# (Avoimet) Rajapinnat hankinnoissa

—

9.6.2016 / Oiva akatemia / ICT-hankinnat -koulutus

# Kaupungin tietotekniikkaohjelma 2015-2017

Tietohallinnon kehittämistä ohjaavat periaatteet

- Palvelujen yhteentoimivuutta, tiedon saatavuutta ja avointa kehitystä tuetaan lisäämällä digitaalisten tietosisältöjen ja tietoteknisten rajapintojen saatavuutta.

# Kaupungin tietotekniikkaohjelma 2015-2017

Kaupungin ICT-ostokäyttäytymistä ja ICT-ostojen ohjausta kehitetään tavoitteena **toimittajariippuvuuden vähentäminen** ja **kilpailun lisääminen**.

- Tietojen saatavuutta parannetaan kaupungin oman organisaation sisällä rakentamalla avoimia rajapintoja olemassa oleviin järjestelmiin ja edellyttämällä avointen ja dokumentoitujen rajapintojen olemassaolo hankittavissa järjestelmissä.

# Mikä on API?

# Mikä on API? (1) https://fi.wikipedia.org/wiki/Ohjelmointirajapinta

Ohjelmointirajapinta (engl. Application programming interface, API)

on **määritelmä**, jonka mukaan eri ohjelmat voivat tehdä pyyntöjä ja vaihtaa tietoja eli keskustella keskenään.

# Mikä on API? (2)

APIt käytössä kaikkialla:

- HTML5 geolokaatio-API: “kerro tämänhetkinen sijainti maailmassa” https://developer.mozilla.org/en-US/docs/Web/API/Geolocation
- näytönohjaimissa: “piirrä nämä kolmiot läpinäkyviksi” https://en.wikipedia.org/wiki/Vulkan_(API)
- käyttöjärjestelmissä: “siirrä tämä ikkuna vasempaan yläkulmaan” https://en.wikipedia.org/wiki/Windows_API
- verkkopalveluissa: esim. “kerro Helsingin kaupunkipyöräpisteiden tilanne” http://api.digitransit.fi/routing/v1/routers/hsl/bike_rental

# Mikä on API? (3)

Huom!

Vaikka esityksen sisältö periaatteessa pätee yleisesti mihin tahansa ohjelmointirajapintoihin, keskitytään tässä kuitenkin internetin yli käytettäviin, ns. web service -rajapintoihin.

# API kioskin luukkuna

# Data- vs. toiminnallinen API

Rajapinnat voi jakaa kahteen ryhmään:

- Datarajapinnat, vain tarjoilevat dataa pyynnöstä
  - OpenAhjo (http://dev.hel.fi/apis/openahjo, http://dev.hel.fi/paatokset)
  - Palvelukartan toimipisterajapinta (http://www.hel.fi/palvelukarttaws/rest/index_en.html, http://dev.hel.fi/servicemap)
- Toiminnalliset rajapinnat, joiden läpi voi tallentaa järjestelmän tietoja
  - Palautejärjestelmän rajapinta (Open311) (http://dev.hel.fi/apis/issuereporting)

# API-lähtöinen- vs. monoliittiarkkitehtuuri (1)

Monoliittinen arkkitehtuuri, jossa komponentit tiukasti yhteenliitetty (“tight coupling”)

- muutokset kalliita ja vaikeita tehdä
- yleensä se yksi käyttöliittymä ja siihen on tyytyminen
- toimittajariippuvuuden vaara suuri
- esimerkkejä: Ilmanlaatuportaali (http://www.ilmanlaatu.fi), vanha Palvelukartta (http://www.hel.fi/palvelukartta/)

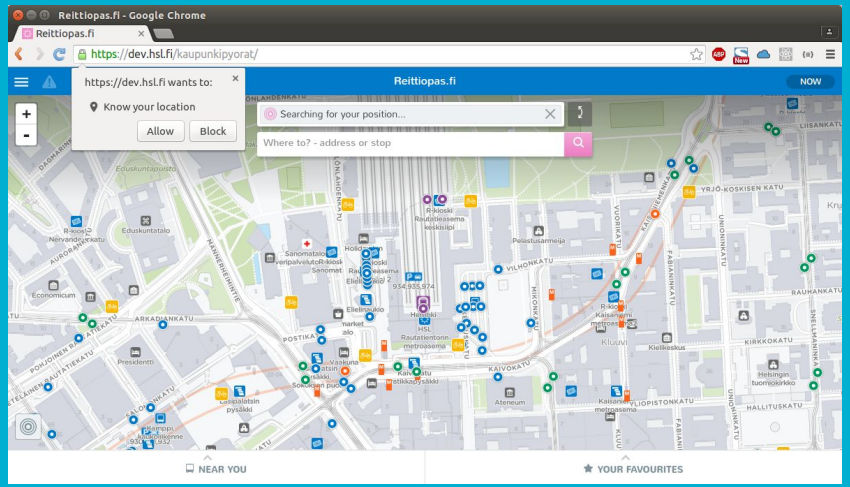
# API-lähtöinen- vs. monoliittiarkkitehtuuri (2)

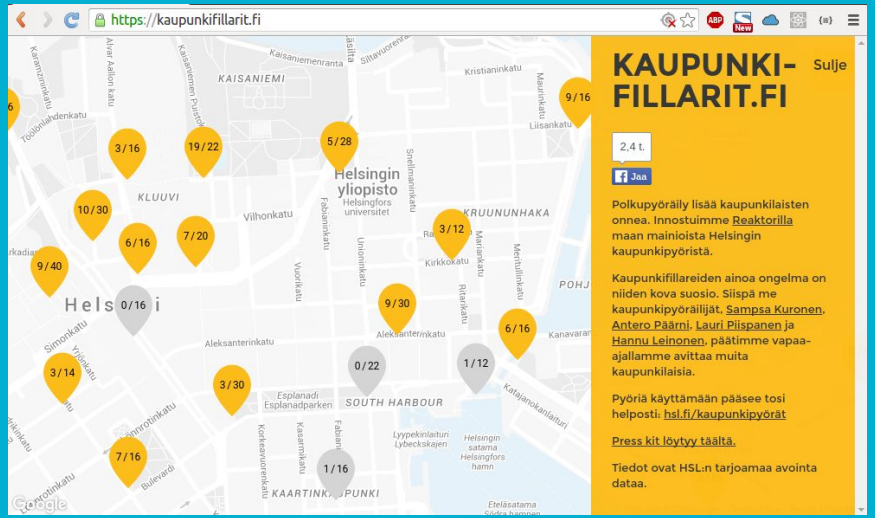
API-lähtöinen arkkitehtuuri, jossa komponentit liitetty toisiinsa rajapintojen avulla (“loose coupling”)

- muutokset komponentteihin vaivattomampia tehdä
- komponenttien vaihtaminen kokonaan mahdollista
- komponenttien kehityksen hajautus eri toimittajille mahdollista
- esimerkkejä: Uusi palvelukartta (http://palvelukartta.hel.fi), Varaamo (http://varaamo.hel.fi)

# API-first kehityksen menestystarina

Vain viikko kaupunkipyörä-APIn julkaisun jälkeen ilmestyi muiden osapuolten kehittämiä käyttöliittymiä



# Mikä on **avoin** API?

# Avoin rajapinta (1) http://www.avoinrajapinta.fi

Määritelmä:

1. Avoimesti dokumentoitu
2. Testattava ja käyttöönotettava

# Avoin rajapinta (2) http://www.avoinrajapinta.fi

HUOM!

Avoin rajapinta

≠

Avoin data tai avoin pääsy tuotantojärjestelmään

# Avoin rajapinta (3) http://www.avoinrajapinta.fi

1. Avoimesti dokumentoitu

- **Ajantasainen** dokumentaatio julkisesti jaossa
- AINAKIN: rakenteisella API-kuvauskielellä tehty (esim. swagger, api blueprint)

  esim. http://dev.hel.fi/apis/linked-events/
- MIELELLÄÄN: luonnollisella kielellä kirjoitetut ohjeet
- Ei Word-dokumentteja!

# Avoin rajapinta (4) http://www.avoinrajapinta.fi

2. Testattava ja käyttöönotettava

- Rajapintaan on pystyttävä tekemään testimielessä pyyntöjä
- Jos vaaditaan rekisteröitymistä, hyväksynnän oltava automaattista
- Jos data on suljettua (tai järjestelmä on esimerkiksi irti julkisesta internetistä), täytyy rajapintaa silti pystyä testaamaan joko:
    - testipalvelimella, joka sisältää realistista testidataa, tai
    - mahdollistamalla järjestelmän asentaminen paikallisesti (+ realistista testidataa)

# Avointen rajapintojen edut kaupungilla

- Vähentää riippuvuutta yksittäisestä toimittajasta
- Integraatiot muihin järjestelmiin helpompia ja halvempia tehdä
- Mahdolliset synergiaedut organisaatiorajojen yli (helpompi harmonisointi, liiketoiminta helpottuu jos muualla saman apin toteuttavat järjestelmät)

# Avointen rajapintojen haasteet kaupungilla

- Toimittajien haluttomuus
  - tulkinnat rajapinnan avoimuuden kriteereistä
- Vanhojen tietojärjestelmien muokkaamisen vaikeus
  - ikivanhat (ja mahdollisesti suljetut) tiedostomuodot
- Vaikeaa virallisesti käyttää kaupungin muita rajapintoja, jos niille ei anneta palvelutasolupausta (esim. SSO)

# APIen hallinta

- APIen hallintateknologiat tulevat kyseeseen jos APIn suosio kasvaa hyvin suureksi
- Rate limiting / throttling
- API-avaimet ja käyttäjäksi rekisteröityminen
- Mittaaminen ja analytiikka
- Kaupungille rakenteilla API-hallintajärjestelmä, johon toivottavasti voidaan liittää mielivaltaisia kaupungin rajapintoja

# Standardit

- Tietomallien standardit
  - esim. open311 (http://www.open311.org/)
- APIen “standarditoimintatavat”
  - sivutus
    - jos kutsut palauttavat isoja määriä tietoa, on vastaukset hyvä jakaa “sivuiksi”
  - versiointi
    - jos rajapinnan määritelmään tehdään muutoksia, jätetään vanha API kuitenkin elämään
  - tuetut formaatit
    - nykyaikaisten APIen pitäisi palauttaa ainakin JSONia ja mielellään myös XML:lää sekä mahdollisesti muita riippuen tietosisällöstä

# Miten hankkia?

# Huomioitavaa hankkiessa / API olemassa olevaan järjestelmään

- Tietomallin speksaus
  - käyttäjäystävällisyys (olisi hyvä kerätä kommentteja potentiaalisilta käyttäjiltä)
  - käyttötapaukset (filtterit, aggregointi, datan pyytäminen eri tavoin järjestettynä)
  - tarvitseeko palvella eri käyttäjäryhmiä? (esim. virkamiehen tarvitsee nähdä tietomallista kenttiä, joiden pitää olla piilotettuna yleisöltä)
- Avoimuuden kriteerien täyttyminen
  - julkisesti saatavilla oleva dokumentaatio, mielellään heti ensimmäisistä versioista lähtien
  - testattavuus

# Huomioitavaa hankkiessa / kokonaan uusi järjestelmä

- API-first kehitystapa eli “syödään omaa koiranruokaa”
  - Määritellään käyttötapaukset eli rajapintojen tarve ja mitkä rajapinnoista halutaan avoimiksi
  - Järjestelmän dataan pääsy ainoastaan rajapintojen kautta
  - Käyttöliittymä ja taustajärjestelmä+API(t) hyvä erottaa alusta alkaen toisistaan riippumattomiksi komponenteiksi
    - Helpointa tehdä ketterästä ja iteratiivisesti
    - Mahdollista tilata komponentit eri toimittajilta, joka vaatii aktiivista koordinaatiota ja toimittajien yhteistyötä

# Kysymyksiä?