukrsia oikeilla ja väärillä tunnuksilla kirjautuminen.

Opinnäytetyössä testasin HL7 FHIR-rajapintoja, jotka perustuvat REST-rajapintoihin. REST-rajapinnat hyödyntävät HTTP-protokollan toimintoja, joista yleisimpiä ovat GET, POST, PUT, DELETE ja PATCH. Näitä toimintoja hyödynnetään myös SoapUI:ssa REST-rajapintojen testaamisessa. Liitteessä 1 on esitetty ajanvarauksen rajapintapyynnön tietosisältö. Tietosisällössä on HL7 FHIR standardin määrittämät resurssisisällöt, joita ovat esimerkiksi osallistuvat tekijät (potilas, lääkäri, sijainti), ajanvarauksen aloitus- ja lopetusaika, palveluluokka ja vastaanotontyyppi. Lisäksi standardissa on määritetty tyyliopas resurssien sijainneille www-sovelluspalvelimella. Esimerkiksi potilaan tietoja pystytään hakemaan GET-toiminolla ”https://server/path/Patient” sijainnista parametrien avulla. (HL7, 2019f.)

SoapUI:ssa rajapintaan lisätään REST-pyyntöjä, joihon määritetään HTTP-toiminto sekä parametrejä. Parametreihin määritetään nimi, arvo, tyyppi sekä taso. Lisäksi tietosisältöjä pystytään lisäämään niihin pyyntöihin, jotka lähettävät tietoa serverille. SoapUI:ssa on muutama valmiiksi määritetty tietosisältö esimerkiksi JSON ja XML. Käyttäjä voi määritellä myös muun tietosisällön.

Testitapauksissa rajapintapyyntöihin pystyy lisäämään väitteitä, jotka tarkistavat saadun vastauksen. Väitteinä voi olla esimerkiksi tietyn sisällön oleminen vastauksessa, kelpaavat ja kelvottomat HTTP-tilakoodit sekä vastauksen saaminen tietyssä ajassa. Lisäksi käyttäjä voi itse luoda komentosarja, joka tarkistaa vastauksen.

Kuvassa 5 on esimerkki Youtube Data API:n rajapintapyynnöstä. Kuvan merkinnässä 1 on pyynnön parametrit, jotka ovat key, q ja part. Key on autentikointiin liittyvä API Key, jonka Google vaatii rajapintapyynnöissä. Ilman avainta vastauksena saa yleensä ilmoituksen ”Unauthenticated Use Exceeded” eli tunnistautumattomia käyttöjä on tehty liian monta. Pyynnön parametri q on hakusana, jolla haetaan videoita, kanavia tai toistolistoja. Part parametrillä voidaan määrittää, mitä resursseja vastaukseen sisällytetään. Tässä tapauksessa part parametrin arvoksi on asetettu snippet, joka tuo kaikki resurssin ominaisuudet. Kuvan merkinnässä 2 pystyy lisämään muita autentikointitapoja, otsikoita tai liitteitä. Työssäni otsikoihin lisättiin autentikointitapoja, kuten JSON Web Tokeni. Kuvan merkinnässä 3 on rajapinnalta saatu vastaus. Tässä esimerkissä on haettu ”CGI Suomi” hakusanaa ja sen perusteella on saatu hakutuloksia. Rajapintapyynnön vastaus esitetään joko XML-, JSON-, HTML- tai Raw-muodossa. Vastauksen otsikoita, liitteitä, SSL -tietoja voi selata vasemmalta alareunasta. Rajapintapyyntö lisätään testitapaukseen, jossa pystytään määrittämään väitteitä. Sovellus vertaa saatua vastausta annettuihin vätteisiin ja ilmoittaa, onnistuiko vai epäonnistuiko testitapaus. (Youtube Data Api 2019.)



Kuva 5. SoapUI -rajapintapyynnön esimerkkikuva.

Opinnäytetyössä rajapintapyynnön saaduista vastauksista piti siirtää tietoa seuraavaan pyyntöön. SoapUI:ssa pystyy siirtämään muuttujia vastauksesta projektin, testiohjelman tai testitapauksen mukautettuihin muuttujiin. Lisäksi Groovy Scriptillä voi tehdä komentosarjan, joka esimerkiksi hakee vastauksesta tarvittavan elementin, muokkaa sitä ja palauttaa muokattuihin muuttujiin. Rajapintapyynnön parametriin tai tietosisältöön kirjoitetaan mukautettua muuttujaa pyytävä pyyntö (${#project#customPropertieName}).

Opinnäytetyössä tein Groovy Script komentosarjan, joka muutti nykyisen päivän alku- ja loppuajan (00:00 ja 23:59) millisekunneiksi Epoch-ajasta (1.1.1970 00:00:00 UTC+2). Kuvassa 6 komentosarja hakee nykyisen päivän tietyssä muodossa sekä alku- ja loppuaika ovat valmiiksi määritetty. Tämän jälkeen yhdistettiin nykyinen päivämäärä alku- sekä loppuaikaan. Lopuksi molemmat ajat muutettiin millisekunneiksi Epoch-ajasta ja siirrettiin SoapUI:n projektin muuttujiin.

```groovy
/*
This script creates begin date and end date as millis since the UNIX epoch (January 1, 1970 00:00:00 UTC).
Script gets current date, start time and end time are defined.
Current date and start time are combined and converted into millis.
Current date and end time are combined and converted into millis.
Sends startMillis and endMillis to project custom properties.

Author: Ville Tuominen
*/

Date date = new Date()
def datePart = date.format("yyyy-MM-dd")
def startTime = "00:00:00"
def endTime = "23:59:59"
def dateFormat = "yyyy-MM-dd HH:mm:ss"
def beginDate = datePart + " " + startTime
def endDate = datePart + " " + endTime

def startMillis = Date.parse(dateFormat, beginDate).time
def endMillis = Date.parse(dateFormat, endDate).time

testRunner.testCase.testSuite.project.setPropertyValue("beginDate", startMillis.toString())
testRunner.testCase.testSuite.project.setPropertyValue("endDate", endMillis.toString())
```

Kuva 6. Groovy Script komentosarja, joka muuttaa nykyisen päivämäärän alku- ja loppuaika millisekunteiksi.

4.2 Autentikointi

SoapUI:ssa rajapintapyyntöön voi määrittää autentikoinnin. Autentikointitapoja ovat perinteinen kirjautuminen käyttäjätunnuksella ja salasanalla, NTLM, SPNEGO-Kerberos ja OAuth 1.0 ja OAuth 2.0. Rajapintapyynnön otsikoihin (Header) sekä tietosisältöön pystyy myös määrittämään autentikointiin liittyviä määritteitä. Tietyissä tapauksissa tarvitaan myös HTTP-yhteyden pysyvän kokoajan päällä, koska autentikointiin liittyviä evästeitä (Cookies) tarvitaan rajapintapyynnöissä. Testitapauksen asetuksista pystyy määrittämään HTTP-yhteyden pysyvän koko testitapauksen ajan päällä.

Opinnäytetyössä hyödynnettiin myös JWT:tä eli JSON Web Token -autentikointia, jonka tarkoituksena on varmentaa, että käyttäjällä on oikeus haluttuun kohteeseen ja määritetään rajapintapyynnön otsikkoon. JWT jaetaan kolmeen osaan: otsikko, tietosisältö ja allekirjoitukseen. Nämä kolme osaa koodataan yhteen ja erotellaan toisistaan pisteellä. Kuvassa 7 on esitetty kolme osaa sekä yhdistetty koodi.

”Otsikko”

```
{
 "alg": "HS256",
 "typ": "JWT"
}
```

”Tietosisältö”

```
{
 "doctor": "Example",
 "name": "Doctor Example",
 "id": 1234567890
}
```

”Allekirjoitus”

```
HMACSHA256(
 base64UrlEncode(header) + "." +
 base64UrlEncode(payload),
 SecretSignature
)
```

”Yllä olevat koodattuna”

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkb2N0b3IiOiJFeGFtcGxlIiwibmFtZSI6IkRvY3RvciBFeGFtcGxlIiwiaWQiOjEyMzQ1Njc4OTB9.OBIX8rNEvTleqxc2BAhVWyy1O0031ZU-uu-FeYa7U6A

Kuva 7. JWT -esimerkki otsikosta, tietosisällöstä ja allekirjoituksesta sekä koodattuna.

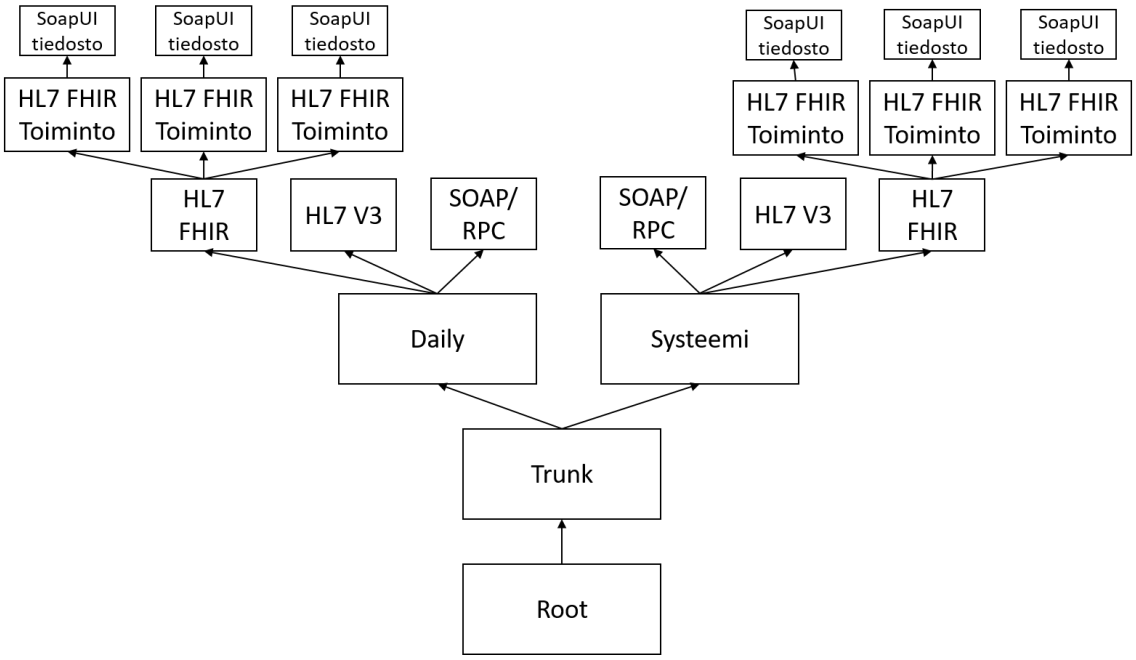
Groovy Scriptillä luotiin komentosarja muuttamaan tarvittavia tunnistautumisia Base64-muotoon. Kuvassa 8 haetaan projektin muuttuja, joka muutetaan Base64-muotoon ja lopuksi palautetaan projektin muuttujaan.

```
/*
This script will get custom property from project and encode (Base64) its content.
After encoding it will be returned to project properties.
Author: Ville Tuominen
*/

def customProperty = context.expand('${#Project#customProperty}')

// Encode customProperty to base64.
def encodedCustomProperty = accessContext.bytes.encodeBase64().toString()

testRunner.testCase.testSuite.project.setPropertyValue("encodedCustomProperty", "client/" +encodedCustomProperty)
```

Kuva 8. Groovy Script komentosarja, joka muuttaa projektin muuttujan Base64-muotoon.

4.3 Versionhallinta

Versiohallintaan luotiin kansiorakenne potilastietojärjestelmän ympäristöjen perusteella. Näitä ovat muun muassa Trunk, Next, julkaistava versio sekä edellinen versio. Lisäksi jokaiseen kehitysympäristöön kuuluu myös Daily- ja Systeemi-versiot, joihin luotiin testattavien rajapintojen kansiot, joita ovat muun muassa HL7 FHIR, HL7 V3 ja SOAP/RPC. Testattavien rajapintojen kansioihin luotiin vielä testattavien toimintojen kansiot, joiden sisällä on SoapUI:n luoma projektitiedosto. Kuvassa 9 esitetään Trunkin Dailyn ja Systeemin kansiorakennetta. Kansioiden rakenteet ovat identtiset, mutta SoapUI tiedostoissa on www-sovelluspalvelin eri. Kansiorakenne on luotu toiminnottain, koska testiautomaation aikana voi tapahtua virhe. Virheenkorjaus on helpompaa, kun tiedetään suoraan, mikä toiminto on aiheuttanut virheen. Lisäksi tehtäviä on helpompi jakaa eri tietokoneille sekä eri toimintoja voidaan hyödyntää myös muissa testiautomaatioissa. Esimerkiksi luodaan rajapintapyynnöillä testidataa, jonka jälkeen potilastietojärjestelmää testataan Ranorexilla.



Kuva 9. Trunkin Daily- ja Systeemi-version kansiorakenne.

4.4 Testitapauksien automatisointi ja raportointi

Testitapausten automatisointiin hyödynnettiin Jenkins -automaatiotyökalua. SoapUI:n testiohjelmia ja testitapauksia pystytään suorittamaan Windowsin komentorivillä. Komentorivin komennolla pystytään määrittämään SoapUI ajamaan testiohjelma tai yksittäisiä testitapauksia sekä luomaan raportti tuloksista. Kuvassa 10 on esitetty komennot, jolla testitapaus saadaan ajettua. Ensimmäisellä rivillä haetaan hakemisto, johon SoapUI on asennettu. Toisella rivillä avataan komentorivi sekä määritetään testiohjelman ja testitapauksen nimi, tallennuspaikka testiraportille, SoapUI:n asetukset tiedoston sijainti sekä SoapUI:n luoma XML-tiedosto projektista.

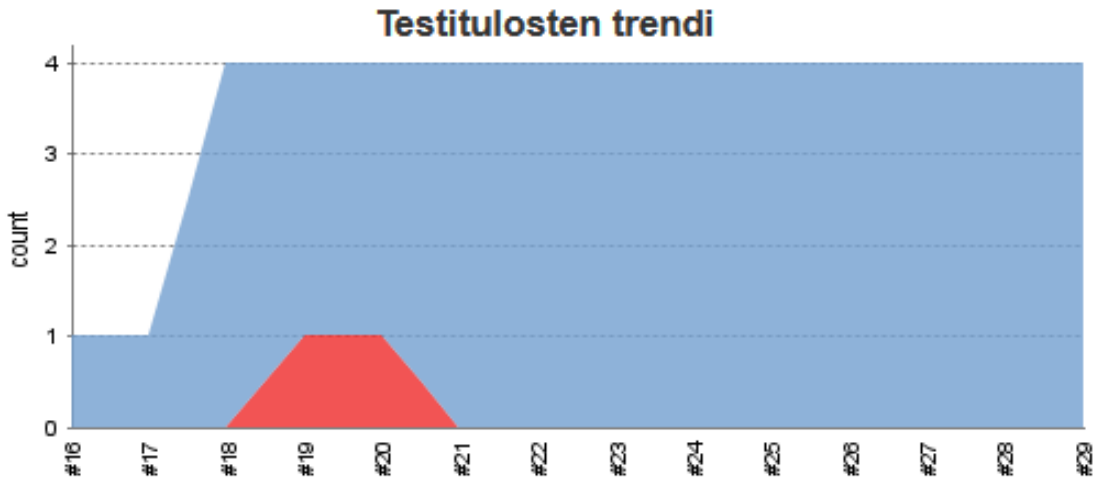
```
cd C:\Program Files\SmartBear\SoapUI-5.5.0\bin\

cmd.exe /C testrunner.bat -sLogin -c"Succesful login" -r -j -fC:\ohj\jenkins.start\workspace\SoapUI\SoapUI_FHIR\SoapUI_logs -tC:\Users\ville.tuominen
\soapui-settings.xml "C:\ohj\jenkins.start\workspace\SoapUI\SoapUI_FHIR\SoapUI\Kayttajatason_kutsu\FHIR\Trunk\Systeemi\Rajapintatestausta.xml
```

Kuva 10. Jenkinsissä käytetyt komennot, joilla saadaan testiohjelman testitapaus suoritettua sekä luotua raportti tuloksesta.

Jenkinsissä pystytään asettamaan, milloin ja mihin aikaan testejä halutaan suorittaa. Pääsääntöisesti testit ajetaan arkipäivisin yöllä sekä tarvittaessa Jenkinsille annetaan käsky suorittaa testi. Esimerkiksi, kun testitapauksia ollaan muokattu ja halutaan tietää, meneekö testit edelleen hyväksytysti läpi. Jenkinsiin määritetään myös SoapUI:n tiedoston sijainti. Tässä hyödynnetään versionhallintaa, mistä Jenkins hakee uusimman version tiedostosta. Koska SoapUI:n luoma XML-tiedosto on hyvin pieni voidaan kyseinen tiedosto hakea aina täysin uutena kopiona.

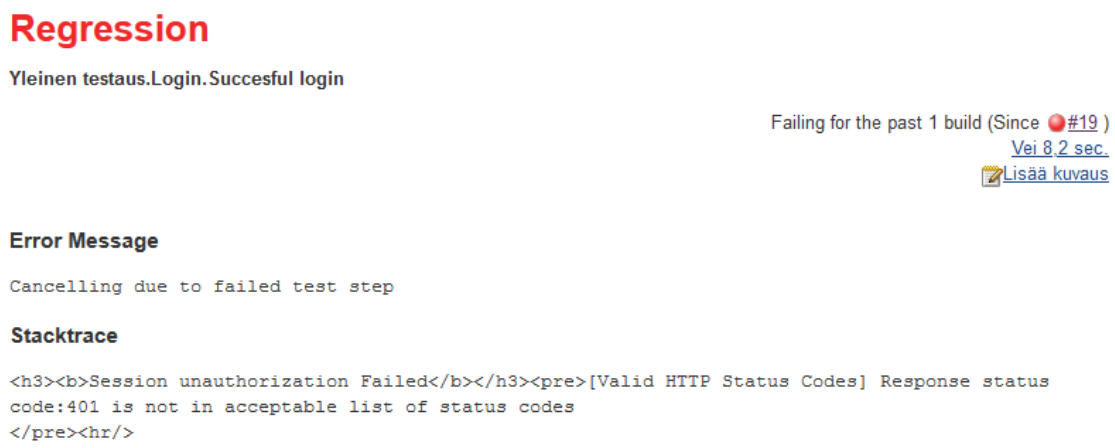
Jenkinsille voidaan myös asettaa tarvittavia jälkitehtäviä. Pääsääntöisesti nämä tehtävät ovat testiraportin julkaiseminen sekä tiedon lähettäminen sähköpostiin. Tieto lähetetään sähköpostiin niissä tilanteissa, kun jokin testitapaus on epäonnistunut.

Jenkins luo SoapUI:n testiraportista kaavion projektin etusivulle. Kaaviosta näkee suoritettujen testien määrän, onnistuneet testit ovat sinisellä ja epäonnistuneet punaisella. Kaaviota klikkaamalla pääsee testituloksiin. Kuvassa 11 on esitetty Jenkinsin luoma kaavio suoritettujen testiajojen onnistuneista sekä epäonnistuneista testeistä.



Kuva 11. Jenkinsin luoma kaavio testituloksista.

Jokaisesta ajetusta testauksesta voi tarkastella testituloksia. Jenkinsissä epäonnistuneesta testitapauksesta näkee helposti, mikä on mennyt pieleen. Esimerkiksi kuvassa 12 näkee, että kirjautuminen on epäonnistunut, koska vastaukseksi on saatu luvaton pääy sekä HTML-tilakoodi 401, joka ei ole hyväksyttyjen tilakoodien listalla. Testituloksista näkee myös jokaisen testin keston.



Kuva 12. Jenkinsissä epäonnistuneen testitapauksen virheilmoitus.

Jenkins lähettää sähköpostiviestin koko testin lokista, josta pystyy tutkimaan, miten testi on edennyt. Tämä ei kuitenkaan ole mielestäni tehokas tapa tutkia, mikä on mennyt pieleen. Jenkinsissä pystytään määrittämään jälkitehtäviin muokatun sähköpostiviestin, jota en kerinnyt tutkimaan tässä työssä.

5 YHTEENVETO

Opinnäytetyön tavoitteena oli luoda potilastietojärjestelmän rajapinnoille testiautomaatio. Tulokseksi saatiin luotua toimiva prosessi potilastietojärjestelmän rajapintojen testiautomaatiolle. Työssä hyödynnettiin SoapUI -rajapintatestaustyökalua, TortoiseSVN -versionhallintaa sekä Jenkins -automaatiotyökalua. SoapUI:lla pystyttiin testaamaan käyttäjätasolla ja sanomaliikenteen kautta potilastietojärjestelmän HL7 FHIR -rajapintoja sekä tarkistamaan, että rajapinta toimii oikein. TortoiseSVN:ään saatiin luotua kattava kansiorakenne testattavia ympäristöjä ja rajapintoja varten. Kansiorakennetta pystyy muokkaamaan vielä tarvittaessa. Jenkinsiin saatiin luotua automaatio testaamaan rajapintoja sekä luomaan testiraportit SoapUI:n raportista. Jenkinsissä testiraportista näkee yksittäisen testitapauksen tuloksen sekä testin kuluneen ajan.

Jatkokehityksenä potilastietojärjestelmästä löytyy myös muita rajapintoja, kuten SOAP/RPC-rajapintoja, joita olisi hyvä testata. Sanomaliikennettä testattiin työssä vain valtuustietojen avulla. Sanomaliikenteen testaamiseen on useampia tapoja, joita olisi myös hyvä testata. Rajapintojen suorituskyky- ja tietoturvatestaukset olisivat myös yksi testattava kohde. Lisäksi rajapintapyynnöillä pystyttäisiin laajentamaan nykyisiä Ranorexilla luotuja testiautomaatioita luomalla niihin testidataa valmiiksi rajapintapyynnöillä. Testidatan luonti rajapintapyynnöillä onnistuu todella nopeasti, mikä laajentaisi sekä nopeuttaisi testausta. SoapUI -työkalun maksullisessa versiossa pystyy myös laajentamaan vielä rajapintatestausta. Maksullinen versio tarjoaa laajemmat työkalut toiminnan ja Data-Driven testaukseen.

LÄHTEET

Atlassian 2019. What is version control. Viitattu 15.11.2019 https://www.atlassian.com/git/tutorials/what-is-version-control..

Avoin rajapinta 2014. Avoimen rajapinnan määritelmä. Viitattu 04.11.2019 http://avoinrajapinta.fi/

CGI 2018. Code Corner: Pieni API-sanakirja. Viitattu 04.11.2019 https://www.cgi.fi/fi/blogi/pieni-api-sanakirja.

CGI 2019. Historia Suomessa, Viitattu 29.0.1.2020 https://www.cgi.fi/fi/cgi-yrityksena/historia-suomessa.

DevQA 2019. Test Automation Advantages and Disadvantages. Viitattu 20.11.2019 https://devqa.io/test-automation-advantages-and-disadvantages/.

Digia 2019. API:T ovat modernin integraatiostrategian ydin. Viitattu 07.11.2019 https://blog.digia.com/rest-api.

Edureka 2019a. What is Jenkins? Viitattu 05.12.2019 https://www.edureka.co/blog/what-is-jenkins/.

Edureka 2019b. What is the key difference between Declarative pipeline and scripted pipeline. Viitattu 08.12.2019 https://www.edureka.co/community/54705/difference-between-declarative-pipeline-scripted-pipeliine.

Guru99 2019. Automation testing tutorial: What is, Process, Benefits & Tools, Viitattu 20.11.2019 https://www.guru99.com/automation-testing.html.

HL7 2019a. Appointment example request. Viitattu 06.1.2020 http://hl7.org/fhir/appointment-example-request.json.html.

HL7 2019b. About HL7. Viitattu 11.11.2019 http://www.hl7.org/about/index.cfm?ref=common.

HL7 2019c. Esittely. Viitattu 11.11.2019 http://www.hl7.fi/esittely/.

HL7 2019d. FHIR. Viitattu 12.11.2019 https://www.hl7.org/implement/standards/product_brief.cfm?product_id=343.

HL7 2019e. Rajapintakartta. Viitattu 12.11.2019 http://www.hl7.fi/hl7-rajapintakartta/.

HL7 2019f. RESTful API. Viitattu 06.01.2020 https://www.hl7.org/fhir/http.html.

JavaTpoint 2019. Jenkins – Reporting. Viitattu 08.12.2019 https://www.javatpoint.com/jenkins-reporting.

Jenkins 2019a. etusivu. Viitattu 04.12.2019 https://jenkins.io/.

Jenkins 2019b. Pipeline. Viitattu 08.12.2019 https://jenkins.io/doc/book/pipeline/.

Perforce 2017. What is SVN (Subversion). Viitattu 16.11.2019 https://www.perforce.com/blog/vcs/what-svn.

Perforce 2018. Git vs SVN – What is the difference. Viitattu 15.11.2019 https://www.perforce.com/blog/vcs/git-vs-svn-what-difference.

SearchSoftwareQuality 2019. What is Automated Testing and How Does it Work? Viitattu 21.11.2019 https://searchsoftwarequality.techtarget.com/definition/automated-software-testing.

Smartbear 2019. What is Automated Testing? Viitattu 16.11.2019 https://smartbear.com/learn/automated-testing/what-is-automated-testing/.

SoapUI 2019. Example REST Test in SoapUI. Viitattu 26.11.2019 https://www.soapui.org/resources/tutorials/rest-sample-project.html.

SoapUI 2019. Getting Started With SoapUI. Viitattu 25.11.2019 https://www.soapui.org/getting-started/introduction.html.

SoapUI 2019. Open Source. Viitattu 25.11.2019 https://www.soapui.org/open-source.html.

The Marketing Technologist 2017. Moving API requests in React to Redux-Saga's. Viitattu 08.11.2019 https://www.themarketingtechnologist.co/moving-api-requests-in-react-to-redux-sagas/.

TortoiseSVN 2019a. About TortoiseSVN. Viitattu 30.11.2019 https://tortoisesvn.net/about.html.

TortoiseSVN 2019b. Subversion-käyttöliittymä Windows-ympäristöön, julkaistu 30.09.2019. Viitattu 30.11.2019 https://dotsrc.dl.osdn.net/osdn/storage/g/t/to/tortoisesvn/1.13.1/Documentation/TortoiseSVN-1.13.1-fi.pdf.

Youtube Data Api 2019. Request Search List, Viitattu 07.01.2020 https://developers.google.com/youtube/v3/docs/search/list.

HL7 FHIR ajanvarauksen rajapintapyynnön tietosisällöstä


```
{
  "resourceType": "Appointment",
  "start": "2019-12-12T08:30:00+02:00",
  "end": "2019-12-12T08:45:00+02:00",
  "participant": [
    {
      "actor": {
        "reference": "Patient/Example"
      },
      "required": "required",
      "status": "accepted"
    },
    {
      "actor": {
        "reference": "Practitioner/Example"
      },
      "required": "required",
      "status": "accepted",
      "type": [
        {
          "coding": [
            {
              "code": "ATND",
              "system": "http://terminology.hl7.org/CodeSystem/v3-ParticipationType"
            }
          ]
        }
      ]
    },
    {
      "actor": {
        "reference": "Location/Example"
      },
      "required": "required",
      "status": "accepted"
    }
  ],
  "serviceCategory": {
    "coding": [
      {
        "code": "gp",
        "display": "General Practice",
        "system": "http://example.org/service-category"
      }
    ],
    "text": "General Practice"
  }
}
```


Liite 1. Esimerkki HL7 FHIR ajanvarauksen rajapintapyynnön tietosisällöstä. (HL7 2019a.)