TEKNIIKKA JA LIIKENNE

Tietotekniikka

Ohjelmistotekniikka

INSINÖÖRITYÖ

WEB-SOVELLUSKEHITYS PYTHON/DJANGO-YMPÄRISTÖSSÄ

Työn tekijä: Konstantin Tulainov
Työn ohjaaja: Kirmo Uusitalo
Työn ohjaaja: Ilpo Kuivanen

Työ hyväksytty: ___. ___. 2011

Ilpo Kuivanen

ALKULAUSE

Tämä insinöörityö tehtiin Keypro Oy:lle. Työharjoittelun aikana yrityksessä tehtiin uuden nettipohjaisen palvelun ja kehityksen aikana tuli paljon uusia mielenkiintoisia asioita. Tässä insinöörityössä kuvataan toteutetun palvelun kehitystekniikoita. Kiitän projektissa mukana olleita.

Vantaalla 04.05.2011

Konstantin Tulainov

TIIVISTELMÄ

Työn tekijä: Konstantin Tulainov

Työn nimi: Web-sovelluskehitys python/django-ympäristössä

Päivämäärä: 04.05.2011 **Sivumäärä:** 26 s. + 1 liite

Koulutusohjelma: Tietotekniikka

Suuntautumisvaihtoehto: Ohjelmistotekniikka

Työn ohjaaja: Ilpo Kuivanen

Työn ohjaaja: Kirmo Uusitalo

Tässä työssä käydään läpi web-sovellusten kehitysprosessi python/django-ympäristössä, verrataan kyseisen tekniikan muiden mahdollisten menetelmien kanssa. Työn pohjana on Keypro Oy:n kehittämä kaivulupa.fi- projekti.

Työssä kerrotaan Pythonin ja Djangon keskeisistä ominaisuuksista sekä toteutetusta sovelluksesta esimerkkikoodein sekä kuvin.

Toteutettu sovellus tarjoa kaivajille palvelun, jonka kautta he pääsevät tarkastelemaan johtotietoja kaivualueella sekä tilata näyttöjä. Palvelu kattaa koko Suomen, mukana ovat myös verkko-omistajat. Sovellusta pääsee käyttämään kuka tahansa, mutta kaikkea hyötyä tuottavat toiminnot vaativat sovellukseen rekisteröitymisen ja jotkut sovelluksen osat ovat maksullisia. Sovelluksessa on myös eritasoisia käyttäjiä kuten ylläpito, operaattori, näyttäjä, kaivaja.

Kun sovellus on käytettävissä internetselaimella, ei käyttäjien ole tarvetta asentaa erillisiä ohjelmia tietokoneellensa. Näin käyttäjä voi helposti kirjautua sivustoon ja vaivattomasti hoitaa työtään.

Avainsanat: python, django, web-sovellus, framework, ohjelmointi

Metropolia

ABSTRACT

Name: Konstantin Tulainov

Title: Web-application development using Python/Django framework

Date: 04.05.2011 **Number of pages:** 26 s. + 1 liite

Department:

Information Technology

Instructor: Ilpo Kuivanen

Instructor: Kirmo Uusitalo

In this work is described web-application development process using Python/Django framework, comparing this technique with other possibilities.

The work is based on developed ‘kaivulupa.fi’ project.
Here is described Phyton and Djangon most important options and possibilities also described developed application by example code and pictures.

Developed application offers to diggers services will help them to check information about underground networks and to order displayer visit. Service works in entire Finland and includes network owners involved. Everyone can use application, but useful things require registration and some parts of the application require payment. The application has also some different luser levels like admins, operators, displayers and diggers.

When application is available in internet browser, the users are not required to
install any programs and users can easily login and do their own job.

Keywords: python, django, web-application, framework

LYHENNELUETTELO

ORM (Object-relational mapping) - ohjelmointitekniikka joka yhdistää tietokannat olio-pohjaisiin objekteihin luomalla virtuaalitietokannan.

MVC (Model-View-Controller) - ohjelmistoarkkitehtuurityyli, jonka tarkoituksena on käyttöliittymän erottaminen sovellusaluetiedosta.

API (Application programming interface) - ohjelmointirajapinta, jonka kautta eri ohjelmat voivat tehdä pyyntöjä ja vaihtaa tietoja eli keskustella keskenään.

URL (Uniform Resource Locator) - on merkkijono, jolla kerrotaan tietyn tiedon paikka.

HTML (Hypertext Markup Language) - on avoimesti standardoitu kuvauskieli, jolla voidaan kuvata hyperlinkkejä sisältävää tekstiä eli hypertekstiä.

DOM (Document Object Model) - ohjelmointirajapinta, joka mahdollistaa HTML-dokumenttien sisällön muokkauksen. JavaScriptin kanssa sillä voidaan toteuttaa vuorovaikutteisia www-sivuja, jotka eivät vaadi jatkuvaa palvelinyhteyttä.

CSS (Cascade Style Sheets) - WWW-dokumenteille kehitetty tyyliohjeiden laji.

AJAX (Asynchronous JavaScript And XML) - on joukko web-sovelluskehityksen tekniikoita, joiden avulla web-sovelluksista voi tehdä vuorovaikutteisempia.

JSON (JavaScript Object Notation) - on yksinkertainen tiedonsiirtomuoto, jota on helppo käyttää JavaScript-ohjelmissa. Nimestään ja JavaScript-perustastaan huolimatta JSON on JavaScriptistä riippumaton, eli sitä voidaan käyttää myös muilla ohjelmointikielillä.

XML (eXtensible Markup Language) - on merkintäkieli tai standardi, jolla tiedon merkitys on kuvattavissa tiedon sekaan.

GPL (General Public License) - on vapaiden ohjelmistojen julkaisemiseen tarkoitettu lisenssi, joka antaa kenelle tahansa oikeuden käyttää, kopioida, muuttaa ja jakaa edelleen ohjelmia ja niiden lähdekoodia.

LDAP (Lightweight Directory Access Protocol) - on hakemistopalvelujen käyttöön tarkoitettu verkkoprotokolla.

CMS (Content Management System) - sivuston sisällönhallintajärjestelmä.

WSGI (Web Server Gateway Interface) - standartiväylä Python:n ohjelman ja Apache:n palvelimen välillä.

CGI (Common Gateway Interface) - on tärkeä Web-ympäristön tekniikka, jonka avulla selain voi välittää dataa palvelimella suoritettavalle ohjelmalle.

WGS84 (World Geodetic System) - on Yhdysvaltain puolustusministeriön määrittelemä ja ylläpitämä koordinaattijärjestelmä ja siihen liittyvä geoidimalli. GPS-järjestelmä perustuu WGS 84 –koordinaattijärjestelmään.

KKJ - kansallinen kartastokoordinaatistojärjestelmä.

ELY-keskus - elinkeino-, liikenne-, ja ympäristökeskus.

SSL (Secure Sockets Layer) - on salausprotokolla, jolla voidaan suojata Internet-sovellusten tietoliikenne IP-verkkojen yli.

SISÄLLYS

1 JOHDANTO

Web-sovellus on aivan kuten mikä tahansa perinteinen tietokoneohjelma, mutta se sijaitsee jossain verkossa ja kaikki käyttäjät pääsevät sitä käyttämään. Yrityksillä on usein pyrkimyksiä muuttaa ohjelmistonsa Internet-pohjaisiksi tai kehittämään ainakin versioita, jotka voi netin kautta käyttää.

Nettipohjaisten sovellusten kehittämiseen on olemassa erilaisia tekniikoita. Yleistä on, että web-sovellus on tavallisessa selaimessa pyörivä ohjelma. Kaikkien teknikoiden pohjana on kuitenkin ”asiakas-palvelin”-periaate. Selain on silloin ”client” ja palvelin on ”server”. Paikallisessa koneessa voi olla myös mikä tahansa muu ohjelma kuin selain. Eli kaksi sovelluksen osaa keskustelee keskenään ja samanaikaisesti näitä pyyntöjä ja vastauksia voi olla yhtäpaljon kuin on käyttäjiä. Palvelin pystyy käsittelemään tulevia pyyntöjä niin monta kuin näitä on, mutta palvelimen suorituskyvyn on oltava silloin hyvällä tasolla. Sovelluksen tekninen rakenne on myös yksi suorituskykyyn vaikuttajista.

Tietoturvallisuuteenkin on kiinnitettävä erityistä huomiota. Tähän on jo valmiina erilaisia tapoja. Osa ratkaisuista tulee palvelinympäristön mukana, mutta kehittäjän on huomioitava, mitkä sovelluksen kohdan on suojattava ja millä tavalla, mitä kaikkia pyyntöjä palvelin saa vastaanottaa ja mitä näihin pitäisi vastata tai vastataanko ollenkaan.

Erilaisia palvelinympäristöjä on iso määrä, ja oikean valinta on tärkeää. Oikeaan asiaan vaikuttavat monet asiat kuten tavoitteet, joihin pyrkii, kaikki mahdolliset kehitysongelmat, kehitysnopeus, ylläpito, kustannukset ja muut sellaiset.

Yritysesittely

Keypro Oy on vuonna 1995 perustettu tietotekniikka-alan yritys. Yrityksessä työskentelee noin 60 järjestelmäasiantuntijaa Joensuun ja Vantaan toimipisteissa. Keypro Oy:n erikoisosaaminen löytyy seuraavista alueista:

- paikka- ja tietojärjestelmien kehitys ja käyttöönotto
- käytössä olevien verkkotietojärjestelmien tehostaminen

- verkkotietoaineistopalvelut.

Yrityksen asiakkaat ovat tele- ja tietoliikenneoperaattorit, vesi- ja sähkölaitokset, kunnat ja kaupungit. Markkina-alueena on koko Suomi.

Yritys käyttää suuremmassa osassa ohjelmistosta pohjana ”python/django” -teknologiaa ja sen lisäksi montaa muuta tekniikkaa palveluiden pyörittämiseen.

Tavoitteet ja menetelmät

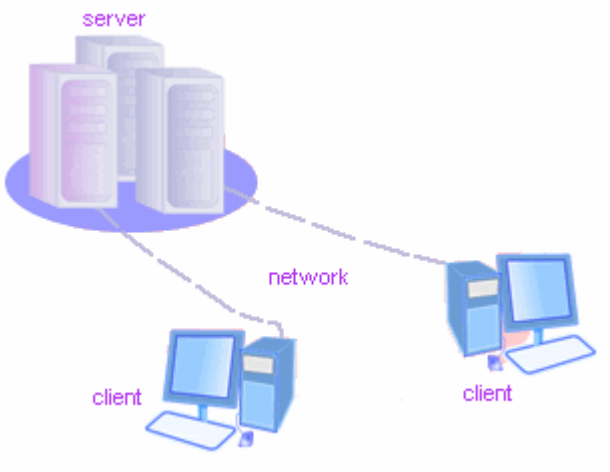
Tämän työn tavoite on kuvata Framework-toimiminnan yleisesti ja Djangon web-sovelluskehitysmenetelmän Keypro Oy:n kehittämässä ”kaivulupa.fi”-palvelussa.

Työn pohjana on toimivan nettipalvelun kehitysprosessi tyhjästä julkaisuversioon asti.

2 WEB-SOVELLUSKEHYKSEN TOIMINTAPERIAATE

2.1 Web-sovelluksen toiminta

Web-sovellus on erikoistapaus hajautetusta tietojärjestelmästä. Web-sovelluksessa käyttöliittymänä toimivat selaimessa näytettävät sivut, jotka tuotetaan jollakin palvelimella ja välitetään tietoverkon kautta selaimelle (kuva 1). Tuotettavat sivut perustuvat yhdeltä tai useammalta palvelimelta saatavaan tietoon ja ne tuotetaan vastauksena selaimen lähettämään aineistopyyntöön. Suuri etu tässä on se, että käyttäjät eivät ole riippuvaisia käyttöjärjestelmästä.



Kuva 1, Client-Server

Suurin etu web-sovelluksen toiminnallisuuden rakenteessa on se, että tämä sovellus päästään suorittamaan selaimen perusominaisuuksilla. Sovellus suunnitellaan vain kerran palvelinympäristöön, ja se on kaikkialla käytettävissä, mutta selaimessa suorittavien osien erilaiset ratkaisut kuten HTML, DOM, CSS voivat jatkossa olla ongelmallisia kehityksessä ja ylläpidossa. Käyttäjän omat selaimen viritykset kuten kirjainkoon, värin muuttaminen, javascripttien tuen poisto voi estää sovelluksen tietyn osan toimintaa.

Toinen tapa rakentaa web-sovellusten käyttöliittymiä on sellaisten tekniikoiden käyttö kuten Java Applet, Adobe Flash tai Silverlight. Suurin osa

selaimista tukee näitä teknologioita ja ne tarjoavat sovelluskehittäjille enemmän mahdollisuuksia käyttöliittymän tekemisessä ja pystyvät kiertämään monia ristiriitaisia selaimen asetuksia. Mutta haittana voi olla silloin että asiakkaan päässä selain ei tue kyseistä tekniikkaa.

2.2 Client-Server-rakenne

2.2.1 Palvelin

Palvelin voi olla moniosainen usean ohjelmiston ja jopa usean laitteen kokonaisuus. Yksinkertaisimmillaan se on perinteinen www-palvelin, esimerkiksi Apache, joka ottaa vastaan aineistopyyntöjä ja osaa toimittaa asiakkaalle palvelimen tiedostojärjestelmästä valmiina löytyviä staattisia sivuja sekä käynnistää ohjelmia tuottamaan dynaamisia, ei valmiina olevia sivuja. Palvelin voiva olla myös monipuolisempi sovelluspalvelin, joka edellisten toimintojen lisäksi osaa hoitaa esimerkiksi istunnonhallintaa, kuormantasausta ja oikeuksien valvontaa sekä yhteyksiä erilaisiin ulkopuolisiin tietolähteisiin kuten tietokantapalvelimiin. Palvelin voi lähettää selaimelle myös jonkun muun dokumentin kuin html-sivun, esimerkiksi PDF-, JSON- tai XML- tiedoston.

Paljon hyötyä saadaan kuormituksen jaosta, jos asiakasmäärä on suuri ja jokainen käyttää omaa selainta tietyn ohjelman osan suorittamisen, palvelin ei silloin kuormitu niin pahasti.

Palvelinpuolen Web-pohjaisissa sovelluksissa voidaan luoda käyttäen erilaisia tekniikoita millä tahansa ohjelmointikielellä, joka voi tulostaa dataa konsoliin. Ohjelmointikieliä voivat olla ASP, ASP.NET , C/C++, Java, Perl, PHP, Python, Ruby.

2.2.2 Asiakas

Asiakasosassa on toteutettu käyttöliittymä. Sen avulla lähetetään palvelimelle pyyntöjä ja myös näytetään vastaanotetut palvelimen aineistot.

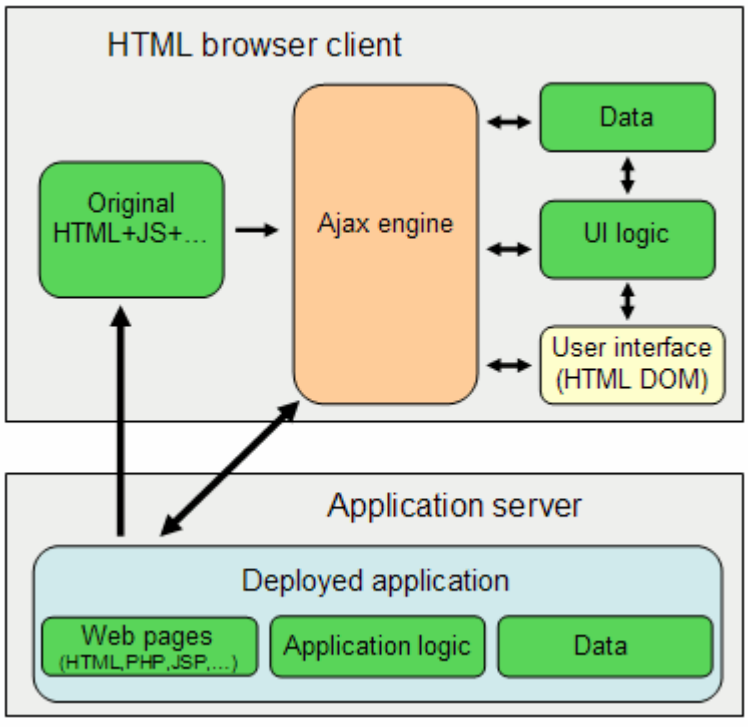
Tällä hetkellä hyvin menestynyt uusi lähestymistapa web-sovellusten kehittämisessä on nimeltään AJAX. Kun käytetään Ajaxia, Web-sovelluksen

sivuja ei ladata kokonaisuudessaan, vaan ainoastaan tarvittavat lisätiedot haetaan palvelimelta ja päivitetään vain osan sivua ilman, että sivu päivittyy kokonaan. Sellainen tapa tekee sivuston enemmän interaktiivisemmaksi, tuottavammaksi ja nopeammaksi.

Client-puolella käytetään yleensä HTML:ää sekä CSS:ää. Kyselyjen käsittelyyn ja selainriipumattomutta varten käytetään: ActiveX:ää, Adobe Flashia, Adobe Flexia, Javaa, JavaScriptiä, ja Silverlightia.

2.2.3 Ajax

Ajax (Asynchronous JavaScript And XML) on joukko web-sovelluskehityksen tekniikoita, joiden avulla web-sovelluksista voi tehdä vuorovaikutteisempia. Alkuperäisessä merkityksessään AJAX:lla on alun perin viitattu tekniikkaan, jossa verkkosivulla JavaScript:llä asynkronisesti tehtävistä HTTP-pyynnöistä palautetaan XML-merkkausta. Nykyisin Ajax-tekniikoilla viitataan yleisesti samankaltaiseen toimintatapaan: Ajaxissa selainohjelma vaihtaa pieniä määriä dataa palvelimen kanssa taustalla niin, ettei koko verkkosivua tarvitse ladata uudelleen joka kerta käyttäjän tehdessä muutoksen. Tekniikan päämääränä on siis lisätä verkkopalvelun vuorovaikutteisuutta, nopeutta ja käytettävyyttä [1].



Kuva 2, Ajax Client-Server- toiminnassa

Ajax koostuu seuraavien tekniikoiden yhdistelmästä (kuva 2):

Palvelin joka lähettää selaimelle HTML-sivun, jossa on HTML-koodia, JS, CSS ja kaikkea muuta, muun muassa itse Ajax-kone(Dojo-, jQuery-, ExtJs-toteutuksella), joka vastaa käyttöliittymän muutoksista ja tiedon päivittämisestä, sekä palvelimelle tiedon lähettämisestä ja palvelimeltä tiedon vastaanottamista. Tiedonvaihtoformatti on yleensä XML, JSON tai jokin muu tavallinen teksti. XML ja JSON mahdollistavat objektien lähettäminen palvelimelta selaimelle ja toisiinpäin.

Ajaxin etuja ovat:

- Tietoliikenteen säästäminen, koska haetaan ja ladatan vain niitä tietoja joita oikeasti tarvitaan tai joita on tarkoitus muuta käyttöliitymässä ilman että sivu päivittyy kokonaan.
- Palvelimen kuormituksen vähentäminen, esimerkiksi sähköpostilaatikon selailun yhteydessä, palvelin tallettaa tiedon siitä tietokantaan ja lähettää takaisin selaimelle tiedon hyväksytysti suoritetusta operaatiosta, eli toista kertaa koko sivua ei tehdä eikä lähetetä palvelimelle.
- Käyttöliitymän nopeuttaminen, koska sivulla päivitetään vain muuttunut tieto, käyttäjä näkee tuloksen nopeammin.

Löytyy myös puutteitakin:

- Selaimen perustyökalujen integroinnin puute, eli selain ei rekisteröi dynaamisesti luotuja sivuja ja esimerkiksi silloin ”edellinen”-nappi ei toimi, mutta on olemassa skripteja jotka ratkaisevat tämän ongelman.
- Dynaamisesti ladattu tieto ei ole hakukonerobottien käytettävissä. Hakukoneet eivät voi suorittaa Javascript:ä, tämän takia sovelluskehittäjän on huolehdittava millä toisella tavala sivuun saisi näkyvyyttä.
- Vanhat tavat tehdä sivuston tilastoja eivät toimi, koska tieto päivittyy ilman URL:n päivittämistä.
- Projekti monimutkaistuu, sovelluksen logiikka järjestäytyy uudelleen, monet tietokäsittelijät siirtyy palvelimelta selainosaan.
- Ja tietenkin vaatii JavaScript-tuen oltavan päällä selainasetuksissa.

On olemassa paljon JavaScript-kirjastoja, jotka tukevat Ajax-periaatetta: Dojo, jQuery, Prototype, ExtJs.

2.3 Sovelluskehykset (Frameworks)

Sovelluskehys tarkoittaa ohjelmistotuotetta, joka muodostaa rungon sen päälle rakennettavalle tietokoneohjelmalle. Ohjelmistokehys on ohjelmoinnin apuväline, jonka tarkoituksena on nopeuttaa uusien ohjelmistotuotteiden valmistusta. Kehys tarjoaa valmiiksi rakennettuja tietokoneohjelman osia, joita ei tarvitse kirjoittaa uudelleen ohjelmistokehityksen aikana – tämä nopeuttaa kehitystyötä. Tavallisesti ohjelmistokehystä ei voi käyttää sellaisenaan suoritettavana ohjelmana, vaan varsinainen toimiva lopputuote saadaan aikaan rakentamalla uusi ohjelma kehyksen päälle.

Monet ohjelmistokehykset ovat oliopohjaisia ja niitä voidaan käyttää WWW-sovellusten kehittämiseen, mutta muitakin käyttötarkoituksia on olemassa. Ohjelmistokehyksiä on toteutettu mm. PHP-, Java-, Ruby- ja Python-ohjelmointikielillä.

Tunnettuja ohjelmistokehyksiä ovat esimerkiksi CakePhp, Zend, Cocoon, Django, Jade, Ruby on Rails, TurboGears, Microsoft .Net Framework.

2.4 Sovelluskehysten turvallisuus

Web-sovelluskehyksillä on yleensä erilaisia omia mekanismeja hoitamaan turvallisuuteen liittyviä ongelmia. Ne suojavat resursseja ulkopuolisilta käyttäjiltä ja luvattomalta koodilta. Selainpuolella käyttäjän turvallisuutta voidaan säätää internetselaimen turvallisuusasetuksilla, mutta palveluntarjoajalle tärkeintä on palvelimen suojaus.

Tietokantasyötteiden käsittelyssä (sql injection -haavoittuvuudet) ja xss-aukoissa (cross-site scripting) piilevät vakavimmat puutteet, joihin kehittäjien kannattaa kiinnittää huomiota.

Kehittäjien on huomioitava myös esimerkiksi missä muodossa salasanat säilytetään, missä muodossa ne välitetään verkossa.

Session kaappaus ja Cross-site Request väärennys voi olla myös tietoturva-aukko, mutta yleensä kehysympäristöt tarjoavat valmiita työkaluja tai ratkaisuja niiden hoitamiseen.

Kaikki riippuu web-sovelluksen kehittäjästä, käyttääkö hän sovelluskehystä vai ei. Sovelluskehykset helpottavat kehitystä ja turvallisuusasioiden hoitamista tarjoamalla työkaluja.

3 PYTHON / DJANGO -SOVELLUSKEHYS

3.1 Yleiskatsaus

Django on ilmainen sovelluskehys web-sivustoihin, joka on toteutettu Python kielellä. Alussa projekti luotiin ylläpitämään ja hallitsemaan uutissivustoja jonka omisti The World Company, USA.

Django-sivusto rakennetaan yhdestä tai useammasta sovelluksesta, jotka tehdään erillisinä kokonaisuuksina ja liitetään toisiinsa. Se on tämän kehyksen suurin ero toisiin web-sovelluskehyksiin kuten esimerkiksi Ruby on Railsiin. Myös URL:n käsittelijät konfiguroidaan koodilausekkeilla, eikä niitä oteta automaattisesti kontrollereiden hierarkiarakenteesta.

Tietokantatoimintoja varten django käyttää omaa ORM:ää, jonka kautta mallit toteutetaan luokilla ja kuvataan tietokanta-schemalla.

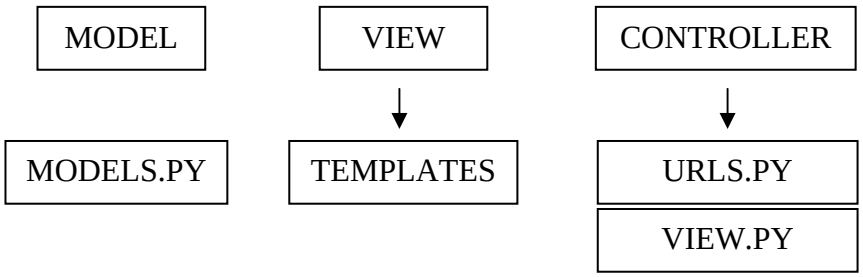
Djangon kautta voidaan periaatteessa hyödyntää nettipohjaisessa sovelluksessa kaikki Python- ohjelmointikielen mahdollisuudet. Itse Python on nuori ohjelmointikieli, se on on monipuolinen ja tulkattava. Pythonia pidetään helppona oppia sen yksinkertaisen syntaksin ja korkean tason tietorakenteiden takia. Monet suosittelevat sitä ensimmäiseksi ohjelmointikieleksi.

Python-kieli tukee useita ohjelmointiparadigmoja; erityisesti sitä voi käyttää oliopohjaisena, proseduraalisena tai funktionaalisena ohjelmointikielenä. Yleisesti ottaen luonteenomaista Pythonille on pyrkimys selkeään ja luettavaan ohjelmakoodiin. Kauneuden ja yksinkertaisuuden tavoitteminen ovat keskeinen osa kielen suunnittelufilosofiaa [2].

Python-tulkki ja -kirjastot on kehitetty avoimen lähdekoodin projektina, ja niitä levitetään Pythonin oman lisenssin alaisena, joka on yhteensopiva myös GPL-lisenssin kanssa. Pythonin lisenssi sallii lisäksi kaikenlaisen kaupallisen käytön ja jopa kaupallisen uudelleenlevittämisen.

3.2 Arkkitehtuuri

Djangon arkkitehtuuri on model-view-controller-tyyppinen. MVC klassisen mallin kontrolleri vastaa Djangon View-tasoa, View vastaa Djangon Template järjestelmää. Tämän takia Djangon arkkitehtuuria sanotaan yleensä Model-Template-View:ksi (kuva 3).



Kuva 3, Model-Template-View

Selaimelta tulevan pyynnön kasittelyprosessi näyttää seuraavalta: annettaessa selaimelle osoiterivin Django tarkistaa url.py:ssä olevan tiedon, löytyykö vastaavalle url:lle funktiota, joka sen käsittelee. Funktiot sijoitetaan view.py tiedostoon tai muualle mihin tahansa tiedostoon, jos vaan annetaan ohjelmalle tietoa sijaintipaikasta. Funktio tarvittaessa tietyn modelin kautta saa tietoja tietokannasta ja vastaavasti tallentaa niitä kantaan, tai muuten tekee joitakin laskennallisia toimintoja. Lopputuloksena funktio ja vastaavasti palvelin palauttaa selaimelle esimerkiksi html-sivun johon on sijoitettu muuttujien kautta näytettävä tieto käyttäjälle.

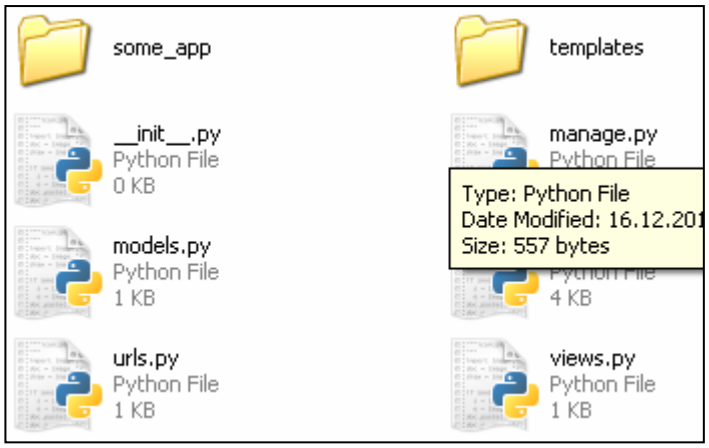
Djangon kehitys alkuvaiheessa uutissivustoja varten on vaikuttanut paljon arkkitehtuuriin. Django tarjoaa työkaluja, joiden avulla päästään todella nopeasti kehittämään tietosivustoja. Kehittäjän ei tarvitse huolehtia sivuston ylläpitotoiminnoista. Djangossa on sisäänrakennettuja osioita hallintaa varten, joka voi tarvittaessa liittää sivustoon, sekä samalla sevoi hallita useampia sivustoja samalla palvelimella.

Hallintotyökalun avulla pystyy luomaan, muokkaamaan ja poistamaan mitä tahansa sivuston objekteja sekä kirjoittaa lokitiedostoja kaikista toiminnoista. Käyttäjä-, ryhmä- ja oikeuksienhallinta on myös olisi valmiina mukana.

Djangon rungon päälle saa rakennetua toimivan sovelluksen todella nopeasti. Django-kehykselle on kehitetty monia muita kirjastoja ja apuvälineita tiedon käsitelyyn.

3.3 Django-esimerkki

Perusohjelmakokoonpano voisi olla seuraava (kuva 4): template-kansio jossa on html-koodauksella *.html mallitiedostoja, models.py-tiedosto, jossa on luokkilla kuvatut objektit, views.py tiedosto jossa on kaikki logiikka ja funktioita jotka palauttavat html-sivuja selaimelle, urls.py tiedosto jossa yhdistetään palvelimelle tullut url-pyyntö ja views.py:ssä olevat toiminnot, jotka suoritetaan.



Kuva 4, esimerkki Django-palvelimellä olevistä tiedostoista

Esimerkiksi (kuva 5 ja kuva 6) annettaessa selaimessa osoitteeksi ”http://127.0.0.1/clock” suoritetaan views.py:ssä olevaa show_time() funktiota joka palauttaa selaimelle time_now.html sivun kellonajalla.

```
1  from django.conf.urls.defaults import *
2  from clock.views import *
3
4
5  # Uncomment the next two lines to enable the
6  # from django.contrib import admin
7  # admin.autodiscover()
8
9  urlpatterns = patterns('',
10     (r'^clock/', show_time),
11
```

Kuva 6, Urls.py tiedosto

```
1  from django.shortcuts import render_to_response
2  import datetime
3
4  def show_time(request):
5      time = datetime.datetime.now()
6
7      return render_to_response('time_now.html', locals())
8
9
```

Kuva 7, Views.py tiedosto

3.4 Mahdollisuuksista

Djangolla on monia sellaisia työkaluja ja ratkaisuja, joita löytyy melkein kaikilta muilta kehyksilta, mutta tietenkin erikoisuuttakin on. Djangoon kuten Pythooniinkin on kootu kaikki tarpeelliset työkalut, huomioitu muiden kehysten ongelmat ja epäkohdat.

Luetellaan keskeiset Djangon ominaisuudet:

- oma ORM, Api tietokantaa varten, joka hoitaa kyselyjen muodostamisen
- oisäänrakennettu ylläpidon käyttöliittymä
- URL käsittelijä
- laajennettava template-järjestelmä omilla tageilla
- välimuistin tuki
- monikielisyyden tuki
- sovellusten arkkitehtuuri on sellainen, että sovelluskokonaisuuksia mahdollista liittää mihin tahansa sivustoon
- geneeriset näytöt
- autentikointi, liitettävät ulkopuoliset moduulit LDAP, OpenId ja muut
- suodattimet (middleware), rakentamiseen kyselyjen lisäkäsittelijöitä, esim. välimuisti, tietojen pakkaaminen, URL normalisointi, anonyymit sessiot.
- kirjasto lomakkeiden käsittelyyn (Forms), voidaan rakentaa lomakkeita tietokantakuvauksen mukaan, voidaan periä sekä laajentaa olemassa olevat.
- sisäänrakennettu automaattinen dokumentointi

Erilaiset kehyksen komponentit ovat yhteydessä vähän toisiinsa. Tämän takia on helppo vaihtaa tietty komponentti toisella, esimerkiksi sisäänrakennettujen template-järjestelmän sijaan voi käyttää Mako tai Jinja, jotka ovat Python-pohjaisille sovelluksille tarkoitettuja template-järjestelmiä.

Kaikki Djangon valmiit osat ovat helposti laajenettavissa ja muokattavissa omissa ratkaisuissa ilman, että alkuperäisen kirjaston lähdekoodin muutetaisiin. Esimerkiksi Djangon hallintotyökalu on muokattavissa sekä rakenteellisesti että myös ulko-näöllisesti.

Django-rungolla on kehitetty monia valmiita ratkaisuja vapaalla lisenssilla, kuten verkkokaupat ja CMS:t.

3.5 Palvelimen konfigurointi

Django on suunniteltu Apache-palvelimelle, moduulilla mod_python, käyttäen tietokantana PostgreSQL. Kytkemällä WSGI:n tuki, Django voi toimia FastCGI:n, mod_wsgi, SCCG ohjauksella Apachella tai millä tahansa palvelimella (lighttpd, nginx). Django voi käyttää myös muita tietokantoja kuten esimerkiksi MySQL:ää, SQLitea, Oraclea.

Django-kehyksen kokonpanossa on oma web-palvelin kehitystä varten. Palvelin tunnistaa automaattisesti tiedostojen muutokset ja käynnistyy uudelleen, mikä nopeuttaa kehitysprosessia, mutta palvelin toimii silloin debug- tilassa ja sopii vain kehitystä ja vianetsintää varten.

Django konfiguroidaan sille tarkoitetussa tiedostossa settings.py:ssä. Siinä kuvataan tietokantayhteydet, luetellaan käytettävät kehyksen osat ja sovellukset, määritellään globaali-muuttujia, asetetaan template-tiedostojen sijaintipolku. Esimerkki tiedostossa (liite 1) ovat keskeiset yleiset määritykset Django-palvelimen käynnistystä varten.

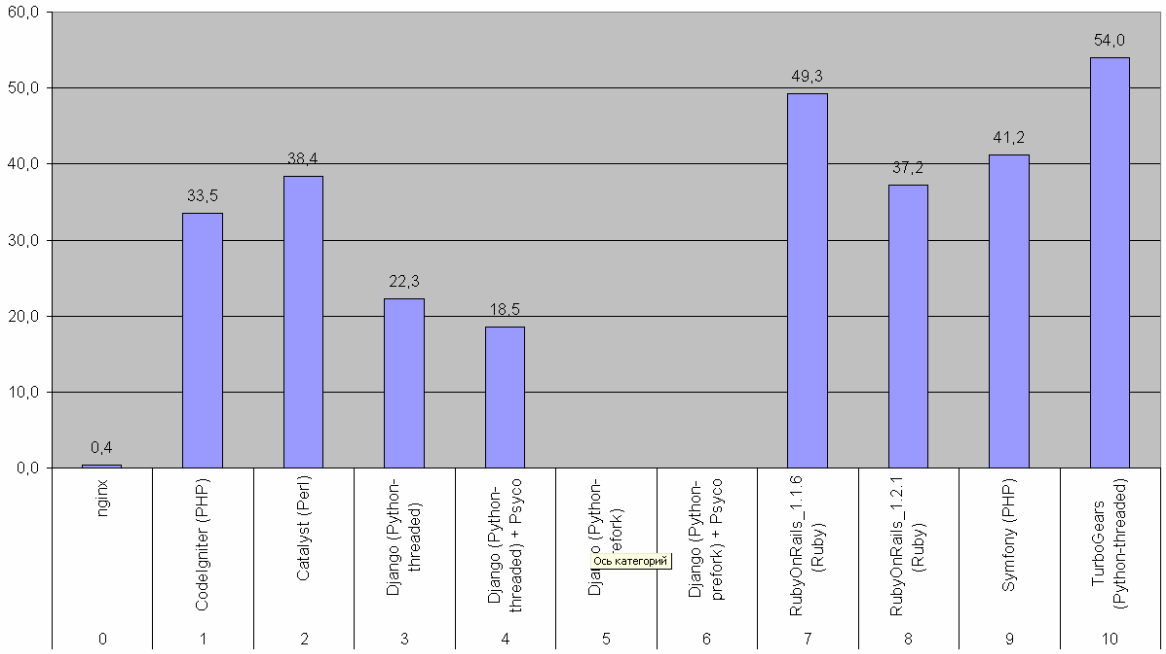
3.6 Djangon vertailu toisiin kehyksiin

Erilaisia web-sovelluskehyksien testaustuloksia löytyy etsimällä netistä. Djangoa vertaillaan muun muassa sellaisiin kuin CakePHP, Catalyst(Perl),

RubyOnRails, TurboGears(Python). Djangon vahvuuksiin tulosten perusteella kuuluu vähäinen prosessorikuormitus (kuva 8, numerot 5 ja 6), pieni transaktioihin käytetty aika sekä palvelimen vastausaika on nopea, eli voidaan sanoa, että Django on ihan huippuluokan web-sovelluskehys[3].

Tietenkin helikopteria ja lentokonetta on vaikea verrata keskenään, mutta perusasetuksilla kevyen sivuston voi testata eri kehyksillä suorituskyvyltä.

Jokainen web-sovellus on kuitenkin erilainen ja tiettyyn tarkoituksen suunnattu kokonaisuus ja on viritettävä niin että hyötyä on mahdollisimman paljon.



Kuva 8, Djangon prosessorikuormitus, paikka 5 ja 6 (Psyco-laajennuksella)

Php-sovelluksista isona miinuksena on se että kehittäjän on aina käytettävää kehyksen arkitehtuurissa määriteltyä tapa kirjoittaa koodia, muuten sovelluksen rakenne voi helposti mennä sekaisin ja on vaikeasti ylläpidettävä ja laajennettavaa. Djangosta voi sanoa että jopa huonon ohjelmakoodin on helppo hahmottaa ja korjata, sekä laajentaa ja ylläpitää.

3.7 Djangon soveltuvuus toteutettuun kaivulupa.fi-projektiin

Moni muu Keypron ohjelmistosta on rakennettu Djangolla. Tässäkin projektissa sen käyttö on järkevää. Yritys on valinnut kehykseksi Djangon, sillä kehitysnopeus on huomattavasti parempi kuin muilla alustoilla, jonkun asian korjaaminen tai vaihtaminen toisella onnistuu helposti ilman, että koodia muokataan monissa paikoissa. Myös sovelluksen ylläpito ja vianetsintä on helppoa. Monet tarvittavat osat ovat jo valmiina paketissa ja vaativat vain pientä viritystä.

Kaivulupa.fi:n beta-version oli tarkoitus toteuttaa nopeasti ja sivuston olisi tarkoitus olla mahdollisesti kevyt ja nopea niin, että kaivaja pystyy tarvittaessa kämmentietokoneella jossain työmaalla tarkastelemaan maanalaisten kaapeleiden johdonomistajia. Djangohan sopii siihen erinomaisesti.

4 KAIVULUPA.FI-PROJEKTI

4.1 Kuvaus palvelusta

4.1.1 Mikä on kaivulupa.fi?

Kaivutöihin ryhtyminen edellyttää sekä johtoselvitystä että lupaa aloittaa kaivutyöt ja viestintämarkkinalaissa säädetään johdonomistajan velvollisuudesta antaa tarvittavat tiedot kaivutöiden suorittajalle.[4]

Kaivulupa.fi on netissä toimiva johtoselvityspalvelu.

Kaivulupa.fi-palvelun täysi hyödyntäminen edellyttää käyttäjätunnusta. Palvelu on kaivajalle maksuton, mutta rekistöröitymisen yhteydessä on hyväksyttävä palvelun käyttöehdot.[5]

Johdonomistajien palvelussa mukana oleminen mahdollistaa toimintatapojen tehostamisen liittyen suunnitteluun, dokumentointiin, työnohjaukseen sekä näyttötoimintojen organisointiin ja seurantaan.

4.1.2 Keskeiset mahdollisuudet

Osoitehaku

Osoitehaulla saadaan osoitteen perusteella Googlen karttanäkymän, jossa voidaan tarvittaessa käyttää katunäkymä, josta on hyötyä kaivajille. Osoitteen haettaessa tarjotaan osoitevaihtoehtoja automaattisesti.

Kartta- ja katunäkymä

Kartta- ja katunäkymät on toteutettu Googlen ilmaisen palvelun avulla, alkuvaiheessa on kaikkien käyttäjien vapaasti käytettävissä. Kirjautumista vaativissa palveluissa käytetään omia karttaratkaisuja.

Kaivualueen piirto karttapohjalla

Kaivupaikkaa rajataan kartalla, piirtämällä kaivualue. Kaikki kaivualueelle osuvat verkkot ja niiden omistajatiedot näytetään.

Johtoselvitys

Verkko-omistajien johtoselvityksen puhelinnumeroita ja muuta kaivamiseen liittyvää tietoa, kuten pakkonäytön tarve saa palvelun kautta.

Karttojen tilaus

Palvelun kautta saa karttatulosteen, josta näkyy kaikkien johdonomistajien verkkojen sijainti kaivualueella, jos johdonomistaja on antanut kaivulupa.fi:lle riittävän aineiston siitä.

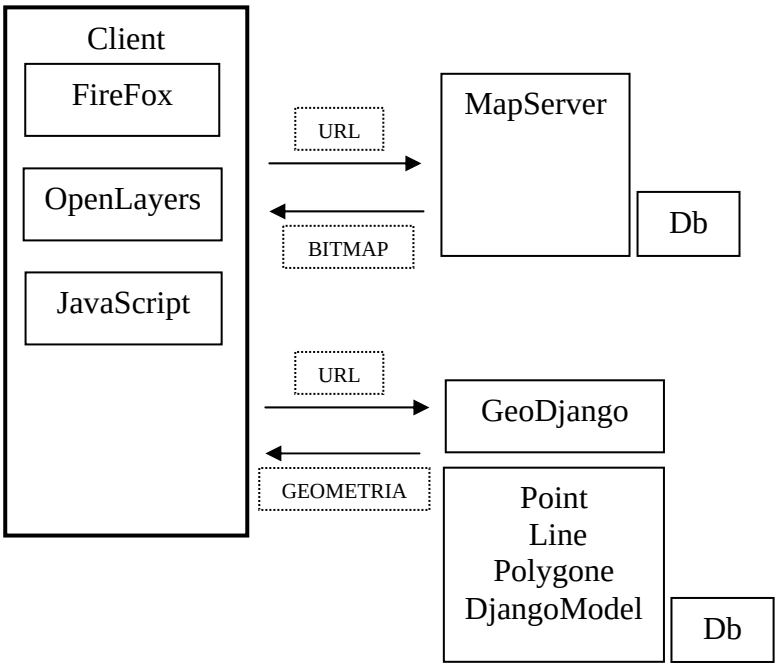
Näyttöjen tilaus (mahdollisuus yhteisnäyttöihin)

Pakollisen näytön alueilla johdonomistaja tekee verkkonäytön kaivajalle. Sellaista tapahtuu yleensä siellä, missä verkon omistaja ei tarkkaan tiedä, missä johdot kulkevat tai missä kaapeleiden määrä tai laatu edelyttää.

4.2 Miten tämä kaikki toimii

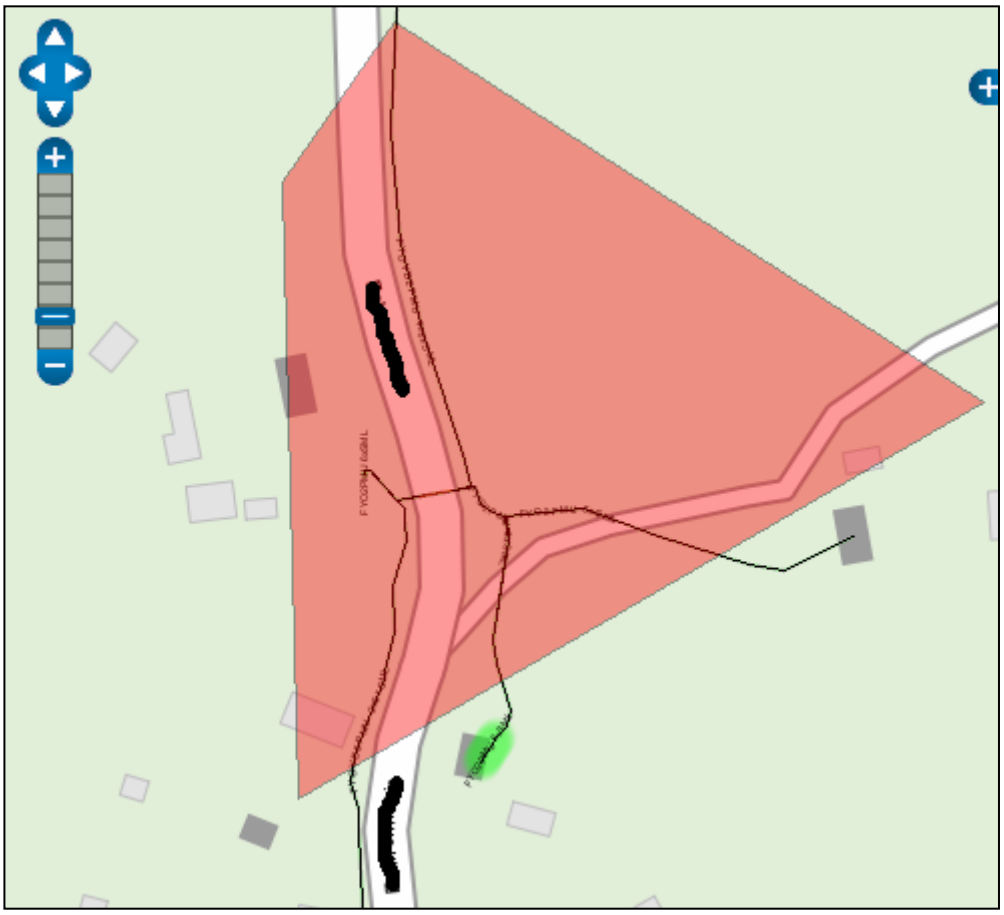
Mukana toiminnassa on muutama ratkaisu kytkettynä yhteen. Web-sovelluskehys, eli runko jonkun päälle rakennetaan kaikki muu. MapServer, joka voi muodostaa kartasta ja johtotiedoista bitmap-kuva, hakee ja tallentaa niitä tietokantaan. OpenLayers ratkaisu on Javascript kirjasto, joka pyörii paikallisessa tietokoneessa ja vastaa objektien piirtämisestä, karttakerrosten näytöstä. GeoDjango, Djangon laajennus geometristen kuvioden käsittelyyn.

Kuvitellaan että halutaan sivulle kartta-alueen, josta näkyy alueella olevat johdot. Käydään prosessi läpi (kuva 9).



Kuva 9, Selaimen yhteys MapServeriin ja Django-palvelimeen

Sivu latautuu, sivussa on JavaScripti joka määrittelee missä kohdassa kartta näkyy ja kutsuu OpenLayers kirjaston näyttämään kartan aluerajauskoordinaateilla. OpenLayers silloin lähettää kyselyn MapServerille, joka hakee kannasta karttaa ja johtotiedot rajatulla alueella ja lähettää karttakuvan vastauksena. Jos tarvitaan johtotietoja objekteina, joita on mahdollista muokata lähetetään johtotietopyyntö Django:n palvelimelle tai muualle ulkopuoliselle palvelulle. GeoDjango ottaa vastaan aluerajauskoordinaatit muodostaa kyselyn ja hakee tarvittavat johtotiedot kannasta ja palauttaa selaimelle tuloksen. OpenLayers piirtää karttakerroksen päälle johtoja (kuva 10).



Kuva 10, johdot kartalla. Taustakartan © Maanmittauslaitos 2011

Mutta johdot voivat olla myös MapServerin muodostamassa staattisessa bitmap-kuvassa riippuen siitä halutaanko ne muokattavaksi. Suorituskyvyn johdosta ei ole järkevä näyttää muutama sata tai tuhatta johtoa objekteina.

Projektissä on käytetty paljon jo valmiita ratkaisuja, joko ulkopuolisia vapaalla lisenssilla olevia työkaluja tai Keypron omia Core-kirjastoja.

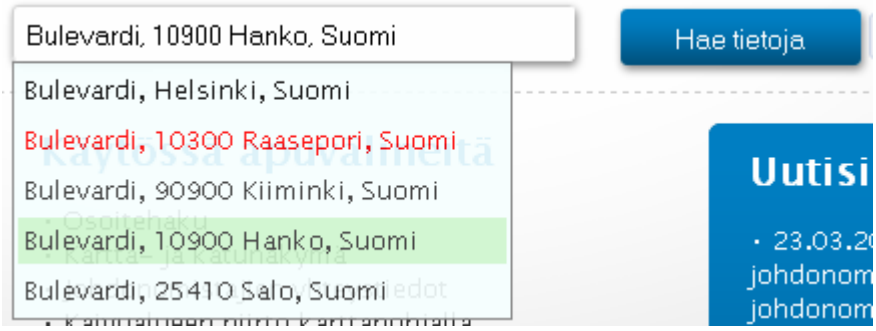
4.3 Toteutuksen rakenne

4.3.1 Osoitehaku

Palvelun käyttö ja kaikki muut jatkossa olevat toiminnot alkavat osoitehausta. Osoitehaku on toteutettu käyttämällä Googlen ilmaista kartta-api:ta, Google edellyttää, että sen ilmaiset palvelut käytetään vain sivustoissa, jotka ovat myös ilmaisia ja kaikki pääsevät esteittä käyttämään niitä. Niin Googlen palveluita on otettu käyttöön vain näissä sovelluksen

osissa, jotka ovat kaikille ilmaisia. Jatkotoiminnoissa on käytetty OpenLayers-karttoja.

Osoitehaku on rakennettu Javascriptillä Dojo-kirjastolla. Siinä on käytetty niin sanottuja widgetteja, jotka ovat riippumattomia erillisiä kokonaisuuksia. Omia widgettejä luomalla näitä voi myöhemmin sijoittaa mihin tahansa sivustoon. Sekä osoitehaku että Googlen karttanäkymä ovat erikseen tehtyjä kokonaisuksia. Osoitehaku toimii autocomplete muodossa (kuva 7), osoite-ehdotuksia haetaan Googlen Geocooding-palvelun kautta, samalla palvelulla konvertoidaan osoite koordinaateiksi ja saadaan ne ulos karttanäkymää varten. Karttanäkymään syötetään parametriksi koordinaatit ja saadaan kartan kohdistuvan syötettyyn osoitteeseen.



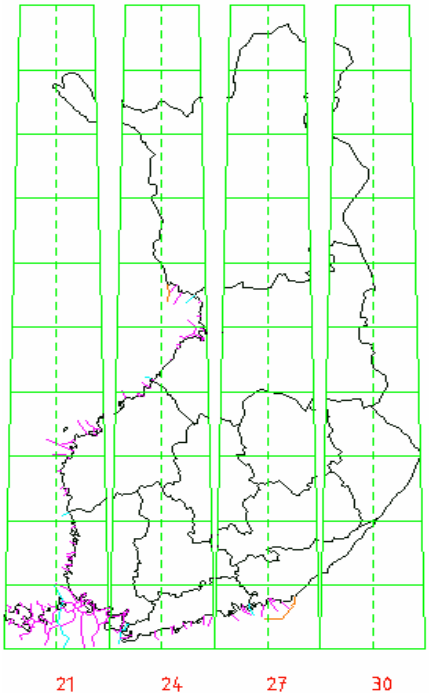
Kuva 11, hakuvaihtoehtojen näyttö

Googlen kartta on valittu käytettäväksi sen takia, että siinä on mukana katunäkymä, jolloin kaivajalle on helpompi hahmottaa paikkaa valokuvien perusteella.

4.3.2 Koordinaatistot

Kaikki muut karttatoiminnot palvelussa sitten alkavat Googlen koordinaateista, jotka ovat WGS84 maapallokoordinaattijärjestelmää. Maanmittaustöissä käytetään tasokoordinaatistoja, tyypillisesti Transverse Mercator-projektiota 3 asteen projektiokaistoin (Kuva 12). Käynnissä on parhaillaan siirtyminen KKJ-koordinaattijärjestelmistä yleiseurooppalaiseen Euref-pohjaiseen järjestelmään. Palvelimella Googlen WGS84-koordinaatit

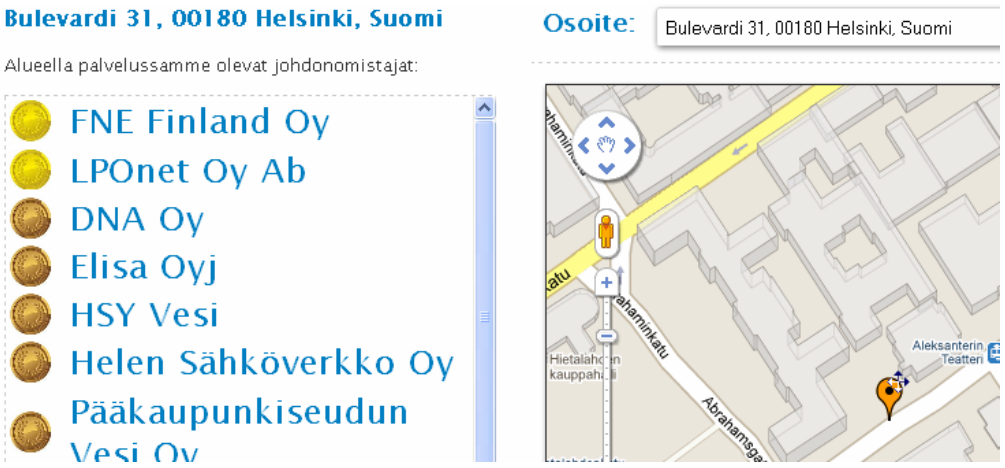
muunnetaan käytettyyn tasokoordinaatistoon kun siirrytään kirjautumista vaativiin osiin. Koordinaattimuunnoksissa käytetään PROJ.4 kirjastoja.



Kuva 12, kaistajako

4.3.3 Perustietojen näyttö

Ilmaisessa versiossa ilman rekisteröitymistä käyttäjä saa näkyville vain lähialueen verkon omistajanimet ja omistajien tasot (kuva 13).



Kuva 13, johdonomistajien näyttö

Tasoja on kolme: pronssi, hopea ja kulta. Niiden mukaan johdonomistajilla on erilaisia mahdollisuuksia palvelussa.

Yhdessä tunnissa rekisteröimätön käyttäjä voi tehdä vain muutamia kyselyitä. Näin palvelu on suojattu ulkopuolisilta kyselyiltä, koska palvelu on tässä vaiheessa kuitenkin ilmainen ja kaikkien käytettävissä, mutta mikään muu ulkopuolinen ohjelmisto ei pysty vastaanottamaan palvelimelta useita vastauksia.

4.3.4 Johtoselvitys

Johtoselvitys tapahtuu tuottamalla pdf-karttaotteen kaivualueen leikkaavista johdoista (kuva 10) ja tulosteen johdonomistatiedoista ja kaapeleiden määrästä.

4.3.5 Rekisteröinti ja autentikointi

Rekisteröitiin ja autentikointiin on käytetty Djangon valmiita ratkaisuja, lomakkeita ja kirjastoja. Djangolla on oma html-template rekisteröintiä, autentikointia sekä salasanaa palauttamista varten Rekisteröinti tapahtuu lomakkeen täydentämisellä ja lomake lähetetään ylläpidon hyväksyttämiseksi, ylläpidolta tulee sähköposti käyttäjätunnusten aktivoinnista.

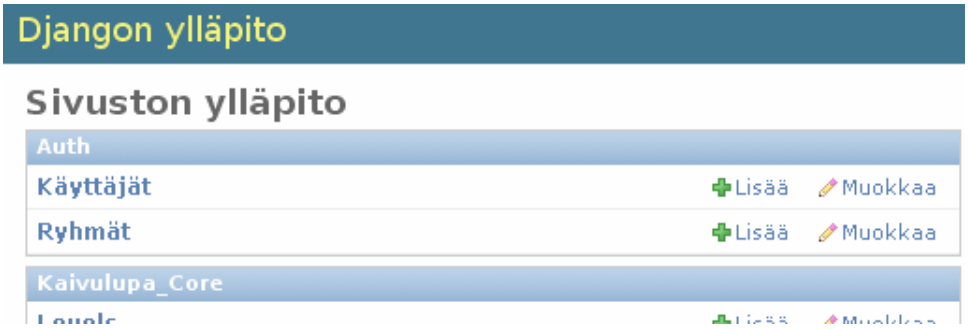
Lomakkelle on tehty oma objekti, joka talletetaan tietokantaan, djangon ylläpito-osiossa, joka on mukana valmiina, ylläpito-henkilökunta näkee lähetetyn lomakkeen ja voi tarvittaessa korjata tietoja ja aktivoida käyttäjätilin. Silloin syntyy Djangon User-niminen objekti ja tallentuu tietokantaan. Djangon valmiin ratkaisun kanssa on helppo hallita käyttäjiä, rooleja, oikeuksia.

Autentikointi tapahtuu myös valmiilla työkalulla. Django hoitaa lomakkeiden tarkastukset automaattisesti ja salaa salasanoja. Autentikointiin on ostettu käyttöön https:n SSL sertifikaatti. Niin kukaan ulkopuolinen ei voi varastaa salasanaa, jos käytetään langatonta verkkoa kirjautumiseen.

Sekä rekisteröinnissä että hallintapaneelissa on käytetty JavaScriptiä tietojen piilottamisen ja näyttämiseen. Kaikki muu on toteutettu template-järjestelmää käyttämällä.

4.3.6 Ylläpitäjän työkalut

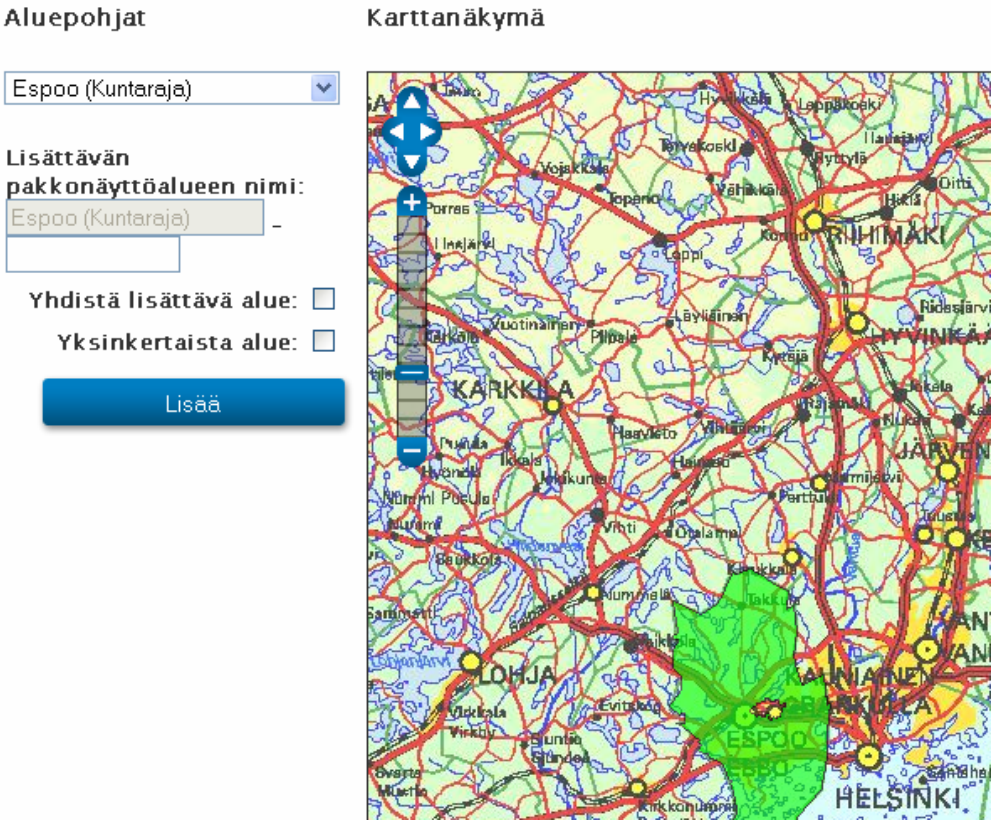
Djangon admin-panelissa ovat heti valmiina näkyvissä käyttäjätilit ja ryhmät (kuva 13). Mikä tahansa muu objekti voi laittaa sinne muokattavaksi, sellaisen toiminnon määrittäminen tapahtuu admin.py-tiedostossa.



Kuva 13, Djangon ylläpito

Myös mikä tahansa muu ylläpito-toiminto määritetään tässä tiedostossa, kuten esimerkiksi käyttäjän aktivointi, käyttöehtojen hyväksyminen, käyttöehtojen tiedoston lataaminen palvelimelle, organisaatioiden hallinta, oikeustasojen muokkaaminen.

Sivustolle on tehty myös erillinen hallintapaneeli verkko-omistajille ja kaivajille. Siinä ei ole käytetty Djangon ylläpitoratkaisuja vaan tehty sivustoon rekisteröityneille käyttäjille ”Oma sivu” -valinta, jossa kaivajat pystyy yksinkertaisesti muuttamaan joitakin omia profiilitietoja ja johdonomistajat lisäämään toiminta-alueita, näyttöalueita, muokkaamaan organisaatiotietoja, tulostamaan kyselyraportteja (kuva 14).



Kuva 14, pakkonäytealueen rajaus kartalla

Maksullisessa palvelussa on käytetty OpenLayers-ratkaisuja, jotka on toteutettu JavaScriptillä. OpenLayersilla on myös tehty kaivualueen piirto kartalla. Piirtoalueen koordinaatit talletetaan spatiaalitietokantaan.

4.3.7 Spatialivarastot

Sijaintitietoja ja geometrisia kuvioita varastoidaan tietokantoihin perinteisen numeraalisen tiedon tapaan, mutta spatiaalisen tiedon tallentaminen vaatii tallennusjärjestelmältä, tietokantaskeemalta ja tietokannanhallintajärjestelmältä tiettyjä ominaisuuksia, jotka mahdollistavat spatiaalisen tiedon tehokkaan käsittelyn. Tietokannaksi on valittu Oracle, joka sopii tähän erinomaisesti.

Oracle Spatial tukee useita erilaisia primitiivisiä geometriatyyppejä sekä näistä koottuja kokoelmia. Kaksiulotteiset pisteet muodostuvat X- ja Y-koordinaattiparista, jonka avulla ilmaistaan yleensä itä- ja

pohjoiskoordinaatteja. Viivajonot muodostuvat yhdestä tai useammasta pisteparista, jotka määrittelevät viivasegmentin. Polygonit muodostuvat yhdistetyistä viivajonoista, jotka muodostavat suljetun kehän. Näiden primitiivisten tyyppien avulla kuvataan varsinaista spatiaalista tietoa. Esimerkiksi polygoneilla kuvataan aluetta ja viivalla johtoa.

4.3.8 GeoDjango

GeoDjango on Djangon laajennuskirjasto, sekä tärkeä komponentti MapServerin ja sivun(OpenLayers) välissä. GeoDjango muodosta kyselyitä sopiviksi ja lähettä ne edelleen karttapalvelimelle. GeoDjango osaa käsitellä geometrisia primitiivejä ja verrata niitä, esimerkiksi ylittääkö viiva jonkun toisen viivan tai neliön, onko yksi primitiivi toisen sisällä. Nämä primitivi-objektit on helppoa säilyttää tietokannassa. GeoDjango myös tarvitaan spatiaalitietokannan kanssa työskentelyä varten.

5 JOHTOPÄÄTÖKSET

Vaikka Django-kehys on nuori, siinä on huomattu kaikkien jo kauan olleiden sovelluskehysten ongelmat. Djangon kehitys jatkuu koko ajan ja uusia versiota ilmestyy. Myös kehykselle on jo tehty pajon laajennusosia ja lisäkirjastoja. Kehyksen käyttöönotto on nopea, kehitys on helppo ja hyvin ylläpidettävä. Uusien osien liittäminen projektiin onnistuu hyvin.

Django tukee aika monta muita ratkaisuja, kuten tässä projektissä tarvittiin GeoDjangon laajennusta. Myös Json-, Xml- objektien käsittelyyn on monta kirjastoa, sellaisten tuki tässä projektissa on ollut välttämätön ehto.

Django-ympäristössä Kauvulupa.fi- projektin toteutus on ollut vauhdikkasta ja sujunut ihan hyvin sekä ohjelmiston koodi on järjestelmällistä ja helposti ylläpidettävä.

VIITELUETTELO

[1] *Ajax: A New Approach to Web Applications, Verkkotietolähde, 27.03.2011*
http://www.adaptivepath.com/ideas/essays/archives/000385.php.

[2] *The Zen of Python*, *Verkkotietolähde, 02.04.2011*
http://www.python.org/dev/peps/pep-0020.

[3] *The performance test of 6 leading frameworks*, Verkkotietolähde, 14.04.2011
http://www.alrond.com/en/2007/jan/25/performance-test-of-6-leading-frameworks/.

[4] *Viestintämarkkinalaki,* 111 §, *Verkkotietolähde, 03.05.2011*
http://www.finlex.fi/fi/laki/ajantasa/2003/20030393.

[5] *Kaivulupa.fi*, *Verkkotietolähde, 02.04.2011*
http://beta.kaivulupa.fi.

Djangon settings.py diedoston esimerkki

```
import os
PROJECT_DIR = os.path.dirname(__file__)

DEBUG = True
TEMPLATE_DEBUG = DEBUG
ADMINS = ()
MANAGERS = ADMINS

DATABASE_ENGINE = 'postgresql_psycopg2'
DATABASE_NAME = 'X'
DATABASE_USER = 'X'
DATABASE_PASSWORD = 'X'
DATABASE_HOST = ''
DATABASE_PORT = ''

LANGUAGE_CODE = 'fi'
SITE_ID = 1
USE_I18N = True

MEDIA_ROOT = os.path.join(PROJECT_DIR, 'media/').replace('\\','/')
ADMIN_MEDIA_ROOT = os.path.join(PROJECT_DIR, 'media/').replace('\\','/')
MEDIA_URL = '/media/'
ADMIN_MEDIA_PREFIX = '/admin_media/'

SECRET_KEY = '_^+2&-153t=v#4g)h(!pu=wp^0@j1!*9lv#@n!67)^nx8u#3vk'

TEMPLATE_LOADERS = (
    'django.template.loaders.filesystem.load_template_source',
    'django.template.loaders.app_directories.load_template_source',
)

MIDDLEWARE_CLASSES = (
    'django.middleware.common.CommonMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
)

ROOT_URLCONF = 'urls'

TEMPLATE_DIRS = (
    os.path.join(os.path.dirname(__file__), 'templates').replace('\\','/'),
)

INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'json'
```