Osaamista
ja oivallusta
tulevaisuuden
tekemiseen

Olli Oittinen

# Häiriötiedotesovelluksen suunnittelu

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

15.5.2020

| | |
|---|---|
| Tekijä<br>Otsikko<br><br>Sivumäärä<br>Aika | Olli Oittinen<br>Häiriötiedotesovelluksen suunnittelu<br><br>34 sivua + 3 liitettä<br>15.5.2020 |
| Tutkinto | insinööri (AMK) |
| Tutkinto-ohjelma | Tieto- ja viestintätekniikka |
| Ammatillinen pääaine | Ohjelmistotuotanto |
| Ohjaajat | Lehtori Juha Kämäri<br>Ohjelmistoinsinööri Harri Halttunen |


Insinöörityön aiheena ja tavoitteena oli suunnitella häiriötiedotesovellus huomioiden siihen liittyviä komponentteja rajapinnoista lähtien ja kattaa mahdollisimman laajasti integroitavia osia, kuten käyttäjänhallintaa ja lokitusta. Tavoitteena oli myös antaa sovelluksen kehittäjälle informaatiota mahdollisimman käsitellyssä muodossa perusteluineen, mutta ottamatta kantaa itse toteutukseen tai muihin ratkaisuihin. Työn tilaajana oli suomalainen liikepankki.

Suunnitelman toteutus koostui pääosin yrityksen sisäisiin dokumentteihin, mutta työtä laajennettiin henkilökunnan esittämien huomioiden perusteella. Suunnitelmaan koostettiin nykyisistä käytännöistä ratkaisuja tulevaan sovellukseen. Insinöörityöhön kerättiin myös nykyisten käytäntöjen pohjalta mietittyihin päätelmiin.

Suunnitelman ansiosta sovelluksen toteutukseen on olemassa raamit, joiden puitteissa kehittäjän on helppo lähteä toteuttamaan ratkaisua. Kehittäjälle jää näin myös vähemmän pohdittavaa erilaisista ratkaisutavoista.

Suunnitelman täysimittainen soveltaminen kehitystyöstä käytäntöön on vielä kirjoitusvaiheessa suunnittelun alla. Liiketoiminnan tulee ensin hyväksyä häiriötiedotesovelluksen kehittäminen, eikä suoranaista näkyvää säästöä työn tekemisestä synny välittömästi. Yrityksessä on kuitenkin pohdittu vastaavan ratkaisutavan toteuttamista, joten on todennäköistä, että suunnitelmaa hyödynnetään tulevaisuudessa.



| | |
|---|---|
| Avainsanat | häiriötiedote, REST, LDAP, API, lokitus |

| | |
|---|---|
| Author<br>Title<br><br>Number of Pages<br>Date | Olli Oittinen<br>Designing a Service Disruption Message Application<br>34 pages + 3 appendices<br>15 May 2020 |
| Degree | Bachelor of Engineering |
| Degree Programme | Information Technology |
| Professional Major | Software Engineering |
| Instructors | Juha Kämäri, Lecturer<br>Harri Halttunen, Senior Software Engineer |

The topic and goal of this thesis was to design a service disruption message application, considering related components ranging from application programming interfaces to a wide array of services (e.g. user control and logging) to be integrated. Another goal was to give the future developer information related to the software in a concise manner, but without forcing the use of certain tools or methods. The commissioner of the study was a Finnish commercial bank.

Compilation and the resulting plan were mostly done by using the internal documentation and the notes and comments of the staff in the company. Current methodology and conclusions drawn from current means of publishing a disruption message were also considered, whilst attempting to provide a myriad of options for the future developer.

As a result, there now exists a concrete plan with an underlying framework on how to develop said software, promoting an action driven development when the plan is eventually utilized. It should be considerably easier for the developer to implement solutions regarding the software, as the planning has been done for them.

As for the usability – future is, as of writing, unknown for actual implementation. As the gain for developing a stand-alone application for use is unclear for all parties, the time to develop said software is unknown. Business and corporate office will be the party that eventually approves the scheduling for the development of the service disruption message application. It should be noted, however, that the company has sought out for such an application for quite a long time, thus the adaptation of the set design is likely.

| | |
|---|---|
| Keywords | disruption message, REST, LDAP, API, logging |

## Sisällys

## Lyhenteet

API Application Programming Interface. Ohjelmointirajapinta, joka määrittelee, miten ohjelmisto tarjoaa tietoja ja palveluita.

CORS Cross Origin Resource Sharing. Mekanismi, jonka hyödyntää HTTP-otsikkotietoja vain tiettyjen resurssien (kuten muiden verkkosivujen) sallimiseen.

HTTP-otsikkotieto
HTTP-pyyntöön lisätty kenttä, jonka avulla voidaan täsmentää joitakin verkkosivujen teknisiä tietoja.

IMOC Incident Manager on Call. Yrityksessä muun muassa häiriöistä vastuussa oleva henkilö, kun Customer Service ei ole paikalla.

JIRA Tehtävienhallintaohjelmisto.

Kanava Yrityksen sovellusalusta. Käytettyjä alustoja ovat muun muassa AFI (yrityksen verkkosivusto), VEPA (verkkopankki) sekä Mobile (matkapuhelinsovellus).

LDAP Lightweight Directory Access Protocol. Hakemistopalvelujen käyttöön tarkoitettu verkkoprotokolla. LDAP:n yleisin käyttötarkoitus on käyttäjäntunnistus ja käyttöoikeuksien tarkistaminen.

UID Unique Identifier. Ainutlaatuinen tunniste, jonka avulla tietoja voidaan yksilöidä.

REST Representational State Transfer. HTTP-protokollaan perustuva arkkitehtuurimalli ohjelmointirajapintojen toteuttamiseen.

Splunk Datahallintaohjelmisto.

| | |
|---|---|
| SSO | Single Sign-On, kertakirjautuminen. Menetelmä, jossa pääsy useisiin palveluihin toteutetaan yhdellä autentikoinnilla. |

# 1 Johdanto

Insinöörityö käsittelee häiriötiedotesovelluksen suunnittelua. Häiriötiedotteilla tarkoitetaan tässä työssä tiedotteita, jotka näkyvät kaikille työn tilaajana toimineelle suomalaisen liikepankin asiakkaille, kun ilmoituksia annetaan julkisuuteen. Esimerkiksi verkkopankin palvelinhäiriö tai verkkosivujen käyttökatkos ovat mahdollisia häiriöitä, joista tiedotetaan.

Yrityksellä on käytössään useita julkisia kanavia (VEPA eli verkkopankki, Mobile eli mobiilisovellus jne.), joihin tulee väistämättä häiriöitä. Tällä hetkellä jokaisella kanavalla on oma häiriötiedotejärjestelmänsä, jonka avulla tiettyyn kanavaan voidaan laittaa erillinen häiriötiedote.

Henkilö, joka haluaa ilmoittaa useaan eri paikkaan kohdistuvasta häiriöstä, joutuu tekemään saman ilmoituksen moneen otteeseen: esimerkiksi suunniteltu verkkopankin päivitysoperaatio joudutaan ilmoittamaan erikseen mm. VEPA- sekä Mobiili- ja AFI-järjestelmiin (AFI ≈ yrityksen verkkosivustot). Tämä vie resursseja muista toimenpiteistä.

Yrityksessä on kuitenkin jo jonkin aikaa toivottu menetelmää, jossa yksi häiriötiedote voidaan lähettää samanaikaisesti moneen eri kanavaan, jolloin työmäärä vähenisi ja häiriötiedotteiden käsittely helpottuisi.

Yritys uudistaa insinöörityötä kirjoittaessa AFI-kanavaansa, joten hetki häiriötiedotteita käsittelevän suunnitteluun oli otollinen: uudistuksen jälkeen on helpompi rakentaa uutta. Yritykselle tehtävässä suunnitelmassa käsiteltiin vain julkisia häiriötiedotteita.

## 2 Nykyiset järjestelmät

Yrityksen käytössä on useita eri kanavia, joita yritys hyödyntää palveluidensa ylläpitoon ja kehittämiseen. Oli siis loogista aloittaa työ perehtymällä nykyisten häiriötiedotejärjestelmien toimintaan hieman tarkemmin.

Jokaisella yrityksen kanavalla on yrityksen sisäisesti omat tiiminsä, jotka ovat vastuussa kanavien kehittämisestä. Esimerkiksi web team on vastuussa AFI-kanavasta, ja mobile team on luonnollisesti vastuussa mobiilisovelluksien kehittämisestä. Kehittäjätiimi ei kuitenkaan ole vastuussa häiriötiedotteiden tekemisestä, vaan häiriötiedotteiden laatiminen, eli *julkaisu* delegoidaan IMOC-henkilöille ja Customer Service -puolen henkilöille.

Nykyisessä tilanteessa häiriötiedotteet joudutaan viemään jokaiseen yrityksen käyttämään kanavaan erikseen. Jos häiriö siis koskee vaikkapa verkkopankkia, AFI-kanavaa sekä jotakin mobiilisovellusta, joutuu häiriöistä vastuussa oleva taho tekemään erillisen häiriötiedotejulkaisun jokaiseen kanavaan erikseen. Tämä ei sinällään haittaa, jos häiriötiedote joudutaan tekemään vain yhteen kanavaan, mutta on tehotonta laatia erikseen häiriötiedotteita moneen eri kanavaan kerralla. Kanavilla on omat julkaisumenetelmänsä, jotka saattavat poiketa tai olla poikkeamatta toisistaan.

Yrityksen sisäisessä käytössä on valmiita tekstipohjia käytettävänä, joihin on liitetty tunniste löytämisen helpottamiseksi; kuitenkin sellaisenaan tapa on kömpelö ja aikaa vievä. Tämän lisäksi tekstipohjat on dokumentoitu yrityksen omilla wikisivuilla, ja tiedon etsiminen saattaa olla kiiretilanteessa tuskastuttavan hidasta. Ongelmaksi on myös huomattu, että sivujen vanheneminen ja deprekoituminen (eli käytännössä hylkääminen) johtavat tiedon hajaantumisen eri paikkoihin – esimerkiksi AFI-kanavaan liittyvien häiriötiedotteiden osalta pari vuotta sitten deprekoituneeksi merkittyä wikisivua päivitetään edelleen, vaikka uudelle sivustolle on deprekoituneeksi merkityn sivun ylälaidassa linkki. Tietoa ei ole kootusti missään; omille kanaville on tehtynä omat wikisivunsa, joiden alisivuilta joudutaan kahlaamaan häiriötiedotteiden sisältöä.

Seuraavana on lyhyesti mainittuna nykyisiä häiriötiedotteiden julkaisumenetelmiä kanavakohtaisesti.

### 2.1 Verkkopankki- ja AFI-kanavien häiriötiedotejärjestelmät

Verkkopankin ja AFI-kanavan osalta käytetään Liferayn tuotteilla tehtyä alustaa. Liferay on yhdysvaltalainen ohjelmistoyritys, jonka alustaa voidaan kuvata verkkosivujen sisällönhallintajärjestelmäksi [1]. Häiriötiedotteen julkaisusta vastaava henkilö joutuu siis hyödyntämään Liferayn käyttöliittymää ja manuaalisesti julkaista häiriötiedote kumpaankin kanavaan.

### 2.2 Mobiilikanavien häiriötiedotejärjestelmä

Mobiilisovellukset on tuotettu natiiveina sovelluksina, joten niiden osalta häiriötiedotejärjestelmä on käytännössä itse rakennettua. Kanaviin ei siis liity mitään verkkopankin tai muun kanavan kaltaista ulkopuolista sisällöntuotannon kaltaista ratkaisua, vaan jokainen häiriötiedote annetaan viestinä sovellusta käyttäessä sille varatussa tilassa.

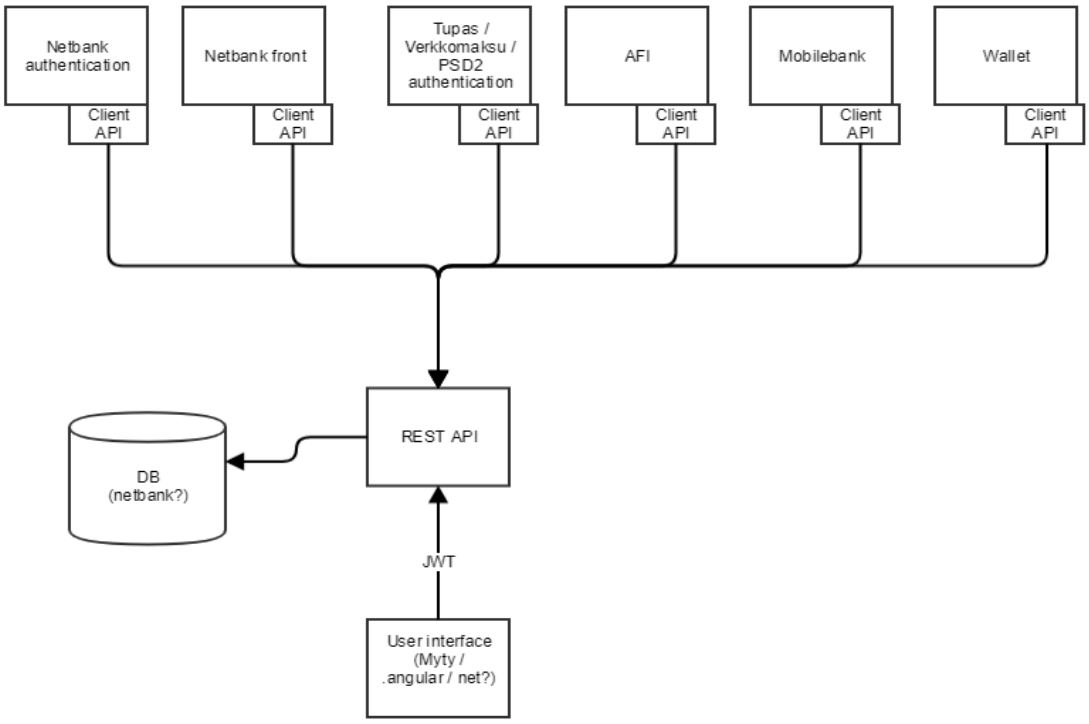
# 3 Uuden häiriötiedotesovelluksen suunnitelma

Uuden häiriötiedotesovelluksen kehittämistä on suunniteltu jo pitkään. Aikaisimmat tiedot sovelluksen kehittämiseen on esitetty jo vuoden 2019 alkupuolella, ja alustava pohjarakenne tehtiin pian tämän jälkeen. Sovellusta ei vain ole koskaan tehty, sillä sen tekemiseen ei ole ollut resursseja saatavilla, eikä suunnitelmaa koskaan viety sen pidemmälle.

Suunnitellulle häiriötiedotesovellukselle määriteltiin kuitenkin jo alkuun neljä eri vaatimusta:

- luoda häiriötiedotteita
- poistaa häiriötiedotteita
- muokata häiriötiedotteita
- ajastaa häiriötiedotteita.

Vaatimusten lisäksi alkuperäiseen suunnitelmaan kehitettiin alustava ratkaisu sovelluksen toteuttamiseksi, joka löytyy kuvasta 1.




Kuva 1. Alustava kokonaiskuva sovelluksen kokonaisuudesta

Kuvassa häiriötiedotesovellusta hyödyntävillä järjestelmillä on yhteys REST-rajapinnan kautta tietokantaan. Häiriötiedotesovellukselle kehitettäisiin käyttöliittymä, jonka avulla sovelluksen käyttäjä käyttäisi REST-rajapintaa hyväkseen tietueiden siirtämisessä tietokantaan. Jokaiseen kanavaan jouduttaisiin kuitenkin erikseen luomaan oma asiakasrajapinta (engl. *client API*), jonka kautta häiriötiedotesovelluksen tietokannan tietoja saataisiin käytettyä.

## 3.1 Häiriötiedotesovelluksen tekemisen perustelut ja hyödyt

Yrityksellä on jo pitkään ollut erilaisia häiriötiedotekanavia, joihin väistämättä tulee häiriöitä. Häiriöt johtuvat sekä inhimillisistä syistä että syistä, joihin on ollut mahdotonta vaikuttaa yrityksen omasta toimesta. Jotkin häiriöt kohdistuvat asiakkaisiin asti, ja yritys pyrkii luonnollisesti minimoimaan vahinkoja mahdollisimman ripeällä aikataululla.

Työhön käytettiin pääosin yrityksen henkilökunnan esittämiä määritelmiä, näkemyksiä sovelluksen kokonaisuudesta [2;3], ja yrityksen sisäisiä dokumentteja nykyisistä käytännöistä. Lisäksi hyödynnettiin verkosta saatavaa materiaalia asioiden perusteluun ja hyväksyntään.

Insinöörityön luonteen vuoksi häiriötiedotesovelluksen lopullinen toteutusmuoto pyrittiin jättämään mahdollisimman vapaatulkintaiseksi. Tarkoituksena ei ollut rajoittaa sovelluksen kehittäjiä tiettyyn arkkitehtuuriin, työskentelytapaan tai muuhun ratkaisuun, vaan laatia raamit, joiden puitteissa toimia.

Selkeimmät hyödyt tästä insinöörityöstä kohdistuvat kehittäjille, sillä heidän on huomattavasti helpompaa tehdä sovellus (tai muu ratkaisu), kun siihen on jo olemassa suunnitelma. Toisaalta työstä hyötyvät loppupeleissä myös IMOC ja Customer Service, joihin lopullinen ratkaisu pääosin kohdistuu.

Selkeää suoraa taloudellista merkitystä työllä ei ole; sovellus joudutaan kuitenkin vielä tekemään, testaamaan ja tekemään mahdollisimman helppokäyttöiseksi (lue: yksinkertaiseksi), mikä taas itsessään on haastavaa ja aikaa vievää. Kehittäjien vaatima aika ja kustannukset on poissa muista projekteista, joten selkeää aikataulua ei välttämättä saada välittömästi.

Vaikka nykyinen järjestelmä saataisiinkin korvattua uudella, eivät sen ylläpito ja kustannukset sellaisenaan katoa. Häiriötiedotteet voidaan suunnitelman ansiosta yhtenäistää sellaisiksi, joissa samaa häiriötiedotetta hyödynnetään useassa eri kanavassa samanaikaisesti. Lopullinen ratkaisu on helppokäyttöisempi ja nopeampi kuin nykyinen järjestelmä, joten häiriötiedotesovelluksen avulla säästetty aika voi olla merkittävä.

## 3.2 Häiriötiedotesovelluksen kokonaiskuva

Häiriötiedotesovelluksen tarkoituksena on tarjota rajapinta, jota hyödyntämällä jokainen yksittäinen kanava pystyy hakemaan tietokannasta häiriötiedotteet, ja niistä saatujen tietojen perusteella luomaan ja päivittämään ilmoituksia.

Alkuperäisen suunnitelman mukaan sovelluksella tuli myös poistaa häiriötiedotteita sitä käyttävästä kanavasta. Kävi kuitenkin nopeasti ilmi, ettei häiriötiedotteita kannattaisi sellaisenaan poistaa samalla tietokannasta, jotta häiriötiedotteita voitaisiin käsitellä myös eräänlaisena historiikkina. Liiallista tallennusta haluttiin kuitenkin välttää, sillä tästä oli heikkoja kokemuksia – eräässä sovelluksessa oli tallennettu jokainen sovellukseen liittyvä pyyntö ja käsittely, jolloin lopputuloksena oli tietokannan nopea laajeneminen.

Kaikissa nykyisissä asiakkaalle näkyvissä häiriötiedotteissa on mukana nk. Content-kenttä (Kuvaus-kenttä), otsikkotieto, jonka perusteella häiriötiedotteet valitaan, sekä sisäinen tunniste-ID, jonka avulla em. tietoja yhdistellään tietokannassa. Luonnollisesti mukana kulkee myös ajankohta (tai ajankohtia), mutta sitä ei eksplisiittisesti lisätä – muutamin poikkeuksin.

Häiriötiedotteen vähittäisvaatimuksia ovat siis

- otsikko
- kuvaus
- ajankohta, sisältäen sekä päivämäärän että kellonajan sekä
- sisäinen uniikki tunniste.

Otsikko ja kuvaus tuli kuvata kolmella eri kielellä – suomi, ruotsi ja englanti, jotta saavutettaisiin kaikki yrityksen asiakkaat. Tämä oli ollut tuleva käytäntö jo jonkin aikaa, mutta kaikkiin häiriötiedotteisiin ei ollut englanninkielisiä käännöksiä, ja häiriötiedotteet oli tehty vain suomeksi ja ruotsiksi.

Suunnitelmaa laatiessa kävi myös ilmi, että häiriötiedotteen senhetkistä tilaa tulee pystyä hallinnoimaan, eli häiriötiedote tulee pystyä merkitsemään sekä alkavaksi että päättyneeksi. Helpoin esittämistapa, ja tapa joka suunnitelmaan päätyi lopulliseksi ratkaisuksi, oli kronologinen asteikko: Tuleva – Meneillään – Päättynyt.

Päättymisajankohtaa ei tarkoituksella otettu mukaan vähittäisvaatimuksiin: on lähestulkoon mahdotonta tietää, milloin häiriö oikeasti päättyy – ajastetuissakin (tulevissa) tiedotteissa päättymisaika on parhaimmillaan arvio. Tästä huolimatta oli syytä mahdollistaa ajastettujen häiriötiedotteiden automaattinen päättyminen; häiriön kesto usein tiedetään melko hyvin esimerkiksi verkkopankin suunnitelluissa päivityksissä, jolloin tiedote

voidaan asettaa päättymään automaattisesti. Viestin kuvauksessa on tällöin tosin syytä mainita, että katkos päättyy viimeistään häiriötiedotteen poistuessa.

Koska on työlästä kirjoittaa jokainen häiriötiedote manuaalisesti aina kun sellainen halutaan julkaista, on huomattavasti nopeampaa ja tehokkaampaa käyttää valmiita häiriötiedotteita, joita (mahdollisesti täydentämällä) saadaan toivottu lopputulos. Tällöin käännöksistäkään ei tarvitse erikseen huolehtia, sillä ne saadaan automaattisesti häiriötiedotepohjasta.

Osa yrityksen nykyisistä kanavista käyttävät id-tunnisteita omissa häiriötiedotteissaan, jonka pohjalta esitäytettiin sovelluksen käyttäjälle tekstipohja. Oli siis selvää, että uuden häiriötiedotesovelluksen tuli pystyä hyödyntämään vastaavaa ominaisuutta: tietyn tunnisteen perusteella näytettäisiin tietty häiriötiedote.

Lopullisessa ehdotetussa ratkaisussa oli tehdä häiriötiedotepohjia, joiden perusteella voitaisiin julkaista häiriötiedotteita. Pohjien avulla tietokantaan tallennettaisiin joitakin huolella valikoituja häiriötiedotteita, jotka olisivat sisällöltään riittävän yleisiä, jotta niitä voitaisiin käyttää monessa eri kanavassa, mutta samalla sellaisia, että niistä tulisi selkeästi esille tarpeellinen tieto. Valmiiden häiriötiedotteiden ansiosta tiedotteet itsessään toimivat tällöin eräänlaisena historiikkina, eikä erillistä historialokia tarvittaisi.

Joissakin häiriötiedotteissa on erittäin tärkeää, että esimerkiksi kellonaika ja päivämäärä ovat merkitty oikein, joten niiden esittäminen ennen julkaisua erillisessä kentässä otettiin huomioon. Suunnitelman mukaan sovellus tekee tietojen pohjalta automaattisesti uuden tiedotteen kenttiä täydentämällä. Täydennettävissä tiedotteissa oli erittäin tärkeää tarkistaa jo hyvissä ajoin, että ajankohta tulee varmasti merkittyä tiedotteeseen ja että paikkateksti (engl. *placeholder text*) muuttuu varmasti oikeaksi ennen julkaisua. Suunnitelman mukaisesti sovellus itse tarkistaa oikeellisuuden eikä jätä tätä tiedotepohjan luojan vastuulle, vaikkakin pohjan tekijän on myös hyvä tarkistaa ajankohdan täsmääminen.

Häiriötiedotepohjassa paikkatekstillä tarkoitetaan sellaista kohtaa tekstissä, joka tulee korvata uudessa häiriötiedotteessa. Suunnitelmassa häiriötiedotepohjaan lisätään kulmasuluilla ympäröity avainsana, jotka tiedotetta luotaessa korvataan toisella arvolla. Esimerkiksi häiriötiedotteen alkuajankohta lisätään pohjaan avainsanalla <alkuaika>, jolloin

tiedotteessa itsessään ei tätä avainsanaa näy, vaan sen tilalle tulee vaikkapa 31.5.2030 klo 15.30. Tiedotteen julkaisija voi kuitenkin valita, tuleeko kellonaika näytettävään häiriötiedotteeseen mukaan vai ei. Vastaava toimenpide tehdään luonnollisesti myös päättymisajankohdalle.

Suunnitelman mukaan häiriötiedotepohjan laatijan on syytä varmistaa, että paikkatekstit tulevat kaikkiin kieliversioihin. Linjaus koskee myös sellaisia häiriötiedotteita, joissa otsikko muuttuu käytettävien ajankohtien mukaan, joten jää tiedotteen julkaisijan vastuulle päättää, tuleeko kellonaika mukaan itse tiedotteeseen vai ei.

Valmiiden tiedotteiden ansiosta julkaisu yksinkertaistuu sellaiseksi, jossa käyttäjän tarvitsee helpoimmillaan vain valita häiriötiedote, julkaistavat kanavat ja alkamisajankohta. Tällöin julkaisuvaiheessa mitään muuta kuin alkamisajankohtakenttää ei voida muuttaa, jolloin inhimillisiä virheitä voidaan ehkäistä mahdollisimman pitkälle, mikä tekee sovelluksesta käyttäjäystävällisemmän. Häiriötiedotteista vastuussa olevat henkilöt voivat helposti ja nopeasti saada uusia tiedotteita julkaistua useaan eri kanavaan samanaikaisesti.

Suunnitelmaa laatiessa löytyi yrityksen omilta wikisivuilta tekstit, jotka ovat nykyisellään käytössä häiriötiedotteissa, tosin ilman englanninkielisiä käännöksiä. Tekstit täytyisi vain viedä tietokantaan, jolloin niiden käyttö olisi yksinkertaista.

### 3.3 Tiedon tallennus, luonti ja järjestely

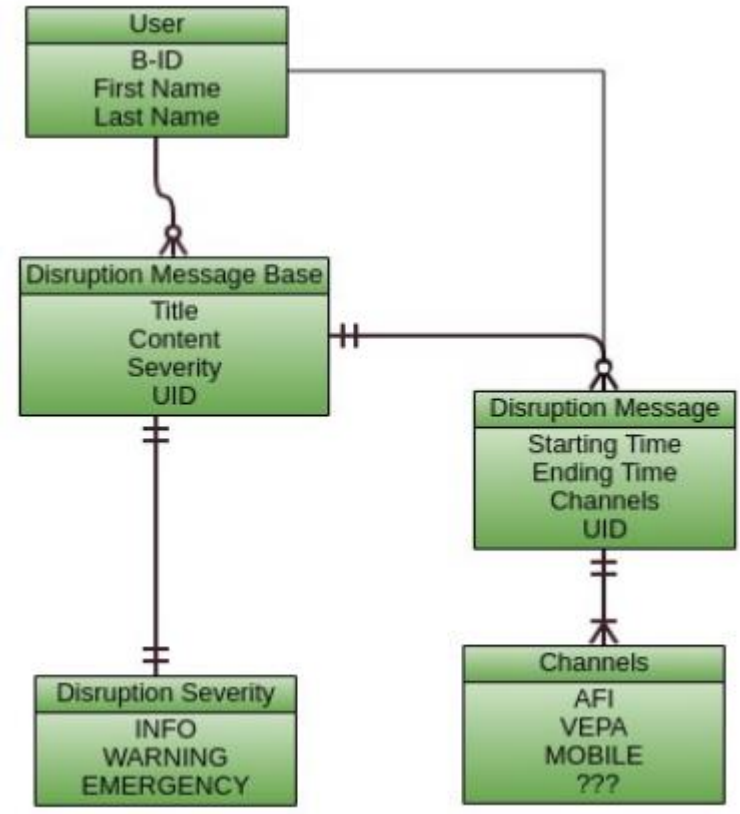
Vaatimusten perusteella voitiin muodostaa tietokantarakenne, joka käsittelee myös kaikki erikoistapaukset. Selkein ratkaisu oli hyödyntää SQL-kantaa tietokantana [2;3], sillä jokaisen tietueen arvo jokaisessa kohdassa on aina tiedossa, eikä tietokantaan odoteta kohdistuvan suuria muutoksia. On myös kriittistä, että tieto pysyy eheänä ja yhtenäisenä. Tällöin voidaan helpommin noudattaa ACID (*Atomicity, Consistency, Isolation, Durability*; suom. atomisuus, eheys, eristyneisyys, pysyvyys) -periaatteen mukaista lähestymistapaa, jossa jokainen tietokannan käsittely tulee hoidettua korrektisti, myös häiriötilanteissa [4].

Atomisuus takaa, että jokainen transaktio (hakujen ja tallennusten sarja) joko onnistuu tai epäonnistuu kokonaisuudessaan, eikä osittaisia muutoksia hyväksytä. Eheydellä

taataan, että tietokanta siirtyy transaktioiden myötä yhdestä eheästä tilasta toiseen samanlaiseen. Eristyneisyydellä tarkoitetaan transaktioiden käsittelyä omina itsenäisinä komponentteinaan; ne eivät vaikuta toisiinsa ja toimivat toisistaan riippumatta. Pysyvyys taas puolestaan takaa, että transaktion jälkeen sen aiheuttamat muutokset jäävät voimaan eivätkä katoa järjestelmästä mihinkään edes häiriötilanteissa.

Suunnitelmaa tehdessä kävi nopeasti ilmi, että häiriötiedotesovelluksessa ei ole syytä käyttää NoSQL-pohjaista ratkaisua [5.;6.]: on kriittistä, että tieto säilyy tismalleen haluttuna eikä ”sinne päin”. Kun tiedetään, että data on aina tietynlaista, on syytä hyödyntää SQL-pohjaista ratkaisua. On tosin syytä huomata, ettei tämä tapa ole ainoa: mikäli hakukyselyistä halutaan vaikkapa tekstipohjaisempia, löytyy siihen muita mahdollisuuksia. Näitä mahdollisuuksia ei kuitenkaan lähdetty sen enempää tutkimaan, sillä SQL-ratkaisu tuntui kaikista osapuolista järkevimmältä.

Koska erotus häiriötiedotepohjiin ja häiriötiedotteisiin tehtiin jo varhain suunnitelmaa tehtäessä, oli yksinkertaista mallintaa tietokantarakenne, jota häiriötiedotesovellus käyttäisi. Suunnitelman lopullinen ratkaisu on esiteltynä kuvassa 2, mutta tätä ennen ratkaisua muokkailtiin useaan otteeseen, ja esimerkiksi käyttäjätaulun lisäys suunnitelmaan tuli vasta LDAP-integraation (engl. *Lightweight Directory Access Protocol*) yhteydessä.




Kuva 2. Esitetty tietokantarakenne

Häiriötiedotteiden vakavuusaste otettiin mukaan varmuuden vuoksi, sillä sen käytöstä ei ole täyttä varmuutta. Nykyisissä häiriötiedotteissa on käytetty kaksi- tai useampiportaista järjestelmää, mutta niiden hyödyntäminen on jäänyt vajaaksi. Esimerkiksi AFI-kanavassa vakavuusaste on merkitty erikseen asteikolla hätätiedote ja häiriötiedote, mutta vain jälkimmäistä on käytetty. Mikään nykyinen tiedote ei siis ole kyseessä olevassa kanavassa hätätiedote.

Kanavataulua ei haluttu ottaa häiriötiedotepohjaan suoraan, vaan jokaisen häiriötiedotteen kohdalla julkaistavat kanavat haluttiin määrittää erikseen. Kanavat puolestaan määritettiin jo alkuperäisessä suunnitelmassa (kuva 1), tosin tähän pohdittiin myös FTN-alustaa eli Luottamusverkostoa (engl. *Finnish Trust Network*) [7]. FTN-alustan hyödyntäminen häiriötiedotesovelluksen osalta jäi epäselväksi, joten sen liittäminen sovellukseen päätettiin jättää ajatustasolle.

Peruskomponentit ovat jokaisen tiedotteen kohdalla samat: kantaan tallennetaan häiriötiedotepohjia, joiden perusteella yhteen tauluun tallennetaan häiriön otsikko, sisältö eli kuvaus, ja häiriön vakavuusaste. Yhdestä häiriötiedotepohjasta voidaan saada monta eri häiriötiedotetta, jonka ajankohdat ja kanavat voivat olla keskenään erilaisia.

Suunnitelmassa kävi ilmi muutamia seikkoja tietokannan tietueiden kanssa: ensinnäkin nk. teknistä ID:tä ei haluttu sellaisenaan ratkaisuksi UID:ksi, sillä tekninen ID (lue: tietokannan generoima valmis ID) ei kerro sellaisenaan yhtään mitään tietueesta, jolloin ID:n arvo jää lähes olemattomaksi muuten kuin yksilöinnissä.

Toisekseen: nykyisellään häiriötiedotetyyppejä on kymmenkunta. Häiriötiedotepohjien osalta teknisen ID:n ongelma ratkaistiin jaottelemalla häiriötyypit karkealla tasolla tunnistautumishäiriöihin, kanavakohtaisiin häiriöihin, korttihäiriöihin ja muihin häiriöihin. Alkutunnisteet tapauksille olivat AUTH (tunnistautumishäiriöt), CARD (korttihäiriöt), DIGI (digitaalisiin palveluihin kohdistuvat häiriöt, eli kanavakohtaiset häiriöt) sekä OTHR (muut häiriöt). Suunnitelmassa huomattiin myös, että häiriötiedotteiden laajuus voi myös olla yhdistelmä edellä mainittuja häiriötyyppejä.

Kolmanneksi huomattiin, että olisi mahdollisesti tarvetta jaotella erikseen sellaisenaan julkaistavat häiriötiedotteet ja julkaistessa muokatut häiriötiedotteet jo avaimen perusteella, joten päätettiin lisätä yksikirjaiminen arvo avaimen alkuun, joka kertoisi, onko pohjaa ollut tarvetta muokata julkaistaessa vai ei. Ratkaisuksi tuli kaksi eri vaihtoehtoa; joko N-kirjain, merkiten muokkauksen tarpeettomuutta (*No Editing*), ja R-kirjain, jolla muokkaus merkittäisiin tarvittavaksi (*Requires Editing*). Pohjan UID voitiin tällöin luoda tietty alku ja lisätä siihen juokseva numero, esimerkiksi NAUTH-12345.

Häiriötiedotteiden teknisen ID:n ongelman osalta ratkaisuksi esitettiin nk. yhdistelmäavainta (engl. *Composite Key*) [8.], jossa avaimeen sisältyy kaksi osaa: ensimmäiseen osaan sisällytetään käytetty häiriötiedotepohja ja häiriön vakavuusaste, ja toinen osa voidaan generoida esimerkiksi tekniseksi tunnisteeksi. Näin saadaan yhdellä avaimella kerrottua runsaasti tietoa, mutta jokainen tietue on kuitenkin yksilöity. Tällöin lopullisen avaimen muodoksi tulisi esimerkiksi NAUTH-12345-INFO-AZ7PH6.

Häiriötiedotteiden lopullisen rakenteen (taulukko 1) perusteella voidaan näyttää käyttäjille kaikki tiedotteeseen liittyvät asiat kerralla, ja toisaalta tietyt tietueet voidaan jättää näyttämättä asiakkaalle päin.

Taulukko 1. Häiriötiedotteen kokonaisrakenne

| **Tietue** | **Tyyppi** | **Huomioita/Arvot** |
|---|---|---|
| UID | String | Kaksiosainen: alkuosa pohjan, vakavuusasteen ja käytettyjen kanavien perusteella, ja loppuosa yleinen, jonkin perusteella generoitu → yksilöi tiedotteen |
| Vakavuusaste | Enum | Pohjan perusteella toisesta taulusta |
| Otsikko | String | Kolmella eri kielellä: suomi, ruotsi, englanti |
| Kuvaus | | |
| Alkamisajankohta | Date | ISO-8601 -mukainen |
| Päättymisajankohta | | |
| kanavat | Enum | Lista. Voi olla useita arvoja, mutta vähintään yksi |

Ajankohdan asettamiseksi esitettiin ISO-8601 -standardin mukaista aikamääritelmää, jolloin tekninen esitysmuoto on VVVV-KK-PP**T**HH:MM:SS[**Z**/±HH:MM], jossa T-kirjain erottaa päivämäärän ja ajan toisistaan, ja Z-kirjain edustaa koordinoitua yleisaikaa (UTC, engl. *Coordinated Universal Time*). Koordinoitu yleisaika voidaan myös vaihtaa vastaamaan aikavyöhykkeiden aikaansaamaa aikaeroa; tällöin esimerkiksi ensimmäinen huhtikuuta vuonna 2000 klo 18:00 Suomen aikaa kirjoitetaan muotoon 2000-04-01T18:00:00+02:00. Standardia käytetään jo muun muassa Mobiilipankin puolella, eikä ole viisasta käyttää erilaisia tapoja esittää aikaa eri sovelluksissa.

Suunnitelmassa pyrittiin myös ottamaan huomioon nykyinen arkkitehtuuri: kuvassa 2 näkyvää User-taulua ei tarkoitettu vietäväksi tietokantaan sellaisenaan, vaan käyttäjän tiedot voitaisiin hakea LDAP-integraatiolla. LDAP on verkkoprotokolla, jonka yleisin käyttötarkoitus on käyttäjäntunnistus ja käyttöoikeuksien tarkistaminen. LDAP:n avulla voidaan siis varmistaa, kuka sovellusta saa käyttää, mikä on suotavaa häiriötiedotteiden julkaisemisessa. Näin oli tehty lukuisten muidenkin järjestelmien kanssa (mainittakoon esimerkkinä sähköposti), joten olisi erikoista olla hyödyntämättä mahdollisuutta tässäkin.

Häiriötiedotteen laatija haetaan siis LDAP-integraatiolla olemassa olevista järjestelmistä. Laatijatietoa ei kuitenkaan ole merkitty tiedotteen lopulliseen rakenteeseen tarkoituksella – kyseessä oleva tieto haetaan eri paikasta.

Taulukossa ei ole mainittu tiedotteen historiasta tarkoituksella. Häiriötiedotteen historia riippuu käytettävästä tietokannasta, mutta harvoin on tapauksia, joiden historian säilyttäminen sellaisenaan olisi tarpeellista. Tietokantaan voidaan tallentaa senhetkisen version numero, joka on itsepäivittyvä juokseva numero. Historiatietokannan ylläpitäminen on tosin työlästä ja hintavaa: tästä oli esimerkkejä jo aikaisemmin mainitussa esimerkissä, joten (täyttä) historiaa tulee itsessään välttää.

Suunnitelmaa laatiessa huomattiin myös, että tietokantaa kannattaa ”siivota” riittävän tiheästi, jolloin taataan tiedon ajankohtaisuus – on hyvin epätodennäköistä, että yli vuotta vanhemmat häiriötiedotteet tuovat lisäarvoa sellaisenaan. Kanta määritettiin itsepuhdistautuvaksi, jolloin kuusi kuukautta vanhempien tiedotteiden tiedot poistetaan kannasta. Tarkoituksena on kannan hallittavuus: kanta pysyy kohtalaisen pienenä, eikä se pääse pirstaloitumaan. Tällöin myös riski, että UID-kentän käyttämät arvot tulisivat käytettyä useaan kertaan, saadaan pienennettyä – mikä ei kuitenkaan realistisesti ajateltuna ollut riski muutenkaan.

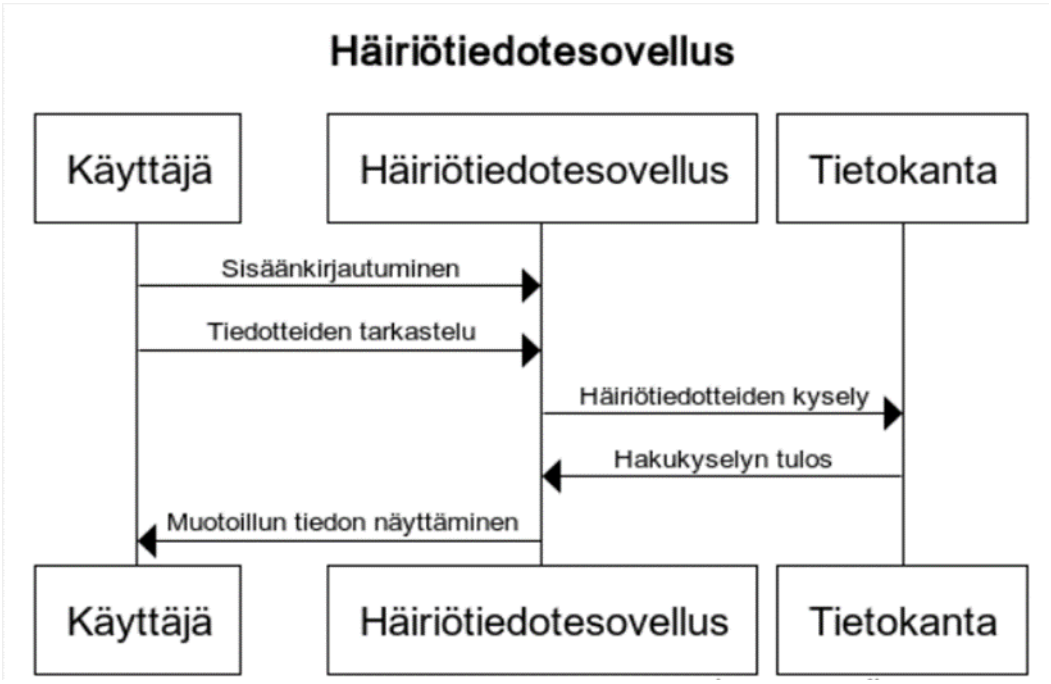
### 3.4 Sovelluksen rooli, käyttö, ja integrointi

Jo suunnitelman alussa huomattiin, että joissakin häiriötiedotteissa saattaa esiintyä lihavoitua tai kursivoitua tekstiä tai hyperlinkkejä, jolloin asiakkaalle näkyvissä häiriötiedotteissa nämä tulee myös luonnollisesti näkyä. Häiriötiedotesovelluksen tulisi siis jotenkin pyrkiä näyttämään korostettua tekstiä, ja näihin yleisin ratkaisu on jonkinlainen merkintäkieli (engl. *markup language*).

Häiriötiedotesovelluksen suunnitelmaa suunnitellessa päädyttiin kahteen eri merkintäkieleen: JSON:iin ja XML:ään [2;3]. Merkintäkielten valinta perustuu nykyisissä järjestelmissä käytettäviin – yritys käyttää pääsääntöisesti alustoillaan JSON- ja/tai XML-merkintäkieltä. Kyseisissä merkintäkielissä pystyy parsimaan ja käsittelemään lihavoitua ja kursivoitua tekstiä, minkä lisäksi hyperlinkkien upotus onnistuu molemmissa. Tietokannasta löytyvät tietueet tarjotaan siis näissä kielissä, jolloin tietokantakyselyn vastauksena palautetaan joko JSON- tai XML-tyyppistä dataa, mikä riippuu käyttäjästä ja hänen toiveistaan.

Datamuodon käsittelemiseksi päädyttiin hyödyntämään HTTP-otsikkotietoja (engl. *header*). Vaihtoehtoinen menetelmä olisi ollut käyttää URI:n muodostuksessa tiettyä päätettä (esimerkiksi https://esimerkkisivu.fi/häiriöt/tulevat.xml), mutta ensimmäinen lähestymistapa on kuitenkin suositeltu menetelmä: otsikkotiedossa voidaan tällöin kuljettaa Accept-otsikkotietoa [9].

Kenties tärkein rooli, joka häiriötiedotesovelluksella on, on häiriötiedotteiden tarkastelu (kuva 3). Ideana oli, että häiriötiedotesovellus hakisi ajankohtaisinta tietoa tietokannasta joko jättämättä mitään välimuistiin, tai vertailemalla välimuistin ja tämänhetkistä (tietokannasta löytyvää) tietoa.




Kuva 3. Sekvenssikaavio häiriötiedotteiden tarkastelusta

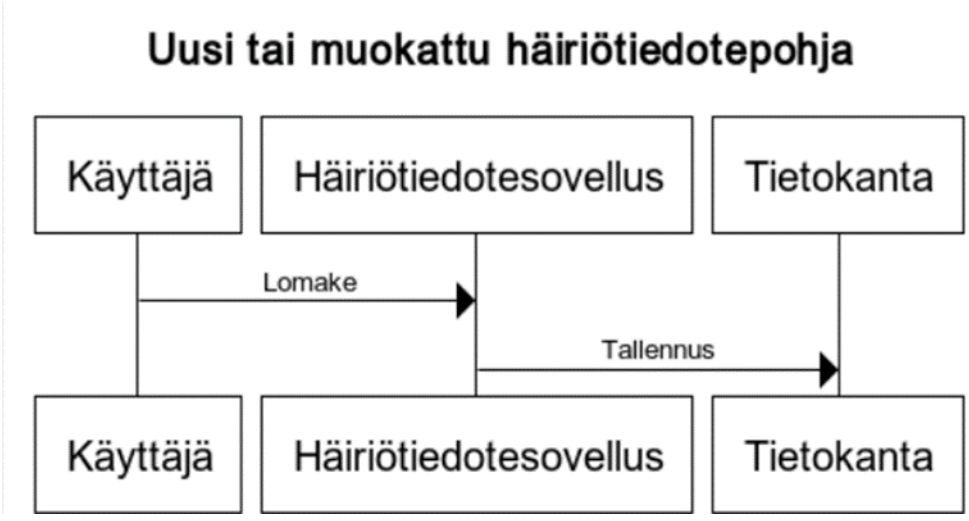
Häiriötiedotteiden tarkastelun ensimmäisen vaihtoehdon vahvuutena on, että voidaan varmistua tiedon ajankohtaisuudesta. Vaihtoehdon haasteena voi olla esimerkiksi sovelluksen käyttö alueella, jossa on huonolaatuinen web-yhteys tai jos tietokanta paisuu kovin suureksi – mikä ei tosin ole realistista, mutta ajan myötä tämäkin olisi teoreettinen mahdollisuus. Mainitut kohdat ovat kuitenkin käytännössä merkityksettömiä, sillä häiriötiedoteilmoituksen laatijat (joko Customer Service tai IMOC) ovat aina alueella, jossa edellä mainittua huonoa web-yhteyttä olisi olemassa, eikä tietokanta tule olemaan niin

suuri, että tiedon hakeminen kestäisi pitkään. Realistisin ongelma olisi palvelimien kaatuminen, mutta siinä vaiheessa ongelma on laajempi kuin käsitellyn sovelluksen kannalta olisi tarvetta paneutua.

Toinen vaihtoehto oli vertailla sovelluksesta jo löytyvät häiriötiedotteet ja näyttää uudet ja päivitetyt häiriöilmoitukset korostetummin. Vaihtoehto on käyttäjäystävällisempi, mutta teknisesti “vaarallisempi” vaihtoehto: mitä jos käyttäjä ei pystykään lataamaan tietoja vaikkapa sovelluksen sisäisen virheen takia? Käyttäjälle ei tällöin näy mitään uusia tietoja, mikä voi olla haitallista. Vaihtoehtoa kuitenkin harkittiin.

Muita ongelmia todennäköisesti esiintyy vasta sovelluksen käytön lomassa, eivätkä ne todennäköisesti liity suoraan mainittuihin ongelmiin – ongelmien korjaukseen tulee kuitenkin varata aikaa riippumatta valitusta tavasta. Varsinaista ratkaisua tähän ongelmaan ei sellaisenaan löytynyt, mutta todennäköisin lähestymistapa tulee olemaan ensimmäinen vaihtoehto, sillä sen mahdolliset haasteet ovat vertailun jälkeen arvioitu pienemmiksi – käytännössä katsoen lähes olemattomiksi.

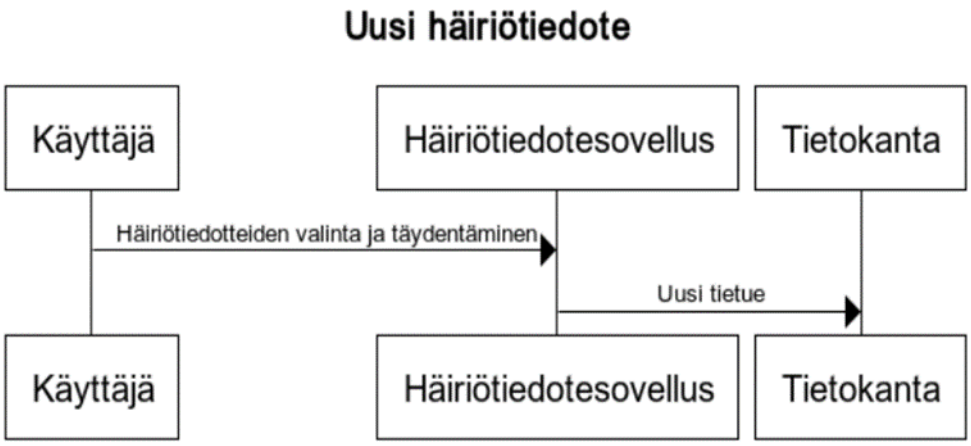
Häiriötiedotesovelluksen toinen rooli on lisätä tietokantaan häiriötiedotepohja täyttämällä lomake, jonka perusteella rakennetaan uusi tietue. Toimenpiteen sekvenssikaavio on esitettynä kuvassa 4.




Kuva 4. Uusi häiriötiedotepohja

Käyttäjä siis tekee uuden lomakkeen häiriötiedotesovellukseen, tekee muutoksia, lisää tekstit ja mahdolliset paikkatekstit, tarkastaa työnsä ja tallentaa lopulta valmiin pohjan tietokantaan. Tämän jälkeen tietokannasta löytyvää pohjaa voidaan hyödyntää nopeasti ja tehokkaasti muutamalla klikkauksella, ja saada uusi häiriötiedote nopeasti useaan eri kanavaan. Suunnitelmassa huomioitiin myös, että vasta uutta häiriötiedotetta julkaistaessa tulee lisätä häiriötiedotteen kanavat: jos ne olisivat olleet häiriötiedotepohjaan sidonnaisia, olisi tiedotteita haastavampi julkaista vain tiettyihin kanaviin.

Häiriötiedotesovelluksen on luonnollisesti pystyttävä julkaisemaan uusi häiriötiedote, jolloin sekvenssi määräytyy pitkälti kuvan 5 mukaisesti.

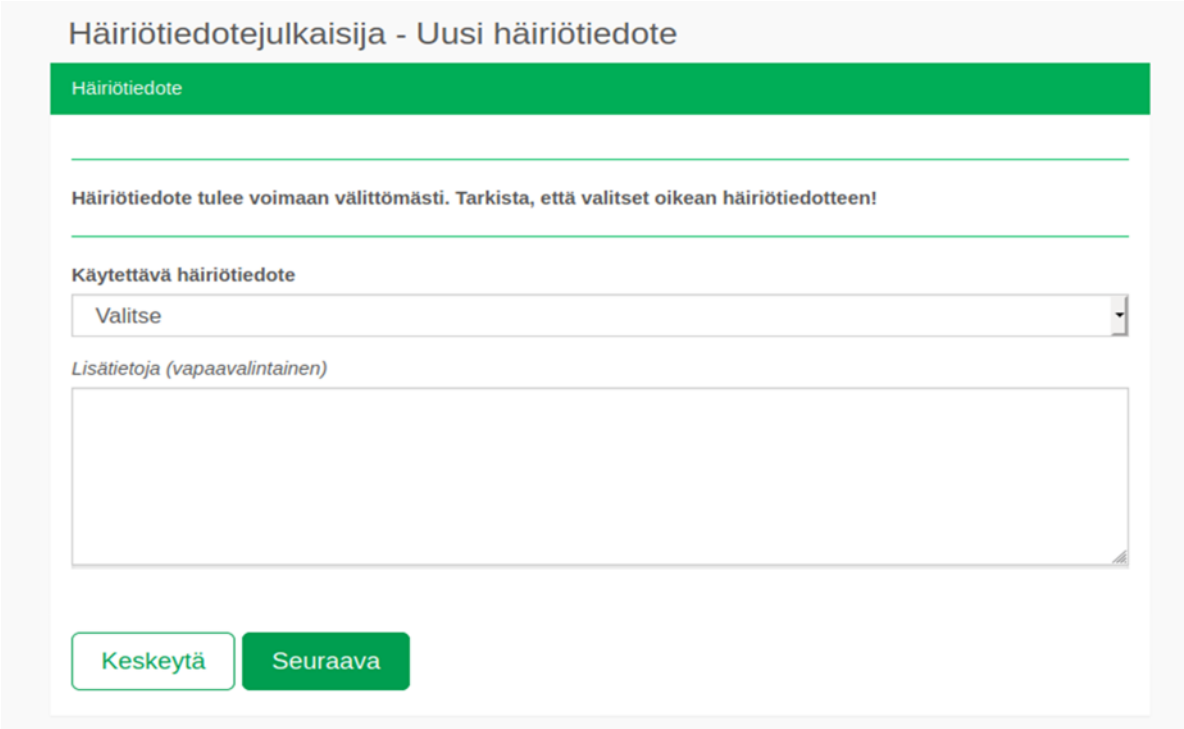



Kuva 5. Uusi häiriötiedote

Käyttäjä valitsee häiriötiedotepohjan häiriötiedotesovelluksesta ja täydentää sitä tarpeen vaatiessa. Tämän jälkeen käyttäjä julkaisee uuden häiriötiedotteen, ja häiriötiedotesovellus luo samalla uuden tietueen tietokantaan.

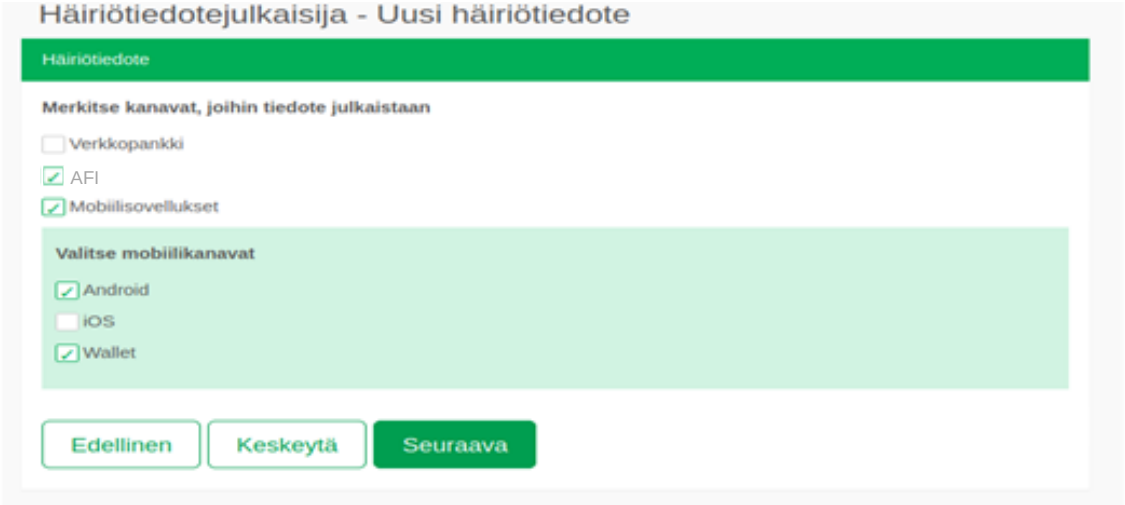
Häiriötiedotesovelluksesta ei tehty toimivaa prototyyppiä työn luonteen vuoksi, mutta käyttöliittymää varten tehtiin muutamia esittelymalleja (engl. *mock-up*). Esittelymalleja on esiteltyinä kuvissa 6,7, 8 ja 11. Suurin osa esitysmalleista tehtiin Angular-ohjelmistokehyksellä [10] TypeScript- ja HTML-kieliä käyttäen hyödyntäen yrityksen sisäisiä Angular-komponentteja. Angular on laajasti käytössä yrityksen sisäisessä kehityksessä. Konseptitasolla joidenkin versioiden tekemiseen käytettiin Confluence-ohjelmiston tarjoamaa Gliffy-työkalua.

Esimerkkitoteutus uuden häiriötiedotteen julkaisusta on esitetty kuvassa 6, mutta suunnitelmassa painotettiin, ettei kuvaa tulisi käyttää sellaisenaan mallina itse toteutuksessa. Ideana oli myös, että ennen kuvassa esiintyvää häiriötiedotteen valintaa käyttäjälle annettaisiin vaihtoehtoja häiriötiedotteen ajankohdan valintaan. Kuvan mukaisessa tilanteessa ajankohdaksi on valittu välittömästi voimaan tuleva häiriötiedote.




Kuva 6. Mahdollinen ratkaisu uuden häiriötiedotteen julkaisemiseen

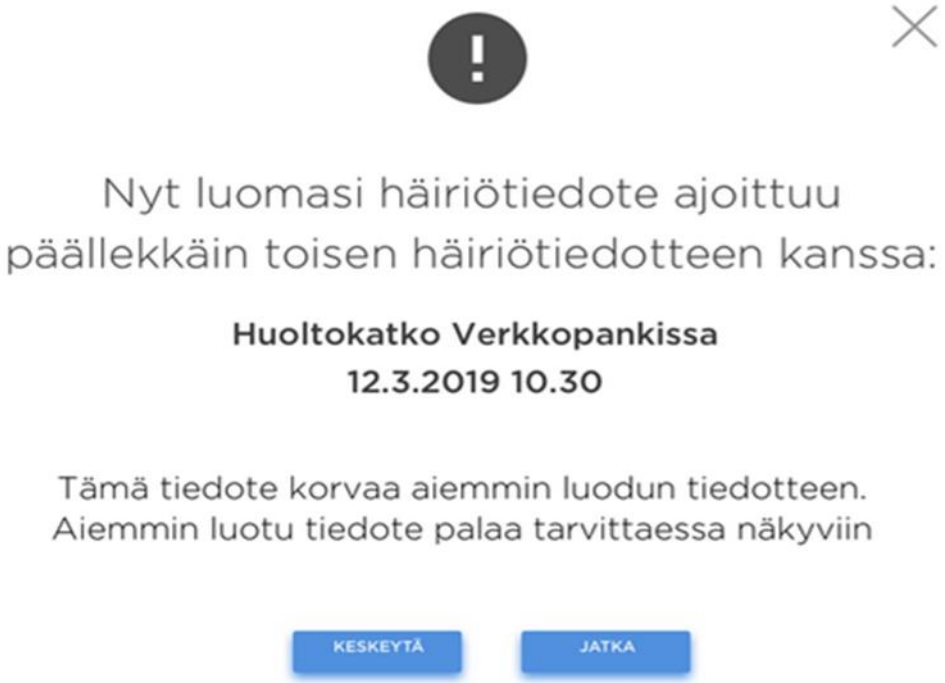
Kuva 7 on jatkoa kuvassa 6 esitetylle tapahtumalle, jossa käyttäjä on luomassa uutta häiriötiedotetta. Kuvassa 7 käyttäjä valitsee kanavat, joihin häiriötiedote on tarkoitus julkaista.




Kuva 7. Julkaistavien kanavien valinta häiriötiedotesovelluksessa

Ideana on, että kanavien valinnan jälkeen käyttäjälle tarjotaan näkymä, jossa hän pääsisi tarkastelemaan julkaistavaa häiriötiedotetta kokonaisuudessaan: kuvaus, otsikko, päivämäärä, julkaisuaika…

Huomattiin tosin, että joissakin yrityksen käyttämissä kanavissa on tuki vain yhdelle häiriötiedotteelle kerrallaan, minkä takia sovelluksen tulee pystyä myös tarkastamaan uusien häiriötiedotteiden päällekkäisyys automaattisesti ja ilmoittaa käyttäjälle tästä, kuten kuvassa 8. Tällöin sovelluksen tulee tarjota mahdollisuus joko hyväksyä tai hylätä tuleva tiedote.




Kuva 8. Päällekkäinen häiriötiedote

Uutta päällekkäistä häiriötiedotetta hyväksyessä sovelluksen tulee kuitenkin automaattisesti korvata päättyvä tiedote aikaisemmalla – riippuen aikaisemman tiedotteen voimassaolosta. Mikäli aikaisempi häiriötiedote (tiedote A) on jo merkitty päättyneeksi, tulee sovelluksen poistaa myös tuoreempi tiedote (tiedote B) – tässä tilanteessa kumpaakaan häiriötiedotetta ei näy sen jälkeen, kun tiedote B poistuu. Mikäli vain uudempi tiedote (B) on merkitty päättyneeksi, tulee sovelluksen korvata tiedote aikaisemmalla tiedotteella, joka jo kerran näytettiin. Tässä tilanteessa tiedote A tulee uudestaan näkyviin, kun tiedote B poistuu.

Häiriötiedotteita ei kuitenkaan pidä poistaa tietokannasta häiriötiedotesovelluksella itsellään – häiriötiedotteet vain piilotetaan asiakkaiden näkyvistä.

Sovelluksen tuli tarjota mahdollisuus suodattaa häiriötiedotteita tiettyjen parametrien perusteella, eli joko aikaperusteisesti (esim. vain viimeisen kk ajalta; myös häiriön tilan perusteella – eli meneillään, päättynyt tai suunniteltu) tai kanavapohjaisella suodatuksella (esim. vain VEPA tai VEPA ja AFI).

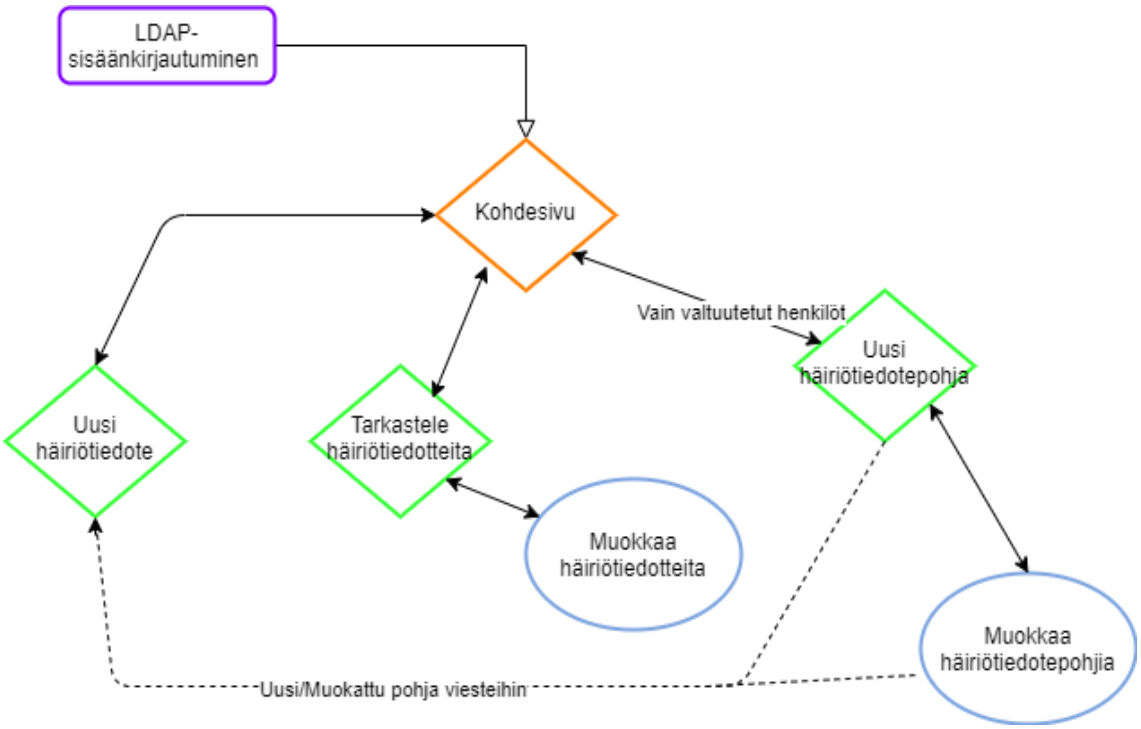
Häiriötiedotesovelluksen tulee myös (LDAP:lla haettujen oikeuksien perusteella) tarjota mahdollisuus olemassa olevien häiriötiedotepohjien muokkaukseen. LDAP:lla tarkoitetaan protokollaa, jonka tarkoituksena on käsitellä (yleisimmin) TCP/IP-yhteyden välityksellä hakemistopalveluihin pääsyä. [11.] Hakemistopalvelut puolestaan kartoittavat verkkoresurssien nimet niitä vastaaviin verkko-osoitteisiin. Tarkoituksena on siis hallinnoida käyttäjien pääsyä johonkin järjestelmään joko automaattisesti tai (harvemmin) manuaalisesti. Suunnitelmaan sisällytettiin LDAP-integrointi hyödyntämällä yrityksessä jo olemassa olevia tunnuksia SSO:n (*Single Sign-On*) avulla, kuten mm. sähköpostin kanssa oli tehty.

Häiriötiedotesovellukselle määriteltiin kolme erillistä ryhmää, joiden oikeuksien perusteella häiriötiedotteita ja -pohjia voitiin tarkastella ja muokata. Kaikilla käyttäjillä on suunnitelman mukaan oikeus tarkastella häiriötiedotteita, mutta tämän lisäksi yleisimmälle *User*-käyttäjälle määriteltiin vain uusien häiriötiedotteiden luonti ja päivitys. *Manager*-käyttäjälle annettiin tämän lisäksi myös häiriötiedotepohjien käsittelyoikeudet ja harvinaisimmalle *Admin*-käyttäjälle määriteltiin oikeudet kaikkeen toimintaan häiriötiedotesovelluksen puitteissa.

Jaottelun perusteella Customer Service -käyttäjille voidaan antaa *User*-oikeudet, sillä heidän tarvitsee vain hyödyntää olemassa olevia häiriötiedotepohjia, ei luoda uusia. *Manager*-käyttäjiksi määriteltiin mm. IMOC-henkilöt, ja *Admin*-käyttäjiksi varattaisiin vain henkilöt, joilla on suora pääsy tietokantaan sekä lähdekoodiin, ja näin ollen olisivat vastuussa järjestelmän ylläpidosta.

Kuvassa 8 on esitettynä häiriötiedotesovelluksen käytön työnkulku. Kokonaisuuteen kuuluu siis vain muutama erilainen visuaalisesti erilaista komponenttia, vaikka taustalle tuleekin huomattavasti enemmän arkkitehtuuria.

Ennen häiriötiedotesovelluksen käyttöä siihen kirjaudutaan sisään omilla tunnuksilla (violetti ruutu). Sisään- ja uloskirjautuminen on tarkoitettu toteutettavaksi LDAP-integraatiolla.

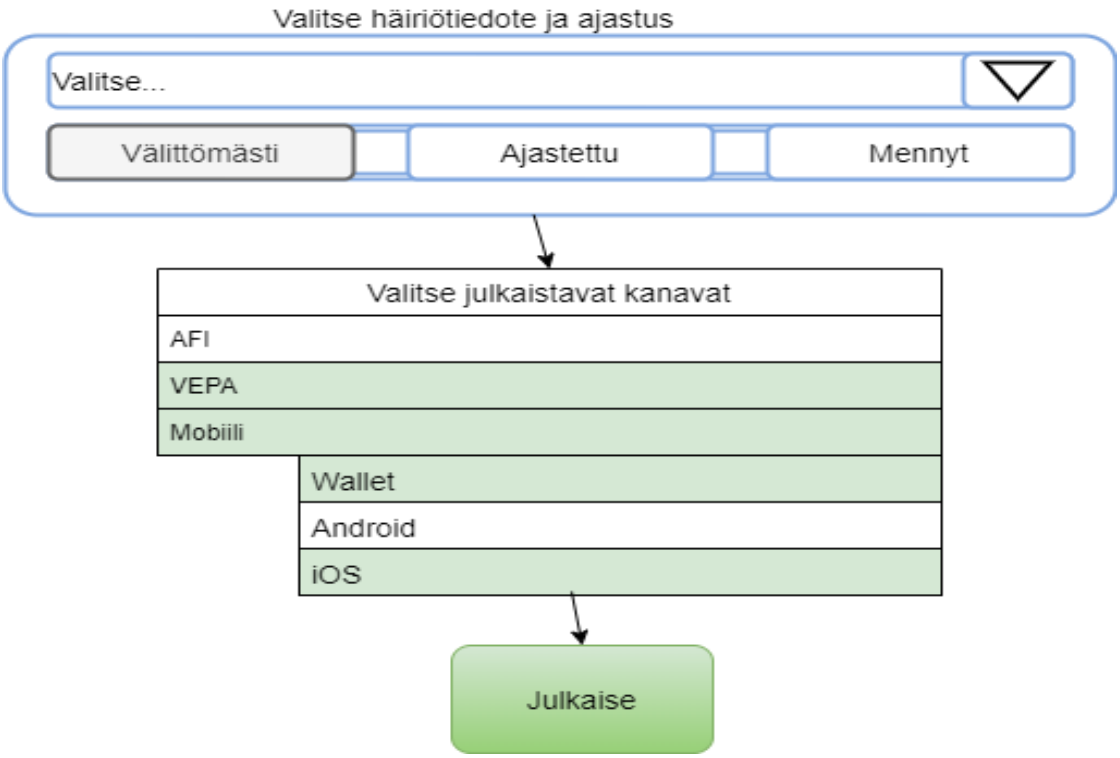



Kuva 9. Kokonaiskuva häiriötiedotesovelluksen työnkulusta

Tämän jälkeen kohdesivulla (keltainen ruutu) näytetään kahdesta neljään eri vaihtoehtoa, riippuen käyttäjän oikeuksista:

- Luo uusia häiriötiedotteita. (*User, Manager & Admin,* kuva 9)
- Muokkaa olemassa olevia häiriötiedotteita. (*Manager & Admin*)
- Tarkastele häiriötiedotteita. (*User, Manager & Admin*, kuva 10)
- Hallinnoi häiriötiedotepohjia. (*Admin*)

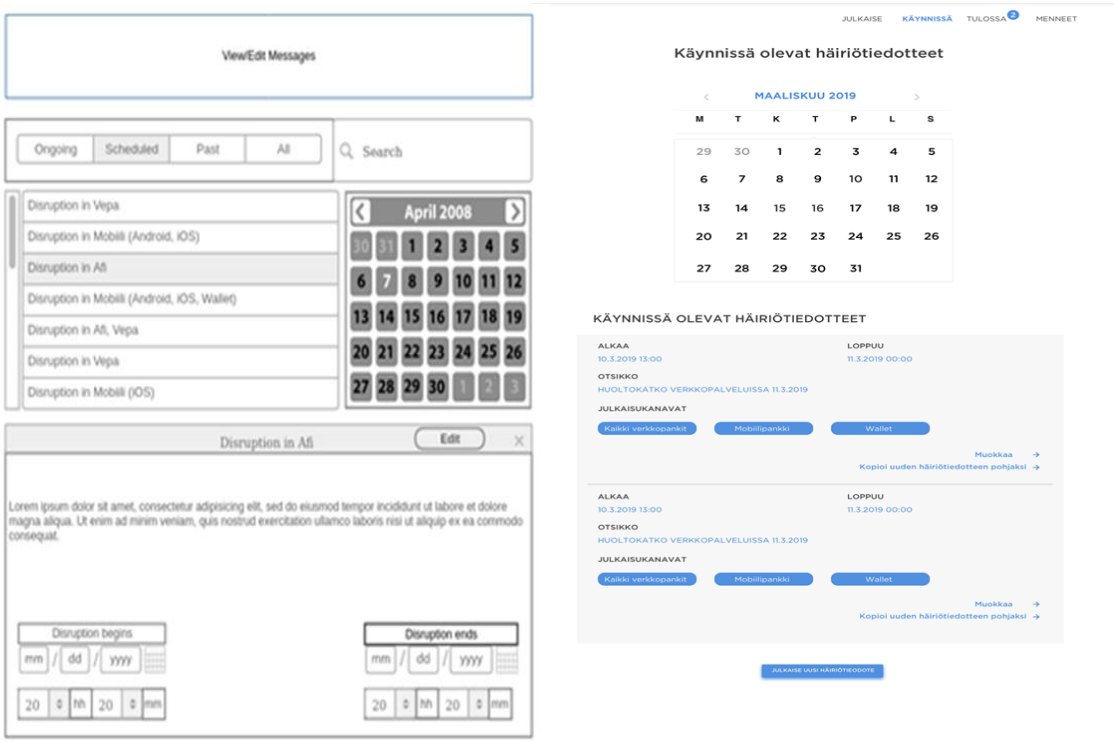
Suunnitelmassa huomioitiin myös tarve kirjautua ulos missä tahansa vaiheessa, ja käytetyn kielen vaihtaminen lennossa jollakin painikkeella. Tämän lisäksi huomattiin myös, että jokaisessa näkymässä olisi hyvä olla käyttöohjeet käyttäjää varten.




Kuva 10. Uuden häiriötiedotteen luonti kokonaisuudessaan häiriötiedotesovelluksessa

Uutta häiriötiedotetta laatiessa käyttäjä valitsee ensin kuvassa 9 esitetyn työnkulun mukaisesti käytettävän häiriötiedotepohjan ja valitsee kanavat, joihin häiriötiedote julkaistaan. Häiriötiedotteeseen tulee myös valita julkaisuajankohta, mikäli kyseessä on ajastettu (eli tuleva) häiriö. Valintojen jälkeen tarjotaan mahdollisuus tarkastella julkaistavaa häiriötiedotetta ja palata edelliseen näkymään (tietojen muokkaukseen).

Kuvassa 10 on esiteltynä näkymävaihtoehtoja häiriötiedotteiden tarkasteluun. Suunnitelmassa huomioitiin erilaisten suodatustapojen käyttöä muun muassa aikaperusteisesti ja kanavakohtaisesti. Kuvassa on myös esitetty erilaisia tapoja esittää jokin tietty häiriötiedote yleisnäkymästä, mutta esitellyt ratkaisut ovat kuitenkin jätetty vain malleiksi – kehittäjän lopulliseen toteutukseen ei haluttu paneutua liiaksi.




Kuva 11. Häiriötiedotteiden tarkasteluvaihtoehtoja häiriötiedotesovelluksessa

Suunnitelmassa otettiin myös huomioon yksittäisen häiriötiedotteen tarkastelu ja muokkaus. Tarkastelunäkymässä näytetään kaikki tiedotteeseen liittyvät tiedot, ja näin ollen yksittäisen häiriötiedotteen tietojen muokkaus tulisi sallia vain tässä näkymässä: tällä voidaan varmistua siitä, että muokataan vain oikeaa tiedotetta. Huomattiin myös, ettei erotusta tiedotteen ajankohtien välillä tarvitse muokkauksessa tehdä, sillä kyse oli vain suodatuksesta.

Lokituksella (engl. *logging*) tarkoitetaan yleisesti sellaista toimenpidettä, jossa jostakin asiasta pidetään kirjaa. Tieto- ja viestintätekniikassa lokituksella viitataan aivan samalla tavalla lokin kirjanpitoon, ja voi yksinkertaisimmillaan tarkoittaa vaikkapa viestin kirjoittamista lokitiedostoon [12].

Häiriötiedotesovellusta suunniteltaessa ilmeni, että on hyödyllistä listata erinäisiä tietoja lokille, kuten esimerkiksi virheilmoitukset ja sovelluksen käyttöön liittyvät tehtävät. Lokitus määriteltiinkin jo melko aikaisessa vaiheessa jakautumaan näihin kahteen eri lokiin:

virhelokiin (engl. *error log*) ja käyttölokiin (engl. *audit log*). Virhelokille merkittäisiin sovelluksen sisäiset virheet ja ongelmatapaukset, kuten vaikkapa epäonnistuneet kyselyt tietokannasta sekä ongelmat uuden häiriötiedotteen luonnissa. Käyttölokille puolestaan kirjattaisiin asiat, jotka liittyvät sovelluksen käyttöön: mm. käyttäjä, käsiteltävä häiriötiedote sekä häiriötiedotteiden luonti- ja muokkaustapahtumat ajankohtineen. Käyttölokin ei kuitenkaan ole tarkoitus palvella historialokina, joka kertoisi kaikki häiriötiedotteisiin liittyvät kyselyt, muutokset tai muut vastaavat tapahtumat. Käyttölokin avulla nk. jäljitysketju tai kirjausketju (engl. *audit trail*) tulee huomattavasti helpommaksi: sisään kirjautuneita käyttäjiä voidaan valvoa tehokkaasti, kun jokainen kirjautuminen tallentuu [13].

Yritys käyttää nykyisellään Splunk-järjestelmää lokituksessaan [2], joten ei ollut juurikaan syytä pohtia muita vaihtoehtoja. Järjestelmä on nykyisellään ollut vahvassa käytössä sisäisesti.

## 3.5 Hyödynnettävät rajapinnat

> Ohjelmointirajapinta (Application Programming Interface, API) määrittelee, miten ohjelmisto tarjoaa tietoja tai palveluita sovelluksille tai muille tietojärjestelmille. [14.]

Häiriötiedotesovellukseen liittyy olennaisesti kaksi eri rajapintaa, joiden kautta tietoa käsitellään: häiriötiedotesovelluksen oma rajapinta sekä REST-rajapinta. REST-rajapinta kuitenkin hajautettiin kahteen erilliseen rajapintaan, joten kaiken kaikkiaan rajapintoja on käytännössä kolme, vaikkakin REST-rajapinnat ovat keskenään lähes samanlaisia.

Häiriötiedotesovelluksen omalla rajapinnalla tarkoitetaan rajapintaa, joka mahdollistaa uusien häiriötiedotteiden julkaisun, tiedotteiden muokkauksen sekä poistamisen. Kanavat, joissa tiedote julkaistaan, riippuu tiedotteesta ja halutusta häiriötiedotteen julkaisuratkaisusta.

Rajapinnan pohjamalliksi valittiin perinteinen CRUD-malli. Tätä mallia tosin muokattiin loppupeleissä sellaiseksi, jossa häiriötiedotepohjaa ei tulisi koskaan voida poistaa tietokannasta. Syynä tähän oli datan yhtenäisyys; häiriötiedotteita tulisi voida tarkastella myös sisällöltään, vaikka se olisi jo merkitty päättyneeksi. Häiriötiedotepohja olisi siis aina olemassa, vaikkei sitä käytettäisi kuin kerran.

Tämän seurauksena syntyi kuitenkin haaste: kaikkia häiriötiedotepohjia ei välttämättä haluta näyttää käyttäjille uutta häiriötiedotetta luotaessa, jos pohjia olisi suuri määrä. Ratkaisuksi kehiteltiin merkitsemistapa, jonka avulla häiriötiedotesovellus voisi piilottaa pohjan käyttäjiltä, vaikka se teknisesti olisikin olemassa. Tämän ”lipun” avulla häiriötiedotetta julkaistaessa lista siis näkyy hallittavissa olevan kokoisena, eikä yksittäisen pohjan etsimiseen kuluteta turhaan aikaa.

REST on akronyymi, jonka lyhenne tulee sanoista Representational State Transfer. REST on hajautettujen hypermedioiden arkkitehtuurillinen tyyli, jolla on oma kuusikohtainen rajoitelistansa. Usein REST-tyyli on yksinkertaistettu HTTP-mallin mukaiseksi, mutta näin ei kuitenkaan ole; REST on tässä yhteydessä, ja erittäin yksinkertaistettuna, vain *tapa käyttää HTTP:ta*. REST-mallissa resurssit ja niiden representaatio on eriytetty toisistaan, jotta resurssien sisältö voidaan esittää monessa eri muodossa, kuten HTML, XML, PDF, JSON ja niin edelleen [15].

REST-rajapintojen tuli tukea vain (HTTP) GET-kyselyitä rajapinnan julkisuuden vuoksi [16.] – häiriötiedotteiden kannattaa kuitenkin näkyä myös ulospäin, sillä muuten ei sovelluksesta olisi mitään hyötyä. Tämä kuitenkin synnytti haasteen tekniseen toteuttamiseen: miten muiden kuin ajankohtaisten häiriötiedotteiden näkyminen ulospäin saataisiin estettyä? Alun perin yksi rajapinta hajautettiin tämän haasteen vuoksi kahteen, joista toinen olisi vain sisäiseen käyttöön tarkoitettu, ja toinen tarjoaisi vain julkisia häiriötiedotteita. Taulukossa 2 on esitettynä julkisen REST-rajapinnan mallin ratkaisu.

Taulukko 2. Julkinen REST-rajapinta

| /disruptions | | | | | |
|---|---|---|---|---|---|
| /all | /afi | /vepa | /mobile | /wallet | /{id} |
| /{id} | | | | | |

Ratkaisussa palautetaan vain tällä hetkellä voimassa olevat häiriötiedotteet ja tarjotaan mahdollisuus rajata häiriötiedotteita kanavakohtaisesti. Yksittäisiin häiriötiedotteisiin päästäisiin käsiksi id-parametrin avulla, muutoin palautettaisiin lista häiriötiedotteita – tai tyhjä lista, jos niitä ei ole. Koska mitään aikaperusteista rajausta ei olisi, ei sitä myöskään pääsisi pyyntöön lisäämään.

Suunnitelmaan sisältyi lisäksi REST-rajapinta, joka pystyy käsittelemään kaikki häiriötiedotteet. Tässä rajapinnassa tarjotaan mahdollisuus kaikkien häiriötiedotteiden hakuun, mutta rajapinnalla itsellään voidaan rajata häiriötiedotteiden ajankohtaa karkealla tasolla, toisin kuin julkisessa versiossa. Rajapinnassa on myös mukana yrityksen sisäiseen viestintään eli intraan tarkoitetut häiriötiedotteet. Sisäiseen viestintään tarkoitettuja häiriötiedotteita ei tarjota julkisessa rajapinnassa.

Rajapintojen julkisuutta päätettiin rajoittaa myös infrastruktuurissa: päätelaitteille näkyviä tuloksia rajataan mm. CORS (*Cross Origin Resource Sharing*) -rajoituksin [3.;17.] ja *system-id*- ja *application-id*-tunnuksilla HTTP-otsikkotiedoissa, kuten nykyisissä yrityksen käyttämissä lomakkeissa on tehty. *System*- ja *application*-id:t tulisi tällöin *whitelistata* [18.], eli sallia vain ne pyynnöt, jotka tulevat tietyillä tunnisteilla.

Tiedotteiden sisällön tuli kuitenkin olla rajapinnasta riippumaton. Näin ollen kummassakin rajapinnassa tarjotaan sisällöllisesti samat asiat, mutta erityistä huolta täytyi pitää siitä, ettei vain intraan meneviä häiriötiedotteita pääse tutkimaan julkista rajapintaa käyttäen.

Rajapintojen käytössä esiintyy väistämättä virheitä; mikään käytettävä sovellus ei ole täysin varma. Virhekäsittelyä on kuitenkin pyritty helpottamaan: esimerkkipyynnöt kumpaankin rajapintaan löytyvät liitteistä 1 ja 2.

Häiriötiedotesovellusta suunnitellessa paljastui, että julkisen rajapinnan puolella on viisainta olla palauttamatta mitään muuta kuin pelkkä virhekoodi 400, jolloin mahdollista urkintaa rajoitetaan mahdollisimman pitkälle. Tällöin virhelokille jäävässä tiedossa kerrotaan tarkemmin, mistä virhe johtuikaan; virhe voi olla myös vaikkapa koodilla 401 tai 403, mutta tätä tietoa ei haluta antaa ulospäin. Virheinformaatiota ei siis tarjota sellaisenaan ”julkisuuteen”, vaan se käsitellään sisäisesti tarkemmin. Muut virhekoodit niputetaan tämän virhekoodin (400) taakse [3].

Liitteen 1 taulukossa on mainittuna tapaus, jossa käynnissä olevia häiriöitä ei ole; tämä ei sinällään ole virhe, vaan kyseisessä tapauksessa tulisi tällöin palauttaa tyhjä lista. Tapauksessa on palautettu tyhjä arvo, jonka ideana on olla tarjoamatta liikaa informaatiota, jolla koodin toimintaa voisi kalastella [19] (verkkourkinta, tietojenkalastelu, engl.

*phishing*). Tietojen kalastelu on kuitenkin erityisesti pankeille merkittävän tärkeää pyrkiä estämään mahdollisimman monin keinoin, ja vaikka kyseessä onkin vain häiriötiedotteiden mahdollinen kalastelu, saattaa varsinkin pelkästään intraan tarkoitetuissa häiriötiedotteissa kulkea arkaluontoista materiaalia.

# 4 Tulevaisuus ja kehityskohdat

## 4.1 Suunnitelman hyödyntäminen

Kanavien rajat ylittävä yhtenäinen häiriötiedotesovellus on ollut yrityksen potentiaalissa parannuskohdissa jo pitkään. Ei siis ole tuulesta temmattua, että työssä hyödynnettyä informaatiota käytettäisiin itse sovelluksen rakentamisessa – haasteena on varata kehittämiseen aikaa. Sovelluksen liiketoiminnallinen hyöty on melko epäselvää kaikille osapuolille, joten toteutuksen aloittamiseen saattaa kestää jonkin aikaa. Jos sovelluksen tekemiseen kuitenkin päädytään, on suunnitelma jo tehtynä – eli sovellus on jo puoliksi tehty.

## 4.2 Jatkokehitys

Tietty häiriötiedote jossakin kanavassa voisi automaattisesti johtaa vaikkapa tiettyjen kanavien sulkuun ja ilmoitukseen. Tämä ei tosin täysin kuulu sinällään häiriötiedotesovelluksen alkuperäiseen suunnitelmaan (ja esimerkki sinällään kuuluu VEPA-puolen asiantuntijoiden ryhmälle), mutta integraatiota järjestelmiin syvemmin kuitenkin harkittiin jo suunnitelmavaiheessa.

Häiriötiedotesovelluksessa olisi hyvä olla aikakatkaisu väärinkäyttöjen ja muiden ei-haluttujen tilanteiden ehkäisemiseksi. Sovellukseen voidaan rakentaa nykyisistä AFI-lomakkeista löytyvän aikakatkaisumekanismin kaltainen toiminnallisuus. Kyseisessä mallissa käyttäjälle annetaan ensin varoitus aikakatkaisusta, mikäli käyttäjä ei ole ollut aktiivinen hetkeen ja mikäli käyttäjä ei ole reagoinut viestiin ja aika kuluu umpeen, käyttäjä kirjataan ulos automaattisesti.

Aikakatkaisuun liittyen voidaan häiriötiedotesovellukseen rakentaa osittaistallennusta mm. häiriötiedotepohjien ja muutosten osalta. Tällä logiikalla uutta häiriötiedotepohjaa ei tarvitsisi tehdä yhdellä istumalla, vaan asiaan voisi palata, kun kaikki tiedotteeseen liittyvät tiedot ovat selvillä.

### 4.3 Havainnot ja selvittämättömät asiat

Häiriötiedotesovellusta suunnitellessa tuli allekirjoittaneelle myös pohdintaa LDAP-integraatioon ja lokitukseen liittyen – nämä olivat sellaisenaan jääneet melko abstraktille tasolle, mutta suunnitelmaa tehtäessä käsitteet aukesivat huomattavasti ymmärrettävämmälle tasolle. SSO-kirjautuminen oli myös osana tätä kokonaisuutta, ja vaikka sitä käytettiin päivittäin, ei sen merkitys ollut juurikaan avautunut.

Jokaisesta mainitusta asiasta tosin riittäisi aihetta omiin insinööritöihinsä, joten tässä työssä käsiteltynä aiheet ovat käytännössä ymmärretty sellaisenaan, eli toisin sanoen niihin ei ole sen enempää pureuduttu.

Suunnitelmassa ei huomioitu muun muassa käytettävyyttä tai responsiivisuutta, tosin yrityksen toimesta näitä asioita huomioidaan luonnostaan ja suunnitelman tuli olla ottamatta kantaa toteutukseen. Kyseessä oli siis tarkoituksellinen asiakokonaisuuden rajaaminen, jossa huomioitiin, että näiden aiheiden tarkempi kommentoiminen olisi tuonut lisäarvoa työhön.

Häiriötiedotteen visuaalinen puoli jäi myös kevyemmäksi työn luonteen vuoksi. Ainoa esimerkkiratkaisu, joka valikoitui insinöörityöhön asti, oli esitettynä kuvassa 6. Yritys on kuitenkin juuri brändäämässä itseään uudelleen, joten kuvituskuvia sellaisenaan pyrittiin välttämään.

# 5 Yhteenveto

Insinöörityön tavoitteena oli suunnitella suomalaiselle liikepankille häiriötiedotesovellus. Sovelluksen avulla tuli pystyä julkaisemaan häiriötiedotteita moneen eri kanavaan samanaikaisesti. Lisäksi tavoitteena oli ottaa huomioon jo olemassa olevia arkkitehtuureja ja rakennelmia, joita hyödyntämällä sovelluksen käyttö olisi mahdollisimman monipuolista ja selkeää.

Aluksi kerättiin tietoja nykyisistä järjestelmistä, esimerkiksi miten niissä on hoidettu tiedotteiden sisältö ja mahdolliset muuttuvat kentät sekä minkälaisia häiriötiedotetyyppejä on olemassa.

Seuraavassa vaiheessa suunnitelmaan laadittiin nykyisten häiriötiedotteiden perusteella ratkaisu, jossa häiriötiedotteiden rakenne jaettiin kahtia häiriötiedotteisiin ja häiriötiedotepohjiin. Ratkaisun ansiosta yksittäisen häiriötiedotteen voi julkaista moneen eri kanavaan kerralla, mutta tiedotteet on kuitenkin yksilöity. Tämä ratkaisu heijastui myös tietokantarakenteeseen, sovelluksen rajapintoihin ja LDAP-integraation kautta saataviin oikeuksiin. Rajapintatoteutuksissa hyödynnettiin REST-mallia.

Häiriötiedotesovelluksen suunnitelmassa pyrittiin ottamaan huomioon myös mahdollisimman kattavasti virheraportointijärjestelmä. Järjestelmä hajautettiin kahteen eri lokiin, joista kummallakin oli eri käyttötarkoituksensa.

Lopuksi pohdittiin häiriötiedotesovelluksen suunnitelman tulevaisuutta jatkointegraatiolla ja pyrittiin miettiä sellaisia kohtia, joissa saattaisi löytyä vielä kehittämisen varaa. Mainittiin myös sovelluksen toteutuksen epävarmuudesta, mutta pidettiin kuitenkin todennäköisenä, että suunnitelmaa hyödynnettäisiin tulevaisuudessa.

Häiriötiedotesovelluksen suunnitelmalla saatiin aikaiseksi kokonaisuus, jota hyödyntämällä sovelluksen kehittäjällä olisi mahdollisimman yksinkertaista päästä työn tekoon ja käyttää suunnitelmaa pohjana lopullisissa ratkaisuissaan. Työ itsessään keräsi monipuolisesti eri kantoja siihen, minkälaisia muun muassa tietokantarakenteen ja sovelluksen rajapintojen tulisi olla, unohtamatta käyttäjänhallintaa sekä lokitusta.

## Lähteet


1 Liferay DXP for Websites. Verkkoaineisto. <https://www.liferay.com/solutions/websites>. Luettu maaliskuussa 2020.

2 Halttunen, Harri. 2020. Ohjelmistoinsinööri, Helsinki. Keskusteluja tammi-maaliskuussa 2020.

3 Kesti, Jani. 2020. Senior Software Development Engineer, Helsinki. Keskusteluja tammi-maaliskuussa 2020.

4 Wenzel, Kris. 2015. Verkkoaineisto. Essential SQL. <https://www.essentialsql.com/what-is-meant-by-acid/>. Luettu huhtikuussa 2020.

5 Grolinger et al., 2013. Journal of Cloud Computing: Advances, Systems and Applications. Verkkoaineisto. <https://journalofcloudcomputing.springeropen.com/track/pdf/10.1186/2192-113X-2-22>. Luettu toukokuussa 2020.

6 Kingsbury, Kyle 2015. Jepsen: MongoDB stale reads. Verkkoaineisto. <https://aphyr.com/posts/322-call-me-maybe-mongodb-stale-reads>. Luettu toukokuussa 2020.

7 Signicat. Verkkoaineisto. <https://developer.signicat.com/documentation/finnish-trust-network/>. Luettu huhtikuussa 2020.

8 JavaTpoint – SQL Composite Key. Verkkoaineisto.- <https://www.javatpoint.com/sql-composite-key>. Luettu toukokuussa 2020.

9 REST – Content Negotiation. Verkkoaineisto. <https://restfulapi.net/content-negotiation/>. Luettu 21.2.2020.

10 Angular. Verkkoaineisto<https://angular.io/docs>: Luettu kesäkuussa 2019.

11 OpenLDAP Software 2.4 Administrator's Guide – 1.2. What is LDAP? Verkkoaineisto. <https://www.openldap.org/doc/admin24/intro.html Luettu 20.4.2020> Luettu huhtikuussa 2020.

12 Kreps, Jay. 2013. LinkedIn Engineering. The Log: What every software engineer should know about real-time data's unifying abstraction https://engineering.linkedin.com/distributed-systems/log-what-every-software-engineer-should-know-about-real-time-datas-unifying>. Luettu huhtikuussa 2020.

13 Committee on National Security Systems. 2019. Verkkoaineisto. https://web.archive.org/web/20120227163121/http://www.cnss.gov/Assets/pdf/cnssi_4009.pdf>. Luettu toukokuussa 2020.

14 Avoimen rajapinnan määritelmä. 2014. Verkkoaineisto. <http://avoinrajapinta.fi/> Luettu huhtikuussa 2020.

15 REST API Tutorial – What is REST? Verkkoaineisto. <https://restfulapi.net/>. Luettu maaliskuussa 2020

16 REST – HTTP Methods. Verkkoaineisto. <https://restfulapi.net/http-methods/#get>. Luettu huhtikuussa 2020.

17 MDN web docs. Mozilla. Päivitetty 1.5.2020. <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>. Luettu toukokuussa 2020.

18 Mack, Elizabeth. 11.6.2018. Verkkoaineisto. <https://www.springboard.com/blog/what-is-whitelisting/>. Luettu toukokuussa 2020.

19 Moore, James. Verkkoaineisto. <https://www.f-secure.com/en/consulting/our-thinking/phishing-attacks-measuring-your-susceptibility>. Luettu toukokuussa 2020.

**Julkisen rajapinnan kutsu, jolla saadaan kaikki häiriötiedotteet voimassa olevista häiriöistä AFI-kanavassa**

| REST | Kutsuparametrit | Huomioita |
|---|---|---|
| URL | /disruptions/afi/ | *Kanavatietoa muuttamalla voidaan hakea muista kanavista. Tyhjä kanavatieto (vaihtoehtoisesti {/all}) hakee kaikista kerralla* |
| HTTP-metodi | GET | |
| Otsikkotiedot (*Header*) | Ei ole | *Sisältää halutun formaatin (JSON tai XML), system-id:n, ja application-id:n* |
| URL-parametrit | Ei ole | |
| Dataparametrit | Ei ole | *Turhia GET-metodissa* |
| Onnistunut vastaus (engl. *Success response*) | 200:<br>Content: {<br>title:<br>“Häiriötiedote - Verkkopankissa saattaa esiintyä häiriötä.”,<br>start_date: 2000-01-01T00:00:00Z,<br>end_date: 2000-01-03T00:00:00Z,<br>content_fi: “Verkkopankeissa saattaa esiintyä häiriöitä. Vikaa korjataan parhaillaan. Pahoittelemme häiriöstä aiheuttavaa haittaa.”, content_sv: “Det kan förekomma störningar i nätbankerna. Felet åtgärdas som bäst. Vi beklagar besväret som orsakas av störningen.”<br>} | *Tulisi sisältää myös englanninkieliset käännökset – valitettavasti näitä ei kirjoitushetkenä ollut saatavilla* |
| Virhevastaus (engl. *Error response*) | 400 BAD REQUEST:<br>Content: {<br>error: ””<br>} | *Virheet niputetaan virhekoodin 400 alle tietoturvasyistä, ja virhelokilta ilmenee todellinen syy virheen palautukseen.* |
| Esimerkkikutsu (engl. *Sample call*) | ({<br>url: “/disruptions/afi/”,<br>type: “GET”,<br>success: function(r) {<br>  console.log(r);<br>}<br>}); | |

## Rajapintakutsu, jolla saadaan verkkopankin tulevat häiriötiedotteet

| REST | Kutsuparametrit | Huomioita |
|---|---|---|
| URL | /disruptions/vepa/scheduled/ | *Kanavatietoa muuttamalla voidaan hakea muista kanavista, ja aikaparametria muuttamalla muista ajankohdista.* |
| HTTP-metodi | GET | |
| Otsikkotiedot (*Header*) | Ei ole | *Sisältää halutun formaatin, system-id:n, ja application-id:n* |
| URL-parametrit | Ei ole | |
| Dataparametrit | Ei ole | *Turhia GET-metodissa* |
| Onnistunut vastaus<br>(engl. *Success response*) | 200:<br>Content: {<br>title:<br>“Häiriötiedote - Verkkopankissa saattaa esiintyä häiriötä.”,<br>start_date: 2030-01-01T00:00:00Z,<br>end_date: 2030-01-03T00:00:00Z,<br>content_fi: “Verkkopankeissa saattaa esiintyä häiriöitä. Vikaa korjataan parhaillaan. Pahoittelemme häiriöstä aiheuttavaa haittaa.”, content_sv: “Det kan förekomma störningar i nätbankerna. Felet åtgärdas som bäst. Vi beklagar besväret som orsakas av störningen.”<br>} | *Tulisi sisältää myös englanninkieliset käännökset – valitettavasti näitä ei kirjoitushetkenä ollut saatavilla* |
| Virhevastaus<br>(engl. *Error response*) | 400 BAD REQUEST:<br>Content: {<br>error: ””<br>} | *Virheet voidaan niputtaa virhekoodin 400 alle samoin kuin julkisessa rajapinnassa. Tässä rajapinnassa tämä ei kuitenkaan ole välttämätöntä, vaan virhekoodeja voidaan palauttaa kuvaavammilla.* |
| Esimerkkikutsu<br>(engl. *Sample call*) | ({<br>url: “/disruptions/vepa/scheduled/”,<br>type: “GET”,<br>success: function(r) {<br>console.log(r);<br>}<br>}); | |

**Yleisimmät HTTP-virheviestit ja niiden käsittely häiriötiedotesovelluksen REST-rajapinnassa**

| HTTP-virhekoodi | Virheviesti | Yleinen tapaus | Esimerkkitapaus | Huomioita |
|---|---|---|---|---|
| 400 | Bad Request | Palvelimelle lähetetty pyyntö on muodostettu väärällä syntaksilla. | Käyttäjä on kirjoittanut käsin hakupyynnön, joka osoittautuu virheelliseksi | Tulee olla julkisen rajapinnan yleisin palautettava virhekoodi. Muut virheet niputetaan tämän virheen taakse, ja virhelokilta tulee selvitä todellinen virheen syy. |
| 401 | Unauthorized | Käyttäjä yrittää päästä käsiksi resurssiin, johon hänellä ei ole oikeuksia. | LDAP:in ulkopuolinen henkilö tekee pyynnön käynnissä olevien häiriötiedotteiden ulkopuolelta.<br><br>Puuttuva tai virheellinen autentikointi. | Ei tule palauttaa käynnissä olevista häiriötiedotteista. Mikäli käynnissä olevia häiriötiedotteita ei ole, palautetaan 200 (SUCCESS) ja tyhjä lista. Käyttäjää ei pystytä varmentamaan. |
| 403 | Forbidden | Käyttäjä teki validin pyynnön palvelimelle, mutta palvelin estää toimenpiteen käyttäjän oikeuksien puutteen vuoksi. | Käyttäjä on kirjautunut sisään sovellukseen, mutta yrittää tehdä toimenpidettä, johon hänellä ei ole oikeuksia. | ”Tällä tunnuksella ei ole luvallista hakea tietoja”. |
| 404 | Not Found | Käyttäjä sai yhteyden palvelimeen, mutta etsittyä | LDAP:in ulkopuolinen käyttäjä yrittää etsiä tietyllä ID:llä | Myös: etsitään häiriötiedotetta, jota ei ole. |

| | | | | |
|---|---|---|---|---|
| | | resurssia ei löytynyt. | häiriötiedotetta, joka on päättynyt. | |
| 500 | Internal Server Error | Palvelin ei pysty käsittelemään pyyntöä jostakin syystä. On tosin syytä käyttää jotakin tarkempaa viestiä aina kun siihen on mahdollisuus | Yleisvirhe. | Yleisin syy löytyy palvelimen (väärin) konfiguroinnista. Mahdollisuuksien mukaan tätä tulisi palauttaa vain harvoin. |
| 502 | Bad Gateway | Välipalvelin ei saa vastausta loppupään palvelimelta (eli käytännössä tietokannasta) | | Esiintyy lähinnä kuormantasaajalla tai muulla vastaavalla välipalvelimella. |
| 503 | Service Unavailable | Palvelin on ruuhkautunut tai huollettavana | Mm. huoltotoimenpiteistä johtuvat virheet. | Viittaa siihen, että palvelu tulee myöhemmin käytettäväksi |
| 504 | Gateway Timeout | Palvelin ei saa vastausta kuormantasaajalta tai muulta välipalvelimelta | Yhteydet palvelimien välissä ovat hidastuneet huomattavasti. | |