581358 Ohjelmistoarkkitehtuurit

Harjoitus 3

1) Yrityksen tuottaman erään sähköisen palvelun käytöstä kerätään jatkuvasti lokitietoja yhteen tiedostoon. Lokiin kirjautuu tietoja palvelupyyntöjen saapumisajoista, järjestelmän antamien vastausten toimitusajoista, ja mahdollisista poikkeustilanteista (jos pyyntöä ei voitu jostain syystä täyttää). Pyyntöjä on useampaa eri tyyppiä. Kerran vuorokaudessa kaikki kirjaukset sisältävästä lokitiedostosta koostetaan tiivistetty tekstimuotoinen raportti, joka näyttää yhteenvedon pyyntötyypeittäin jaoteltuna ja tunneittain viimeksi kuluneen 24 tunnin ajalta:

   – pyyntöjen lukumäärä tunneittain

   – poikkeusten lukumäärää tunneittain

   – Pyynnön keskimääräinen käsittelyaika (keskiarvo ja mediaani) tunneittain

   Tiedot kirjataan lokitiedostoon yksittäisinä tapahtumina tietuemuodossa (riveinä), joka koostuu vastaanottavan palvelimen tuottamasta palvelupyynnön tunnisteesta (request ID), kirjauksen tyypistä (pyyntötyypin tunnistekoodi, vastaus, poikkeus) ja aikaleimasta (tapahtuma-aika, ei kirjausaika). Palvelupyynnön tunniste kytkee yhteen palvelupyynnön ja sen vastauksen/poikkeuksen. Lokiin kertyy kirjauksia vuorokauden aikana n. 100 000. Huomaa, että kirjausten järjestyksestä lokissa *ei voi* suoraan päätellä tapahtumien järjestystä. Voit kuitenkin olettaa, että request ID on aina yksikäsitteinen tarkasteltavan 24 tunnin aikajaksolla (lokitiedosto nollautuu 24 tunnin välein automaattisesti). Esimerkki lokitiedostosta:

ID		TYPE	TIME
13334287	REQA	1379680320947	
13334287	RESP_OK	1379680321026	
13334288	REQB	1379680321251	
13334390	REQA	1379680321260	
13334390	RESP_OK	1379680321290	
13334288	ERR_110	1379680321266	

   Hahmottele arkkitehtuuri yhteenvetoraportin koostavalle ohjelmistoratkaisulle. On tärkeää, että ratkaisuun on helppo lisätä uusia raporttimuotoja (esim diagrammien tuottaminen koostetusta datasta) sekä tuki uusille pyyntötyypeille. Mitä luennoilla käsiteltyjä *arkkitehtuurityylejä* voisit soveltaa tämän järjestelmän arkkitehtuurissa?

2) Tutustu *Oracle Tuxedo* väliohjelmistoon (middleware):
   http://en.wikipedia.org/wiki/Tuxedo_%28software%29

   Kerro lyhyesti, mikä *Tuxedo* on ja minkälaisiin järjestelmiin se on tarkoitettu (katso myös esimerkiksi *Tuxedo Data Sheet* –mainoslehtistä Oraclen sivuilta)? Mitä luennoilla läpikäytyjä *tyylejä/patterneja* tunnistat Tuxedo –ohjelmiston kuvauksesta wikipedian artikkelista?

3) Viestinvälitykseen perustuvia hajautettujen järjestelmien toteuttamisen mahdollistavia ohjelmistoja kutsutaan usein nimillä *Message Queue* tai *Message Oriented Middleware*. Viestien välityksen semantiikkaan liittyy monia huomioon otettavia seikkoja: katso listaus Wikipedian *Message queue* –artikkelista otsikon *Usage* alta (http://en.wikipedia.org/wiki/Message_queue).

   Ota selvää, miten nämä asiat on ratkaistu *RabbitMQ* (https://www.rabbitmq.com/) – viestinvälityspalvelussa (kaikkiin kysymyksiin ei välttämättä löydy vastausta palvelun dokumentaatiosta).

   Huom! RabbitMQ tukee useita viestinvälitysprotokollia, joten keskity yleisimpään AMQP – protokollaan: https://www.rabbitmq.com/tutorials/amqp-concepts.html

4) Tutustu *BitTorrent* -tiedostonjakoprotokollaan ja vastaa kysymyksiin. Lähteitä:
   http://help.bittorrent.com/customer/en/portal/articles/178790-the-basics-of-bittorrent
   http://en.wikipedia.org/wiki/BitTorrent
   http://www.youtube.com/watch?v=urzQeD7ftbI (YouTube - "Torrents Explained: How BitTorrent Works" by raindrop264)
   a) Mitä tarkoittavat käsitteet *Torrent, Torrent file, Swarm, Seed, Peer, Seeder, Leech*?
   b) Miten Torrent-tiedoston avaava Torrent Client saa tietää, mistä varsinaisen sisältötiedoston osat löytyvät? Eli miten Client saa tietää torrentia jakavien toisten solmujen osoitteet ja niiden jakamat osat?
   c) Mitkä ovat BitTorrent:n (ja yleensä vertaisverkkojen) pääasialliset edut ja toisaalta haitat/uhat?

5) Ensimmäisten harjoitusten 4. tehtävässä pohdittiin harjoitustehtävien kirjausjärjestelmän tuote- ja projektiriskejä. Riskianalyysin tuloksena on nyt saatu lista merkittävimmistä riskeistä (alla). Pohdi, minkälaisilla arkkitehtuuritason suunnitteluratkaisuilla voisi lieventää näitä riskejä. Ota huomioon, että ratkaisusi ovat suhteessa järjestelmän luonteeseen (kyseessä ei ole kriittinen järjestelmä, lähinnä apuväline). Miten arkkitehtuuria (laajasti ymmärrettynä, osana kehitysprosessia) voisi muuten käyttää riskien hallinnan apuna?
   a) Tuoteriskit
      i) Monen alustan tukeminen voi osoittautua vaikeaksi
      ii) Muutokset toimintaympäristössä/muitten järjestelmien rajapinnoissa
      iii) Miten taataan saatavuus 24/7? Oltava jokin luotettava varajärjestelmä.
      iv) Tietojen siirto Kurkeen on hankalaa. Suoraa yhteyttä ei saada järjestelmien välille, vaan tiedon siirto tapahtuu tiedostojen kautta.
   b) Projektiriskit
      i) Ohtu-ryhmien osaaminen vaihtelee paljon. Tehtävä voi osoittautua liian haastavaksi joiltain osiltaan.
      ii) Tiedon kulku eri toteutusryhmien välillä (toteutus peräkkäisissä ohtu-projekteissa).
      iii) Rinnakkain etenevien tiimien kommunikointi ja synkronointi, jos järjestelmä toteutetaan kahden ohtu-ryhmän yhteistyönä.