TONTTIPÖRSSI

LAHDEN AMMATTIKORKEAKOULU
Tekniikan ala
Ohjelmistotekniikan koulutusohjelma
Ohjelmistotekniikan suuntautumisvaihtoehto
Opinnäytetyö
Kevät 2011
Ville Pirskanen

Lahden ammattikorkeakoulu
Tietotekniikan koulutusohjelma

PIRSKANEN, VILLE: Tonttipörssi

Ohjelmistotekniikan opinnäytetyö, 43 sivua

Kevät 2011

TIIVISTELMÄ

Opinnäytetyön tavoitteena oli luoda tonttipörssi-sivusto päijäthämäläisille kunnille. Kuntien tämänhetkiset tonttipörssi-sivustot ovat sekavia, ja niiden ylläpito on vaikeaa. Uuden sivuston on tarkoitus yhdistää eri kuntien sivustot yhdenmukaisiksi ja tuoda sivustoille interaktiivisuutta. Interaktiivisuutta saatiin lisäämällä kartan keskeisimmäksi elementiksi Google Maps -kartta. Sivuston ylläpito pitää myös olla helppoa. Ylläpitoa varten sivustolle toteutettiin hallintapaneeli, jonka kautta sivuston tontteja sekä sisältötekstejä voidaan helposti hallita.

Tonttipörssi tehtiin osana KOTI-vetovoimaa Päijät-Hämeen maaseutukuntiin -projektia. Koti-projektin tavoitteena on lisätä maaseutukuntien vetovoimaa asuinpaikkana ja yritysten sijoittumispaikkakuntina. Tonttipörssin toteutukseen osallistui kolme Lahden ammattikorkeakoulun insinööriopiskelijaa: ohjelmistotekniikan, mediatekniikan ja ympäristöteknologian opiskelijat.

Opinnäytetyössä tutustutaan myös Google Mapsiin ja siihen, miten Google Maps voidaan lisätä sivustolle Google Maps JavaScript API:n avulla. API:sta käydään läpi sen perusominaisuudet ja niitä ominaisuuksia, joita tarvitaan, että Mapsin päälle saadaan rakennettua tonttipörssi-palvelu. Tonttipörssi-sivustosta käydään läpi, miten mikäkin sivuston elementti on tehty ja miten niiden ylläpito onnistuu. Opinnäytetyön tutkimusongelma on selvittää, miten luodaan helposti ylläpidettävä tonttipörssi-sivusto ja miten Google Maps voidaan liittää sen keskeisemmäksi elementiksi.

Avainsanat: tonttipörssi, Google Maps, Google Maps JavaScript API

Lahti University of Applied Sciences
Degree Programme in Information Technology

PIRSKANEN, VILLE: Plot market webpage

Bachelor's Thesis in software engineering, 43 pages

Spring 2011

ABSTRACT

The objective of this thesis was to make a plot market webpage for the municipalities of Päijät-Häme. The current plot market webpages are disorganized and maintenance of the webpages is difficult. The purpose of the new plot market webpage is to make the different municipalities webpages uniform. The webpage was part of a project called KOTI-vetovoimaa Päijät-Hämeen maaseutukuntiin. The goal of this project was to bring information about municipalities of Päijät-Häme to individuals and companies. The research problem of the thesis was how to make easily administrated plot market webpage and how to integrate Google Maps to it.

The new webpage is more interactive than the current webpages. Interactivity is achieved by placing Google Maps as a central element of the plot market webpage. In Google Maps the user can see the plots that a municipality is currently selling. Also maintenance of the new site needs to be easier than with the current sites. For easier page maintenance the site includes an admin panel. In the admin panel the administrator can add, modify and delete plots from the plot market. The administrator can also edit text contents that are displayed on the page.

The first part of the thesis is to familiarize the reader to Google Maps JavaScript API. The thesis instructs how to add Google Maps to a webpage. It also goes through the basics of Google Maps JavaScript API and those features which were used to make the implementation of the plot market to Google Maps possible. The second part of the thesis tells how the plot market webpage was made and how the admin can edit different parts of the page.

Key words: Plot market, Google Maps, Google Maps JavaScript API

SISÄLLYS

1 JOHDANTO

Opinnäytetyön tavoitteena on toteuttaa uudenlainen tonttipörssi-sivusto päijäthämäläisille kunnille. Tonttipörssi-sivusto on sivusto, jolla kunta voi mainostaa myytävänä olevia tontteja asiakkailleen.

Aiemmat kuntien tonttipörssi-sivustot ovat olleet sekavia ja niiden ylläpito on ollut vaikeaa. Uuden tonttipörssin on tarkoitus tuoda apua juuri ylläpidettävyyteen ja saada kunnille yhtenäisen näköiset tonttipörssit. Tonttipörssin ensimmäisenä kohteen käytetään Kärkölän kunnan Louhivaaran aluetta. Myöhemmin tonttipörssi on tarkoitus saada myös käyttöön toisille kunnille ja alueille.

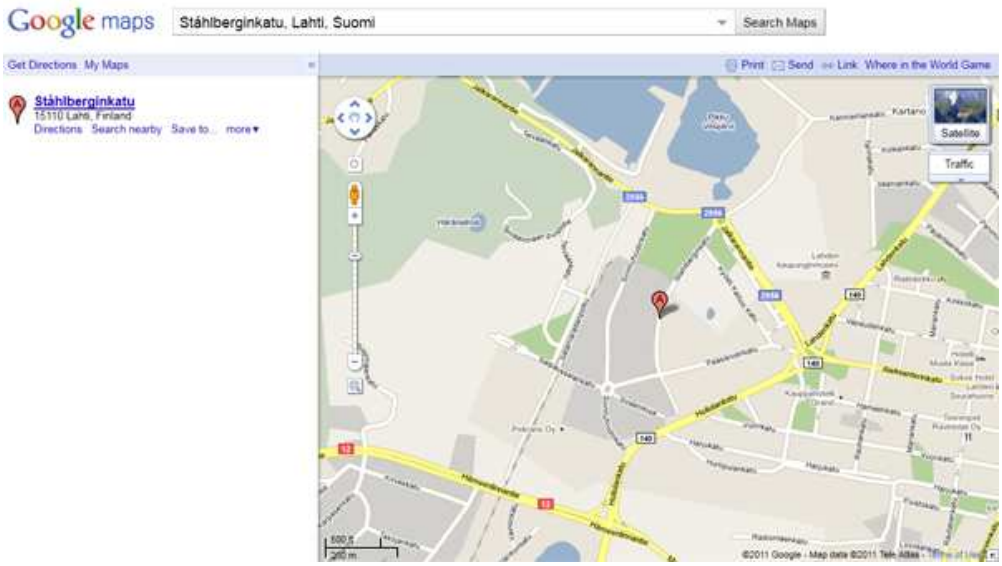
Tonttipörssi tehtiin osana KOTI-vetovoimaa Päijät-Hämeen maaseutukuntiin -projektia. Koti-projektin tavoitteena on lisätä päijäthämäläisten kuntien vetovoimaa yksityisten henkilöiden asuinpaikkana ja yritysten sijoittumispaikkakuntina. Projektissa on mukana seitsemän kuntaa: Asikkala, Hartola, Hollola, Kärkölä, Nastola, Padasjoki ja Sysmä. Koti-projektin avulla kuntien sisäinen sekä kuntien välinen yhteistyö aktivoituu. (Koti-projekti 2011.)

Opinnäytetyössä tutustutaan Google Mapsiin ja sen JavaScript API:iin. Google Mapsista kerrotaan yleisesti, mikä se on ja miten Google Mapsin voi lisätä omalle sivustolle käytettäväksi Maps JavaScript API:n avulla. API:sta käydään myös läpi sen perusominaisuudet ja niitä palveluita, jotka mahdollistavat tonttipörssin rakentamisen Mapsin päälle. Työn tutkimusongelmana on selvittää, miten tehdä helppokäyttöinen ja helposti ylläpidettävä tonttipörssi-palvelu ja miten Google Mapsia voidaan käyttää järjestelmässä hyväksi.

Tässä työssä kerrotaan aluksi luvussa 2 yleisesti, mikä Google Maps on. Luku 3 käsittelee Google Mapsin JavaScript API:a ja sen keskeisimpiä ominaisuuksia. Luvussa 4 keskitytään projektissa luotuun tonttipörssi-palveluun.

2 GOOGLE MAPS YLEISESTI

Google Maps on Googlen tarjoama ilmainen karttapalvelu (kuvio 1). Maps oli alun perin tehty tietokoneen selaimella käytettäväksi, mutta myöhemmin se on laajentunut myös matkapuhelimiin. Maantieteellisestä sijainnista riippuen käyttäjälle tarjotaan mahdollisuuksia erilaisten karttanäkymien ja palveluiden käyttöön. Karttaa voidaan vapaasti zoomata ja liikutella click and drag -menetelmällä. (Google Maps 2011.)



KUVIO 1. Google Maps -sivusto

2.1 Kartan ulkoasu

Erilaisia karttanäkymiä on tällä hetkellä kolme erilaista: perinteinen kartta, satelliittikartta ja hybridi-kartta, joka yhdistää kaksi edellä mainittua karttaa yhdeksi. Perinteinen näkymä on Google Mapsin vakionäkymä. Kartalla näkyy maiden, kaupunkien, merien ja suurempien nähtävyyksien nimet. Näkymästä voidaan valita myös maastotila, jolloin kartalla näkyy myös maan muodot.

Satelliittikartalla käyttäjä näkee avaruudesta käsin otetut satelliittikuvat. Nämä kuvat kattavat koko maapallon. Tarkempia kuvia on otettu ainoastaan valituista

kohteista, kuten maapallon suurimmista kaupungeista, joten jos käyttäjä haluaa tarkastella jotain aluetta lähempää, niin se ei välttämättä ole aina mahdollista (Google Maps 2011).

Hybridi-kartta yhdistää perinteisen kartan ja satelliittikartan. Käyttäjä näkee ruudullaan satelliittikuvat, ja niiden päälle on tuotu perinteisestä kartasta paikkojen tunnisteet.

2.2 Palvelut

Google Maps tarjoaa käyttäjilleen myös erilaisia palveluita. Näistä suosituimpia ovat esimerkiksi ajo-ohjeet, katunäkymä ja liikennetiedot. Palveluita tulee koko ajan lisää ja myös niiden käyttöalueet laajenevat. Googlen tarjoamien palveluiden lisäksi käyttäjät voivat luoda omia palveluitaan käyttämällä Googlen tarjoamia Google Maps API:ja.

3 GOOGLE MAPS JAVASCRIPT API

Google Maps JavaScript API on Googlen vuonna 2005 julkaisema palvelu, jonka avulla käyttäjä voi lisätä omalle sivustolleen Googlen karttapalvelun (Google Blog API v1 2005). API tarjoaa käyttäjälleen useita tapoja muokata karttaa sekä rakentaa erilaisia palveluita kartan päälle (Google JavaScript API 2011).

3.1 JavaScript API:n käyttöehdot ja käytön rajoitukset

Google Maps JavaScript API on vapaasti käytettävissä kaikilla sivustoilla, joita voidaan käyttää maksutta. API:a saa käyttää ei-kaupallisilla ja kaupallisilla sivustoilla, kunhan ne täyttävät Google Maps API:n käyttöehdot. Sivustolla, jolla karttaa käytetään, ei saa harrastaa laitonta toimintaa eikä kartan avulla saa kerätä käyttäjistä henkilökohtaisia tietoja. (Google Terms of Use 2011.)

Google Maps API:sta on myös tarjolla maksullinen Premier-palvelu. Palvelu on tarkoitettu yrityksille, jotka haluavat tukipalveluita Googlelta, täyden kontrollin karttamainostukseen tai käyttää tehokkaammin Googlen tarjoamaa geocoding-palvelua. Hinta Premier-palvelulle on vähintään 10 000 $/vuosi, mutta se määräytyy tarkemmin sivuston käytön mukaan. (Google Premier 2011.)

3.2 Google Maps JavaScript API:n käyttöönotto

API:n keskeinen elementti on kartta itse. Tässä osiossa käydään läpi, miten käyttäjä voi liittää kartan omille sivuilleen ja alkaa käyttää sitä. Samalla myös käydään läpi keskeisimmät kartan lataukseen liittyvät parametrit.

Esimerkkikartan lisäämisestä sivulle on nähtävissä seuraavassa kuvassa (kuvio 2). Kuvan koodi tekee sivustolle 500 px x 500 px laatikon, johon kartta liitetään. Kuvion 2 luoma karttasivu on nähtävissä kuviossa 3.

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <style type="text/css">
5    #map_canvas { width: 500px; height: 500px }
6  </style>
7  <script type="text/javascript"
8          src="http://maps.google.com/maps/api/js?sensor=false"></script>
9  <script type="text/javascript">
10   function initialize() {
11     var latlng = new google.maps.LatLng(60.985, 25.642);
12     var myOptions = {
13       zoom: 16,
14       center: latlng,
15       mapTypeId: google.maps.MapTypeId.ROADMAP
16     };
17     var map = new google.maps.Map(document.getElementById("map_canvas"),
18         myOptions);
19   }
20 </script>
21 </head>
22 <body onload="initialize()">
23   <div id="map_canvas"></div>
24 </body>
25 </html>
```

KUVIO 2. Kartan lisääminen sivustolle

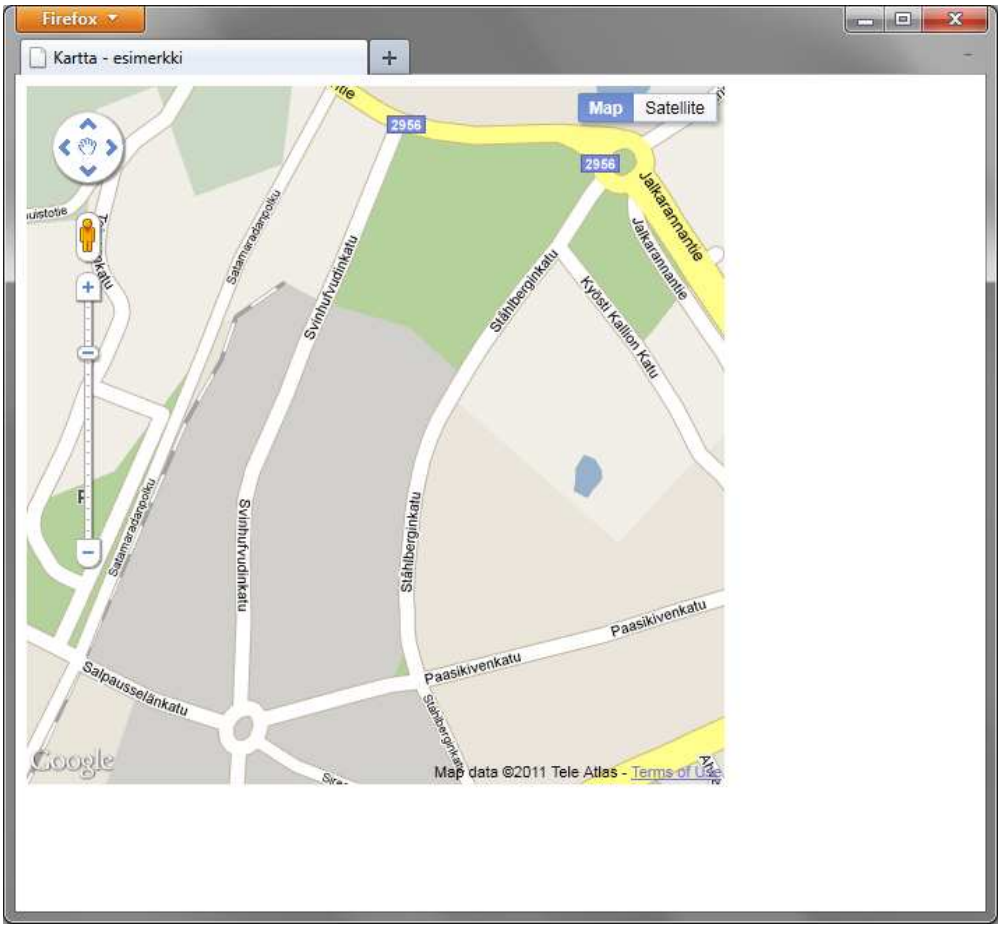
Aivan ensimmäisenä suositellaan määrittämään sivuille DOCTYPE. Tämä tehdään siksi, että sivusto toimisi samalla lailla mahdollisimman monella selaimella. Doctypeksi suositellaan määrittämään HTML5. Doctypen määrittely on kuviossa 2 rivillä 1. (Google JS Tutorial 2011.)

Seuraavaksi sivustolle täytyy tehdä jokin tila, johon kartta liitetään. Kuviossa 2 rivillä 23 luodaan div-elementti karttaa varten. Div-elementille täytyy myös määrittää jokin koko, tai muuten kartan kooksi tulee 0 pikseliä. Kuviossa 2 rivillä 5 määritetään kartan div-elementin kooksi 500 x 500 pikseliä.

JavaScript API:n lataus tapahtuu lataamalla Google Mapsin JavaScript API. Kuviossa 2 riveillä 7 - 8 ladataan API käyttöön. Annettu URL-osoite osoittaa JavaScript-tiedostoon, joka lataa kaikki tarvittavat symbolit ja määritykset, jotta karttaa voidaan käyttää. API:a ladatessa käyttäjän tulee määrittää sensor-parametri. Jos sivustolla tarvitsee tunnistaa käyttäjän maantieteellinen sijainti, tulee sensor-parametri asettaa todeksi. Muulloin parametri voi olla epätosi. API:a

ladatessa voidaan määrittää myös language ja libraries-parametrit. Language-parametrilla voidaan määrittää kartalle jokin tietty kieli. Libraries-parametrin avulla voidaan ladata eri kirjastoja kartalle käytettäväksi. Kirjastoista kerrotaan lisää luvussa 3.2.1 Kirjastot.

Jotta kartta piirtyisi div-elementtiin, täytyy sivuston latauduttua luoda Map-luokasta olio. Kuviossa 2 olion luonti tehdään initialize-funtiossa riveillä 10 - 19. Olio tarvitsee luontivaiheessa määreitä, jotka käydään tarkemmin läpi luvussa 3.3 Map-luokka. On myös tärkeää, että kartta liitetään div-elementtiin vasta, kun sivusto on täysin latautunut. Tämä varmistetaan liittämällä initialize-funktio body-elementin onload-tapahtumaan. Funktion liittäminen on kuviossa 2 rivillä 22.



KUVIO 3. Yksinkertainen karttasivu

3.2.1 Kirjastot

Normaalisti JavaScript API:a ladatessa ladataan vain osa API:n tarjoamista ominaisuuksista. Jos sivustolla halutaan käyttää API:n tarjoamia ulkoisia kirjastoja, niin ne tulee ladata erikseen. Tämä onnistuu antamalla API:n latauksen yhteydessä libraries-parametri. Tällä vältetään suurempien kirjastojen turhaa latausta. (Google API v3 2011.)

Käyttäjä voi ladata yhden tai useamman kirjaston käytettäväksi. Jos halutaan ladata useampi kuin yksi kirjasto kerrallaan, tulee kirjastojen nimet erotella pilkuilla (kuvio 4). Kun kirjasto on ladattu, tulee se käyttöön google.maps.kirjastonNimi nimiavaruuteen. (Google API v3 2011.)

```
<script
  type="text/javascript"
  src="http://maps.google.com/maps/api/js?libraries=geometry,panoramio&sensor=false">
</script>
```

KUVIO 4. Geometry ja panoramio -kirjastojen lataus

Kirjastoja tuotetaan koko ajan uusia, mutta tällä hetkellä Google tarjoaa kolme eri kirjastoa: geometry, adsense ja panoramio.

- Geometry-kirjaston avulla voidaan laskea skalaarisia suureita kartalta, esimerkiksi välimatkoja ja pinta-aloja.
- Adsense-kirjasto tarjoaa mahdollisuuden tuoda Googlen mainoksia kartalle. Jotta adsense-kirjastoa voidaan käyttää, tulee käyttäjällä olla ennestään Googlen AdSense-tili.
- Panoramio-kirjastolla voidaan kartalle tuoda kuvia Googlen tarjoamasta Panoramio-palvelusta. (Google API v3 2011.)

3.2.2 Versiot

Vuonna 2005 Google julkaisi API:n ensimmäisen version (Google Blog API v1 2005). Se korvaantui uudella versiolla jo vuonna 2006 ja nykyisin käytössä oleva version 3 julkaistiin vuonna 2010. Googlen kartat on suunniteltu niin, että myös

tänäkin päivänä voidaan käyttää ensimmäistä versiota, mutta se ei ole suositeltavaa, vaan Google suosittelee käyttämään aina uusinta versiota. (Google JavaScript API 2011.)

Aivan ensimmäisessä versiossa Google oli määrännyt käyttäjiään rekisteröimään API-avaimen. Avaimen avulla Google pystyi tarkkailemaan sivun latauksia. Sivustojen käyttäjille oli myös asetettu päivittäinen kuvien lataus rajoitus. Tämä tarkoitti sitä, että yksittäinen käyttäjä pystyi lataamaan päivässä ainoastaan tuhat eri karttakuvaa. Tämä ei siis tarkoittanut, että yhden avaimen kuvia voidaan ladata tuhat päivässä, vaan yksi käyttäjä pystyi lataamaan tuhat eri kuvaa. Jos tämä rajoitus ylitettiin, niin käyttäjälle annettiin ilmoitus asiasta eikä hän pystynyt enää lataamaan uusia karttakuvia sinä päivänä. (Google API v1 2005.)

Versio 2:n tulon myötä kuvien latausten rajoitus poistettiin, mutta API-avaimen käyttöpakko säilyi. Google edelleen tarkkaili sivuston latauksia avaimen avulla. Jos sivustolle oli odotettavissa yli 500 000 päivittäistä käyttäjää, oli Googlelle syytä ilmoittaa asiasta ennen sivuston julkaisua. (Google Blog API v2 2006.)

Uusimmassa versiossa Google on panostanut API:n toimivuuden nopeuteen. Päältäpäin versio 3 on hyvin samanlainen kuin versio 2, mutta se on suunniteltu erityisesti toimivan nopeasti matkapuhelinten selaimilla kuten esimerkiksi Android- ja iPhone-puhelimilla. Myös ladattavan JavaScript-koodin kokoa on yritetty pienentää, minkä toivotaan nopeuttavan API:n toimivuutta. (Google API v3 2011.)

Uusimmassa versiossa käytetään myös hyödyksi MVC-arkkitehtuuria. Muutokset olioissa, kuten esimerkiksi kartalla, tapahtuvat gettereiden ja settereiden kautta ja näitä muutoksia seurataan tapahtumankäsittelijöillä. API-avaimen käyttöpakko on poistettu uusimmasta versiosta. (Google API v3 2011.)

3.2.3 Versiotyypit

Käyttäjä voi määrätä tarkasti, mitä versiota hän haluaa käyttää, ja usein tämä on myös suositeltavaa. Versioita on tällä hetkellä kaksi erilaista: kehitysversio ja numeroitu versio. Kehitysversio on uusin tämänhetkinen versio, ja sen käyttäjä saa käyttöönsä karttaa ladatessa v-parametrilla (esim. v=3). Tätä versiota kutsutaan myös nimellä nightly-version.

Numeroitu versio on tarkka versio tietystä versiosta. Näitä on kolme erilaista: release-, frozen- ja retired-versiot. Release-versio pitää sisällään nightly-versiosta tuodut muutokset. Aina kun uusi release-versio julkaistaan, aiempi release-versio muutetaan frozeniksi. Frozen-versioon ei enää tehdä koodi muutoksia, joten se toimii aina samalla tavoin. Tämä tarkoittaa myös sitä, että jos versiossa on jokin bugi, niin sitä ei korjata enään tähän versioon. Joka kerta, kun uusi frozen-versio tulee, niin vanha frozen-versio siirtyy retired-versioksi. Jos käyttäjä yrittää käyttää retired-versiota, niin palvelimelta ladataan sen sijaan aina uusin frozen-versio. Numeroidun version käyttäjä saa käyttöönsä määrittämällä v-parametrin arvoksi versio.alaversio (kuvio 5). (Google API v3 2011.)

```
1 http://maps.google.com/maps/api/js?v=3.4&sensor=false
```

KUVIO 5. Numeroidun version valinta

3.2.4 Version valinta

Google on antanut versioiden valintaan seuraavanlaisia ohjeita:

- Tuotantosovelluksen tulee aina käyttää alaversiota (esim. v=3.3).
- Premier-sopimus ei kata nighty-versioita. Premier-palvelun käyttäjien tulee käyttää uusinta release-versiota tai sitä vanhempaa versiota.
- Sovellusta kehittäessä on suositeltavaa valita käyttöön uusin nighty-versio. Jos API:iin lisätään myöhemmin jokin uusi ominaisuus, jota tarvitaan käyttöön, niin silloin tulee aina siirtyä uusimpaan nighty-versioon. Näin

sovelluksen toimivuus pysyy samana koko sen elinkaaren. (Google API v3 2011.)

3.3 Map-luokka

Map-luokka on kaiken keskusta. Siitä voidaan luoda olio, joka liitetään sivustolle johonkin ennalta luotuun elementtiin. Yksinkertaisimmillaan olio tarvitsee vain koordinaatit kartalta, mihin sen keskipiste tulee, kartan zoomaustason sekä karttatyypin (kuvio 2, rivit 11 - 18). Toisin kuin aiemmissa Mapsin versioissa, kartalle tulee antaa jokin karttatyyppi. Aiemmin kartoilla on ollut perustyyppi, mutta nyt se on poistettu käytöstä. (Google JS Tutorial 2011.)

Map-luokan konstruktoriin annetaan kaksi parametriä. Ensimmäinen parametri on HTML-kontaineri, joka on tyypillisimmin div-elementti. Tämä saadaan luomalla div-elementti, jolle annetaan jokin id, ja sen jälkeen hakemalla se esimerkiksi JavaScriptin avulla getElementById()-metodilla. Toisena parametrina annetaan olioliteraali, joka sisältää asetukset, jotka kartta luontivaiheessa saa. Esimerkki olioliteraalista on kuviossa 2 riveillä 12 - 16.

Jotta kartta tietäisi, mihin se keskitetään, tulee sille antaa keskipiste. Tämä tapahtuu luomalla LatLng-objekti. Objektin konstuktoriin annetaan halutun sijainnin latitudi ja longitudi. Esimerkiksi kuviossa 2 rivillä 11 annetut arvot latitudi 60.985 ja longitudi 25.642 keskittää kartan Lahden ammattikorkeakouluun. Tämä luotu LatLng-objekti liitetään kartan konstruktorissa annettavan olioliteraalin center-määreen arvoksi. Kartan zoomaustaso määritetään luvulla 0 - 20, joka asetetaan olioliteraalin zoom-määreen arvoksi.

Jotta kartan tyyppi voidaan määrittää, on luontivaiheessa annettavan olioliteraalin sisällettävä mapTypeId-määre. Erilaisia karttatyyppejä on useita, mutta käytännössä niistä käytetään neljää eri tyyppiä: ROADMAP, SATELLITE, HYBRID tai TERRAIN. Kuviossa 2 rivillä 15 asetetaan kartan tyypiksi roadmap. Roadmap on perinteinen 2D-kartta, joka on myös käytössä vakiona normaalissa Google Mapsissa. Satellite-tyyppi näyttää satelliitilla otetut kuvat kartalla, hybrid

yhdistää satelliittikuvien päälle teiden, kaupunkien, järvien jne. nimet ja terrain näyttää kartalla pinnan muodot. (Google JS Tutorial 2011).

Näillä kolmella (center, zoom, mapTypeId) asetuksella kartta saadaan näkyväksi ja toimivaksi sivuilla. Asetuksia on myös suuri määrä lisää ja niillä voidaan vaikuttaa kartan ulkoasuun ja toimivuuteen. Esimerkiksi minZoom- ja maxZoom-määreellä voidaan määrittää kartalla käytettävien zoomaustasojen raja-arvot tai draggable-määreellä estää kartan liikuttaminen. Myös kartan päälle ilmestyvien kontrollejen määrä määritetään näillä asetuksilla. Kaikkia näitä asetuksia ei tarvitse määrittää, ja niitä voidaan myös myöhemmin vaihtaa ja lisätä.

Joillekin asetuksille on tehty setterit, joilla voidaan kyseistä asetusta muuttaa. Asetukset, joille ei ole tehty erillistä setteriä, voidaan vaihtaa setOptions()-metodilla. Kartalla tapahtuvia muutoksia tarkkaillaan tapahtumankuuntelijoilla. Täysi lista kartan asetuksista ja funktioista löytyy Google Maps JavaScript API:n manuaalista. Kuviossa 6 on esimerkkifunktio, jolla asetetaan map muuttujaan tallennetun map-olion karttatyyppi satelliitiksi ja estetään kartan liikuttelu hiirellä. Karttatyypin vaihtamiseen on tehty oma setteri, joten se voidaan tehdä setMapTypeId()-metodilla, mutta kartan liikutteluun ei ole tehty omaa setteriä, joten sen arvoa joudutaan muuttamaan setOptions()-metodilla.

```
1 function modifyMap()
2 {
3   map.setMapTypeId(google.maps.MapTypeId.SATELLITE);
4   map.setOptions({draggable: false});
5 }
```

KUVIO 6. Kartan asetusten vaihtaminen

3.4 Event-luokat

Selaimessa JavaScript toimii tapahtumapohjaisesti. Tämä tarkoittaa sitä, että JavaScript luo kaikista muutoksista tapahtuman ja olettaa, että ohjelma kuuntelee sitä kiinnostavia tapahtumia. Maps API:ssa on kahden tyyppisiä tapahtumia:

käyttäjän vuorovaikutuksesta aiheutuvia tapahtumia ja Maps API:n oliossa tapahtuvien tilanmuutoksien aiheuttamia tapahtumia. Käyttäjän aiheuttamia tapahtumia ovat esimerkiksi hiiren klikkaus jossain sivun elementissä. Tällöin Maps API monistaa normaalin DOM-tapahtuman API:n omaksi tapahtumaksi ja tämä tapahtuma käsitellään erillään DOM-tapahtumasta. Olion tiloissa tapahtuvat muutokset käsitellään MVC-mallin mukaisesti erillisessä tapahtumankuuntelijassa ja ne ovat nimetty muuttunutarvo_changed. (Google Maps API Events 2011.)

Jokainen Maps API:n olio luo useita eri nimettyjä tapahtumia. Jos ohjelma on kiinnostunut jostain tietystä tapahtumasta, täytyy tapahtumalle luoda tapahtumankuuntelija. Tämä tapahtuu addListener()-metodilla, joka löytyy google.maps.event-nimiavaruudesta. Jokaisella Mapsin oliolla on hieman eri tapahtumat, ja lista näistä tapahtumista löytyy Google Maps JavaScript API:n manuaalista. (Google Maps API Events 2011.)

Jotkin Mapsin oliot on tehty kuuntelemaan käyttäjän tekemiä tapahtumia esimerkiksi hiiren klikkaus tai näppäimistön painallukset. Esimerkkinä google.maps.Marker voi kuunnella esimerkiksi seuraavia tapahtumia: click, dblclick, mouseup, mousedown, mouseover ja mouseout. Nämä muistuttavat paljon normaaleja DOM-tapahtumia, mutta oikeasti nämä ovat osa Maps API:a. Tällä vältetään eri selainten erilainen toimivuus, joka aiheutuu niiden tavasta käsitellä DOM-tapahtumia. Koska tapahtuma tulee Maps API:lta, ei käyttäjän tarvitse tarkistaa, mikä selain on käytössä, ja tehdä jokaiselle selaimelle eri tapahtumankäsittelijää. Näissä tapahtumissa tulee mukana myös jokin tieto tapahtumasta, esimerkiksi hiiren sijainti kartalla. (Google Maps API Events 2011.)

Niin kuin normaalistikin, MVC-oliolla on eri tiloja, näin on myös Maps API:n oliolla. Olioiden tilamuutoksia seurataan tapahtumankuuntelijoilla. Joka kerta kun olion tila muuttuu, luodaan kyseisestä tilan muutoksesta tapahtuma. Esimerkiksi aina kun kartan zoomaustaso muuttuu, luodaan zoom_changed tapahtuma. (Google Maps API Events 2011.)

Vaikka käyttäjän muutoksista aiheutuvat tapahtumat ja MVC-tilamuutostapahtumat näyttävät samoilta, tulee niitä käsitellä hieman erilailla. Kumpiakin tapahtumia kuunnellaan addListener()-metodilla, mutta toisin kuin käyttäjän aiheuttamista tapahtumissa, MVC-tilamuutostapahtumissa ei tuoda mukana mitään informaatiota esimerkiksi hiiren sijainnista. (Google Maps API Events 2011.)

addListener()-metodi ottaa sisäänsä kolme parametriä. Ensimmäisenä on olio, johon tapahtumankuuntelija liitetään. Toisena parametrinä on tapahtuma, jota kuunnellaan, ja kolmantena on funktio, joka suoritetaan, kun toisena parametrina annettu tapahtuma on tapahtunut.

```
1  var map;
2  function initialize()
3  {
4    var latlng = new google.maps.LatLng(60.985, 25.642);
5    var myOptions = {
6      zoom: 16,
7      center: latlng,
8      mapTypeId: google.maps.MapTypeId.ROADMAP
9    };
10
11   map = new google.maps.Map(document.getElementById("map_canvas"),
12       myOptions);
13
14   google.maps.event.addListener(map, 'click', function(event) {
15     alert("Karttaa klikattu kohdassa: " + event.latLng);
16   });
17
18   google.maps.event.addListener(map, 'zoom_changed', function() {
19     alert("Uusi zoomitaso: " + map.getZoom());
20   });
21 }
```

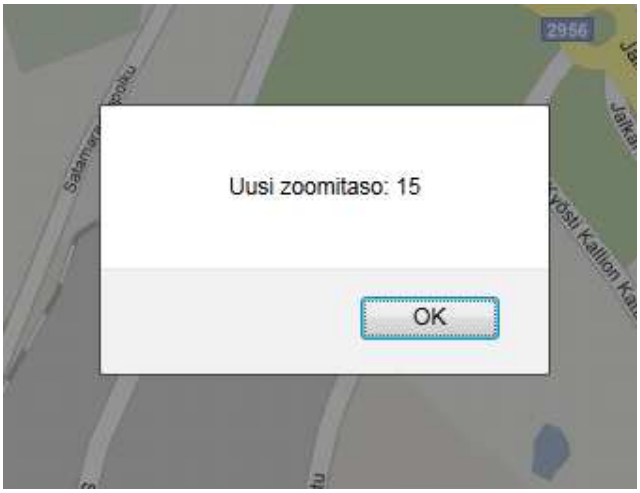
KUVIO 7. Esimerkki tapahtumakuuntelijoista

Kuviossa 7 on yksinkertainen esimerkki siitä, miten kartta-olioon liitetään kaksi erilaista tapahtumaa. Riveillä 14 - 16 liitetään karttaan käyttäjän klikkausta kuunteleva tapahtumankuuntelija. Ensimmäisenä parametrina annetaan map-olio, toisena click-tapahtuma ja kolmantena on funktio, joka suoritetaan klikkauksen tapahduttua. Koska kyseessä on käyttäjän klikkauksesta aiheutuva tapahtuma, niin suoritettavalle funktiolle voidaan antaa tapahtumasta parametri. Tässä tapauksessa

tapahtuma tallentuu event-muuttujaan. Kun käyttäjä on klikannut karttaa, luodaan alert-ikkuna, jossa lukee käyttäjän klikkaaman kohdan koordinaatit (kuvio 8).

KUVIO 8. Click-tapahtuma

Kuvion 7 riveillä 18–20 liitetään karttaan zoomaustason vaihtelua tarkkaileva tapahtumankuuntelija. Koska nyt kyseessä on MVC-olion tilassa tapahtuva muutos, ei tapahtumaa käsittelevälle funktiolle voida antaa mitään parametria. Kun kartan zoomaustaso on vaihtunut, näytetään käyttäjälle alert-ikkuna, jossa lukee uusi zoomaustaso (kuvio 9). MVC-olion tilamuutoksien tarkastelussa täytyy huomata se, että yhtäkään tapahtumankuuntelijaa ei voida liittää tapahtuman alkamiskohtaan vaan tapahtumankuuntelijat suoritetaan vasta kun tapahtuma on tapahtunut. Tämä tarkoittaa esimerkiksi sitä, että jos karttaa zoomataan, niin sellaista kuuntelijaa ei voida luoda, mikä kertoisi zoomaustason ennen zoomauksen aloittamista. Jos vanha zoomaustaso halutaan tietää, on se pitänyt ennalta tallentaa käyttäjän luomaan muuttujaan.



KUVIO 9. zoom_changed-tapahtuma

Vaikka Maps API:lla on omat tapahtumat, voidaan sillä myös kuunnella perinteisiä DOM-tapahtumia. Tämä tapahtuu addDomListener()-metodilla, joka löytyy myös google.maps.event-nimiavaruudesta. Metodi ottaa sisäänsä kolme parametria kuten addListener()-metodikin. Ensimmäinen parametri on tarkkailtava DOM-objekti, esim. window tai document.getElementById(”elementti”). Toisena parametrina on tarkkailtava tapahtuma, ja kolmas on funktio, joka tapahtuman tapahduttua suoritetaan. (Google Maps API Events 2011.)

Esimerkiksi perinteinen JavaScriptin tapa tarkastella, että sivu on kokonaan ladattu (kuvio 10) voidaan korvata Maps API:n addDomListenerillä (kuvio 11). Nämä kaksi scriptiä tekevät täysin saman asian.

```
<script type="text/javascript">
function initialize()
{
  // Kartan luonti
}
</script>
</head>
<body onload="initialize()">
  <div id="map_canvas"></div>
</body>
```

KUVIO 10. Perinteinen tapa sivun latauksen tarkasteluun

```
<script type="text/javascript">
function initialize()
{
  // Kartan luonti
}
google.maps.event.addDomListener(window, 'load', initialize);
</script>
</head>
<body>
  <div id="map_canvas"></div>
</body>
```

KUVIO 11. Maps API:n tapa tarkastella sivun latausta

3.5 Control-luokat

Kontrollit ovat kartan päälle tulevia elementtejä, joiden avulla käyttäjä voi kontrolloida karttaa. API:n avulla voidaan päättää, mitä kontrolleja kartalla näytetään vai näytetäänkö niitä ollenkaan. Jos kartalle ei määritetä minkäänlaisia kontrolleja, niin Maps API hoitaa kontrollien näyttämisen. (Google Maps API Controls 2011.)

API:ssa on suuri määrä erilaisia kontrolleja käytettäväksi:

- Zoom-kontrolli, jolla voidaan vaihtaa kartan zoomaustasoa
- Pan-kontrolli, jolla karttaa voidaan käännellä tietyissä tiloissa
- Scale-kontrolli, jolla näytetään kartan mittakaava
- MapType-kontrolli, jolla voidaan vaihtaa kartan tyyppiä
- StreetView-kontrolli, jolla voidaan näyttää katunäkymä
- Overview-konrolli, joka näyttää tämänhetkisen sijainnin suuremmalla alueella

Näitä kontrolleja ei käsitellä suoraan itsenäisesti, vaan ne liitetään kartta-olion luontivaiheessa annettavaan olioliteraaliin. Tällä voidaan vaihtaa kontrollien näkyvyyttä ja sijaintia kartan päällä. Näkyvyyttä voidaan myös vaihtaa jälkeenpäin kartan setOptions()-metodilla. Vakiona kaikkia kontrolleja ei näytetä ja niiden näkyvyyteen vaikuttaa myös se, käytetäänkö karttaa esimerkiksi tietokoneella vai

kännykällä. Jos käyttäjä haluaa piilottaa kaikki vakiokontrollit, täytyy hänen asettaa kartta-olion olioliteraalissa parametri disableDefaultUI todeksi. Tarkempi lista kontrolleista ja niiden vakionäkyvyyksistä löytyy Google Maps JavaScript API:n manuaalista. (Google Maps API Controls 2011.)

Yksittäisiä kontrolleja voidaan myös muokata. Jokaisella kontrollilla on oma kontrolliControlOptions-asetus, johon halutut muutokset voidaan määrittää. Jokaiselle kontrollilla voidaan asettaa position asetus, joka määrittää kontrollille paikan, jossa se kartan päällä näytetään. Joillakin kontrolleilla on myös eri tyylimääreitä, jotka muuttavat kontrollin ulkoasua esimerkiksi pienemmäksi tai suuremmaksi. Tämä tapahtuu style-määreellä.

Käyttäjä voi myös luoda omia kontrolleja kartalle. Käytännössä kontrollit ovat vain div-elementtejä, joilla on kiinteä sijainti kartan päällä. Jotta käyttäjä voi tehdä oman kontrollin, on hänen ensin luotava div-elementti, annettava sille CSS-tyylimääreet, määritettävä elementille tapahtumankuuntelija ja asetettava div-elementti kartan päälle. (Google Maps API Controls 2011.)

```
<script type="text/javascript">
var map;
var lamk = new google.maps.LatLng(60.985, 25.642);

function MyControl(controlDiv, map)
{
  var controlUI = document.createElement('DIV');
  controlUI.style.backgroundColor = '#fff';
  controlUI.style.padding = '5px';
  controlUI.style.border = '1px solid #000';
  controlUI.style.cursor = 'pointer';
  controlUI.style.textAlign = 'center';
  controlDiv.appendChild(controlUI);

  var controlText = document.createElement('DIV');
  controlText.style.fontFamily = 'Arial';
  controlText.style.fontSize = '12px';
  controlText.innerHTML = 'LAMK';
  controlUI.appendChild(controlText);

  google.maps.event.addDomListener(controlUI, 'click', function() {
    map.setCenter(lamk)
  });
}

function initialize()
{
  var myOptions = {
    zoom: 16,
    center: lamk,
    mapTypeId: google.maps.MapTypeId.ROADMAP,
    disableDefaultUI: true,
    zoomControl: true,
    zoomControlOptions: {
      position: google.maps.ControlPosition.TOP_RIGHT,
      style: google.maps.ZoomControlStyle.SMALL
    },
    scaleControl: true,
    scaleControlOptions: {
      position: google.maps.ControlPosition.BOTTOM_LEFT
    }
  };
  map = new google.maps.Map(document.getElementById("map_canvas"), myOptions);
  var myControlDiv = document.createElement('DIV');
  var myControl = new MyControl(myControlDiv, map);

  map.controls[google.maps.ControlPosition.TOP_LEFT].push(myControlDiv);
}
</script>
```

KUVIO 12. Kustomoidun kartan luonti

Kuviossa 12 luodaan kartta-olio kustomoiduilla kontrolleilla ja yhdellä omalla kontrollilla. Ensimmäisenä piilotetaan kaikki kontrollit (rivi 32). Sen jälkeen asetetaan zoomaus-kontrolli näkyväksi (rivi 33), määritetään sen positio kartan oikeaan yläreunaan (rivi 35) ja annetaan vielä style-määreeksi small, joka muuttaa kontrollin ulkoasua (rivi 36). Lisätään kartalle myös toinen kontrolli, joka kertoo kartan mittakaavan (rivi 38). Tälle kontrollille ei määritetä muuta kuin positio

kartan vasempaan alakulmaan (rivi 40) ja style-määre jätetään antamatta. Jos styleä ei anneta, käyttää Maps sen itse valitsemaansa styleä. Kontrolleja lisätessä tulee huomata se, että jos alussa ei piiloteta kaikkia kontrolleja, niin kartalla saattaa näkyä ylimääräisiä kontrolleja, koska niiden näyttäminen on tässä tapauksessa jätetty Mapsin vastuulle.

Itse tehdyn kontrollin lisäys alkaa kuviossa 12 riviltä 44. Luodaan ensin div-elementti muuttujaan. Tämän jälkeen annetaan tämä div-elementti ja kartta-muuttuja itse tehdylle MyControl-funktiolle (rivi 45). MyControl-funktiossa luodaan ensin div-elementti, annetaan sille tyylimääreet ja kiinnitetään se rivillä 44 luotuun diviin (rivit 7-13). Seuraavaksi luodaan uusi div-elementti, joka pitää sisällään tekstin LAMK. Tämä liitetään edellä luotuun div-elementtiin (rivit 15-19). Lopuksi kontrolliin liitetään vielä tapahtumankuuntelija (rivit 21-23). Nyt kontrolli on valmis liitettäväksi kartalle. Se tapahtuu lisäämällä luotu div kartan controls taulukkomuuttujaan (rivi 47). Nyt kartalla näkyy kolme kontrollia zoomaus-kontrolli, mittakaava-kontrolli ja itse luotu kontrolli (kuvio 13). Itse luotua kontrollia klikkaamalla kartta keskittyy LAMK:iin.

KUVIO 13. Kartta kustomoiduilla kontrolleilla

3.6 Overlay-luokat

Overlayt ovat kartalla olevia objekteja, jotka ovat sidottuna tiettyyn latitudi/longitudi-koordinaattiin. Toisin kuin kontrollit, joilla on kiinteä positio karttaelementissä, ovarlayt liikkuvat kartan mukana karttaa liikutettaessa. (Google Maps API Overlays 2011.)

Kuten myös muihin API:n olioihin, myös overlay-olioihin voi liittää tapahtumankäsittelijöitä. Tämä tapahtuu addListener()-metodilla, mutta ensimmäiseksi parametriksi annetaan overlay-olio. Overlay-olion lisääminen kartalle tapahtuu joko antamalla sille luontivaiheessa map-asetukseen kartta-olio tai käyttämällä setMap()-metodia. setMap()-metodia käyttäessä annetaan metodille parametrinä kartta-olio, esim. setMap(map). Overlay-olion poisto kartalta tapahtuu antamalla setMap()-metodissa kartaksi null.

Maps API:ssa on tarjolla useita erilaisia overlay-luokkia. Yhden pisteen esittämiseen kartalla on tarjolla Marker-olio. Marker on nuoli kartalle, joka osoittaa pistettä, johon se on asetettu. Markerin ulkoasua voidaan muuttaa asettamalla sille ikoniksi jokin kuva.

```
1  <script type="text/javascript">
2  function initialize()
3  {
4    var lamk = new google.maps.LatLng(60.9865, 25.643);
5    var lamk2 = new google.maps.LatLng(60.9865, 25.644);
6    var myOptions = {
7      zoom: 16,
8      center: lamk,
9      mapTypeId: google.maps.MapTypeId.ROADMAP,
10   };
11   var map = new google.maps.Map(document.getElementById("map_canvas"),
12       myOptions);
13
14   var marker = new google.maps.Marker({
15       position: lamk,
16       map: map
17   });
18   var marker2 = new google.maps.Marker({
19       position: lamk2,
20       icon: "house.png"
21   });
22   marker2.setMap(map);
23   google.maps.event.addListener(marker2, 'click', function(event) {
24     alert("LAMK");
25   });
26 }
27 </script>
```

KUVIO 14. Markkereiden lisääminen kartalle

Kuviossa 14 luodaan kaksi markkeria, joista toiselle asetetaan vakiosta poikkeava ikoni ja lisätään siihen tapahtumankuuntelija. Riveillä 14 - 17 luodaan perinteinen markker. Sille annetaan luontivaiheessa kartta (rivi 16), joten se lisäytyy kartalle näkyviin samantien. Riveillä 18 - 21 luodaan markkeri, jolle asetetaan ikoni (rivi 20). Markerille ei anneta karttaa luontivaiheessa vaan se määritetään vasta rivillä 22 setMap()-metodilla. Riveillä 23 - 25 jäkimmäiseen markkeriin lisätään vielä click-tapahtumankuuntelija, joka luo alert-ikkunan kun markeria klikataan. Lopputulos on nähtävissä kuviossa 15.

KUVIO 15. Kuvion 14 luoma kartta

Kahden tai useamman pisteen välistä reittiä kuvataan murtoviivalla. Murtoviivalle annetaan kaksi tai useampi piste, joiden väliin kartalle piirtyy viiva (kuvio 16). Viivan väriä, paksuutta ja läpikuultavuutta voidaan vaihtaa viivan omilla asetuksilla. Kuvion 16 koodin luoma viiva on nähtävissä kuviossa 17.

Viivan pisteet tallentuvat Maps API:n MVCArray-taulukkoon. Taulukon käsittelyyn on tehty metodit getAt(), insertAt() ja removeAt(). Taulukosta ei voi hakea alkioita perinteiseen tapaan mvcArray[i], vaan on käytettävät sen omaa metodia mvcArray.getAt(i). (Google Maps API Overlays 2011.)

```
<script type="text/javascript">
  var polyPath = [
    new google.maps.LatLng(60.9865, 25.643),
    new google.maps.LatLng(60.9865, 25.644)
  ];

  var polyline = new google.maps.Polyline({
    path: polyPath,
    map: map,
    strokeColor: "#FF0000",
    strokeOpacity: 1.0,
    strokeWeight: 2
  });
</script>
```

KUVIO 16. Kahden pisteen murtoviiva

KUVIO 17. Kuvion 16 koodin luoma viiva kartalla

Alueita kartalla voidaan kuvata polygonilla. Murtoviivan tapaan myös polygonin pisteet tallentuvat MVCArray-taulukkoon. Myös polygonin ulkonäköä voidaan muokata sen omilla asetuksilla. Koska polygon-olio luo suljetun alueen, vain kolmen koordinaatin määrittäminen riittää luomaan kolmion (kuvio 18). Käyttäjän ei siis tarvitse määrittää viimeiseksi pisteeksi ensimmäistä pistettä vaan Maps API hoitaa tämän. Kuvion 18 koodin luoma polygoni on nähtävissä kuviossa 19. Alueita voidaan kuvata myös ympyrällä tai suorakulmiolla. Ympyrälle annetaan

keskipiste ja sen säteen pituus ja suorakulmiolle annetaan sen kahden kulman koordinaatit.

```
<script type="text/javascript">
  var polyPath = [
    new google.maps.LatLng(60.9865, 25.643),
    new google.maps.LatLng(60.9865, 25.644),
    new google.maps.LatLng(60.9862, 25.6435),
  ];

  var polygon = new google.maps.Polygon({
    path: polyPath,
    map: map,
    strokeColor: "#FF0000",
    fillColor: "#0000FF",
    fillOpacity: 0.5,
    strokeOpacity: 1.0,
    strokeWeight: 2
  });
</script>
```

KUVIO 18. Kolmen pisteen polygoni

KUVIO 19. Kuvion 18 koodin luoma polygoni

Vaikka polygonilla on kätevä kuvata jotakin aluetta, niin se ei pysty sisältämään kuvaa. Jos kartan päälle halutaan tuoda kuva, niin täytyy käyttää GroundOverlay-oliota. Sen konstruktorissa annetaan kuvan URL-osoite sekä kaksi pistettä, johon kuvan kulmat tulevat. (Google Maps API Overlays 2011.)

Informaation näyttämiseen kartalla Maps API:ssa on tarjolla InfoWindow-luokka. Se avaa kartalle puhekuplan, jonka sisälle voidaan laittaa tietoa. Sen tärkeimmät asetukset ovat content ja position (kuvio 20). Contentiin tulee sisältö, joka puhekuplassa näytetään. Sisältö voi olla joko tekstiä ja jokin DOM-elementti. Positioniin laitetaan kohta, josta puhekupla aukeaa. Kuvion 20 koodin luoma InfoWindow on nähtävissä kuviossa 21. Position voi olla joko LatLng-objekti tai Marker-objekti, jolloin puhekupla aukeaa markerin kohdalle. Jos puhekuplan sijaintia halutaan myöhemmin vaihtaa, voidaan se asettaa toiseen kohtaan InfoWindow.setPosition(latLng)-metodilla tai avaamalla puhekupla uudelleen jonkin markerin päälle InfoWindow.open(marker)-metodilla. (Google Maps API Overlays 2011.)

```
<script type="text/javascript">
  var content = '<div>LAMK</div>';
  var infowindow = new google.maps.InfoWindow({
    content: content,
    position: lamk
  });
  infowindow.open(map);
</script>
```

KUVIO 20. Yksinkertainen InfoWindow

KUVIO 21. Kuvion 20 koodin luoma InfoWindow

Käyttäjä voi myös luoda kokonaan omia overlay-luokkia. Oman overlay-luokan hyöty on siinä, että sen avulla kartan päälle voidaan liittää mitä HTML-elementtejä vain, samoin kuin kustomoidussa kontrollissa.

4 TONTTIPÖRSSI

Tonttipörssi-projekti toteutettiin osana KOTI-vetovoimaa Päijät-Hämeen maaseutukuntiin -projektia. Tarkoituksena oli tehdä verkkosivusto, jolla päijäthämäläiset kunnat voivat markkinoida omia tonttejaan uudella tavalla. Monet aiemmat kuntien tonttipörssit olivat sekavia ja toisistaan poikkeavia, ja näihin haluttiin toteuttaa palvelu, joka olisi yhtenäinen kaikilla kunnilla. Palveluun ei ole tarkoitus kerätä kaikkien kuntien tontteja yhteen tonttipörssiin, vaan jokaisella kunnalla on oma palvelu ylläpidettävänään.

Pienemmissä kunnissa on yleistä, että tonttipörssiä ylläpidetään täysin vapaaehtoistyön varassa. Koska kunnan tonttipörssin ylläpitäjällä ei välttämättä ole taitoa luoda kokonaista tonttipörssi-palvelua, oli projektissa luodun tonttipörssin ylläpito oltava helppoa käyttäjälleen.

4.1 Asiakasvaatimukset

Asiakkaan toiveena oli saada kuntien käyttöön uudenlainen tonttipörssi-sivusto. Sivuston ulkoasun täytyi olla selkeämpi kuin vanhojen tonttipörssien ja sivuston käyttö tuli olla helppoa. Käyttäjien piti pystyä selaamaan tontteja sekä sivustolla tuli olla sisältötekstejä, jotka kertoivat alueesta, jolta tontteja kaupataan. Myös yhteystiedot oli sivustolta löydyttävä.

Sivuston hallinta tuli olla myös helppoa, koska sivustoa saattaa ylläpitää käyttäjä, jolla ei aiempaa kokemusta sivuston ylläpidosta ole. Sivuston keskeisempien elementtien sisältötekstien ja tonttien tietojen tuli olla helposti muokattavissa. Koska palvelu oli mahdollisesti tulossa käyttöön useaan kuntaan, myös sivuston yleistä ulkoasua piti pystyä muokkaamaan.

4.2 Suunnittelu

Asiakkaan hyvin avonaisten tavoitteiden takia, kehitykseen annettiin todella vapaat kädet. Suunnittelun pohjana käytettiin jo olemassa olevia tonttipörssi-sivustoja ja niistä katsottiin, mitä tonttipörssin tulee sisältää.

Tonttipörssin ydin on tietenkin itse tontit. Tonteista mietittiin, mitä tietoja itse tonteista tulee tallettaa ja miten eri tontit asiakkaalle sivustolla näytetään. Kuntien aiemmista staattisista tonttipörssi-sivustoista poiketen sivustolle haluttiin hieman eloa ja interaktiivisuutta. Koska tontit on helpoin tapa esittää kartalla, niin sivustolle oli sellainen saatava. Sivuston kartaksi valittiin Google Maps.

Sivustolle päätettiin myös lisätä tonttien listaus. Listaa piti pystyä järjestelemään halutuilla arvoilla ja myös rajoittamaan listalla näkyviä tontteja. Koska sivuston keskeisimpänä elementtinä haluttiin säilyttää kartta, niin rajoitukset listauksessa haluttiin myös esittää kartalla.

Sivustoa piti myös pystyä hallitsemaan helposti. Tämän takia sivustolle päätettiin tehdä myös hallintapaneeli. Paneelin kautta piti pystyä lisäämään, muokkaamaan ja poistamaan tontteja. Myös sivustolla olevia erinäisten sisältötekstien sisältöä piti päästä muokkaamaan. Koska sivusta haluttiin tehdä vähän eläväisempi, myös sivuston taustakuvaa ja pääväriä piti pystyä vaihtamaan hallintapaneelista.

Koska sivustoa oli rakentamassa yhtä aikaa kaksi eri ihmistä, ohjelmoija ja visualisti, piti työjärjestelyt suunnitella niin, että toisen ei tarvinnut odottaa, kun toinen teki muutoksia sivulle. Tämä onnistui suhteellisen helposti jakamalla sivusto osiin. JavaScriptit omiin tiedostoihinsa ja css-tyylimääritteet omiinsa. Tämä ei aiheuttanut lisätoimenpiteitä suunnittelu vaiheessa, koska sivuston jakaminen osiin on hyvin tyypillistä.

4.3 Tietokannat

Sivusto käyttää hyväkseen kolmea eri tietokantaa: tontit, karttakuvat ja users. Tontit-tietokantaan on tallennettu kaikki tiedot tonteista, karttakuvat-tietokannassa on kartalla näkyvät kuvat sekä users-tietokannassa on käyttäjät, jotka voivat kirjautua hallintasivulle.

Tontit-tietokannassa on kahdeksan kenttää (kuvio 22). TonttiID-kenttä on jokaisen tontin id, jota käytetään ainostaan tonttien tunnistamaan toisistaan ja tätä käytetään hyödyksi sivuston koodeissa ja sitä ei ole mahdollista ylläpitäjän muuttaa. TonttiID-kentän kooksi on annettu 7 merkkiä pitkä kokonaisluku, mikä riittää palvelun tonteille. Tunnus-kenttä on tontin tunnus, jonka ylläpitäjä on tontille syöttänyt. Tunnus voi olla tekstiä tai numeroita, joten sen tyypiksi on annettu varchar(15). Muoto-kenttään tallentuu kartalla näkyvä tonttia kuvaavan polygonin tiedot. Muoto on koodattu käyttämällä geometry-kirjastosta löytyvää koodaus -menetelmää. Tällä voidaan lyhentää polygonin pisteiden koordinaattien viemää tilaa. Esimerkki koodatusta muodosta löytyy kuviosta 25 riviltä 5. Osoite-kenttään tallennetaan tontin osoite, hinta-kenttään tontin hinta, pintaAla-kenttään tontin pinta-ala, rakennusoikeus-kenttään tontin rakennusoikeus ja info-kenttään ylläpitäjän syöttämä mahdollinen lisäinfo tontille.

```
+----------------+--------------+------+-----+---------+----------------+
| Field          | Type         | Null | Key | Default | Extra          |
+----------------+--------------+------+-----+---------+----------------+
| tonttiID       | int(7)       | NO   | PRI | NULL    | auto_increment |
| tunnus         | varchar(15)  | NO   |     |         |                |
| muoto          | text         | NO   |     |         |                |
| osoite         | varchar(30)  | YES  |     | NULL    |                |
| hinta          | decimal(8,2) | YES  |     | NULL    |                |
| pintaAla       | int(7)       | YES  |     | NULL    |                |
| rakennusoikeus | int(5)       | YES  |     | NULL    |                |
| info           | text         | YES  |     | NULL    |                |
+----------------+--------------+------+-----+---------+----------------+
```

KUVIO 22. Tontit-tietokanta

Karttakuvat-tietokantaan tallentuut kartalla näytettävät kuvat. Siinä on viisi kenttää (kuvio 23). Id-kenttää käytetään tunnistamaan eri kuvat toisistaan

kooditasolla, eikä sitä ylläpitäjällä ole mahdollisuutta muuttaa. Url-kenttä pitää sisällään url-osoitteen kyseiseen kuvaan. Lat- ja lng-kentät pitävät sisällään kuvan sijainnin kartalla. Zoom-kenttään tallentuu se zoomaustaso, jolla kuva on kartalle lisätty.

```
+-------+----------------+------+-----+---------+----------------+
| Field | Type           | Null | Key | Default | Extra          |
+-------+----------------+------+-----+---------+----------------+
| id    | int(10)        | NO   | PRI | NULL    | auto_increment |
| url   | text           | NO   |     |         |                |
| lat   | decimal(18,15) | NO   |     |         |                |
| lng   | decimal(18,15) | NO   |     |         |                |
| zoom  | int(2)         | NO   |     |         |                |
+-------+----------------+------+-----+---------+----------------+
```

KUVIO 23. Karttakuvat-tietokanta

Users-tietokanta pitää sisällään käyttäjät, jotka voivat kirjautua hallintasivulle (kuvio 24). Tietokannassa on jokaista riviä kuvaava id-kenttä, käyttäjän nimi ja MD5-salauksella tallennettu käyttäjän salasana.

```
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| id       | int(4)      | NO   | PRI |         |       |
| name     | varchar(15) | NO   |     |         |       |
| password | char(32)    | NO   |     |         |       |
+----------+-------------+------+-----+---------+-------+
```

KUVIO 24. Users-tietokanta

4.4 Dynaaminen tiedonsiirto palvelimen ja sivuston välillä

Dynaaminen tiedonsiirto palvelimen ja sivuston välillä on toteutettu PHP:n ja ajaxin avulla. Ajaxia käytetään, kun sivustolle haetaan tiedot tonteista. Sivuston latauduttua lähetetään ajax-kutsu palvelimen PHP-tiedostolle. PHP-skripti hakee kaikki tietokannasta löytyvät tontit ja tulostaa niistä XML-tiedoston. XML-tiedostossa on jokainen tontti eroteltuna toisistaan (kuvio 25).

```
<zones>
  <zone>
    <tonttiID>21</tonttiID>
    <tunnus>57-1</tunnus>
    <muoto>ivjrJsfuxCf@dC~CcDAEAEAE?EAIAG?I?GAG?KAG?ECGAECGAGCGCGAC</muoto>
    <osoite/>
    <hinta>45340.60</hinta>
    <pintaAla>3309</pintaAla>
    <rakennusOikeus>496</rakennusOikeus>
    <neliohinta>13.70</neliohinta>
    <info><p>Varattu</p></info>
  </zone>
</zones>
```

KUVIO 25. Yhden tontin tiedot XML-tiedostossa

4.5 Sivuston ulkoasu ja käyttö

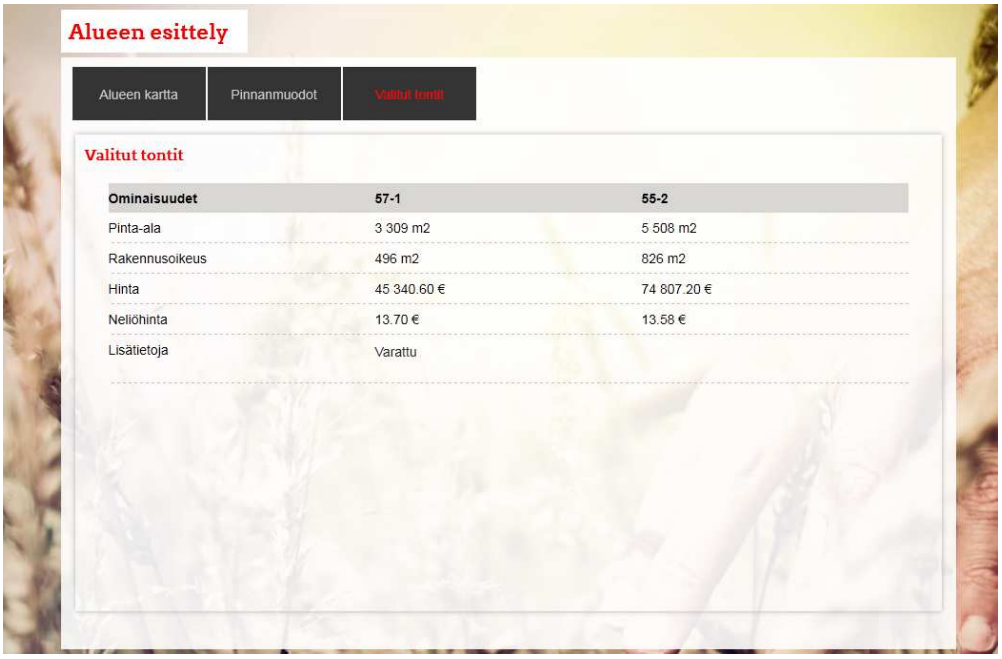
Sivuston ulkoasusta vastasi Lahden ammattikorkeakoulun mediatekniikan opiskelija, joten tässä opinnäytetyössä ei ulkoasu ja sen toteutusta käydä tarkemmin läpi. Sivuston aivan ylälaidassa on kunnan logo sekä kunnan nimi. Nämä ovat vaihdettavissa hallintasivulta. Lisäksi sivun alussa on tietoa alueesta- ja yhteystiedot-kentät (kuvio 26). Näiden kummankin sisältöä voidaan myös vaihtaa hallintasivulta.

KUVIO 26. Sivuston sisältötekstit

Sivuston latautumisen jälkeen kaikki kommunikointi serverin kanssa on toteutettu ajaxilla, joten sivua ei missään vaiheessa ladata kokonaan uudelleen. Kun sivu latautuu ensimmäisen kerran, haetaan sivustolle sisältötekstit erillisistä HTML-tiedostoista ja tontit kartalle sekä tonttien listaukseen PHP-skriptin avulla tietokannasta.

Sisältötekstien alla sivustolla on alue tonttien esittelylle (kuvio 27). Sivustolle tultaessa tässä kohtaa käyttäjälle näkyy kartta, jolta tontteja voidaan selata ja valita tarkempaa tarkastelua varten. Välilehdistä kartan yläpuolella voidaan kartan lisäksi valita pinnanmuodot tai listata valitut tontit. Pinnanmuodoissa käyttäjä näkee alueen pinnanmuotoja kuvaavat 3d-mallinnukset ja valituissa tonteissa käyttäjä voi vertailla kartalta valittuja tontteja (kuvio 28).

KUVIO 27. Kartta, josta valittuna kaksi tonttia

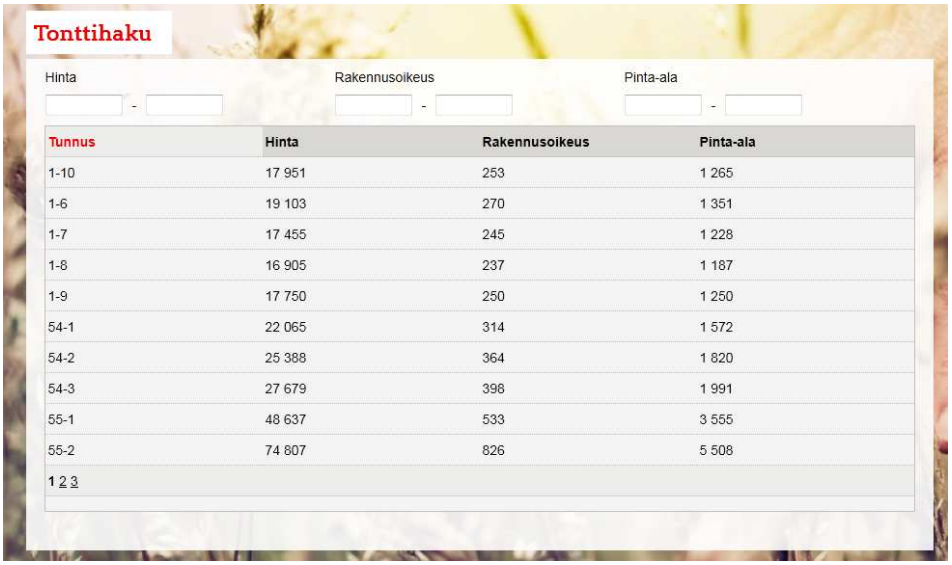


KUVIO 28. Valitut tontit -välilehti

Kartan tontit on toteutettu Maps API:n polygoneilla. Tontteja varten on luotu oma olio, joka pitää sisällään tontin muodon, sen sisältötiedot ja tontin tilan (aktiivinen, passiivinen, piilotettu). Tontit ovat normaalisti passiivisia ja ne näkyvät kartalla vihreällä. Jos hiiri viedään tontin päälle, näytetään käyttäjälle puhekupla, jossa näkyy tontin hinta ja pinta-ala. Tontti voidaan valita aktiiviseksi klikkaamalla sitä, minkä jälkeen tontti näkyy kartalla punaisella. Samalla kun tontti valitaan aktiiviseksi, lisätään se myös valittuihin tontteihin tonttien vertailua varten. Valittuja tontteja voi olla yhtä aikaa valittuna kaksi. Jos käyttäjä valitsee kolmannen tontin, ensimmäisenä valittu tontti poistuu aktiivinen-tilasta. Piilotettu tontti näkyy kartalla harmaalla, eikä sitä voida valita tonttien tarkasteluun. Hiiren ollessa piilotetun tontin päällä näytetään käyttäjälle kuitenkin puhekupla tontista. Kartalla oleva kuva on toteutettu tekemällä oma overlay-luokka.

Viimeisenä osiona sivulla näkyy tonttien listaus (kuvio 29). Listan yläpuolella on kentät näytettävien tonttien rajoittamiseen hinnan, rakennusoikeuden ja pinta-alan mukaan. Hakutuloksen ulosjäävien tonttien tila muutetaan kartalla piilotetuiksi. Tontit järjestyvät listalla normaalisti tunnuksen mukaan aakkosjärjestyksessä.

Järjestystapaa voidaan muuttaa klikkaamalla listan otsikoita. Esimerkiksi klikkaamalla Hinta-otsikkoa, tontit listautuvat hinnan mukaan nousevasti. Jos samaa otsikkoa klikataan toisen kerran, vaihdetaan järjestys laskevaksi. Listalta voidaan myös klikata tontteja. Jos käyttäjä klikkaa jotakin tonttia listalta, niin sivuston kartta keskittyy listalta klikattuun tonttiin. Käyttäjälle näytetään myös samalla tontin infokupla.



Tunnus	Hinta	Rakennusoikeus	Pinta-ala
1-10	17 951	253	1 265
1-6	19 103	270	1 351
1-7	17 455	245	1 228
1-8	16 905	237	1 187
1-9	17 750	250	1 250
54-1	22 065	314	1 572
54-2	25 388	364	1 820
54-3	27 679	398	1 991
55-1	48 637	533	3 555
55-2	74 807	826	5 508

1 2 3

KUVIO 29. Tonttihaku

4.6 Hallintapaneeli

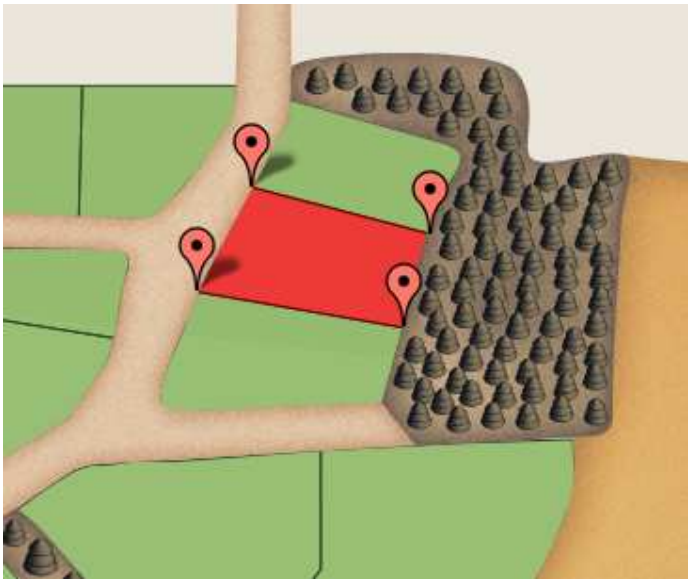
Hallintapaneelin kautta sivuston ylläpitäjä voi hallita sivuston karttaa, sisältötekstejä, vaihtaa kunnan logon, kunnan nimen ja taustakuvan. Kartan muokkauksessa voidaan lisätä ja poistaa tontteja, muokata kartalla näkyviä kuvia sekä vaihtaa sitä koordinaattia, johon kartta sivustolle tultaessa keskittyy. Hallintapaneeliin kirjaudutaan käyttäjätunnuksella, jonka on löydyttävä users tietokannasta.

4.6.1 Tonttien lisääminen ja muokkaus

Tonttien lisäys kartalle on tehty käyttäjälle mahdollisimman helpoksi. Uutta tonttia lisätessä kartalle käyttäjän pitää vain klikkailla hiiren kakkosnäppäimellä tontin rajat ja tontista piirtyy automaattisesti tonttia kuvaava polygoni. Koska polygonilla halutaan kuvata aluetta, tontilla pitää olla vähintään kolme pistettä. Kun käyttäjä on lisännyt tontille kolmannen pisteen, järjestelmä lähettää ajaxin avulla palvelimelle tiedon uudesta tontista.

Palvelimelle lähtevä tieto sisältää ainoastaan luodun tontin pisteiden koordinaatit, jotka on koodattu Maps API:n geometry-kirjaston avulla. Paluuarvona palvelin antaa selaimelle juuri lisätyn tontin ID:n. Nyt käyttäjän selain tietää, mikä käsiteltävän tontin ID on. Jos käyttäjä lisää tontille uuden pisteen, lähetetään palvelimelle tontin ID ja tontin pisteiden koordinaatit. ID täytyy lähettää tässä vaiheessa, koska nyt ollaan muokkaamassa jo tietokannassa olevaa tonttia.

Käyttäjä voi valita tontin klikkaamalla sen polygonia kartalta. Aktiivinen tontti näkyy punaisella. Tontin eri pisteitä kuvataan markereilla (kuvio 30). Markkereiden sijaintia voidaan muuttaa drag & drop -menetelmällä. Käyttäjän vaihtaessa markkeria paikasta toiseen päivittyy tontin muoto tietokantaan automaattisesti. Markerin poistaminen onnistuu klikkaamalla markeria hiiren kakkosnapilla. Tässäkin tapauksessa muuttunut muoto päivitetään tietokantaan. Jos käyttäjä haluaa lisätä uuden pisteen tontille, tulee hänen klikata karttaa hiiren kakkosnapilla. Uutta pistettä kuvataan uudella markerilla ja uusi piste lisätään polygonin ensimmäisen ja viimeisen pisteen väliin.



KUVIO 30. Tontin muokkaus

Tontin poisto onnistuu joko poistamalla tontin markerit tai erillisellä Poista tontti -napilla. Jos käyttäjä poistaa tontista markereita ja jäljelle jäisi vain kaksi markeria, kysytään käyttäjältä, haluaako hän varmasti poistaa pisteen, koska samalla tontti poistuu tietokannasta. Käyttäjän vastatessa kyllä tontti poistetaan tietokannasta. Jos käyttäjä vastaa ei, ei tonttia eikä haluttua pistettä poisteta kartalta.

Tonteille voidaan myös syöttää sisältötietoja, jotka ovat tunnus, osoite, hinta, pinta-ala, rakennusoikeus ja lisäinfo. Sisältötekstien muokkausta varten hallintasivulla on tekstikentät eri tietojen muokkauksia varten.

4.6.2 Karttakuvat

Karttakuvat ovat sivuston kartalla näkyviä kuvia. Jotta kuva voidaan lisätä kartalle, käyttäjän tulee lähettää kuva palvelimelle. Kuvan lähettäminen tehdään HTML-formin avulla. Käyttäjä klikkaa Selaa-näppäintä, minkä jälkeen aukeaa ikkuna, josta valitaan palvelimelle lähetettävä kuva. Kuvan lähetyksen jälkeen uusi kuva ilmestyy listaan, jolla kaikki palvelimelle lähetetyt kuvat ovat listattuna.

Kuvan lisäys kartalle on tehty pääpiirteittäin samanlaiseksi kuin tonttien lisäys. Käyttäjä klikkaa kuvalistasta sitä kuvaa, jonka hän haluaa kartalle lisätä. Sen jälkeen karttaa klikataan hiiren kakkosnapilla ja kuva lisäytyy kartalle. Kuvan koko on liitetty kartan zoomaustasoon. Jos kartta on zoomattuna hyvin lähelle maata, karttakuvasta tulee pieni. Jos zoomaustaso taas on hyvin kaukana maan pinnasta, kuvasta tulee suuri (kuvio 31). Kuvia suunnitellessa pitää ottaa tämä huomioon. Jos halutaan tarkalleen oikean kokoisia kuvia, tulee kuva suunnitella juuri tietylle zoomaustasolle.

Lisättyjen kuvien keskellä näkyy markeri. Markerista kuvaa voi liikuttaa ja klikkaamalla sitä hiiren kakkosnäppäimellä kyseinen kuva poistuu kartalta. Myös kuvalistan kautta voidaan poistaa kuvia. Kuvalistalla jokaisella kuvatiedostolla on Poista-näppäin, jota klikkaamalla kyseinen kuvatiedosto poistetaan palvelimelta. Samalla kun kuvatiedosto poistetaan, poistetaan myös kartalta kaikki karttakuvat, jotka käyttivät poistettua kuvatiedostoa. Jos näin ei tehtäisi, jäisi karttakuvat-tietokantaan karttakuvia, joilla ei olisi kuvatiedostoa.

KUVIO 31. Sama kuva eri zoomaustasoille lisättynä

4.6.3 Muun sisällön hallinta

Hallintapaneelista pystyy myös vaihtamaan sivuston taustakuvan, kartan keskuksen ja hallita sivuston sisältötekstejä. Sivuston taustakuvaa varten uusi taustakuva pitää lähettää palvelimelle. Hallintapaneelissa on listattuna palvelimelle lähetetyt taustakuvat, joista käyttäjä voi valita sen taustakuvan, jota sivustolla halutaan käyttää.

Kartan keskuksen muuttaminen tapahtuu samalta sivulta kuin tonttien lisäys. Hallintasivulla oleva kartta on samankokoinen kuin pääsivustolla näkyvä kartta, jotta kartan keskittäminen olisi helpompaa. Ylläpitäjän tulee siirtää kartta siihen pisteeseen, mihin hän haluaa kartan sivustolla keskittyvän. Kun kartta on oikeassa paikassa, painetaan Päivitä kartan keskus -näppäintä, minkä jälkeen kartan keskus ja zoomaustaso tallennetaan palvelimelle. Seuraavalla kerralla kun sivustolle tullaan, kartta keskittyy automaattisesti hallintasivulta tallennettuun sijaintiin.

Sivuston sisältötekstien muokkaus tehdään TinyMCE:n avulla. TinyMCE on otettu käyttöön, koska sillä ylläpitäjän on helppo muokata sivuston tekstejä juuri sellaiseksi kuin hän haluaa. Ylläpitäjän ei tarvitse tuntea HTML-kieltä saadakseen tekstiin esimerkiksi otsikoita tai lihavointia. Myös linkkien tekeminen TinyMCE:n avulla on helppoa. Tekstit tallennetaan erillisiin HTML-tiedostoihin, joista ne luetaan varsinaiselle sivulle.

4.7 Testaus

Sivuston testaus tapahtui suurimmalta osin projektin sisäisesti. Aina kun uusi ominaisuus lisättiin sivustolle, ilmoitettiin siitä toiselle ja katsottiin, että ominaisuus toimii oikein. Kun suurimmat bugit oli sivustolta korjattu, levitettiin sivuston URL-osoitetta myös pienelle ryhmälle testattavaksi. Testaajina toimi ohjelmistotekniikan oppilaita. Sivusto oli myös ennen lopullista julkaisua Lahden Raksa-messuilla Kärkölän osastolla käytössä. Messuilta ei uusia bugeja järjestelmästä ilmoitettu.

4.8 Sivuston tulevaisuus

Sivusto ei tällä hetkellä ole käytössä, mutta se on tarkoitus siirtää lopulliseen paikkaansa myöhemmin. Siirto tapahtuu siirtämällä sivuston tiedostot uudelle palvelimelle ja luomalla palvelimelle tarvittavat tietokannat.

Sivuston osoitetta on jaettu eri kuntiin, ja se on herättänyt heti kiinnostusta. Mahdollisia eri asiakkaiden toivomia muutoksia on vaikea arvioida, mutta jos eri asiakkailta tulee eri toivomuksia, uudet ominaisuudet sivuille voitaisiin lisätä jonkinlaisella plugin-menetelmällä. Uudet ominaisuudet voisivat esimerkiksi tulla uusiksi välilehdiksi sivustolle.

Järjestelmää voisi myös kehittää toimimaan yhdellä palvelimella, jolla olisi usean eri kunnan sivuja. Kunnilla voisi jokaisella olla eri tunnus ja tunnuksen perusteella näytettäisiin tietyn kunnan sisältö. Järjestelmää kehitettäessä ei tällaiseen ratkaisuun ole pyritty, mutta koska sivuston keskeisimmät tiedot ovat tietokannoissa, niin keskitettyyn järjestelmään siirtyminen ei olisi mikään mahdottomuus. Tämän tekniikan avulla olisi helpompi ylläpitää useaa sivustoa ja mahdollisten bugien korjaus onnistuisi yhdeltä palvelimelta, eikä tarvitsisi jokaisen kunnan sivustoa muuttaa erikseen.

5 YHTEENVETO

Opinnäytetyön tavoitteena oli luoda tonttipörssi-sivusto. Projektia aloittaessa ei vielä tiennyt, mitä tekniikoita sivustolla tullaan tarvitsemaan. Hyvin alkuvaiheessa mukaan kehitykseen tuli Google Maps JavaScript API. Aiempaa kokemusta API:n käytöstä ei ollut, mutta siihen tutustumisen jälkeen oli selvä, että Maps API soveltui hyvin tonttipörssin pohjaksi.

API:a oli helppo käyttää, ja sen dokumentaatio oli hyvin selvä. Googlen omilta sivuilta löytyi myös suuri määrä esimerkkejä eri API:n toiminnoista. API:sta löytyy myös lukuinen määrä eri toimintoja, joita tässä opinnäytetyössä ei käydä läpi, koska niitä ei myöskään tonttipörssi-järjestelmässä tarvittu. Vaikka API:n käytön on helppoa, tarvitsee se käyttäjältään jonkin verran JavaScript osaamista. Yksinkertainen kartta sivustolle on helppo lisätä, mutta mitä monimutkaisempaa palvelua kartalle rakennetaan, sitä laajemmin eri web-tekniikoita pitää tuntea.

Sivuston ainoaksi ongelmaksi jäi sen raskaus. Tämänhetkisillä uusimmilla selaimilla sivusto toimii sulavasti, mutta sivustoa käytettäessä vanhemmilla selaimilla, hidastelua on selvästi huomattavissa. Tämä johtuu sivustolla olevasta suuresta JavaScriptin määrästä. Erityisesti sivun taustakuva tuo sivustolle raskautta, koska se skaalataan JavaScriptin avulla.

Sivuston piti toimia dynaamisesti, joten mukana kehityksessä oli myös ajax. Ajaxin käyttö sivustolla on tehty jQueryn avulla, joka on hyvin tehokas ja ennen kaikkea nopea tapa luoda dynaamisia sivustoja, jotka toimivat usealla eri selaimella samalla tavalla. jQueryn käyttö oli jo ennestään tuttua, joten sen tutustumiseen ei mennyt aikaa.

Alussa asetettuihin asiakasvaatimuksiin päästiin, mutta koska järjestelmä ei vielä ole varsinaisessa käytössä, voi uusia vaatimuksia tulla myöhemmin. Näihin mahdollisesti tuleviin vaatimuksiin on varauduttu rakentamalla sivu niin, että ominaisuuksia olisi helppo lisätä. Myös hallintapaneelin käyttöä on testattu ainoastaan projektin sisäisesti, joten sen käytöstä ei vielä palautetta ole saatu.

Jos aloittaisin projektin teon uudelleen, tekisin sivuston heti alusta alkaen toimimaan yhdellä keskitetyllä palvelimella. Tällä hetkellä, jos sivusto tulee useaan kuntaan käyttöön ja koodeista löytyy bugi, joudutaan se korjaamaan kaikille kunnille erikseen.

LÄHTEET

Google API v1. 2009 [viitattu 25.3.2011]. Saatavissa:
http://code.google.com/apis/maps/documentation/staticmaps/v1/

Google API v2. 2010 [viitattu 25.3.2011]. Saatavissa:
http://code.google.com/apis/maps/documentation/javascript/v2/

Google API v3. 2011 [viitattu 26.3.2011]. Saatavissa:
http://code.google.com/apis/maps/documentation/javascript/basics.html

Google Blog API v1. 2005 [viitattu 25.3.2011]. Saatavissa:
http://googleblog.blogspot.com/2005/06/world-is-your-javascript-enabled_29.html

Google Blog API v2. 2006 [viitattu 25.3.2011]. Saatavissa:
http://googlemapsapi.blogspot.com/2006/04/google-maps-api-version-2.html

Google JavaScript API. 2011 [viitattu 25.3.2011]. Saatavissa:
http://code.google.com/apis/maps/documentation/javascript/

Google JS Tutorial. 2011 [viitattu 26.3.2011]. Saatavissa:
http://code.google.com/apis/maps/documentation/javascript/tutorial.html

Google Maps. 2011 [viitattu 10.3.2011]. Saatavissa:
http://maps.google.com/support/bin/static.py?hl=en&page=guide.cs&guide=21670&from=21670&rd=1

Google Maps API Controls. 2011 [viitattu 27.3.2011]. Saatavissa:
http://code.google.com/apis/maps/documentation/javascript/controls.html

Google Maps API Events. 2011 [viitattu 27.3.2011]. Saatavissa:
http://code.google.com/apis/maps/documentation/javascript/events.html

Google Maps API Overlays. 2011 [viitattu 27.3.2011]. Saatavissa:
http://code.google.com/apis/maps/documentation/javascript/overlays.html

Google Premier. 2011 [viitattu 25.3.2011]. Saatavissa:
http://www.google.com/enterprise/earthmaps/maps.html

Google Terms of Use. 2011 [viitattu 10.3.2011]. Saatavissa:
http://maps.google.com/support/bin/answer.py?hl=en&answer=29435

Koti-projekti. 2011 [viitattu 10.4.2011]. Saatavissa:
http://www.lamk.fi/koti/esittely/