



Osaamista  
ja oivallusta  
tulevaisuuden  
tekemiseen

Janne Heikkilä

# Atlassian Jira -lisäosakehitys

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

9.10.2019

Tekijä Otsikko	Janne Heikkilä Atlassian Jira -lisäosakehitys
Sivumäärä Aika	37 sivua 9.10.2019
Tutkinto	Insinööri (AMK)
Tutkinto-ohjelma	Tieto- ja viestintätekniikka
Ammatillinen pääaine	Ohjelmistotuotanto
Ohjaajat	Lehtori Simo Silander
<p>Insinööritöiden tavoitteena oli tutkia Atlassian Jira -tuotteiden lisäosakehityksen toimintamalleja ja rajoitteita sekä selvittää millaiselle lisäosalle on Jira-käyttäjien keskuudessa kysyntää. Selvityksen pohjalta oli tarkoitus luoda yksinkertainen, Jira Cloud -sovellukseen integroitu ja käyttäjien tarpeeseen vastaava lisäosa.</p> <p>Atlassian tukee lisäosien kehitystä tarjoamalla kolmannen osapuolen käyttöön monia erilaisia rajapintoja ja kirjastoja, joita voi käyttää ulkopuolisissa sovelluksissa tai joilla voi integroida sovelluksia heidän tuotteisiinsa. Lisäosien jakelu on mahdollistettu useisiin heidän tuotteisiinsa integroidulla Atlassian Marketplace -alustalla, jossa on tällä hetkellä yli 4000 eri tuotetta. Tyypillisesti lisäosalla halutaan tuoda Atlassian-sovellukseen uusia ominaisuuksia tai parantaa jo olemassa olevia.</p> <p>Insinööritöiden lopputuloksena syntyi Jira Software Cloud -lisäosa, jonka avulla lisäosaan oikeutetut käyttäjät voivat tarkastella niiden käyttäjäryhmien jäseniä ja jäsenten perustietoja, joihin he myös itse kuuluvat. Lisäosan sisältämältä konfiguraatiosivulta System Administrator -oikeutetut käyttäjät pystyvät määrittämään käyttäjäryhmät, joille lisäosan käyttö on mahdollista. Lisäosa koostuu Jira Cloud -rajapintoihin tukeutuvasta ReactJS-käyttöliittymästä ja Atlassian Connect -kehityksellisestä NodeJS-palvelinsovelluksesta.</p> <p>Sovelluskehitysprosessin raportoinnissa keskitytään esittelemään Jira-integraation näkökulmasta sovellukselle keskeisimmät toiminnallisuudet ja niiden toteutukset. Lisäosasovelluksessa käytetyn JavaScriptin tai sen apukirjastojen, kuten ReactJS:n toimintaan ei työssä perehdytä sen syvällisemmin, jonka takia lukijalta odotetaan perusymmärrystä ohjelmoinnista.</p>	
Avainsanat	Atlassian, Jira, integraatio

Author Title	Janne Heikkilä Atlassian Jira Add-on Development
Number of Pages Date	37 pages 9 October 2019
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Professional Major	Software Engineering
Instructors	Simo Silander, Senior Lecturer
<p>The goal of this thesis was to study procedures and limitations of add-on development for Atlassian Jira products and to determine what kind of add-on is in demand among Jira users. The purpose was to develop a user-friendly Jira Cloud integrated application which responds to the demand found in the study.</p> <p>Atlassian supports third-party add-on development with a set of APIs and libraries that can be used in external applications or to integrate applications with Atlassian products. Most add-ons are distributed through Atlassian Marketplace, which currently includes more than 4000 different applications. Typically, an add-on is intended to add new features to an Atlassian application or to improve an existing one.</p> <p>The final product of this study was a Jira Software Cloud add-on which allows users with add-on access to view the members of the user groups they belong to. Jira system administrators are able to specify the user groups that authorize access to the add-on through the product configuration page. The add-on consists of the ReactJS client application using Jira Cloud APIs and the NodeJS server application with Atlassian Connect framework.</p> <p>This thesis focuses on presenting the core functionalities and their implementations from the Jira integration perspective. The report does not cover the basics of JavaScript or its libraries, such as React.</p>	
Keywords	Atlassian, Jira, integration

## Sisällys

### Lyhenteet

1	Johdanto	1
2	Atlassian	2
2.1	Atlassian yrityksenä	2
2.2	Atlassian Jira	2
2.3	Muut Atlassian-tuotteet	3
3	Jira Software -lisäosakehitys	6
3.1	Ympäristökohtaiset eroavaisuudet	6
3.2	Lisäosakehityksen työkalut	6
3.2.1	Atlassian SDK	6
3.2.2	Jira REST API	8
3.2.3	Atlassian Connect	8
3.2.4	Atlassian Design Guidelines	13
3.2.5	Moduulit	14
3.3	Atlassian Marketplace	16
4	Lisäosan suunnittelu	17
4.1	Kehityskohteen etsiminen ja perustelu	18
4.2	Määrittely	19
4.3	Teknologiat	20
4.4	Ympäristöt	21
5	Lisäosan kehitys	24
5.1	Atlassian Connect Express -kehys	24
5.2	Sovelluksen käyttöliittymä	28
5.2.1	Pääsivu	29
5.2.2	Konfiguraatiosivu	32
6	Yhteenveto	35
	Lähteet	36

## Lyhenteet

ACE	Atlassian Connect Express. Atlassianin kehittäjille tarjoama moduuli lisäosakehityksen tueksi.
API	Application Programming Interface. Ohjelmointirajapinta.
CI	Continuous Integration. Jatkuva integraatio. Automatisoitu prosessi lähdekoodimuutosten integroimiseksi vanhaan koodikantaan.
CD	Continuous Deployment. Jatkuva toimitus. Automatisoitu prosessi muutosten julkaisemiseen.
CORS	Cross-Origin Resource Sharing. Eri domainien väliset resurssipyynnöt mahdollistava selaintekniikka.
CSV	Comma-Separated Values. Tekstitiedostomuoto, jossa taulukkorakenteinen tieto on eroteltu pilkkuja ja rivinvaihtoja hyväksi käyttäen.
JQL	Jira Query Language. Atlassianin oma, SQL-kieltä muistuttava kyselykieli.
MVP	Minimum Viable Product. Pienin toimiva tuote. Sisältää vain toimintaidean kannalta tärkeimmät ominaisuudet.
On-Demand	Pilviympäristössä sijaitseva instanssi, jonka ylläpidosta vastaa palvelun tarjoaja.
On-Premise	Asiakkaan omalle palvelimelle asennettu ohjelmisto.
REST	Representational State Transfer. Arkkitehtuurityyli rajapintojen luontiin.
SaaS	Software as a Service. Ohjelmisto palveluna. Pilvipalvelu, jota ylläpidetään palveluntarjoajan toimesta.

SDK	Software Development Kit. Kokoelma työkaluja sovelluskehityksen tueksi.
SSO	Single Sign-On. Menetelmä, jossa käyttäjä voi kirjautua useaan eri palveluun yhdellä autentikoinnilla.

## 1 Johdanto

Tämän insinöörityön tarkoituksena oli tutkia Atlassian Jira -tuotteiden lisäosakehityksen toimintamalleja ja rajoitteita sekä pilvi- että palvelinympäristöjen osalta. Tutkimuksen pohjalta pyrittiin luomaan pilviympäristöön yksinkertainen lisäosa, joka tuo Jiraan ominaisuuden, jolle on todistettusti kysyntää, mutta jota se ei kuitenkaan vielä tarjoa. Lopullisen lisäosaidean valintaan vaikutti Atlassianin omaan Jira-instanssiin lähetetyn ominaisuustoiveen lisäksi se, että olen myös itse Jiran parissa työskennellessäni törmännyt samaiseen puutteeseen.

Jira tarjoaa käyttäjilleen mahdollisuuden tarkastella eri ryhmiin kuuluvia jäseniä ainoastaan admin-paneelin kautta. Monissa isoissa Jira-projekteissa pääsynhallintaa kuitenkin hoidetaan ryhmien välityksellä, jolloin projektikohtaiselle ylläpitäjälle - jolla ei useinkaan ole pääsyä Jiran admin-paneeliin - voi olla epäselvää, kuinka monella käyttäjällä on pääsy hänen hallinnassaan olevaan projektiin.

Kehitetyn lisäosan tarkoituksena oli antaa yksittäisille käyttäjille mahdollisuus listata samoihin ryhmiin heidän kanssaan kuuluvat käyttäjät ja tallentaa niin halutessaan käyttäjälista tietoineen CSV-muotoisena käyttäjän omalle tietokoneelle. Lisäosan ei tietoturvasyistä haluta näkyvän oletuksena jokaiselle käyttäjälle, jonka takia siihen rakennettiin myös erillinen, admin-paneelin alla sijaitseva konfiguraatiosivu, josta käsin site-admin-oikeutetut voivat määrittää ne käyttäjäryhmät, jotka oikeuttavat lisäosan käyttöön.

Aiheen valintaan vaikutti suuresti työhistoriani Atlassian-tuotteiden parissa, joka alkoi vuonna 2017 aloittaessani Keskolla IT-Traineen nimikkeellä. Opinnäytetyön kirjoitushetkellä olen Atlassian-tuotteiden konsultointipalveluita pääkaupunkiseudulla tarjoavan Avoset Oy:n palkkalistoilla. Molemmissa positioissa Jira on ollut keskeisimmin työhöni liittyvä työkalu.

## 2 Atlassian

### 2.1 Atlassian yrityksenä

Yritys sai alkunsa vuonna 2001, jolloin australialaiset koulutoverit Mike Cannon-Brookes ja Scott Farquhar perustivat kreikkalaisen mytologian titaani Atlaksen mukaan nimetyn Atlassianin. Atlassian korostaa yrityskulttuurissaan ja omien tuotteiden suunnittelussaan avoimen työkulttuurin ja tiimityön merkitystä [1]. Päinvastoin kuin monet kilpailijansa, Atlassian tarjoaa tuotteitaan kaikenkokoisille tiimeille. Lähes kaikissa heidän tuotteissaan hinnoittelumalli on sidottu käyttäjälisenssien määrään. Yrityksen kenties erikoisin piirre on perinteisen myyntitiimin puute: Atlassian luottaa laadukkaiden tuotteiden myyvän itseään.

Atlassian työllistää yli 3000 henkilöä toimistoissaan seitsemässä eri maassa ja heillä on yli 125 000 asiakasta. Atlassianin ekosysteemiin kuuluu nykyään yhteensä 13 ohjelmistotuotetta. Heidän tuotteisiinsa suunniteltujen lisäosien jakelua varten on olemassa Atlassian Marketplace, josta löytyy yli 4000 lisäosaa Atlassianin eri tuotteisiin. Näiden lisäksi Atlassian Community -palstalla on yli 2,6 miljoonaa käyttäjää. [1.] ICT-alan tutkimusyritys Gartner on valinnut Atlassianin monena peräkkäisenä vuotena *Gartner Enterprise Agile Planning Tools Magic Quadrant Leader* -joukkoon [2]. Gartnerin toteuttaman tutkimuksen tarkoituksena on auttaa käyttäjää valitsemaan hänelle sopivin ketterien menetelmien käyttöä ja suunnittelua helpottava ohjelmisto. Atlassianin vahvuuksiksi Gartner mainitsee muun muassa laajan käyttäjäkunnan myötä muodostuneen kustomoitavuuden Atlassian Marketplacen kautta hankittavilla lisäosilla ja Atlassianin ekosysteemiin kuuluvien tuotteiden välisen vahvan integraation.

### 2.2 Atlassian Jira

Jira on Atlassianin vuonna 2002 julkaisema, täysin Java-kielellä toteutettu web-sovellus, joka oli alun perin sovelluskehittäjille tarkoitettu tehtävienhallintaohjelmisto. Vuosien kuluessa Jirasta on muodostunut Atlassianin lippulaivatutuote, joka on nykyään jaettu käyttötarkoituksen mukaan neljään eri kokonaisuuteen:



- *Jira Software* on tarkoitettu alkuperäisen idean mukaisesti sovelluskehityksen tu-  
kityökaluksi. Ohjelmisto antaa käyttäjälle laajan skaalan erilaisia työkaluja, joilla  
esimerkiksi erilaisten ketterien menetelmien soveltaminen onnistuu helposti.
- *Jira Service Desk* on palvelupyyntöjen seurantaan tarkoitettu ohjelmisto. Service  
Desk tarjoaa tiketin jättämiseen erillisen asiakasportaalinäkymän, jonka kautta  
palvelupyyntö voi jättää myös instanssin ulkopuolinen käyttäjä.
- *Jira Core* on kehitetty liiketoimintatiimien tarpeisiin. Sovelluksessa on korostettu  
projektin seuranta, raportointia ja taulunäkymän selkeyttä.
- *Jira Ops* on Atlassianin viimeisin lisäys Jira-tuoteperheeseen, ja se on vielä tä-  
män tekstin kirjoitushetkellä ilmaisessa betatestausvaiheessa. Se on tarkoitettu  
erilaisten ongelmatapausten ja odottamattomien välikohtausten selvitystyöhön ja  
nopeaan ratkaisuun. Ops on saatavilla ainoastaan pilviympäristöihin.

Jira Opsia lukuun ottamatta kaikista edellä mainituista on saatavilla sekä Atlassian-pil-  
viympäristössä pyörivä SaaS-sovellus (On-Demand) että käyttäjän omille palvelimille  
asennettava versio (On-Premise). On-Premise-ympäristöihin saatavilla oleviin tuotteisiin  
kuuluu myös isoille instansseille suunnattu Data Center -versio, jonka tarkoitus on tarjota  
asiakkaalle lisää luotettavuutta ja suorituskykyä isojenkin käyttäjämäärien kanssa. [4.]

## 2.3 Muut Atlassian-tuotteet

Tässä luvussa käydään lyhyesti läpi kaikki Jiran lisäksi Atlassian-ekosysteemiin kuuluvat  
ohjelmistotuotteet ja niiden käyttötarkoitukset. Useimmista tuotteista löytyvät sekä Atlas-  
sianin pilviympäristössä ylläpidetty sovellus että käyttäjän omille palvelimille asennettava  
On-Premise-versio.

## Confluence

Confluence on Atlassianin vuonna 2003 julkaisema sisällönhallintajärjestelmä ja tiimityöskentely-ympäristö, joka antaa tiimeille mahdollisuuden työskennellä yhdessä paikassa reaaliaikaisesti. Confluence-työtilan ja Jira-projektin linkittäminen mahdollistaa projektikohtaisten raporttien ja статистиikan suoraviivaisen dokumentoimisen työtilaan.

## Bitbucket

Bitbucket on Git- ja Mercurial-versionhallintajärjestelmiä tukeva palvelu, jonka Atlassian liitti yrityskaupalla oman brändinsä alle vuonna 2010. Versionhallinnan lisäksi Bitbucket pitää sisällään jatkuvan integraation (continuous integration) ja jatkuvan julkaisun (continuous deployment) automatisoimiseen tarkoitetun Bitbucket Pipelinesin, vahvan integraation Jiraan sekä useita repositorioiden hallintaa helpottavia ominaisuuksia.

## Bamboo

Bamboo on Atlassianin CI/CD-työkalu On-Premise-palvelinympäristöön, jonne Bitbucket Pipelinesia ei ole saatavilla.

## Sourcetree

Sourcetree on ilmainen, graafisen käyttöliittymän tarjoava Git-ohjelmisto.

## Trello

Trello on Atlassianin vuonna 2017 ostama pilvipalvelu, joka tarjoaa käyttäjilleen helppokäyttöisen ja kustomoitavan Kanban-taulun projektinhallintaan.

## Crowd

Crowd on Single sign-on (SSO) -palvelu On-Premise-ympäristöihin, johon Atlassian-työkalujen käyttäjähallinta voidaan keskittää. Palvelu tukee monia eri käyttäjähakemistoja.

## Atlassian Access

Atlassian Access on Atlassian-pilvipalveluissa toimiva Crowdin vastine.

## Statuspage

Statuspage on pilvipalvelu, joka kertoo palveluun linkitettyjen web-sovellusten tilan reaaliaikaisesti. Se mahdollistaa erilaisten valmiiden ilmoituspohjien teon, sähköpostilistat, joilla esimerkiksi katkoksista voidaan informoida tarvittavaa henkilöstöä sekä integraatiot viestisovelluksiin ja Atlassian Opsgenieen.

## Opsgenie

Opsgenie on sovellusten monitorointiin tarkoitettu pilvipalvelu. Palvelu mahdollistaa Statuspagen ja Jira Opsin kanssa kokonaisvaltaisen hälytysjärjestelmän, jossa Opsgenie muodostaa Jira Ops -ongelmatiketin ilmoituksen lauetessa, jonka jälkeen Jira Opsin kautta tehdyt muutokset välittyvät suoraan loppukäyttäjien näkyville Statuspage-sivustolle.

### 3 Jira Software -lisäosakehitys

Tämän luvun tarkoituksena on perehtyä Jira Software -tuotteen lisäosakehityksen toimintamalleihin, pilvi- ja palvelinversioissa kehittämisen eroihin ja Atlassianin kehittäjille tarjoamiin työkaluihin.

#### 3.1 Ympäristökohtaiset eroavaisuudet

Lisäosakehityksessä Jira Cloud- ja Jira Server -versioiden välillä on suuria eroja. Tämä johtuu käytännössä siitä, että Jira Server -lisäosa asennetaan suoraan käyttäjän palvelimella pyörivään Jira-sovellukseen, jolloin sovelluksen omaa Java APIa ja tietokantaa on mahdollista käyttää suoraan lisäosasta. Jira Cloud -sovelluksessa käyttäjällä ei ole suoraa pääsyä edellä mainittuihin, koska Atlassianin SaaS-palvelun palvelinresurssit jaetaan useiden eri instanssien kesken [3]. Jira Cloud -ympäristössä kehittäjät joutuvat näin ollen Java API:n sijaan tukeutumaan REST API:n, jonka päätepisteet eivät kuitenkaan tarjoa aivan yhtä kattavia toiminnallisuuksia Java API:n kanssa.

#### 3.2 Lisäosakehityksen työkalut

Atlassian tarjoaa erilaisia kokoelmia ja rajapintoja, joiden avulla kolmannen osapuolen sovellusten kehittäminen heidän tuotteisiinsa mahdollistetaan. Luvun 3.2 tarkoituksena on luoda yleiskatsaus näihin työkaluihin.

##### 3.2.1 Atlassian SDK

Atlassian plugin Software Development Kit (SDK) mahdollistaa lisäosien rakentamisen ja integroimisen Atlassian Server -sovelluksiin. Jira Server -lisäosakehityksen tueksi SDK tarjoaa kokoelman shell-skriptejä, joiden avulla kehittäjä voi luoda, asentaa tai koota lisäosan. Atlassian SDK vaatii toimiakseen Java SDK:n ja Maven-koontityökalun. Kun SDK on asennettu, Jira-lisäosan alustus onnistuu windows-komentoriviltä komenolla *atlas-create-jira-plugin*. Komentorivi tulee kysymään käyttäjältä *groupId*-, *artifactId*-, *version*- ja *package*-tietueiden arvoja. Asettamalla arvot seuraaviksi, SDK alustaa toimivan lisäosan kuvan 1 mukaisella tiedostorakenteella:

```

groupId:    com.example.plugin
artifactID: examplePlugin
version:    0.0.1
package:    examplePlugin

```

Kuvassa 1 *resources*-kansioiden alla näkyvät *atlassian-plugin.xml*-deskriptoritiedostot ovat lisäosan ja Jira-instanssin yhdistämisen kannalta avainasemassa. Kyseisessä tiedostossa määritellään lisäosan perustiedot, muut kuin Java-kieliset resurssit sekä moduulit, joiden kautta lisäosa integroidaan Jira-sovellukseen. Moduuleita käsitellään tarkemmin luvussa 3.2.5.



Kuva 1. Tiedostorakenne lisäosan alustuksen jälkeen.

### 3.2.2 Jira REST API

Jira REST API:t mahdollistavat Jira-instanssien kanssa kommunikoinnin ilman käyttöliittymää. Rajapinnat pitävät sisällään päätepiteet Jiran yleisimpien toiminnallisuuden käyttöön, jonka myötä ne myös mahdollistavat lisäosakehityksen sellaisissa tapauksissa, jossa lisäosalla ei ole suoraan pääsyä Jiran omaan Java API:in. Tämän kaltaisia tapauksia ovat lisäosakehitys Jira Cloud -tuotteelle, jonka Software as a Service eli SaaS-palvelun taustajärjestelmiin käyttäjällä ei ole pääsyä ja On-Premise-instanssin tietoja tai toiminnallisuuden käyttävän ulkopuolisen integraation rakentaminen. On-Premise-instanssin sisään integroitu lisäosa sen sijaan pääsee käsiksi suoraan REST-rajapintaa monipuolisempaan Java API:in.

Atlassian on kehittänyt omat REST API:t sekä Jira Server että Jira Cloud -versioille. Rajapintojen toiminnallisuudet ovat pääpiirteittäin samoja, mutta ympäristöjen välisten ominaisuuksien takia poikkeuksiaakin löytyy.

### 3.2.3 Atlassian Connect

Atlassian Connect on sovelluskehys, joka avustaa kehittäjää lisäosan ja Jira Cloud -instanssin välisessä integraatiossa, autentikoinnissa ja asennuksessa, eikä se itsessään aseta kehittäjälle kehityskielen tai -ympäristön suhteen vaatimuksia. Atlassian Connect voidaan nähdä Jira Cloud -sovelluksien vastineena Jira Server -sovelluksien Atlassian SDK:lle. Hyvin toteutettu, Atlassian Connectia käyttävä sovellus näyttäytyy loppukäyttäjälle saumattomana integraationa Jiran kanssa. [5.]

Atlassian Connect -sovelluksissa keskeisessä roolissa toimii sovelluksen deskriptoriksi (engl. app descriptor) kutsuttava JSON-muotoiltu *atlassian-connect.json*-tiedosto, joka on käytännössä Jira Cloud -sovelluksen vastine Atlassian SDK:lla rakennettujen Jira Server -lisäosien *atlassian-plugin.xml*-tiedostolle. Deskriptorissa määritellään sovelluksen perustiedot, käytettävät moduulit sekä päätepiteet sovelluksen elinkaariattribuuteille [8]. Kun Jira instanssin admin-käyttäjä asentaa Atlassian Connect -pohjaista sovellusta, asentaa hän todellisuudessa vain deskriptoritiedoston, joka sisältää osoittimen sovelluksen oikeaan sijaintiin.

## Autentikointi

Autentikoinnin eli lisäosan todennuksen tarkoituksena on varmistaa, että Atlassian Connect -lisäosan ja Atlassian Cloud -sovelluksen välisessä kommunikaatiossa ei ole välikäsiä ja että sovelluksen lähettämät HTTP-pyynnöt pysyvät muuttumattomina. Useimmiten autentikointi Atlassian Connect -sovelluksissa on toteutettu JSON Web Tokens eli JWT-menetelmää hyödyntämällä.

JWT-autentikointia käytettäessä tulee sovelluksen deskriptoriin määritellä autentikointityyppi JWT, jolloin Jira lähettää sovellukselle ns. installation handshake, joka sisältää muun muassa autentikoinnin turvallisuustiedot (engl. security context) [6]. Sovelluskehittäjän on tallennettava turvallisuustiedot myöhempää käyttöä varten, jotta sovellukseen tulevien pyyntöjen validointi ja sovelluksesta lähtevien pyyntöjen allekirjoitus onnistuu turvallisuustiedoista löytyvän *sharedSecre* -avaimen avulla.

JWT on String-olio, jonka rakenne koostuu kolmesta, pisteellä erotellusta, base64Url-enkoodatusta tietueesta: otsikosta (engl. header), sisällöstä (engl. payload) ja allekirjoituksesta (engl. signature) [7]. Otsikko sisältää informaation käytettävästä salausalgoritmista, sekä tokenin tyyppin, joka on JWT. Tokenin toinen osa sisältää tyypillisesti autentisuuden varmentamiseen ja tokenin voimassaoloon liittyvät attribuutit. Kolmas osio on salausalgoritmillä salattu yhdistelmä otsikosta ja sisällöstä. Allekirjoitusosion tarkoituksena on varmentaa, että token on pysynyt muuttumattomana transaktion ajan.

Kuvassa 2 on esillä Atlassian Connect -sovelluksen vaatimukset täyttävän JWT-tokenin luontiarvot syötettynä [jwt.io](https://jwt.io)-sivuston generaattoriin. Allekirjoitusosion salausavaimeksi syötetään enkoodattuna teksti "SALAUSAVAIN".

HEADER: ALGORITHM & TOKEN TYPE
<pre> {   "alg": "HS256",   "typ": "JWT" } </pre>
PAYLOAD: DATA
<pre> {   "iss": "jira:1234567",   "iat": 1300819370,   "exp": 1300819380,   "qsh": "f4ca1e41df7bc90c8ab6d0f6207d491cf6dad7c66ea7",   "context": {     "user": {       "accountId": "123456:1234abcd",       "userKey": "janne",       "username": "janneh",       "displayName": "Janne H"     }   } } </pre>
VERIFY SIGNATURE
<pre> HMACSHA256(   base64UrlEncode(header) + "." +   base64UrlEncode(payload),   SALAUSAVAIN ) <input checked="" type="checkbox"/> secret base64 encoded </pre>

Kuva 2. Jwt.io sivuston generaattoriin syötetyt, Atlassian Connect -sovellusten vähimmäisvaatimusten mukaiset arvot, joista JWT-token koostuu.

Kuvassa 3 näkyy generaattorin tuottama enkoodattu lopputulos, jossa rakenteen eri osiot on eroteltu kuvan 2 mukaisin värein.

```

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJqaXJhOjEyMzQ1NjciLCJpYXQiOiEzMDA4MTkzNzAsImV4cCI6MTMwMDgxOTM4MCwicXNoIjoiaZjRjYTFkZjZkdjZkYzZkhYjZkMGY2MjA3ZDQ5MWNmNmRhZDdjNjZlYTciLCJjb250ZXh0Ijp7InVzZXIiOnsiYWVudElkIjoiaMTIzNDU2OjEyMzRhYmNkIiwidXNlcktleSI6Imphbm5lIiwidXNlcm5hbWUiOiJqYW5uZWgiLCJkaXNwbGF5IjoiZSI6IkpXVCJ9.3ZG-QnjdT_HIJv4Y8hbW3HnVYARofRp2TH_w0vVfT9w

```

Kuva 3. JWT-tokenin lopullinen muoto.



## Auktorisointi

Kun sovellus on autentikoitu eli tunnistettu, vaaditaan siltä vielä auktorisointi, jolla varmistetaan käyttöoikeuksien riittävyys. Atlassian Connect -sovelluksessa auktorisointi toteutetaan tyypillisesti deskriptoreissa määriteltävillä staattisilla Scope-arvoilla, tai lisäosasovellusta varten luotavalla käyttäjällä, jolle on annettu tarvittavat oikeudet. Erilaisia Scope-arvoja löytyy yhteensä 8:

- NONE
- READ
- WRITE
- DELETE
- PROJECT ADMIN
- ADMIN
- ACT\_AS\_USER
- ACCESS\_EMAIL\_ADDRESSES

Deskriptorissa Scope-tietue on taulukkomuotoinen, eli se voi sisältää useamman arvon. Tietue määritellään deskriptoritiedoston JSON-objektin ylimmällä tasolla.

Lisäosasovellus on myös mahdollista auktorisoida ottamaan tietyn käyttäjän roolin, jolloin myös oikeudet periytyvät kyseiseltä käyttäjältä. Tämä mahdollistetaan JWT Bearer Token ja OAuth 2.0 -tekniikoiden avulla. [9.]

## JavaScript API

Lähtökohtaisesti Jira Cloud -lisäosasovelluksen ja Jiran välinen kommunikointi ei saman alkuperän käytännön takia (engl. same origin policy) ole mahdollista, koska sovellukset eivät sijaitse samassa palvelinympäristössä. Atlassian Connectin JavaScript API kuitenkin mahdollistaa tämän deskriptorissa määritellyille moduuleille hyödyntämällä tekniikkaa nimeltä *cross-domain messaging* [10]. Tekniikan perusajatuksena on käyttää JavaScriptin *window.postMessage()*-metodia mahdollistamaan tiedonvälitys Atlassian Connectin Inline Frame eli IFrame-dokumentin ja dokumentin parent-elementin välillä, joka on osa Jira-sovellusta. JavaScript-kirjaston käyttöönottamiseksi riittää *all.js* skriptin sisällyttäminen sovelluksen html-koodiin koodiesimerkin 1 mukaisesti.

```
<script src="https://bitbucket.org/atlassian-connect/all.js"></script>
```

Koodiesimerkki 1. All.js-skriptin sisällyttäminen html-koodiin.

JavaScript API tarjoaa myös muita metodeja, joilla kehittäjä pääsee käsiksi Jira-instanssin ominaisuuksiin. Näitä ominaisuuksia ovat mm. evästeet, dialogimodaalit ja notifikaatiot.

### Jatkojalostetut Atlassian Connect -kehykset

Atlassian Connect -kehystä on myös jatkojalostettu eri ohjelmointikielille valmiiksi pake-teiksi, joissa kehittäjän ei tyypillisesti tarvitse enää huolehtia itse autentikointiin liittyvistä asioista tai lisäosasovelluksen rekisteröimisestä Jira-instanssiin. Näistä Atlassianin tu-kemia ja kehittämiä ovat Atlassian Connect Express, joka on NodeJS-ympäristöön poh-jautuva kehys, sekä Javaan pohjautuva Atlassian Connect Spring Boot. Tarjolla on myös useita kolmansien osapuolien kehittämiä kehyksiä muun muassa PHP-, Haskell- ja Scala-pohjaisina.

### Atlassian Connect Express (ACE)

ACE eli Atlassian Connect Express on NodeJS-ympäristöä ja sille tarkoitettua Express-kehystä hyödyntävä väliohjelmiston (engl. middleware) ja apufunktioita sisältävä kirjasto, jonka avulla lisäosakehittäjän on helppo ja nopea alustaa serverillä pyörivä Jira Cloud -lisäosa. Koska tässä opinnäytetyössä käytettiin kyseistä kirjastoa, käymme läpi muuta-mia ACE:n tärkeimmistä ominaisuuksista.

ACE tarjoaa kehittäjälle valmiin ratkaisun JWT-autentikoinnin toteuttamiseen kutsumalla kirjaston *authenticate*-metodia. Kuvan neljä sisältämässä esimerkikoodissa näytetään, miten Express-kehystä ja ACE:n väliohjelmia hyödyntämällä luodaan reititys *app*-nimi-seen malliin (engl. template). ACE hyödyntää malleissa oletuksena Handlebars-si-vumootoria (engl. template engine) *express-hbs*-kirjaston avulla [11].

```
// Route for search page
// Verify that the incoming request is authenticated with Atlassian Connect
app.get('/search', addon.authenticate(), function (req, res) {
    res.render('app');
});
```

Kuva 4. Esimerkki autentikoidun reitin luomisesta ACE:n authenticate-metodia käyttämällä.

Koska Jira-istanssin on päästävä asennettavaan lisäosaan käsiksi internetin kautta, ACE käyttää automaattisesti ngrok-työkalua tunnelin luomiseen lokaalin kehitysympäristön ja internetin välille. Käytännössä ngrok luo julkisen, tietyn ajan voimassa olevan osoitteen pilvipalveluunsa, josta se ohjaa liikenteen kehitysympäristössä pyörivän ngrok-prosessin kautta palomuurin läpi lisäosasovelluksen deskriptoritiedostoon [12].

Kehitysvaiheessa olevan lisäosasovelluksen rekisteröimistä Jira-istanssiin ACE helpottaa automatisoimalla prosessin. Jotta automatisointi toimisi, on käyttäjän luotava tiedostorakenteen juureen *credentials.json*-tiedosto, jossa on määritelty istanssin osoite, sekä admin-oikeutetun käyttäjän käyttäjätunnus ja salasana. Ominaisuuteen on myös sisäänrakennettu tiedostorakenteen tarkkailija, joka muutosten jälkeen uudelleenrekisteröi sovelluksen. Palvelinprosessia sammuttaessa ACE poistaa automaattisesti rekisteröimänsä lisäosan tiedot istanssista. [11.]

### 3.2.4 Atlassian Design Guidelines

ADG eli Atlassian Design Guidelines on kokoelma Atlassianin brändin mukaisia vaatimuksia ja toteutustapoja sovelluksen ulkoasulle ja toiminnallisuuksille, joita yhtiö itse kehitystyössään noudattaa ja joita ulkopuolisen lisäosakehittäjän on myös osittain noudatettava, mikäli lisäosa aiotaan julkaista Atlassian Marketplacella. Koska Jira Cloud ja Jira Server -versiot eivät ole ulkoasuiltaan täysin vastaavia, on molemmille myös omat ADG -kokoelmansa. Näiden määritysten pohjalta Atlassian on rakentanut sovelluskehittäjille kaksi avoimen lähdekoodin UI-kirjastoa:

## Atlaskit

Atlaskit on ReactJS-kirjastolla toteutettu UI-komponenttikokoelma, joka noudattaa Jira Cloud -sovellukselle asetettuja ADG-määritelmiä. Atlaskittiä käyttävät sekä Atlassianin omat sovelluskehittäjät että lisäosasovelluksien kehittäjät. Kokoelmasta löytyy lähestulkoon kaikki Jira Cloud -version käyttöliittymässä toistuvat elementit painikkeista hakukenttiin ja navigointipalkkeihin.

Atlaskitista löytyy myös Reduced UI Pack, joka sisältää nimensä mukaisesti hyvin supistetun määrän Atlaskitin komponentteja: CSS-ruudukon (engl. CSS grid), ikonit ja lomakeelementit. Reduced UI Packin käyttö ei muusta paketista poiketen vaadi React-toteutusta, vaan pelkkä CSS-tiedoston sisällyttäminen HTML-koodiin riittää.

## Atlassian User Interface

AUI eli Atlassian User Interface on HTML- ja CSS-pohjainen tyylikokoelma, joka on pääasiassa käytössä Jira Server -lisäosakehityksessä. Vielä vuonna 2017 jokainen Atlassianin tuote on käyttänyt AUI:ta ulkoasussaan vähintään 40 prosentin käyttöasteella muun muassa sen tarjoamien CSS-reset- ja basic styles -ominaisuuksien takia [14]. Nykään käyttöaste on todennäköisesti alempi, koska myös Atlaskittiin on lisätty tuki kyseisille ominaisuuksille.

Atlassianin pidemmän aikavälin tavoitteena on yhdistää Cloud- ja Server-sovellusten ADG-mallit [13].

### 3.2.5 Moduulit

Jiran moduulien avulla kehittäjät pystyvät integroimaan omat sovelluksensa ja ominaisuutensa Jiraan. Suurin osa moduuleista on Jiran käyttöliittymään lisättäviä linkkejä, sivuja tai paneeleja, mutta myös toiminnallisuuksiin vaikuttavia moduuleja löytyy muutamia. Moduulien määrittely tapahtuu lisäosasovelluksen deskriptoritiedostossa.

Koska käyttöliittymät ja sovellusarkkitehtuurit Jira Cloud- ja Jira Server -sovelluksissa eroavat toisistaan, on myös molempien moduulirakenteissa ja määrittelytavoissa eroja,

vaikka ne perusperiaatteeltaan ovatkin samanlaisia. Käymme nyt läpi moduulirakenteen piirteitä yleisellä tasolla ottamatta suuremmin kantaa sovelluksien eroavaisuuksiin.

Perusmoduulirakenteeseen kuuluu kolme eri komponenttia: web item, web section ja web panel [14].

Web item määrittelee yksittäisen linkin tai painikkeen sille osoitettuun kohtaan Jiran käyttöliittymässä. Koodiesimerkissä 2 on näkyvissä yksinkertainen Jira Cloud -sovelluksen deskriptoritiedostossa JSON-rakenteella määritelty web item, joka lisää Jiran käyttöliittymän navigaatiopalkkiin painikkeen, jota klikkaamalla aukeaa url-tietueessa esiintyvän reitityksen mukainen sivusto. Web itemin toteutumisen ehdoksi on asetettu se, että käyttäjä on kirjautuneena sisään.

```
"modules": {
  "webItems": [{
    "location": "system.top.navigation.bar",
    "url": "/activity",
    "context": "addon",
    "target": {
      "type": "page"
    },
    "icon": {
      "width": 16,
      "height": 16,
      "url": "/icons/activity.png"
    },
    "name": {
      "value": "Activity"
    },
    "key": "activity",
    "conditions": [{
      "condition": "user_is_logged_in"
    }]
  }]
}
```

Koodiesimerkki 2. Moduulin määrittely Jira Cloud -sovelluksen deskriptoritiedostossa.

Web section -peruskomponentin tarkoituksena on koota useampia web item -komponentteja tai vastaavia yhden rajatun osion sisään sovelluksen navigaatiopalkkeihin. Koodiesimerkissä 3 on täydennetty aikaisemman koodiesimerkin koodia niin, että web item sijaitsee nyt admin-toiminnallisuuksien alta löytyvän plugin-valikon uudesta *demo section* -osiosta.

```

"modules": {
  "webItems": [{
    "location": "admin_plugins_menu/demo-section",
    "url": "/activity",
    "context": "addon",
    "target": {
      "type": "page"
    },
    "icon": {
      "width": 16,
      "height": 16,
      "url": "/icons/activity.png"
    },
    "name": {
      "value": "Activity"
    },
    "key": "activity"
  }],
  "webSections": [{
    "key": "demo-section",
    "location": "admin_plugins_menu",
    "name": {
      "value": "UIS Demo"
    }
  }]
}

```

Koodiesimerkki 3. Web section lisättynä aikaisempaan koodiesimerkkiin.

Web panel mahdollistaa uusien osioiden luomisen Jiran käyttöliittymän eri toiminnallisuuksien, kuten issue view -näytymän sisälle. Web panel määritellään samankaltaisesti aikaisempien esimerkkien kanssa.

Edellä mainittujen peruskomponenttien lisäksi Jira sisältää lukuisia edistyneempiä moduuleja mm. Jiran workflow-työnkulun post-function-ominaisuuksien sekä dashboard- ja raporttinäkymien kehittämiseen. Jira Server -sovellus tarjoaa näitä moduuleja huomattavasti Jira Cloudia enemmän ja monipuolisemmin kehittäjän käytettäväksi.

### 3.3 Atlassian Marketplace

Atlassian Marketplace on Atlassianin kehittämä ja useimpiin heidän tuotteisiinsa integroitu kauppapaikka, joka mahdollistaa loppukäyttäjille käytössä olevien sovellusten kustomoinnin sekä ilmaisilla että maksullisilla lisäosilla. Tekstin kirjoitushetkellä kaikkien Marketplace-lisäosien yhteenlaskettu määrä on 4027, joista 1154 on ilmaisia ja loput maksullisia. Maksullisissa tuotteissa hinnoittelu lasketaan Atlassianin tuotteen lisenssin käyttäjämäärän mukaan. Suurin osa tarjolla olevista lisäosista on kolmansien osapuolien kehittämiä tuotteita, mutta listauksesta löytyy myös muutamia lisäosia, joita Atlassian itse

kehittää ja hallinnoi. Kaikkiin Marketplacen lisäosiin on tarjolla ilmainen, vähintään kauden kestävä kokeilujakso.

Sovelluskehittäjän näkökulmasta katsoen Marketplace tarjoaa helpon kanavan tuotteiden markkinointiin alustalla, joka tavoittaa suurimman osan Atlassian tuotteiden pääkäyttäjistä. Julkaiseminen kyseisellä alustalla kuitenkin asettaa kehittäjälle myös velvoitteita sitoutua muun muassa designiin, suorituskykyyn, autentikointiin ja hinnoitteluun liittyviin säädöksiin ja rajoituksiin. Jokainen listattava lisäosa joutuu käymään Atlassianin hyväksymisprosessin läpi [16].

Marketplace-kauppiaan (vendor) ja heidän tuotteidensa luotettavuuden lisäämiseksi Atlassian on luonut muutamia eri tehosteita. Yksittäisten tuotteiden kohdalla yksinkertaisin keino varmistua tuotteesta on lukea käyttäjäarvosteluja, joista useimmiten saa realistisen kuvan lisäosan senhetkisestä toimivuudesta ja mahdollisista ongelmakohtista. Lisäosat tuotteen esittelysivulta selviää myös, kuinka moneen instanssiin kyseinen tuote on sillä hetkellä asennettuna, sekä luonnollisesti myös kaikki versiot, joiden kanssa sovellus on yhteensopiva. Lisäosat tuotteen kehittäjä eli sen kauppias voi myös parantaa omaa statustaan Marketplacen Top Vendor -ohjelman kautta. Päästäkseen top vendoriksi on kauppiaan täytettävä erilaisia Atlassianin määrittelemiä ehtoja, kuten aktiivisten instanssien perusteella laskettava minimikäyttäjämäärä, sekä tarjottava tuotteisiinsa SLA-vaatimukset täyttävä tukipalvelu ja dokumentaatio. Top vendor -ohjelmaan kuuluu myös myyntimääriin sidotut gold- ja platinum-tasot. [17.]

#### **4 Lisäosan suunnittelu**

Luvussa 4 käsitellään lisäosan toimintaidean kehittämistä sekä lisäosan teknistä suunnittelua. Toimintaidean suunnittelussa pyrittiin aluksi kartoittamaan pääosin Atlassian Community -palstalle ja Atlassianin omaan Jira-instanssiin luotujen keskustelujen, pyyntöjen ja parannusehdotuksien perusteella, minkälaiselle lisäosalle yhteisöstä löytyisi selkeästi kysyntää. Suunnittelussa pyrittiin myös määrittämään ja perustelemaan vaatimukset lisäosasovelluksen ensimmäiselle versiolle sekä pohtimaan käytettäviä tekniikoita, teknologioita ja ympäristöjä.

#### 4.1 Kehityskohteen etsiminen ja perustelu

Lisäosan suunnittelu aloitettiin katsauksella Atlassianin henkilökunnan käyttämän Jira-töympäristön suggestion-tyyppisiin tiketteihin. Kyseiseen ympäristöön on Atlassianin kuuluttamien avoimien arvojen mukaisesti annettu näkyvyys kaikille, jotka osaavat navigoida oikeaan url-osoitteeseen. Jiran hakutoiminnon avulla etsittiin kaikki suggestion-tyyppiset tiketit, jotka löytyivät Jira Cloud- tai Jira Server -versioihin liittyvistä projekteista ja joiden status ei viittaa siihen, että Atlassian toteuttaisi kyseistä ehdotusta parhaillaan tai harkitsisi ehdotuksen toteuttamista lähitulevaisuudessa. Koodiesimerkissä 4 on nähtävillä hakuehto Jiran advanced search -ominaisuuden käyttämän Jira Query Language, eli JQL-syntaksin mukaisessa muodossa. Kyseisellä hakuehdolla pystyttiin karkeasti suodattamaan instanssista yhteensä löytyvät 233 295 tikettiä niin, että jäljelle jäi vielä 16 859 potentiaalista ehdotusta.

```
project in (JSWCLOUD, JRASERVER, JRACLOUD, JSWSERVER) AND issuetype = Suggestion AND status not in ("In Progress", Resolved, Closed, "Waiting for Release")
```

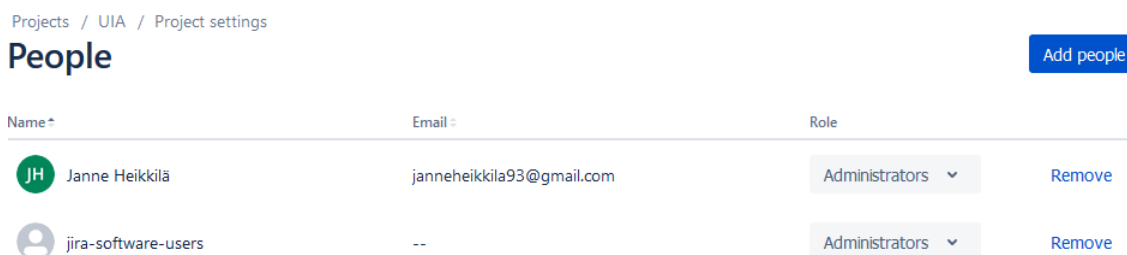
Koodiesimerkki 4. JQL-syntaksin mukainen hakuehto. Koodissa korostettuna oranssilla projektit, vihreällä tikettityypit ja sinisellä statukset, joilla hakua rajattiin.

Jäljelle jääneet tiketit järjestettiin äänimäärän (votes-tietue) mukaisesti isoimmasta pienimpään. Ääniä ja kommentteja tiketeille pystyvät jättämään kaikki Atlassian-tililleen kirjautuneet käyttäjät, joten voimme suoraan olettaa listan kärkipään tikettien olevan toivottuimpia uusia ominaisuuksia. Tämän jälkeen potentiaalista lisäosan kehityskohdetta alettiin etsimään listauksesta manuaalisesti. Prosessi tuotti tulosta ja listauksesta löytyi sekä Server- että Cloud-ympäristöihin liittyviä, ryhmien ja niihin kuuluvien käyttäjien näkyvyyttä koskevia tikettejä. Tämän tyyppiset tiketit kiinnittivät huomioni nopeasti, koska olen itse saanut vastailla aihepiiriin liittyviin kysymyksiin työskennellessäni Atlassian-tuotteiden konsultin roolissa.

Valitun tiketin kehitysehdotus koskee Jiran käyttäjäryhmiä ja yksittäisten projektien oikeushallintaa. Projektien hallinta on tyypillisesti annettu projektien vetovastuullisille, kun taas oikeudet käyttäjähallintaan – johon myös käyttäjäryhmät kuuluvat – on useimmiten vain organisaation Jira-pääkäyttäjillä. Käytännössä tämä tarkoittaa sitä, että projektin vastuhenkilö pystyy lisäämään oikeudet omassa projektissaan käyttäjäryhmälle, mutta



ei pysty näkemään ryhmässä olevia käyttäjiä, ellei hänellä ole myös käyttäjänhallintasi-vustoon oikeuttavia Jiran system administrator -oikeuksia. Käyttäjaoikeuksia voi toki lisätä käyttäjille myös yksitellen, mutta isojen käyttäjämassojen hallitseminen ryhmien kautta on usein huomattavasti yksinkertaisempi vaihtoehto. Kuvassa 5 on havainnollistettuna tapaus, jossa projektin vastuuhenkilö on lisännyt projektinsa admin-rooliin yhden käyttäjän lisäksi yhden käyttäjäryhmän, jonka käyttäjät ja käyttäjämäärä ovat vastuuhenkilöltä täysin pimennossa.



Kuva 5. Kuvankaappaus projektinhallinnasta. Projektin vastuuhenkilö on oikeuttanut projektinsa Admin-rooliin yhden käyttäjän ja yhden käyttäjäryhmän. Ryhmästä ei ole nähtävillä muuta tietoja kuin nimi.

Selailtuani Atlassianin Marketplacea löysin Server-sovellukselle tarkoitetun ilmaisen Group Browser for Jira -lisäosan, joka antaa käyttäjälle näkyvyyden niihin ryhmiin, joiden jäsenenä hän on myös itse. Tämän kaltaisella ominaisuudella pystytään ratkaisemaan kuvailemani näkyvyysongelma, mutta Cloud-sovellukselle vastaavaa lisäosaa ei vielä ole tarjolla.

Lisäosan tarve voidaan siis tiivistetysti perustella Atlassianin Jirasta löytämäni tiketin 175 käyttäjän äänellä, omakohtaisilla kokemuksillani, sekä sillä, että Atlassianin henkilökuntaan kuuluvan Tamara Walshin tiketille antaman vastauksen mukaan Atlassian ei tule itse kehittämään kyseistä ominaisuutta Jira Cloud -sovellukseen lähitulevaisuudessa [18].

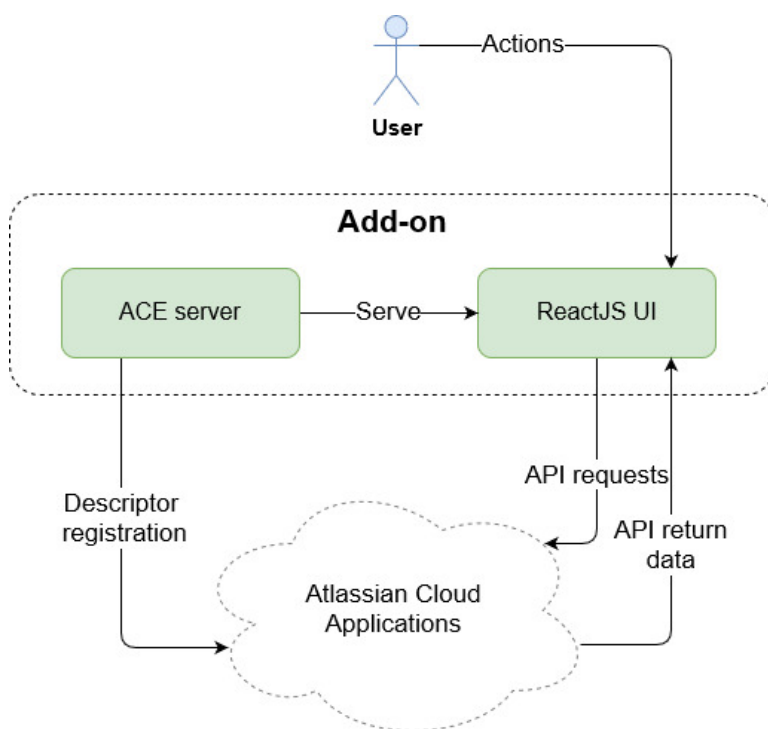
## 4.2 Määrittely

Lisäosan yksinkertaisena perusideana on integroida Jira Cloud -sovellukseen sivu, joka antaa lisäosan käyttäjälle mahdollisuuden tarkastella niiden käyttäjäryhmien jäsentietoja,

joihin hän myös itse kuuluu. Jäsentiedot on pystyttävä tarvittaessa tallentamaan suoraan sovelluksen käyttöliittymän kautta taulukko-ohjelmien kanssa yhteensopivaan tiedostomuotoon, sekä lisäosan käyttöä on pystyttävä rajoittamaan Jiran system administrator -oikeutettujen pääkäyttäjien toimesta ryhmäperusteisesti, jotta mahdollisuudet sovelluksen väärinkäyttöön minimoidaan. Sovelluksen ensimmäisen kehitysversion tarkoituksena on olla pienin toimiva tuote, eli MVP-versio (Minimum Viable Product).

### 4.3 Teknologiat

Lisäosa päätettiin toteuttaa vahvasti JavaScript-ohjelmointikieleen nojaten. Autentikoinnin ja auktorisoinnin toteuttava palvelinsovellus toimii NodeJS-ympäristössä käyttäen hyväksi Atlassianin kehittäjille suuntaamaa, raportin luvussa 3.2.3 esiteltyä Atlassian Connect Express -kirjastoa. Lisäosan käyttöliittymä on rakennettu SPA-arkkitehtuurin (Single Page Application) mahdollistavaa ReactJS-kirjastoa ja ulkoasukomponentteja tarjoavaa, myöskin luvussa 3.2.3 esiteltyä Atlasskit-kirjastoa ja sen React-komponentteja hyödyntäen. Kuvassa 6 on esillä yksinkertaistettu arkkitehtuurikuva sovelluksen tehtävänjaosta.

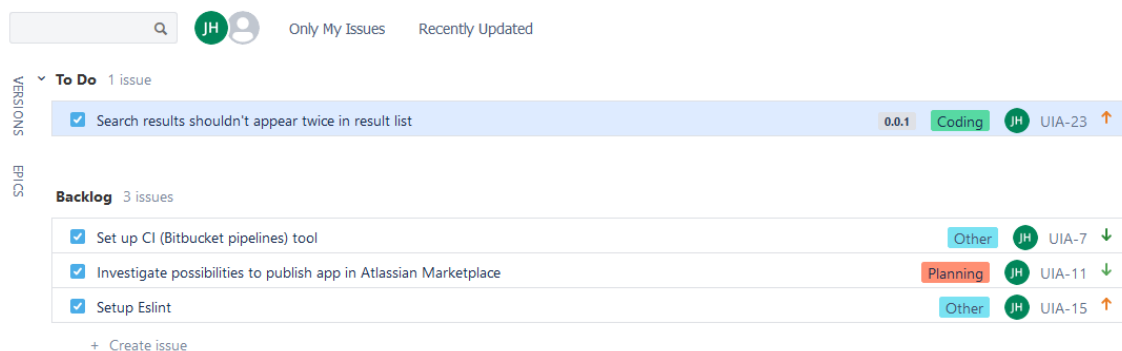


Kuva 6. Arkkitehtuurikuvaus sovelluksen eri osien pääasiallisista tehtävistä.

## 4.4 Ympäristöt

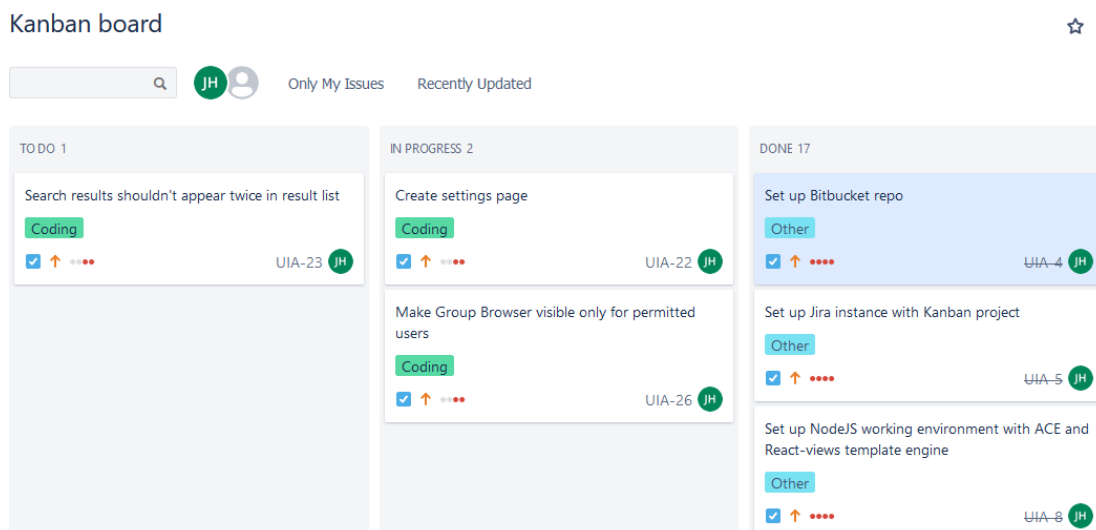
Jira Cloud -lisäosan kehitystä ja testausta varten luotiin uusi Jira Cloud -instanssi, jonka myötä oli myös loogista toteuttaa projektinhallinta saman instanssin sisällä. Koska olin lisäosan ainoa kehittäjä, näin selkeimmäksi vaihtoehdoksi hallita projektia Jiran tarjoaman kanban-aulun (kuva 8) sekä erillisen backlog-näkymän avulla (kuva 7), jonka pääasiallisena tarkoituksena on tässä tapauksessa koota tulevat tehtävät ja selkeyttää niiden priorisointia. Kun tehtävän parissa työskentely aloitetaan, siirretään tiketti backlog-sarakkeesta to do -sarakeeseen, jolloin kyseinen tiketti tulee näkyviin myös projektin kanban-aululle.

### Backlog



Kuva 7. Jiran tarjoama Backlog-näkymä. To Do -sarakeeseen siirrettäessä tiketti siirtyy näkyviin myös Kanban-aululle.

Kuvassa 8 näkyvällä kanban-aululla projektinhallinta haluttiin pitää mahdollisimman yksinkertaisena, jonka myötä taululle luotiin Jiran oletusasetusten mukaiset kolme saraketta, joiden välillä tikettejä liikuteltiin niiden sen hetkisten statusten mukaisesti. Tikeistä on suoraan nähtävissä muun muassa niiden tehtäväkuvaus, kategoria, tavoitteellinen versionumero ja tiketin yksilöivä avain. Tämän lisäksi tiketit ovat sarakkeissaan prioriteettijärjestyksessä niin, että korkeamman prioriteetin tiketit jäävät ylemmäksi.



Kuva 8. Kanban-näkymä, josta selviää tehtävien perustiedot ja niiden sen hetkinen status.

Versionhallintatyökaluksi valittiin Atlassian Bitbucket, koska integraatio Jiran ja Bitbucketin välillä on lähes saumaton. Esimerkiksi tästä voidaan ottaa git commit -toiminto: Kun koodin muutokset päätetään viedä versionhallintaan eli toteuttaa git commit, jätetään commit-viestiksi tiketin yksilöivä avain sekä tiketin mahdolliset kommentit ja/tai sen uusi status. Kuvassa 9 on nähtävissä UIA-14-avaimen tikettiin liitetty commit Jiran näkymästä. Message-sarakkeen alla näkyy aikaisemmin mainittu commit-viesti, jossa tässä tapauksessa määritellään tiketti, johon commit liittyy (UIA-14), tiketin uusi status (#done), sekä tiketin kommenttikenttään jätettävä viesti (#comment ...). Viestin alla näkyvät tiedostot, joihin muutoksia on tehty sekä rivimäärä, joka tiedostosta on poistettu ja lisätty. Tiedostonimeä klikkaamalla Jira ohjaa käyttäjän Bitbucketin käyttöliittymään, jossa muutokset ovat nähtävissä kokonaisuudessaan. Bitbucket tarjoaa myös jatkuvaa integraatiota varten oman, sisäänrakennetun Pipelines-työkalunsa, jota ei lisäosan MVP-vaiheen kehityksessä kuitenkaan vielä otettu käyttöön.

UIA-14: 1 unique commit

JS userinfo-addon		Hide files		
Author	Commit	Message	Date	Files
JH	1428fe4	UIA-14 #done #comment Component created and added in to the layout	11/Mar/19	<a href="#">5 files</a>
+68	-0	package-lock.json		
+2	-0	package.json		
ADDED		views/components/ContentWrapper.js		
+3	-2	views/layout.js		
+25	-2	yarn.lock		

Kuva 9. Jiran käyttöliittymässä näkyvät commit-tiedot.

## 5 Lisäosan kehitys

Luvussa 5 käsitellään lisäosan kehitystyötä erityisesti Atlassian Jira -integraation näkökulmasta, eikä sen tarkoituksena ole pureutua kovinkaan syvällisesti JavaScriptin tai sen pohjalle rakennettujen kirjastojen kuten ReactJS:n toimintaan.

### 5.1 Atlassian Connect Express -kehys

Lisäosan kehitys aloitettiin luomalla valmis, luvussa 3.2.3 esitelty Atlassian Connect Express (ACE) -pohja sovellukselle, jonka jälkeen sovellus oli sovelluspuun juureen credentials.json-tiedostoon asetettavaa Jira-instanssin osoitetta ja sovelluksen haltuun annettavia käyttäjätietoja vaille toimiva. Kuvassa 10 on nähtävillä alustetun ACE-sovelluksen luoma tiedostorakenne. Tämän projektin kannalta suoraan tarpeettomiksi jäivät kansioden public ja views alla olleet tiedostot, koska sovelluksessa käytettiin oletuksena alustettavan Handlebars-sivumoottorin sijaan erillistä selainpohjaista sovellusta ja ReactJS-kirjastoa käyttöliittymän luomiseen.

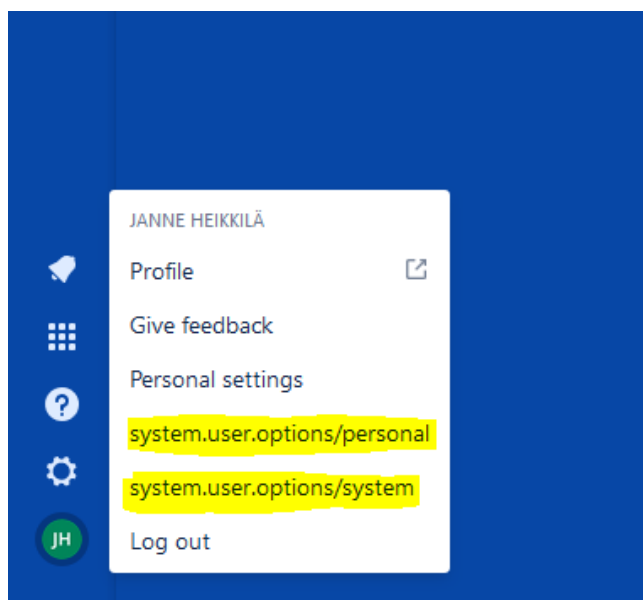
```
1  .
2  |— README.md
3  |— app.js
4  |— atlassian-connect.json
5  |— config.json
6  |— credentials.json.sample
7  |— package.json
8  |— private-key.pem
9  |— public-key.pem
10 |— package.json
11 |— public
12 |   |— css
13 |       |— addon.css
14 |   |— js
15 |       |— addon.js
16 |— routes
17 |   |— index.js
18 |— views
19 |   |— hello-world.hbs
20 |
21 |   |— layout.hbs
22 |
23 |   |— unauthorized.hbs
```

Kuva 10. ACE-kirjaston oletuksena luotu tiedostorakenne.

Lisäosasovelluksen kehittäminen aloitettiin muokkaamalla `atlassian-connect.json`-deskriptoritiedosto halutunlaiseksi. Tiedostosta muokattiin aluksi Jiran Apps-sivulla näkyvät lisäosan perustiedot, joihin kuuluivat sovelluksen yksilöivä avain, nimi, kuvaus, kehittäjä sekä linkki sovellukseen ja sen kotisivuille.

Deskriptorin taulukkomuotoiseen *scopes*-tietueeseen lisättiin siellä oletuksena olevan *read*-oikeuden lisäksi *admin*-oikeus, koska sovellus käyttää Jira Cloud REST API:n *GET /rest/api/3/group/member* -päätepistettä. Rajapinnan dokumentaatio määrittelee jokaisen päätepisteen kohdalla sen tarvitsemat oikeudet ja tässä tapauksessa *admin*-oikeus on auktorisoinnin kannalta välttämätön.

Deskriptoriin tulee määrittää myös moduulit, joiden avulla nidotaan yhteen Jira ja kehitettävä Atlassian Connect -sovellus. Moduulien sijaintien etsimiseen Jira-sovelluksesta on kehitetty Web Fragment Finder -niminen ilmainen lisäosa, joka paljastaa moduulien käyttöliittymään sijoittamisen mahdollistavat osoitepolut niiden omilla paikoillaan. Kuvan 11 esimerkissä näkyy, miten Web Fragment Finder paljastaa polun, jolla pystyn liittämään hakusivun moduulin sille suunnitellulle paikalle sivun vasemmassa alalaidassa sijaitsevan käyttäjän profiilikuvakkeen alle.



Kuva 11. Web Fragment Finder -lisäosan paljastamat sijaintopolut korostettuna kuvassa keltaisella.

Opinnäytetyösovelluksen ensimmäisen version deskriptoriin tuli lisätä kaksi eri *generalPages*-moduulia, koska työhön sisältyi sivu, jossa varsinainen ryhmien haku tapahtuu, sekä admin-valikon alla sijaitseva sivu, josta system administrator -oikeutetut käyttäjät voivat lisätä eri ryhmille käyttöoikeuksia sovellukseen. Admin-valikossa sijaitsevalle sivulle tuli myös yhtenäisen tyylin säilyttämiseksi luoda oma web section -moduuli, joka on tarkoitettu osioimaan tämä lisäosa erilleen muista lisäosan nimen mukaisella yläotsikolla.

Moduulin sijainti- ja perustietojen lisäksi moduuleille asetettiin erilaisia ehtoja (conditions), jotka vaikuttavat moduulien näkyvyyteen eri käyttäjien kohdalla. Settings-sivulle päästääkseen käyttäjällä on oltava Jiran system administrator -oikeudet. My groups -sivun moduulille, joka sisältää lisäosan päätoiminnallisuuden, asetettiin kaksi eri ehtoa: Käyttäjän on oltava kirjautuneena sisään, sekä käyttäjän on kuuluttava johonkin kuvassa 12 määritellyn *allowedGroups*-avaimella löytyvän app properties -tietueen sisältämiin ryhmiin. Mikäli käyttäjä ei täytä moduulille asetettuja ehtoja, moduulin linkki ei ole hänelle näkyvissä ja yritettäessä mennä suoraan linkin osoitteeseen, sovellus antaa access denied -viestin.

Yllä mainittu app properties on Atlassian Connect -kehyksen ominaisuus, jonka avulla lisäosat voivat tallentaa avain-arvo tietueita sovellukseen ilman erillistä tietokantaa [19]. App properties -tietueisiin pääsee käsiksi REST-rajapinnan kautta, mutta ainoastaan sellaisella Atlassian Connect -sovelluksella, jonka deskriptoriin määritelty *addonKey*-tietue vastaa haettavaa kohdetta. Ryhmien tallentamista App Properties -tietueisiin käsitellään käyttöliittymään liittyvässä osiossa myöhemmin.



```

"generalPages": [
  {
    "key": "my-groups-page",
    "location": "system.user.options/personal",
    "name": {
      "value": "My groups"
    },
    "url": "/groups",
    "conditions": [
      {
        "condition": "user_is_logged_in"
      },
      {
        "condition": "entity_property_contains_any_user_group",
        "params": {
          "entity": "addon",
          "propertyKey": "allowedGroups",
        }
      }
    ]
  },
  {
    "key": "my-groups-settings",
    "location": "admin_plugins_menu/my-groups-section",
    "name": {
      "value": "Settings"
    },
    "url": "/settings",
    "conditions": [{
      "condition": "user_is_sysadmin"
    }]
  }
]

```

Kuva 12. Kuvassa on nähtävissä JSON-muotoisen deskriptoritiedoston rakenne sivumoduulien osalta kokonaisuudessaan

Koska lisäosassa päätettiin Handlebars-sivumoottorin sijaan käyttää erillistä käyttöliittymäsovellusta ja ReactJS-kirjastoa, voitiin kaikki Handlebars-viittaukset poistaa palvelinsovelluksen *app.js*-tiedostosta sekä *index.js*-tiedostossa sijaitseviin sivureitityksiin tarjolla ReactJS-sovelluksen koontitiedosto Handlebars-tiedoston renderöinnin sijaan.

## 5.2 Sovelluksen käyttöliittymä

Sovelluksen käyttöliittymä päätettiin toteuttaa JavaScriptillä ja suosituilla, SPA-arkkitehtuurin mahdollistavalla ReactJS-kirjastolla. ReactJS:lle ominaista on toteuttaa sovellus käyttäen mahdollisimman yksinkertaisia, yhtä ominaisuutta toteuttavia ja helposti uudestaan käytettäviä komponentteja [20]. Perinteisen CSS-mallin sijaan sovelluksen tyylien luomiseen otettiin mukaan ReactJS:n apukirjastoksi tarkoitettu Styled Components -kirjasto, joka mahdollistaa CSS-pohjaisten tyylielmenttien luomisen JavaScriptillä. Tämän lisäksi käytössä on aikaisemmin esitelty Atlaskit-kirjasto, joka tarjoaa valmiita, Atlassianin tyylisääntöjä noudattavia komponentteja.

Käyttöliittymäsovelluksessa ReactJS:n taustalla oleellisimmissa rooleissa toimivat Webpack-moduulipaketit sekä Babel-kääntäjä. Babelin tehtävänä on kääntää uudemman sukupolven JavaScript, eli EcmaScript2015 (ES6) ja sitä uudemmat versiot vanhempienkin selaimien kanssa varmasti yhteensopivaan ES5-syntaksiin. Webpack sen sijaan ajaa konfiguraatitiedostossa määritetyt asetukset ja testit, joihin myös Babelin käyttö on määritelty. Webpackin ajettua konfiguraationsa onnistuneesti läpi syntyy tuotoksena konfiguraatitiedostossa output-objektissa määriteltyyn paikkaan koontitiedosto, joka sisältää tiivistetysti kaiken käyttöliittymän koodin muodossa, jota kaikki nykyaikaiset selaimet ymmärtävät.

MVP-mallin määrittelyyn vastaavaa käyttöliittymää suunniteltaessa pyrittiin toteuttamaan yksinkertainen, mutta kuitenkin jatkossa helposti laajennettava sovellus. Sovellus sisältää suunnitelman mukaisesti erilliset sivut käyttäjäryhmien hakemiselle sekä sallittujen käyttäjäryhmien määrittelemiselle. Sivut käyttävät mahdollisuuksien mukaan yhteisiä, valmiiksi tyylielmentyjä komponentteja, jotka eivät pidä sisällään tilanhallintaa, eli ovat ns. stateless-komponentteja. Tästä hyvänä esimerkkinä voimme pitää kuvassa 13 esiintyvää *ContentWrapper.js*-komponenttia, jonka tarkoitus on nimensä mukaisesti luoda sivujen sisällölle yhteinen pohja. Komponentti käyttää hyväksi Atlaskitin sisällön asettelun, osiointin ja responsiivisuuden toteuttamisen avuksi tarjoamia Grid- ja GridColumn-komponentteja sekä Styled Components -kirjaston tapaa luoda tyylielmenty div-elementti. Komponentissa on myös määritelty *{ children }*-muuttujalla, että sille syötetään *props*-arvona se data, jota *ContentWrapper*-komponentin halutaan ympäröivän. *Propsit* ovat ReactJS:n sisäinen ominaisuus datan välittämiseen eri komponenttien välillä.

```

1  import React from 'react';
2  import styled from 'styled-components';
3  import { Grid, GridColumn } from '@atlaskit/page';
4  import { gridSize } from '@atlaskit/theme';
5
6  const Padding = styled.div`
7    margin: ${gridSize() * 4}px ${gridSize() * 8}px;
8    padding-bottom: ${gridSize() * 3}px;
9  `;
10
11 export default ({ children }) => (
12   <Grid>
13     <GridColumn>
14       <Padding>{children}</Padding>
15     </GridColumn>
16   </Grid>
17 );

```

Kuva 13. Esimerkki yksinkertaisesta ja helposti uudelleenkäytettävästä ContentWrapper-komponentista.

Tilanhallinta käyttöliittymäsovelluksessa on page-kansion alle kootuilla sivuilla, jotka näin ollen ovat ns. stateful-komponentteja. Näiden sivujen tarkoituksena on muodostaa sovelluksen näkymät nitomalla yhteen näkymän tarvitsemat komponentit ja välittämällä komponenteille niiden tarvitsema data oikeassa muodossa. Tarvittava data saadaan pääasiassa kommunikoimalla Jiran REST API:n kanssa. Sovelluksen kasvaessa suuremmaksi on varmastikin syytä jakaa näitä toiminnallisuuksia useampien eri komponenttien alle, mutta sovelluksen ollessa vielä suhteellisen pienikokoinen tälle ei nähty tarvetta.

Seuraavissa alaluvuissa katsaukset yllä mainittuihin näkymiin ja niiden tärkeimpiin ominaisuuksiin.

### 5.2.1 Pääsivu

Toinen stateful-komponenteista on sovelluksen päänäkymän kokoava *GroupsPage.js*. Päänäkymään renderöidään sovelluksen nimen ja versionumeron sisältävä otsikko (header), haettavan ryhmän nimeä automaattisesti täydentävä hakukenttä, hakutuloksiin

haluttujen tietueiden valintapainikkeet, Search ja Export to CSV -painikkeet sekä hakutulokset näytävä, lajiteltavissa oleva lista (kuva 14).



Kuva 14. Sovelluksen päänäkökulma. Vasemmassa laidassa näkyvä sininen navigaatiopalkki on osa Atlassian Jiraa.

Kuva 15 on kuvankaappaus *GroupsPage.js*-komponentin elinkaaren (lifecycle) *componentDidMount()*-vaiheesta, jota React-kirjastoa käytettäessä kutsutaan heti, kun komponentti on ladattu (mounted) [21]. *AP.request()*-kutsu käyttää Jiran JavaScript API:n *request*-metodia, joka mahdollistaa IFramen sisällä renderöitävän komponentin XMLHttpRequestin, eli XHR-kutsun ilman tarvetta Cross-Origin Resource Sharing (CORS) -tekniikalle [22]. Ensimmäisellä pyynnöllä haetaan Jiran REST API:sta käyttäjän omat tiedot, joiden joukosta löytyvän *accountID*-arvon perusteella haetaan toisesta saman rajapinnan päätepisteestä ainoastaan ryhmät, joihin käyttäjä kuuluu. Lopuksi kutsun palauttamat ryhmät lisätään Reactin state-olioon *groups*-taulun alle omiksi objekteikseen. Myöhemmässä vaiheessa nämä ryhmät välitetään propseina *AutoSelect*-komponentille, joka taas muodostaa tämän datan perusteella listan, josta käyttäjä voi valita haluamansa ryhmät.

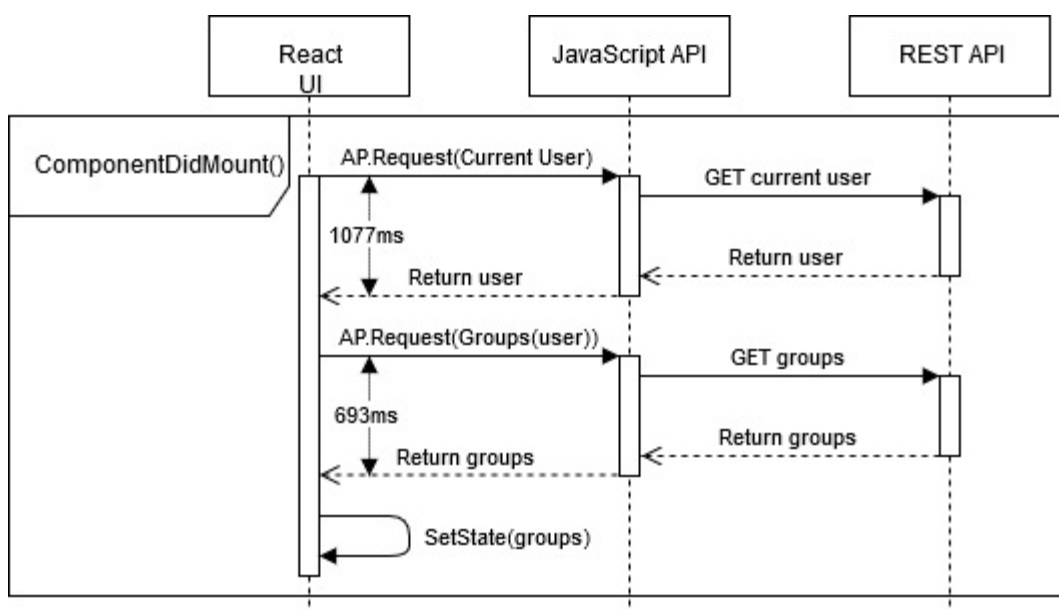
```

componentDidMount() {
  AP.request('/rest/api/3/myself')
    .then(data => JSON.parse(data.body))
    .then(data => AP.request('/rest/api/3/user/groups?accountId='+data.accountId))
    .then(data => JSON.parse(data.body))
    .then(data => this.setState({ groups: data.map(group => ({
      value: group.name,
      self: group.self,
    }))))
    .catch(e => alert(e.err));
}

```

Kuva 15. Kuvankaappaus GroupsPage.js-näkymän komponentista, joka hakee Jiran REST -rajapinnasta tiedot ryhmistä, joihin käyttäjä itse kuuluu.

Kuvassa 16 on nähtävillä *ComponentDidMount()*-tapahtuman rajapintakutsut vasteaikoi-  
neen havainnollistettuna sekvenssikaavion avulla.



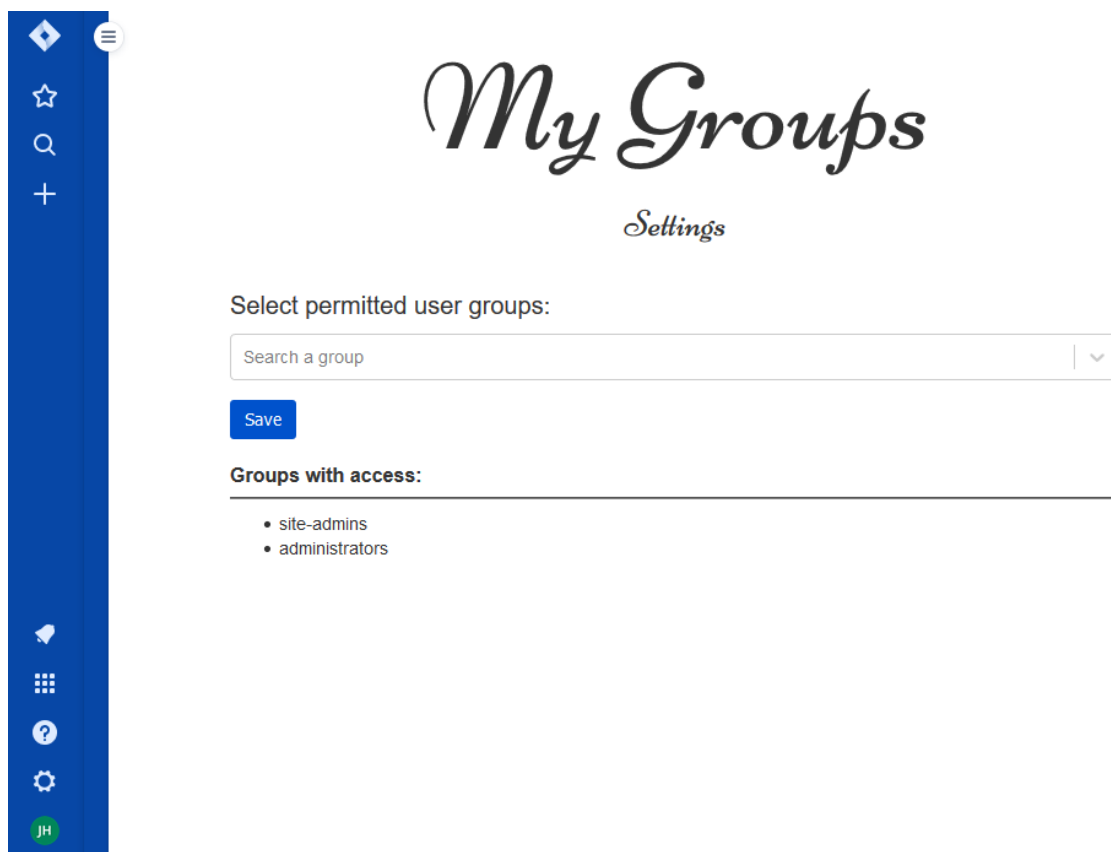
Kuva 16. Havainnollistus aina GroupsPage.js-komponentin latautuessa tapahtuvasta käyttäjien ja -ryhmien hausta JavaScript API:n välityksellä.

Kuvassa 15 näkyvien API-kutsujen lisäksi *GroupsPage.js* kommunikoi Search-painikkeen klikkauksesta myös saman rajapinnan *GET /rest/api/3/group/member* -päätepisteen kanssa, jonka tarkoitus on hakea valittujen ryhmien käyttäjätiedot ja tallentaa ne edellä olevan kuvaesimerkin kaltaisesti state-olioon, josta tiedot voidaan propseina välittää järjesteltävän listarakenteen muodostavalle *ResultList.js*-komponentille.

Lisäosa mahdollistaa käyttäjätietojen viemisen Jirasta CSV-tiedostoon käyttäliittymän Export to CSV -painikkeesta. Ominaisuus on toteutettu käyttäen CSV-muotoilun muodostamisen kanssa hyväksi Node Package Manager -palvelusta löytyvää *convert-array-to-csv*-pakettia. Ominaisuus on myös rakennettu ottamaan huomioon käyttäjän valitsemat tietueiden suodatusasetukset, jonka ansiosta ainoastaan sillä hetkellä aktiivisiksi tietueiksi valitut kirjoitetaan ladattavaan CSV-tiedostoon.

### 5.2.2 Konfiguraatiosivu

Ainoa konfiguraatiovalikolle kohdistettu vaatimus sovelluksen ensimmäisessä versiossa oli mahdollistaa Jira System Administrator -oikeutetuille käyttäjille sovelluksen käyttöön oikeuttavien ryhmien valitseminen, jonka takia kyseinen näkymä jäikin MVP-versioon varsin yksinkertaiseksi (kuva 17). Konfiguraatiovalikon käyttäliittymään lisättiin sama hakukenttäkomponentti kuin päänäkymäänkin, mutta tässä tapauksessa kentän tuli luonnollisesti näyttää kaikki ryhmät riippumatta siitä, kuuluuko admin siihen itse vai ei. Ryhmien valitsemisen ja tallentamisen jälkeen sallittujen ryhmien nimet listataan hakukentän alapuolelle. Ryhmän poistaminen listasta tapahtuu ryhmän nimeä klikkaamalla.



Kuva 17. Konfiguraatiovalikon käyttöliittymä, jossa on myös näkyvissä sovelluksen käyttöön oikeuttavat ryhmät.

Konfiguraatiosivun pinnan alla ainoa päänäkymässä käytetyistä tekniikoista ja työkaluista poikkeava asia lienee Jiran REST API:n app properties -päätepisteen käyttö. Kuten ACE-kehystä käsitelleessä luvussa 5.1 todettiin, app properties -päätepisteen avulla Atlassian Connect -sovelluksesta voidaan tallentaa tietoja ns. sovellusta vasten ilman erillistä tietokantaa. Konfiguraatiosivulla ominaisuutta hyödynnettiin tallentamaan tiedot sallituista ryhmistä kuvan 18 mukaisella PUT-metodilla, sekä hakemaan GET-metodilla aikaisemmin tallennetut ryhmät ja täydentämällä ne sivun listaukseen aina *SettingsPage.js*-komponentin latautuessa. Kuvassa 17 näkyvillä ryhmävalinnoilla lisäosa-sovelluksen app properties -tiedot palauttavat GET-kutsulla alla olevan objektin:

```
{
  "key": "groupsWithAccess",
  "value": {
    "groups": "site-admins,administrators"
  },
  "self": "https://userinfodemo.atlassian.net/rest/atlassian-connect/1/addons/user-lists-addon/properties/groupsWithAccess"
}
```

```
setProperties = () => {  
  const { addonProperties } = this.state;  
  const bodyData = `{"groups": "${addonProperties}"}`;   
  AP.request({  
    url: '/rest/atlassian-connect/1/addons/user-lists-addon/properties/groupsWithAccess',  
    type: 'PUT',  
    contentType: 'application/json',  
    data: bodyData,  
    success: function(responseText){  
      console.log(responseText);  
    },  
    error: function(xhr, statusText, errorThrown){  
      console.log(arguments);  
    }  
  });  
}
```

Kuva 18. App properties -tiedot tallentava API-kutsu.



## 6 Yhteenveto

Työn tavoitteena oli perehtyä Jira-lisäosakehityksen toimintamalleihin ja teknologioihin, etsiä Atlassian yhteisöistä sellainen tarve uudelle ominaisuudelle, jota Atlassian itse ei aio lähiaikoina toteuttaa ja rakentaa tämän tarpeen täyttävä lisäosa. Tarpeeksi määriteltiin ominaisuus, jonka avulla käyttäjä pystyy Jiraan integroidun käyttöliittymän kautta tarkastelemaan ja viemään CSV-muotoiseen tiedostoon niiden käyttäjäryhmien jäsenten perustietoja, joihin hän myös itse kuuluu. Määrittelyyn kuului myös konfiguraatiosivu, jonka kautta System Administrator -käyttäjät pystyvät valitsemaan ja tallentamaan ne käyttäjäryhmät, jotka oikeuttavat lisäosan käyttöön.

Lisäosasovellus pyrittiin tekemään helposti jatkokehitettäväksi, ja sovelluksen ollessa vielä MVP-versio on potentiaalisia kehityskohteitakin monia: nykyiseen export-toimintoon voitaisiin CSV:n lisäksi lisätä muitakin vaihtoehtoja, sovellukselle voitaisiin rakentaa visuaalisesti näyttävämpi ja informatiivisempi ulkoasu ja konfiguraatiosivulla voitaisiin mahdollistaa export-toiminnon rajoittaminen vain tietyille käyttäjäryhmille sekä sellaisten käyttäjäryhmien määrittäminen, joihin ei lisäosankaan kautta voisi saada näkyvyyttä. Edellä mainittujen uusien ominaisuuksien lisäksi sovelluksen kasvaessa sen suorituskykyä olisi varmasti aiheellista testata ja optimoida.

Koen työlle asetettujen tavoitteiden täyttyneen. Lisäosa sisältää määrittelyt täyttävät perusominaisuudet, jonka myötä se olisi valmis käyttäjien testattavaksi. Käyttäjätestivaiheen jälkeen osattaisiin paremmin kertoa, vastaako lisäosa käyttäjien todellisia tarpeita tämän ominaisuuden suhteen riittävän hyvin ja mihin suuntaan kehitystä pitäisi kohdentaa. Atlassian Marketplace julkaisua ajatellen omat resurssini ovat Marketplacen koko huomioon ottaen yksinkertaisesti liian pienet. Optimaalinen tilanne olisi saada lisäosa ensin pienen käyttäjäryhmän kokeiltavaksi ja vasta sen jälkeen laajempaan jakoon.

## Lähteet

- 1 A brief history. 2019. Verkkoaineisto. Atlassian. <<https://www.atlassian.com/company>>. Luettu 4.4.2019.
- 2 Gartner's 2018 Magic Quadrant for Enterprise Agile Planning Tools and the state of agile. 2018. Verkkoaineisto. Atlassian. <<https://www.atlassian.com/blog/jira-software/gartner-enterprise-agile-planning-tools-magic-quadrant>>. Luettu 6.4.2019.
- 3 Understanding Atlassian in the cloud. 2017. Verkkoaineisto. Atlassian. <<https://developer.atlassian.com/cloud/jira/service-desk/architecture-overview/>>. Luettu 6.4.2019.
- 4 Scale Jira with Data Center. 2019. Verkkoaineisto. Atlassian. <<https://www.atlassian.com/enterprise/data-center/jira>>. Luettu 8.4.2019.
- 5 Integrating with Jira Cloud. 2019. Verkkoaineisto. Atlassian. <<https://developer.atlassian.com/cloud/jira/platform/integrating-with-jira-cloud>>. Luettu 9.4.2019.
- 6 Authentication for Connect apps. 2018. Verkkoaineisto. Atlassian. <<https://developer.atlassian.com/cloud/jira/platform/authentication-for-apps/>>. Luettu 9.4.2019.
- 7 Sevilleja, Chris. Getting to Know JSON Web Tokens. 2015. Verkkoaineisto. <<https://scotch.io/tutorials/the-anatomy-of-a-json-web-token>>. Luettu 9.4.2019.
- 8 App Descriptor. 2017. Verkkoaineisto. Atlassian. <<https://developer.atlassian.com/cloud/jira/platform/app-descriptor/>>. Luettu 9.4.2019.
- 9 Security for Connect Apps. 2018. Verkkoaineisto. Atlassian. <<https://developer.atlassian.com/cloud/jira/platform/security-for-connect-apps/>>. Luettu 10.4.2019.
- 10 About the JavaScript API. 2019. Verkkoaineisto. Atlassian. <<https://developer.atlassian.com/cloud/bitbucket/about-the-javascript-api/>>. Luettu 10.4.2019.
- 11 atlassian-connect-express: Node.js package for Express.js based Atlassian Add-ons. 2019. Verkkoaineisto. Atlassian. <<https://bitbucket.org/atlassian/atlassian-connect-express/src/3f57a65201beca90dc49bef3a0e36ee67f0f88da/README.md?at=master>>. Luettu 14.4.2019.

- 12 What is ngrok. 2019. Verkkoaineisto. Ngrok. <<https://ngrok.com/product>>. Luettu 14.4.2019.
- 13 Spencer, Emilee. Updated best practices for Server app design. 2019. Verkkoaineisto. <<https://dac-blog.us-west-2.dev.public.atl-paas.net/blog/2019/01/atlassian-design-guidelines-announcement/>>. Luettu 14.4.2019.
- 14 About Jira modules. 2019. Verkkoaineisto. Atlassian. <<https://developer.atlassian.com/server/jira/platform/about-jira-modules/>>. Luettu 17.4.2019.
- 15 Cheung, Jony. 2018. Verkkoaineisto. <<https://community.developer.atlassian.com/t/how-do-you-develop-and-use-atlaskit-aui/18933>>. Luettu 14.4.2019.
- 16 App approval guidelines. 2018. Verkkoaineisto. Atlassian. <<https://developer.atlassian.com/platform/marketplace/app-approval-guidelines/>>. Luettu 4.5.2019.
- 17 Top Vendor program. 2018. Verkkoaineisto. Atlassian. <<https://developer.atlassian.com/platform/marketplace/top-vendor-program/>>. Luettu 4.5.2019.
- 18 Dave Meyer. 2018. Verkkoaineisto. <<https://jira.atlassian.com/browse/JRA-CLOUD-23783>>. Luettu 3.8.2019.
- 19 Entity Property. 2019. Verkkoaineisto. Atlassian. <<https://developer.atlassian.com/platform/marketplace/top-vendor-program/>>. Luettu 12.9.2019.
- 20 Thinking in React. 2019. Verkkoaineisto. Facebook Inc. <<https://reactjs.org/docs/thinking-in-react.html>>. Luettu 12.9.2019.
- 21 React.Component. 2019. Verkkoaineisto. Facebook Inc. <<https://reactjs.org/docs/react-component.html#componentdidmount>>. Luettu 14.9.2019.
- 22 Request. 2019. Verkkoaineisto. Atlassian. <<https://developer.atlassian.com/cloud/jira/platform/jsapi/request/>>. Luettu 14.9.2019.