Web-sovelluskehitys Vibe.d-alustalla

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Tietotekniikan koulutusohjelma

Riihimäki, kevät 2017

Janne Lehtimäki

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietotekniikan koulutusohjelma
Riihimäki

Tekijä Janne Lehtimäki **Vuosi** 2017

Työn nimi Web-sovelluskehitys Vibe.d-alustalla

TIIVISTELMÄ

Tässä opinnäytetyössä luodaan katsaus D-ohjelmointikieleen ja web-sovelluksissa käytettäviin tekniikoihin. Lukijalle esitellään Vibe.d-kirjasto ja Dietmallit. Lisäksi tutustutaan kilpaileviin web-sovelluskehyksiin ja muihin ohjelmointikieliin.

Työtä varten toteutettiin web-selaimessa toimiva monen käyttäjän tekstieditorisovellus käyttäen Vibe.d-sovelluskehystä. Olin käyttänyt D-kieltä jo aiemmin, mutta Vibe.d-kirjasto ei ollut entuudestaan tuttu. Kehitysympäristön asennusta ja editorin rakennetta käydään läpi tarkemmin työn loppupuolella.

Työn tuloksena selvisi, että Vibe.d soveltuu web-sovelluskehitykseen jo nykyisellään, mutta ei ole vielä kaikilta osin yhtä hiottu kuin suuremmat sovelluskehykset, kuten Django. Myös D:n saatavuus eri alustoille on rajatumpi kuin esimerkiksi Pythonilla.

Opinnäytetyön tarkoituksena on antaa lukijalle käsitys Vibe.d-kirjastosta ja yleiskuva D-ohjelmointikielestä ja sen soveltuvuudesta web-kehitykseen.

Avainsanat Vibe, D, web-sovellus, WebSocket

Sivut 36 sivua

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Information Technology
Riihimäki

Author Janne Lehtimäki **Year** 2017

Subject Web application development with Vibe.d

ABSTRACT

This thesis provides an overview to the D programming language and methods used in web application development. Reader is introduced to Vibe.d web application framework and Diet HTML templates. Competing web frameworks and programming languages are introduced as well.

As the practical part of the thesis a multiuser cloud text editor was developed using Vibe.d. I had used D language a bit before but had no previous experience with Vibe.d. Inner workings of the editor and toolchain setup are covered in last part of the thesis.

Results show that Vibe.d is suitable for web application development as it is now but it's not as polished as bigger frameworks like Django. Some platforms are missing proper D runtime support which is a limiting factor.

Goal of the thesis is to give reader an overview of Vibe.d web framework and the D programming language and it's suitability to web development.

Keywords Vibe, D, Web application, WebSocket

Pages 36 pages

SISÄLLYS

Sanasto

ABI	Application Binary Interface	Matalan tason rajapinta ohjelmamoduulien välillä. Siinä määritellään mm. funktioiden kutsumistapa ja datatyyppien koot.
AJAX	Asynchronous JavaScript and XML	Joukko tekniikoita, joilla websivu voi siirtää dataa taustalla asynkronisesti.
AOT	Ahead-of-time compilation	Prosessi, jossa tavukoodi käännetään natiivikoodiksi AOT-kääntäjällä.
CSS	Cascading Style Sheets	HTML-sivujen tyylimäärittelyt.
CTFE	Compile Time Function Evaluation	Mekanismi, joka mahdollistaa D-funktion suorituksen käännösaikaisesti.
JIT	Just-in-time compilation	Prosessi, jossa ohjelman lähdekoodi käännetään suorituksen aikana.
JSON	JavaScript Object Notation	Tapa esittää dataa ihmisille helposti luettavassa muodossa.
NoSQL	Not only SQL	Käsite, joka kuvaa tavallisesta relaatiotietokannasta poikkeavia tietokantoja.
ORM	Object-relational mapping	Tapa muuttaa tietoa kahden yhteensopimattoman tyyppijärjestelmän välillä. Voidaan toteuttaa vaikkapa luokkana, jonka metodit tekevät kyselyitä relaatiotietokantaan.
REST	Representational State Transfer	HTTP-protokollaan perustuva tapa toteuttaa rajapintoja.
Redis	REmote DIctionary Server	Muistissa toimiva tietovarasto tietorakenteille.

SSL	Secure Sockets Layer	SSL on standardi salatun HTTPS-yhteyden muodostamiseen.
UFCS	Uniform Function Call Syntax	Tapa funktiokutsujen kirjoittamiseen selkeämmin ketjuttaessa D-funktioita.

1 JOHDANTO

Opinnäytetyön tarkoituksena on tutustuttaa lukija D-ohjelmointikieleen ja web-sovellusten kehittämiseen Vibe.d-kirjaston avulla. Työn alussa esitellään D-ohjelmointikieli ja D:n standardikirjasto yleisellä tasolla. Samalla käydään myös läpi kielen historiaa ja D:n hyötyjä ja haittoja verrattuna muihin kieliin. Tämän jälkeen tutustutaan Vibe.d-kirjastoon sekä kilpaileviin kirjastoihin ja yleisiin web-tekniikoihin.

Vibe.d on kirjasto asynkronisten web-sovellusten toteuttamiseen. Sillä voi tehdä tavanomaisia HTTP/HTTPS-pohjaisia sovelluksia ja REST-rajapintaa käyttäviä sovelluksia. Kirjastoa voi myös käyttää osana graafista työpöytäsovellusta, ja sillä voi luoda matalan tason socket- ja WebSocket-yhteyksiä. Vibe.d tukee MongoDB- ja Redis-tietovarastoja natiivisti.

Työn loppupuolella luvussa viisi tutustutaan D-käännösympäristön asentamiseen ja esimerkkiprojektina toteutetun Vibe.d-sovelluksen rakenteeseen. Sovellus käyttää HTML5-tekniikoita ja jQuery-kirjastoa selainkäyttöliittymässä. Sovelluksen toiminnallisuus on rajattu, joten lähdekoodin pitäisi olla selkeästi luettavaa. Esimerkkisovelluksen lähdekoodi on saatavilla osoitteesta https://www.dropbox.com/sh/3ixxnw7n1b04rs9/AADyx6vOWPUPCaa75yaHHkvra?dl=0.

Opinnäytetyön kohdeyleisönä ovat henkilöt, jotka haluavat tutustua D-kieleen tai web-sovellusten kehitysprosessiin Vibe.d:n avulla. Opinnäytetyön alussa käydään D:tä läpi yleisellä tasolla, mutta esimerkkiprojektin ymmärtämiseksi on hyödyllistä osata myös JavaScriptin perusteet.

2 D-KIELI

D on yleiskäyttöinen ohjelmointikieli. Se on ottanut vaikutteita varsinkin C++:sta, sillä D:n kehitys alkoi yrityksestä uudelleensunnitella C++:aa. Kielessä on kuitenkin myös muista ohjelmointikielistä, kuten Pythonista ja C#:sta tuttuja ominaisuuksia. Näitä ovat muun muassa automaattinen muistinhallinta ja ohjelmamoduulit. D on ABI-yhteensopiva C:n kanssa ja tukee myös rajallisesti C++:aa ja Objective-C:tä (D Language Foundation, n.d.).

D on käytössä skriptikielenä Remedy Entertainmentin Quantum Break -pelissä (DConf2016 2016). Facebook on käyttänyt sitä vuodesta 2013 alkaen (DrDobbs 2013). Muita käyttäjiä ovat esimerkiksi eBay ja Sociomantic.

2.1 Historiaa

D-kielestä on kaksi versiota: vanhempi D1 ja nykyisin käytössä oleva D2. Walter Bright aloitti kielen kehittämisen joulukuussa 1999 ja ensimmäinen julkaisu tapahtui kaksi vuotta myöhemmin joulukuussa 2001 (Bright, 2014). Vuonna 2007 kielestä julkaistiin versio 1.0 ja samalla alkoi D2:n kehitys. Samana vuonna Andrei Alexandrescu liittyi kielen toiseksi pääsuunnittelijaksi. D1:n ylläpito lopetettiin joulukuussa 2012 (Alexandrescu, 2011).

Kielen standardikirjastosta on olemassa kaksi eri versiota: oliopohjainen Tango joka on saatavilla vain vanhempaan D1:een, ja Phobos, jota nykyinen D2 käyttää. Tangosta on olemassa versio D2:lle, mutta sitä ei tueta virallisesti. Standardikirjasto ja D:n ajonaikainen kirjasto on erotettu toisistaan D2:ssa. Lisäksi D2:ssa on uusia ominaisuuksia, jotka eivät ole D1:n kanssa yhteensopivia.

2.2 Kielen ominaisuuksia

D:n syntaksi on hyvin tutun oloista C:n sukuisia kieliä aiemmin käyttäneelle, mutta siinä on joitakin eroavaisuuksia joista tärkeimpiä pyrin käymään tässä läpi.

Tulkattavat ohjelmointikielet lukevat lähdekoodia ja suorittavat sitä ajonaikaisesti. Koodi toimii missä tahansa ympäristössä, josta löytyy kielen komentotulkki. Esimerkiksi JavaScript on tulkattu kieli.

D on käännettävä ohjelmointikieli. Käännettävissä kielissä lähdekoodi täytyy kääntää objektitiedostoiksi ja linkittää suoritettavaksi tiedostoksi tai jaetuksi kirjastoksi ennen käyttöä. Käännetyt ohjelmat toimivat vain alustalla jolle ne on käännetty. Etuna tulkattuihin kieliin on nopea suoritus.

Näiden lisäksi on kieliä joiden kääntäjät tuottavat alustariippumatonta tavukoodia, joka suoritetaan virtuaalikoneessa. Microsoftin .NET-alusta ja Adoben Flash ovat hyviä esimerkkejä. JIT-käännös ja AOT ovat tekniikoita, jotka mahdollistavat tavukoodin suorituksen lähes natiivinopeudella. JIT-käännös tarkoittaa suorituksen aikana dynaamisesti tapahtuvaa koodin käännöstä, kun taas AOT-käännöksessä tavukoodi käännetään natiiviksi AOT-kääntäjällä ennen suoritusta. (BlogGeek.me 2014.)

D:n perustietotyypit ovat:

- void (ei tyyppiä), bool (totuusarvo)
- Kokonaisluvut: (u)byte, (u)short, (u)int, (u)long
- Liukuluvut: float, double, real
- Imaginääriluvut: ifloat, idouble, ireal
- Kompleksiluvut: cfloat, cdouble, creal
- Merkit: char, wchar, dchar

Muuttujan tyypiksi ja funktion paluuarvoksi voidaan D:ssä määrittää myös auto, jolloin kääntäjä päättelee oikean tyypin. Muuttujalle voi laittaa immutable-tyyppimääreen, jolloin muuttuja voidaan alustaa vain kerran eikä arvoa voi tämän jälkeen muokata. const-tyyppimääre toimii kuten immutable, mutta ainoastaan sen ohjelmalohkon sisällä jossa muuttuja on määritelty.

2.2.1 Funktiot

CTFE on kielen mekanismi, joka antaa kääntäjän suorittaa funktioita käännösaikaisesti automaattisesti, mikäli funktion arvot tiedetään ennen suoritusta:

```
double n = uniform(9.0, 18.0); //satunnainen
```

```
static compiletime = sqrt(9.0);
writeln("Runtime: ", sqrt(n));
```

UFCS mahdollistaa funktioiden kutsumisen luettavammalla tavalla ja helpottaa niiden ketjutusta. Esimerkiksi foo(bar(x)) on mahdollista kirjoittaa muotoon x.bar().foo(). Tämän lisäksi funktiot, joilla ei ole argumentteja voidaan kirjoittaa ilman sulkuja seuraavasti:

```
double x = 9.0;
double y = x.sqrt; // sqrt(x)
```

2.2.2 Virheenkäsittely

D:ssä virheet voi käsitellä poikkeuksien avulla try-, catch- ja finally-lohkojen sisällä. Toinen vaihtoehto on käyttää **scope**-avainsanaa, joka suorittaa ehdollisesti koodia lohkon päättyessä:

- scope(exit) – koodi suoritetaan aina
- scope(success) – koodi suoritetaan, kun virheitä ei tapahtunut
- scope(failure) – koodi suoritetaan poikkeuksen sattuessa

Seuraava koodinpätkä tulostaa ”ZYX” suorituksen onnistuessa.

```
scope(success) writeln("X");
{
   scope(exit) writeln("Y");
   writeln("Z");
}
```

2.2.3 Template

Template eli malli on kielen rakenne, joka mahdollistaa funktioiden ja luokkien toimimisen geneerisillä tyypeillä. C++:n mallien tapaan myös D:ssä metaohjelmointi on mahdollista. D:n standardikirjasto käyttää malleja runsaasti. Seuraavassa koodiesimerkissä on yhtä tyyppiä käyttävä malli. Mikään ei kuitenkaan estä useamman tyypin käyttämistä. Malleja käytettäessä myös automaattinen tyypinmääritys on mahdollista.

```
T foo(T)(T a, T b)
{
  return a + b;
}
void main()
{
  assert( foo!int(1,2) == 3);       // int foo(int a, int b)
  assert( foo!float(1.2,1.8) == 3);
  assert( foo(1.3,2) == 3.3);
}
```

2.2.4 Mixin

Mixin on kielen ominaisuus, joka mahdollistaa generoidun koodin lisäämisen lähdekoodin sekaan. Mixin voi olla merkkijono, jolloin teksti lisätään sellaisenaan, tai se voi olla malli. Mikäli mixin-malli sisältää kutsuja ulkoisiin ohjelmamoduuleihin, siinä täytyy sisällyttää moduuli paikallisesti ennen funktiokutsuja (Çehreli 2016, 555).

```
mixin template PrintMixin(T)
{
  void foo(T x)
  {
    import std.stdio;
    writeln("Value: ", x);
  }
}
mixin PrintMixin!int;
mixin("int X = 5;");
void main()
{
  foo(X); // Value: 5
}
```

2.2.5 Range

Range on abstraktio perättäisten alkioiden käsittelyyn. Rangen tyyppi määräytyy sen jäsenfunktioiden perusteella.

InputRange on rangetyypeistä yksinkertaisin ja se tukee ainoastaan perättäisten alkioiden lukemista. Sen täytyy toteuttaa seuraavat jäsenfunktiot:

- **empty**, joka tarkistaa onko range tyhjä
- **front**, joka palauttaa ensimmäisen elementin
- **popFront**, joka poistaa ensimmäisen elementin

ForwardRange toteuttaa edellä mainitut ja **save**-funktion, joka palauttaa kopion rangen nykyisestä kohdasta. BidirectionalRange toteuttaa näiden lisäksi **back** ja **popBack**-toiminnot. RandomAccessRange on monipuolisin ja vaatii edellisten lisäksi opIndex-operaattorin kuormituksen, ja mikäli range ei ole päättymätön sille pitää määritellä myös length, joka palauttaa pituuden. (Çehreli 2016, 562-563.)

Standardikirjastossa useat funktiot toimivat rangeilla. Taulukon osaa eli taulukkoviipaletta voi käyttää RandomAccessRangena std.array-moduulin avulla. Moduuli toteuttaa vaaditut funktiot taulukkoviipaleille.

2.2.6 Merkkijonot

Useassa ohjelmointikielessä on oma luokka merkkijonoille. D:ssä merkkijonot ovat char-, wchar- tai dchar-taulukoita. Merkkijonot ovat UTF-enkoodattuja, ja muita merkistökoodauksia käytettäessä dataa on hyvä käyttää suoraan ubyte[] tai char* -tyypeillä.

Mikäli merkkijonossa on erikoismerkkejä ja käytössä oleva merkkijonotyyppi on liian pieni kaikkien merkkien tallentamiseen, merkkijonotaulukon indeksin käyttäminen ei välttämättä tuota toivottua tulosta (Çehreli 2016, 61-62). Tällöin voidaan käyttää **walkLength**-funktiota merkkijonon pituuden selvittämiseksi ja **std.utf.toUTFindex**-funktiota merkkijonon indeksin muuttamiseksi UTF-taulukkoindeksiksi.

2.2.7 Fiber

Fiber on eräänlainen kevytversio tavallisesta säikeestä. Se käyttäytyy tavallisen säikeen lailla, mutta toimii yksittäisen säikeen sisällä. Fiberin suorituksen voi keskeyttää kutsumalla erityistä **yield**-funktiota, joka pysäyttää suorituksen tiettyyn kohtaan. Esimerkiksi Unity3D-pelimoottori käyttää sitä samanaikaisuuden simuloimiseksi skriptissä.

```
import std.stdio;
import core.thread : Fiber;

void func()
{
    writeln("Yksi");
    Fiber.yield();
    writeln("Kaksi");
}
void main()
{
  Fiber fiber = new Fiber(&func);

  fiber.call();
  writeln("main 1");
  fiber.call();
  writeln("main 2");
  assert(fiber.state == Fiber.State.TERM);
}
```

2.3 Käännöstyökalut ja standardikirjasto

Digital Mars D (DMD) on kielen virallinen referenssikääntäjä x86-arkkitehtuurille. Saatavilla on myös GNU-projektin GDC ja LLVM-pohjainen LDC, jotka tukevat useampia arkkitehtuureja. D:tä voi käyttää myös skriptinomaisesti rdmd-työkalulla, joka automatisoi käännösprosessin ja ajaa ohjelman.

2.3.1 DUB

DUB on komentorivityökalu D-projektin käännösprosessin ja riippuvuuksien hallintaan. Kaikilla DUB-projekteilla on JSON- tai SDLang-muotoinen asetustiedosto, jossa määritetään muun muassa paketin yleiset tiedot, käännöstyyppi, riippuvuudet ja tiedostopolut. DUB tukee DMD, GDC ja LDC -kääntäjiä. Sillä voi myös generoida projektitiedostot VisualD-, SublimeText- ja CMake-projekteille. Työkalun kotisivulla on lista kaikista saatavilla olevista paketeista ja käyttäjät voivat lisätä sinne uusia paketteja.

2.3.2 Phobos

Phobos on D-kielen standardikirjasto. Kirjasto on jaettu kolmeen nimiavaruuteen. Phoboksen päämoduuli on **std**, josta löytyvät kaikki yleisimmin tarvittavat moduulit. **etc**-nimiavaruus sisältää rajapinnan muutamalle C/C++-kirjastolle ja **core**-nimiavaruudessa on ajonaikaisen kirjaston matalan tason funktioita. Phobos on saatavilla Linux-alustalle jaettuna kirjastona.

Kirjaston std.algorithm-moduuli sisältää yleiskäyttöisiä algoritmeja hakuun, vertailuun, elementtien iterointiin, lajitteluun, elementtien muokkaamiseen ja joukkojen käsittelyyn. Useat niistä tukevat predikaatteja eli totuusarvon palauttavia funktioita, joilla määritellään jokin haluttu ehto tai järjestys ennen funktion parametreja:

```
int[] a = [1, 2, 3, 1, 5];
count!"a > b"(a, 1); //3
a.sort!("a > b");    //[5,3,2,1,1]
```

Phoboksessa on rajapinta osalle C:n otsikkotiedostoista core.stdc-moduulissa. Loogisten säikeiden lisäksi Phobos tarjoaa matalamman tason työkalut säikeiden luomiseen ja hallintaan core.thread-moduulissa, ja niiden synkronointiin core.sync-moduulissa. Alla olevassa taulukossa (Taulukko 1) on lueteltu joitakin keskeisiä moduuleja.

Taulukko 1. Phobos.

std.container	geneerisiä tietorakenteita
std.concurrency	loogiset säikeet
std.digest	tiivistealgoritmit
std.parallelism	symmetrinen multiprosessointi (SMP)

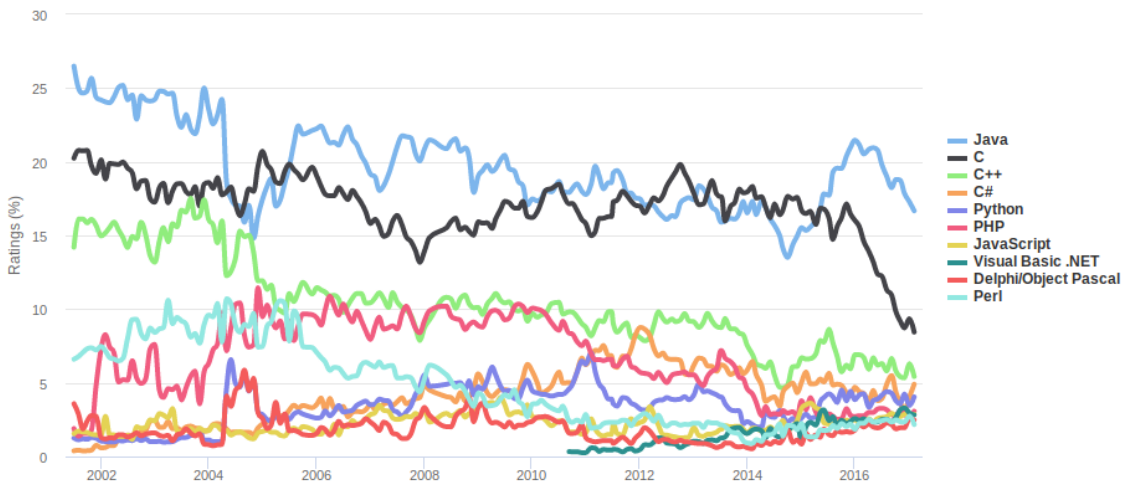
std.socket	socket-primitiivit
std.range	perättäinen alkioiden käsittely
std.json	JSON
std.utf	UTF-merkkijonojen käsittely
std.outbuffer	datan serialisointi

2.4 Muita ohjelmointikieliä

Yksi syy miksi päädyin Vibe.d-sovelluskehykseen enkä esimerkiksi Pythonia käyttävään Djangoon, oli halu testata kuinka hyvin D soveltuu web-kehitykseen. D:n heikkona puolena voisi pitää web-kehityksessä sitä, että ohjelma täytyy kääntää eikä näin ole yhtä helposti siirreltävä palvelimelta toiselle. Hyvänä puolena D:ssä on sen vähäinen muistintarve ja monipuolisuus.

TIOBE ylläpitää ja päivittää kuukausittain ohjelmointikielten suosiota kuvaavaa indeksiä. Helmikuussa 2017 viisi suosituinta kieltä ovat indeksin mukaan Java (16,68 %), C (8,45 %), C++ (5,43 %), C# (4,90 %) ja Python (4,04 %). C:n sukuiset kielet ovat perinteisesti hyvin suosittuja, mutta varsinkin C:n suosio on laskenut viimeisen vuoden aikana (Kuva 1). D on listan 22. sijalla 1,21 %:n osuudella. Heinäkuuhun 2016 verrattuna sen sijoitus on noussut yhdellä sijalla ja osuus kasvanut 0,18 prosenttiyksikköä. (TIOBE 2017.)

W3Techsin mukaan PHP on palvelinpuolen ohjelmointikielistä edelleen ylivoimaisesti käytetyin 82,5 % osuudella. Tulos voi olla PHP:n kohdalla liioiteltu, sillä monen tekniikan kohdalla ei voida luotettavasti tunnistaa millä palvelinpuolen tekniikalla sivusto on toteutettu. PHP:ta seuraavat Microsoftin ASP.NET-teknologia (15,2 %) ja Java (2,7 %), jotka ovat suosittuja yrityskäytössä. (W3Techs 2017.)



Kuva 1. Suosituimmat ohjelmointikielet helmikuussa 2017 (TIOBE 2017).

3 VIBE.D

Vibe.d on kirjasto asynkronisten web-sovellusten kehittämiseen. Tavallisesti asynkronisessa sovelluksessa käytetään takaisinkutsufunktioita, joita ohjelma kutsuu aina operaation valmistuessa. Vibe pyrkii tekemään asynkroniset kutsut käyttäjältä piilossa, jolloin operaatiot ovat näennäisesti synkronisia. Vibe.d toteuttaa tämän fiber-kevytsäikeillä (rejectedsoftware e.K, n.d.).

Kirjastolla tehdyt web-sovellukset käyttävät sisäänrakennettua HTTP/HTTPS-palvelinta. Vibe sisältää myös välityspalvelimen ja sillä voi tehdä HTTP-pyyntöjä ulkoisille palvelimille curl-kirjaston avulla. Kuormantasaus useamman palvelimen välillä on kehitteillä, mutta ei ole tätä kirjoittaessa vielä täysin valmis.

Vibe:ssa on korkean tason tuki web- ja REST-rajapinnoille. Se tukee myös WebSocket-yhteyksiä ja tavanomaisia socket-yhteyksiä. Kirjastoa voi käyttää työpöytäsovelluksen graafiseen käyttöliittymään integroituna Windows- ja X11-ympäristöissä.

3.1 Diet-mallit

Diet on HTML-sivupohjien tekemiseen tarkoitettu yksinkertaistettu versio HTML:stä. Syntaksiltaan se on suurimmalta osalta Pug-yhteensopiva, joten Pugin dokumentaatio pätee pääosin (Vibed.org 2017). Diet-mallit käyttävät HTML-tagien sijaan sisennystä hierarkian määrittelyssä. Tageille voi määritellä attribuutteja sulkujen sisällä. Mallille voi myös antaa muuttujia parametrina D-koodista.

```
doctype html
```

```
html(lang="en")
    head
        title= myVar
    body
- foreach( i; 0 .. 10 )
    - auto num = i+1;
            p= num
        h1& Numbers
```

Malleissa voi kutsua D-koodia ja luoda kokonaisia D-funktioita. Suodattimien avulla malliin voi upottaa myös CSS-, JavaScript- ja Markdown-koodia.

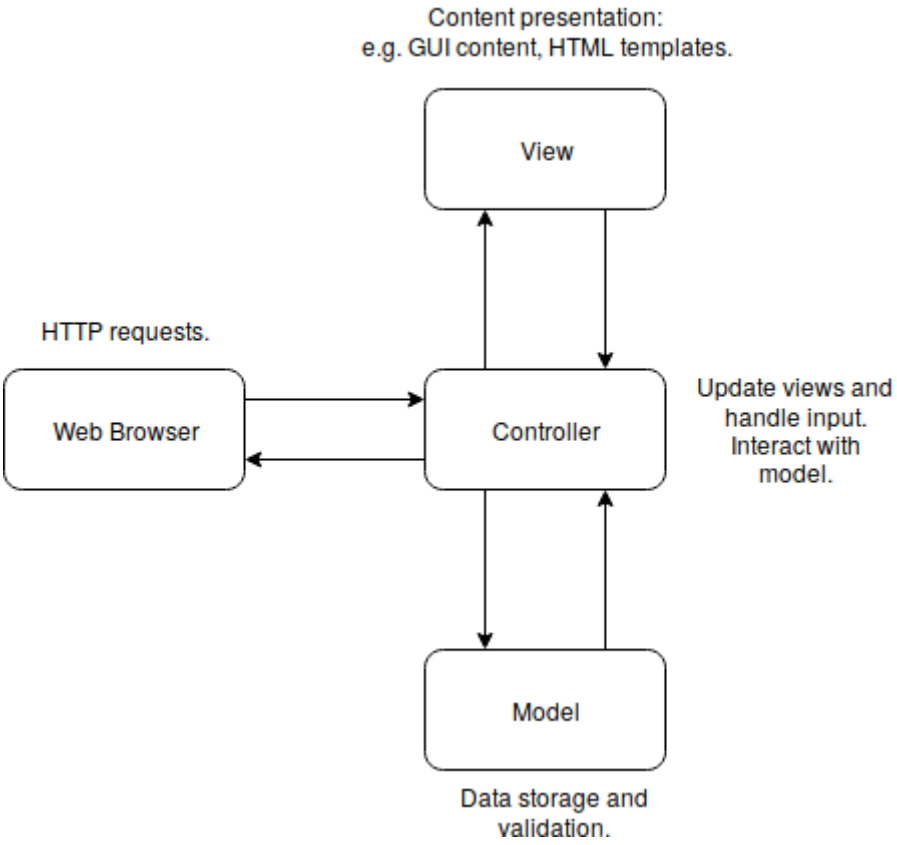
Diet-malli näytetään **render**-funktion avulla pyyntöä käsitellessä. Mallille syötetyt muuttujat luetaan käännöksenaikaisesti, joten niiden tyyppi on oltava tiedossa. Jokaisen malliin tehdyn muutoksen jälkeen ohjelma täytyy kääntää uudelleen. Tämän takia JavaScript- ja CSS-koodi kannattaa sijoittaa ulkoiseen tiedostoon.

Lokalisointi muille kielille on mahdollista GNU gettext -kirjaston avulla. Käännöstä varten sovellukseen on lisättävä TranslationContext, jossa määritellään saatavilla olevat kielet. Tämän jälkeen käännettävien tagien perään merkitään ”&” ja käännettävän merkkijonon tunniste. Gettextin käyttämän po-käännöstiedoston polku on myös lisättävä DUB-projektin asetustiedostoon.

3.2 Muita web-sovelluskehyksiä

Muita web-sovelluskehyksiä Vibe.d:n lisäksi ovat esimerkiksi AngularJS (JavaScript), Zend Framework (PHP) ja ASP.NET (C#). Yrityksissä käytetään paljon myös Javaa. HotFrameworksin mukaan Spring Framework on edelleen suosituin Java-pohjainen sovelluskehys (HotFrameworks 2017).

Useat sovelluskehykset käyttävät MVC-arkkitehtuuria, jossa toiminnallisuus ja näkymä ovat erotettuna toisistaan (Kuva 2).



Kuva 2. Model-View-Controller.

Otetaan esimerkkinä Django, joka on suosittu MVC-arkkitehtuuria noudattava Python-sovelluskehys. Django-sovelluksessa halutut URL-kuviot määritellään erillisessä tiedostossa ja URL-käsittelijä ohjaa kyselyn oikealle view-funktiolle säännöllisen lausekkeen perusteella. Vibe.d:ssa reitit voi määrittää URLRouter-luokan avulla samankaltaisesti, tai vaihtoehtoisesti luomalla uusi luokka, joka asetetaan toimimaan web- tai REST -rajapintana. Tällöin rajapintaluokan julkisista funktioista tulee reittejä, joiden tyypit määräytyvät niiden nimen, parametrien ja attribuuttien mukaan. Djangossa Model-luokat käyttävät ORM:ia ja vastaavat tietokannan tauluja. Vibe.d sovelluksella ei ole oletuksena tietokantaa, vaan käyttäjä määrittää sen tarvittaessa itse.

Djangon middleware-luokat voivat lisätä toiminnallisuutta pyynnön käsittelyn eri vaiheisiin tehden siitä modulaarisen. Vibessa ei tätä kirjoittaessa ole middleware-tukea. Djangossa on HTML:n generointiin Django Template Language -kieli, joka pääpiirteittäin vastaa Dietin ominaisuuksia.

Python-tulkki löytyy nykyään useimmilta palvelimilta jo valmiina ja muille suosituille kielille, kuten PHP:lle ja Javalle on yleensä valmis asennuspaketti. Eräs Viben heikkouksista onkin sen kohtuullisen suppea käyttöjärjestelmätuki, sillä jotkin järjestelmät eivät vielä tue D:tä kunnolla tai niille ei ole vielä asennuspaketteja. Kunnollisen ja kattavan dokumentaation löytäminen on myös jossain määrin hankalampaa suositumpiin sovelluskehyksiin verrattuna.

4 YLEISTÄ WEB-SOVELLUKSISTA

Verkkosivut on yleensä rakennettu HTML-, JavaScript- ja CSS-tekniikoilla. Web-sovellus on palvelimella toimiva ohjelma, joka tarjoaa dynaamista sisältöä selaimelle staattisten sivujen sijaan. Useimmiten web-sovellus käyttää tietokantaa käyttäjätietojen tai muiden tietojen tallentamiseen. Tietokanta voi olla joko tavallinen relaatiotietokanta tai NoSQL-ratkaisu.

4.1 Asynkroniset kutsut

Web-sovelluksissa aikaa vievät operaatiot suoritetaan yleensä asynkronisesti. Ajax-kutsut, tiedostojen käsittely ja socket-operaatiot käyttävät käyttäjän määrittelemää takaisinkutsufunktiota, jota kutsutaan kun operaatio on valmis. Synkronisessa kutsussa ohjelma jäisi odottamaan toimenpiteen valmistumista, mutta asynkronisessa kutsussa ohjelman suoritus jatkuu taustalla kunnes operaatio on valmis ja takaisinkutsufunktiota kutsutaan.

4.2 URL-reititys

Usein web-sovellukset toimivat URL-reitityksellä. Jokaiselle reitille määritellään takaisinkutsufunktio, jota kutsutaan kun vierailija käy tietyssä osoitteessa. Reitit ovat usein GET- tai POST-kyselyille. Vibe.d-sovelluksessa reiteillä on req- ja res-muuttujat, jotka sisältävät HTTP-pyynnön tilan.

```
void index(HTTPServerRequest req, HTTPServerResponse res)
{
  render!("index.dt");
}
shared static this()
{
  auto router = new URLRouter;
  router.get("/", &index);
  auto settings = new HTTPServerSettings;
  settings.port = 8080;
  listenHTTP(settings, router);
}
```

Staattiset tiedostot, kuten kuvat voidaan asettaa noudettavaksi erilliseltä HTTP-palvelimelta sovelluksen sijaan. Tämä tehdään käyttäen käänteistä välityspalvelinta, jota esimerkiksi Nginx tukee. Tiedostojen nouto ulkoisesta sijainnista voidaan tehdä nopeussyistä, jolloin käänteinen välityspalvelin toimii eräänlaisena välimuistina, tai kun halutaan tasata kuormaa

useammalle palvelimelle. Tämä laskee websovelluksen prosessointikuormaa ja parantaa kyselyjen vasteaikaa. (Visolve 2017.)

4.3 Tietovarastot

Tietovarasto on paikka johon tallennetaan dataa pysyvästi. Niitä ovat esimerkiksi tietokannat ja tiedostot. Vibe.d tukee MongoDB ja Redis NoSQL-tietovarastoja. Relaatiotietokannassa kyselyt suoritetaan tavallisesti SQL-kielellä, kun taas NoSQL-palveluissa tietoa voi hakea eri tavoilla, esimerkiksi dokumenttipohjaisesti.

4.3.1 MongoDB

MongoDB on avoimen lähdekoodin dokumenttipohjainen NoSQL-tietokanta. MongoDB-kanta koostuu kokoelmista, jotka koostuvat dokumenteista. Dokumentit tallennetaan BSON-muodossa. BSON eli Binary JSON on JSON-dokumenttien binäärimuotoon tallentamiseen tarkoitettu tiedostomuoto. Tästä johtuen BSON on vahvemmin tyypitetty kuin JSON, eikä siinä ei ole yksittäistä numerotyyppiä vaan koosta riippuvat kokonais- ja liukulukutyypit. Samassa kokoelmassa olevilla MongoDB-dokumenteilla voi olla eri rakenne. Tavallisten relaatiotietokantojen tapaan tietueilla voi olla indeksejä, ja jokaisella dokumentilla on ObjectId, joka toimii pääavaimena. MongoDB soveltuu hyvin prototyypitykseen joustavan dokumenttirakenteensa vuoksi.

4.3.2 Redis

Redis on muistissa toimiva tietovarasto tietorakenteille. Redistä voi käyttää nopeana tietokantana tai välimuistina. Se voi myös välittää viestejä julkaisija-tilaaja-mallilla, jossa Redis-asiakas voi tilata halutun viestityypin tai julkaista viestin tietylle kanavalle. DB-Enginesin mukaan Redis on suosituin avain-arvotietovarasto (DB-Engines 2017).

Redis tukee useita datatyyppejä. String on niistä yksinkertaisin ja moni muu tietorakenne rakentuu niistä. Tavanomaisten tietorakenteiden kuten listojen lisäksi Redis tukee joitakin abstrakteja tietorakenteita. Tällainen on esimerkiksi HyperLogLog, jolla voi laskea uniikkeja asioita.

4.4 Istunnot ja evästeet

Eväste on web-palvelimen käyttäjän tietokoneelle säilömää dataa. HTTP on tilaton protokolla, ja evästeiden avulla istunnosta saadaan tilallinen. Eväste lähetetään HTTP-pyynnön otsikkotiedoissa jolloin käyttäjän selain tallentaa sen. Evästeet voivat olla istuntokohtaisia tai pysyviä. Eväste poistetaan kun se vanhentuu. Eväste voi säilöä vaikkapa ”muista minut”-toiminnon kirjautumistietoja. Euroopan unionissa evästeiden säilömiseen on

pyydettävä käyttäjältä lupa (europa.eu 2016). Tämän voi tehdä esimerkiksi bannerilla, kun käyttäjä saapuu sivustolle.

Istunto on käyttäjän ja palvelimen välinen pysyvä yhteys. Istunnon muodostamisen aikana luodaan uniikki istuntoavain, jonka avulla palvelin voi käyttää pyynnöissä istuntokohtaisia muuttujia paljastamatta niitä selaimelle.

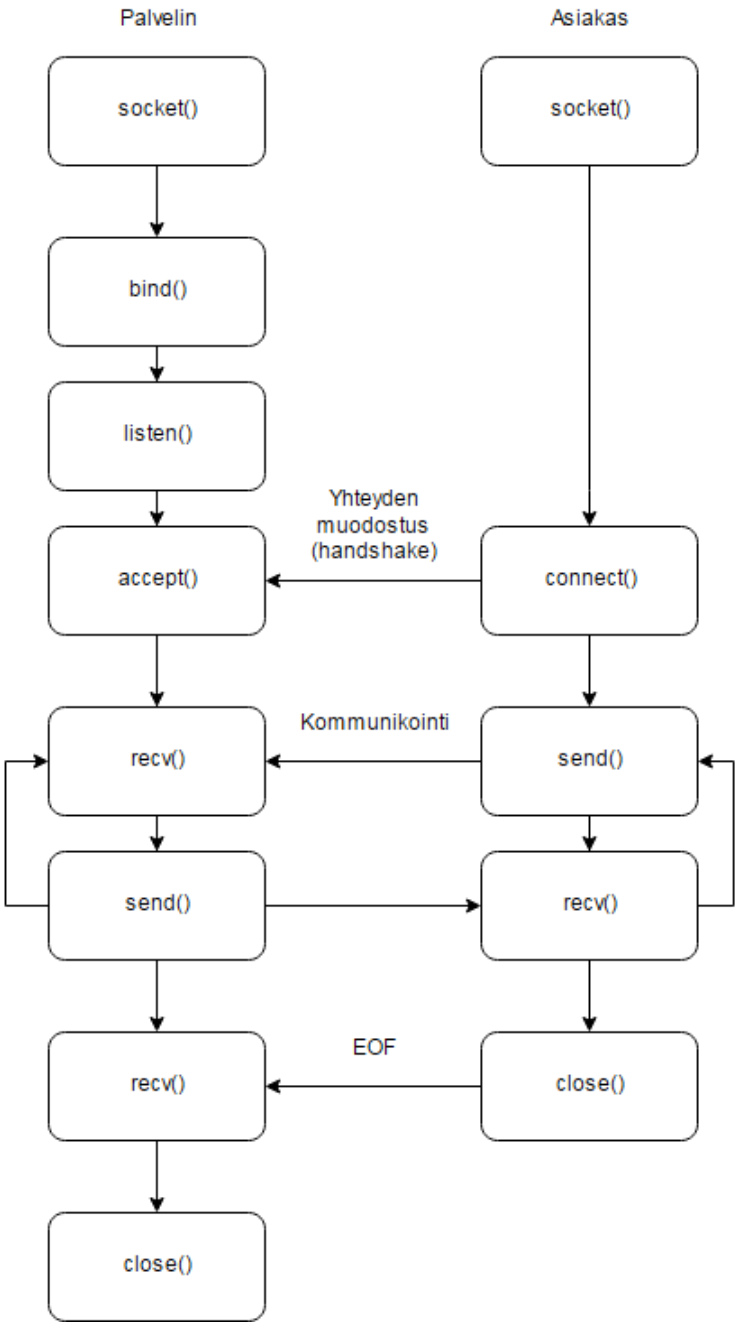
4.5 Socket

Socket on verkkoyhteyden paikallinen päätepiste. Socket voi toimia verkossa tai paikallisesti, jolloin sitä voi käyttää viestimiseen muiden ohjelmien kanssa IPC-mekanismina.

Verkossa toimivat sovellukset käyttävät yleisimmin TCP- tai UDP-protokollaa kommunikointiin socketilla. TCP on tilallinen protokolla, jossa on ruuhkautumisen hallinta ja virheenkorjaus. Sitä käytetään kun pakettien täytyy siirtyä luotettavasti, tai kun niiden tarvitsee saapua tietyssä järjestyksessä vastaanottajalle. UDP on tilaton protokolla, jota käytetään kun tarvitaan matalaa vasteaikaa. UDP ei takaa pakettien lähetyksen onnistumista eikä siinä ole virheenkorjausta. (TechNet 2003.)

Unix-pohjaisissa käyttöjärjestelmissä Unix domain socket mahdollistaa kaksisuuntaisen viestinnän paikallisesti. Unix-socketit käyttävät osoiteavaruutena tiedostojärjestelmää IP-osoitteen ja portin sijasta, joten niitä käytettäessä täytyy huomioida tiedoston käyttöoikeudet.

Palvelin-socketia luodessa socket täytyy ensin sitoa IP-osoitteeseen ja porttiin. Yksittäinen portti voi olla kerrallaan vain yhden sovelluksen käytössä, joten bind() epäonnistuu jos portti on jo käytössä. Bind-kutsun jälkeen palvelin voi alkaa kuunnella saapuvia yhteyksiä. Uuden yhteyden saapuessa se täytyy hyväksyä ennen viestinvaihtoa. Synkronisessa palvelimessa accept() odottaa kunnes yhteys saapuu. Yhdistämisprosessi on havainnollistettu alla olevassa kuvassa (Kuva 3).

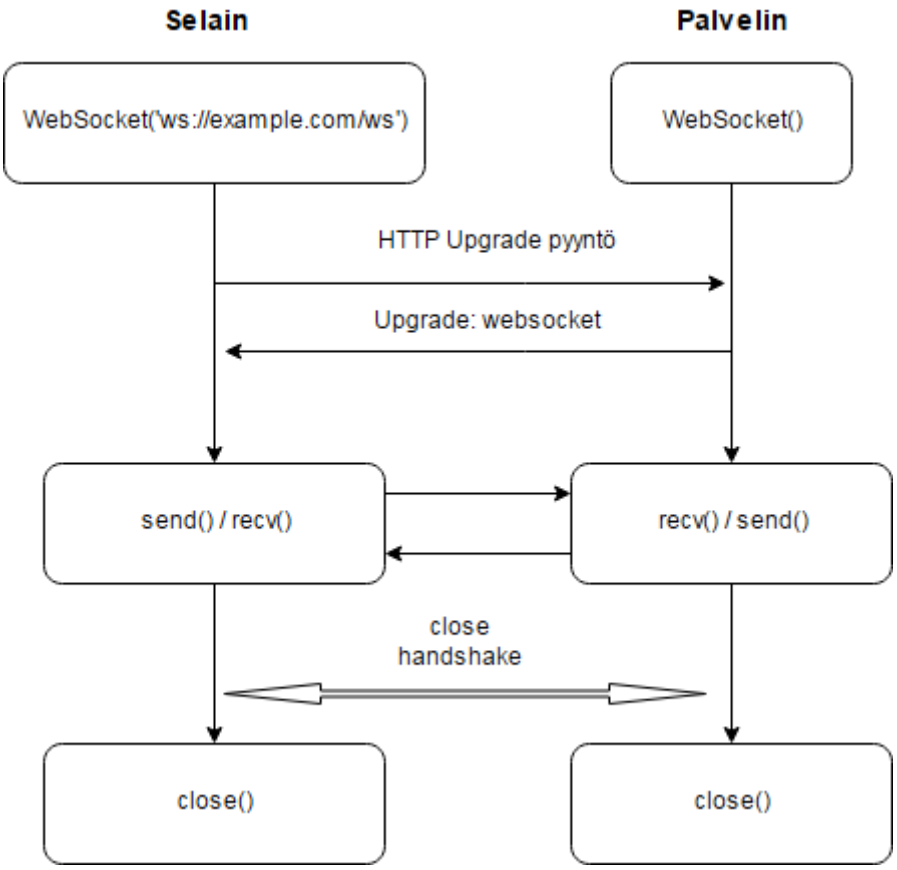


Kuva 3. Yhdistämisprosessi.

4.6 WebSocket

WebSocket on HTML5-rajapinta, joka mahdollistaa perinteisen socketin kaltaisen kommunikoinnin web-selaimen ja palvelimen välillä. WebSocket on Can I Use -sivuston mukaan tuettuna kaikissa suurimmissa selaimissa (caniuse.com 2017). Vielä muutama vuosi sitten tuki oli suppeaa, mutta nykyisin Github ja monet muut sivustot ovat siirtyneet WebSocketiin. WebSocket-yhteydet käyttävät TCP-yhteyden päällä toimivaa ws-protokollaa tai salattua wss-protokollaa.

Yhteyden muodostaminen tapahtuu lähettämällä HTTP Upgrade -pyyntö palvelimelle ja päivittämällä yhteys. Jos yhteys on salattu, selain lähettää myös Sec-WebSocket-Key-otsakkeen ja palvelin vastaa Sec-WebSocket-Accept-otsakkeella. Yhteyttä sulkiessa kummatkin osapuolet lähettävät close frame -viestin. Alla on havainnollistettu tavanomainen salaamattoman yhteyden muodostus (Kuva 4). (CaitieM 2013.)

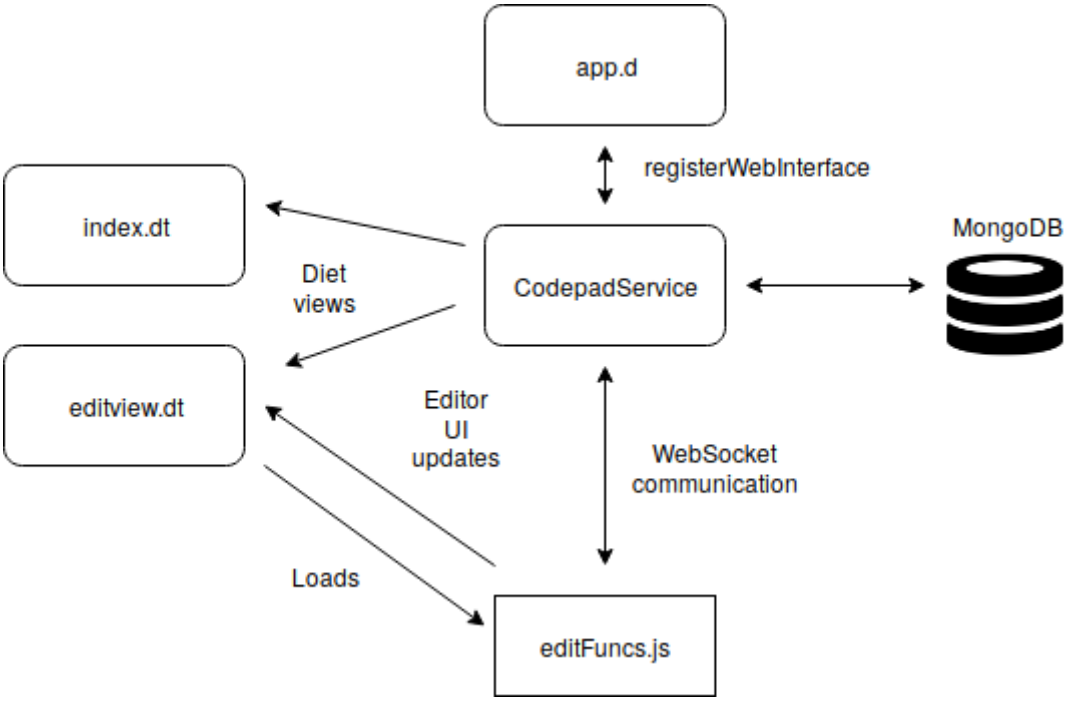


Kuva 4. WebSocket-yhteys.

5 ESIMERKKIPROJEKTI: TEKSTIEDITORI PILVESSÄ

Tässä kappaleessa käydään läpi selaimessa toimivan tekstieditorin toteutus esimerkkiprojektina. Web-sovellus koostuu ”huoneista”, joihin käyttäjä voi liittyä. Huoneessa olevat käyttäjät voivat muokata avointa dokumenttia samanaikaisesti. Käyttäjien sijainti ja valinnat synkronoidaan ja visualisoidaan muille huoneessa olijoille.

Editorin palvelin toteutetaan käyttäen Vibe.d-sovelluskehystä ja MongoDB NoSQL-kantaa. Selaimessa käytetään ja JavaScriptiä ja jQuery-kirjastoa toimintojen toteuttamiseen. Itse editori toteutetaan Ace JavaScript-komponentilla. Tämä mahdollistaa yli 100 kielen syntaksikorostuksen, jonka lisäksi Acessa on hakutoiminto ja syntaksivirheiden merkitseminen ainakin JavaScript-kielelle. Kommunikointi editorin ja palvelimen välillä tapahtuu HTML5 WebSocketin avulla ja sivujen lataus HTTP:lla. Esimerkkisovelluksen rakenne on kuvattu alla (Kuva 5).

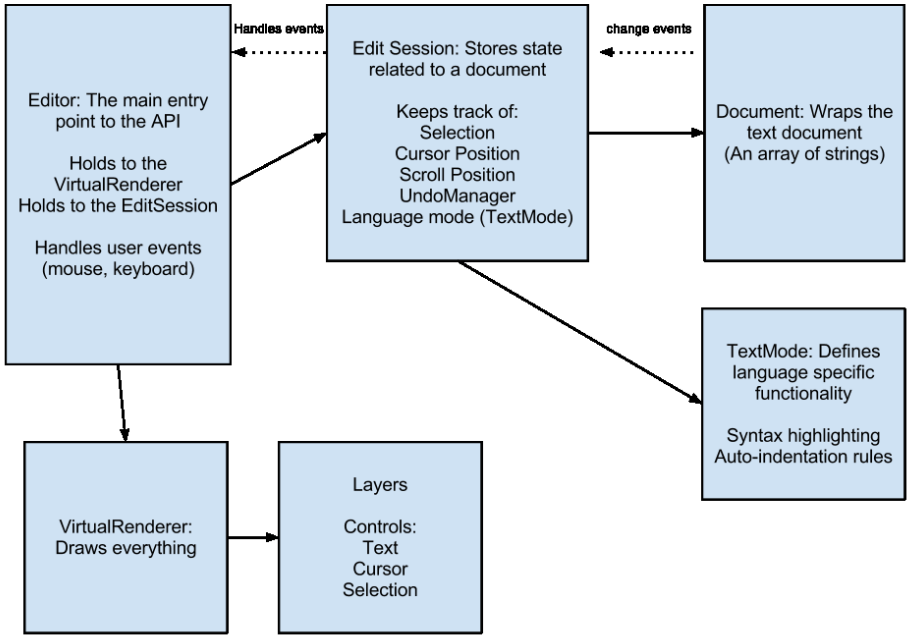


Kuva 5. Esimerkkisovelluksen rakenne.

5.1 Ace

Ace on JavaScript-tekstieditorikomponentti, jota Cloud9 IDE, Inc. ja Mozilla kehittävät. Editorissa on korostustuki yli 110 kielelle. Acesta löytyy myös paljon muita toimintoja, kuten haku säännöllisillä lausekkeilla, automaattisisennys ja tekstin drag ’n drop -tuki.

Valitsin Acen projektiin sen monipuolisuuden vuoksi. Siinä moni toiminto, kuten haku tulevat sisäänrakennettuna ja usealle toiminnolle löytyy lähes suora tuki rajapinnasta. Acen rakenne on kuvattu alla olevassa kaaviossa (Kuva 6). Ace tulee esimerkkisovelluksen mukana ja löytyy public-kansiosta.



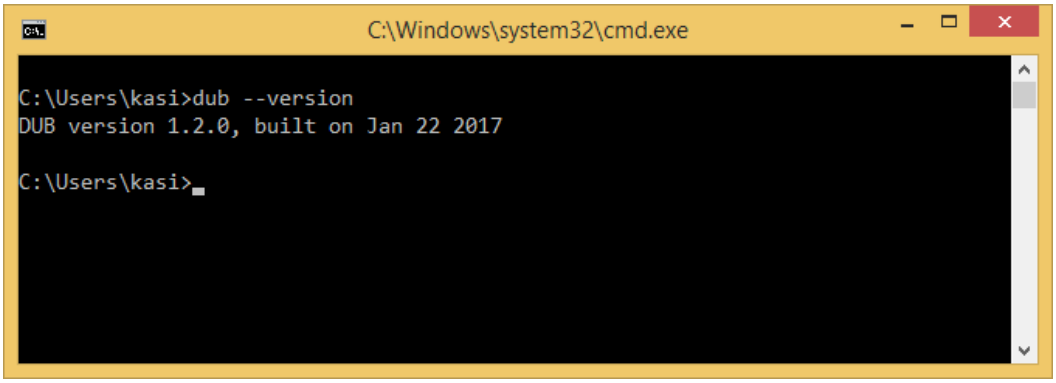
Kuva 6. Ace API:n rakenne (Cloud9 n.d.).

5.2 Kehitysympäristön asentaminen

Valitsin editoriksi Visual Studio Coden, mutta vastaavat lisäosat ovat saatavilla myös Atomiin. Asennusohjeet ovat Windows 8.1 -virtuaalikoneessa testattuja. Linux-järjestelmässä tarvittavat paketit löytyvät suoraan pakетinhallinnasta.

5.2.1 D-kääntäjän asennus

Projektissa käytetään Digital Marsin DMD-kääntäjää. Ladataan ja asennetaan se osoitteesta https://dlang.org/download.html#dmd. Käännösprosessin ja riippuvuuksien hallintaan käytetään DUB-työkalua, joten asennetaan se seuraavaksi osoitteesta https://code.dlang.org/download. DUBin toimintaa voi testata komentoriviltä (Kuva 7).



Kuva 7. Toimiva DUB-asennus.

5.2.2 Visual Studio Code

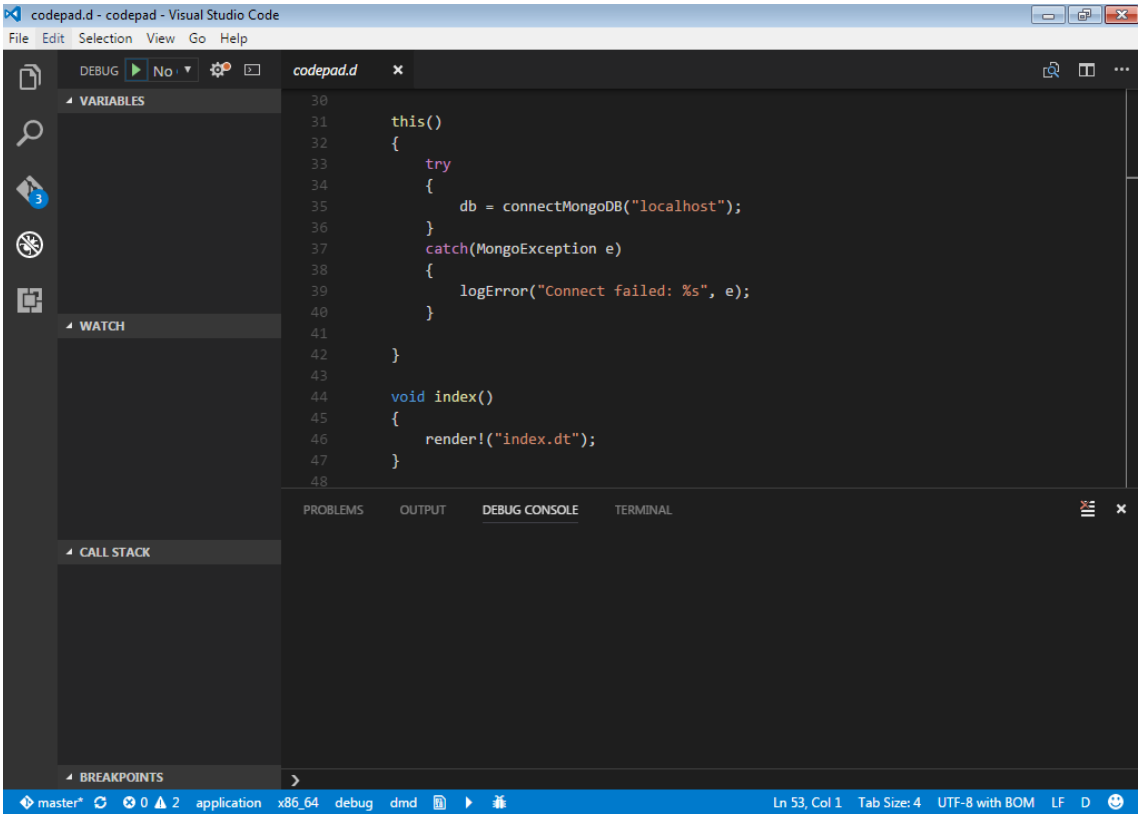
Visual Studio Code on Microsoftin kehittämä koodieditori Windows-, Linux- ja Mac OS -alustoille. Editoriin voi lisätä ominaisuuksia asennettavien lisäosien avulla ja esimerkiksi tuki D-kielelle on toteutettu lisäosana.

Aloitetaan lataamalla ja asentamalla asennuspaketti osoitteesta https://code.visualstudio.com/Download. Tämän jälkeen voidaan asentaa ”D Programming Language (code-d)”, ”Native Debug” ja ”Diet Templates” -lisäosat Visual Studio Coden paketinhallinnasta. Editorin uudelleenkäynnistyksen jälkeen code-d asentaa workspace-d:n ja sen tarvitsemat ohjelmat. Workspace-d on taustalla toimiva palvelin, joka yhdistää seuraavat työkalut yhdeksi DUBin hallinnoimaksi ympäristöksi jota lisäosat voivat käyttää komentoviestien avulla:

- DCD - automaattisen täydennyksen palvelin
- Dfmt - koodin formatoija
- D Scanner - koodin analysoija

Onnistuneen asennuksen jälkeen D-lähdekoodia muokattaessa ikkunan alalaidassa näkyy valittu käännöskonfiguraatio ja painikkeet ohjelman

kääntämiseen, ajamiseen ja virheenkorjaukseen (Kuva 8). Jos lähdekoodi käyttää versionhallintaa sen tila näkyy myös.



Kuva 8. Visual Studio Code.

5.2.3 MongoDB

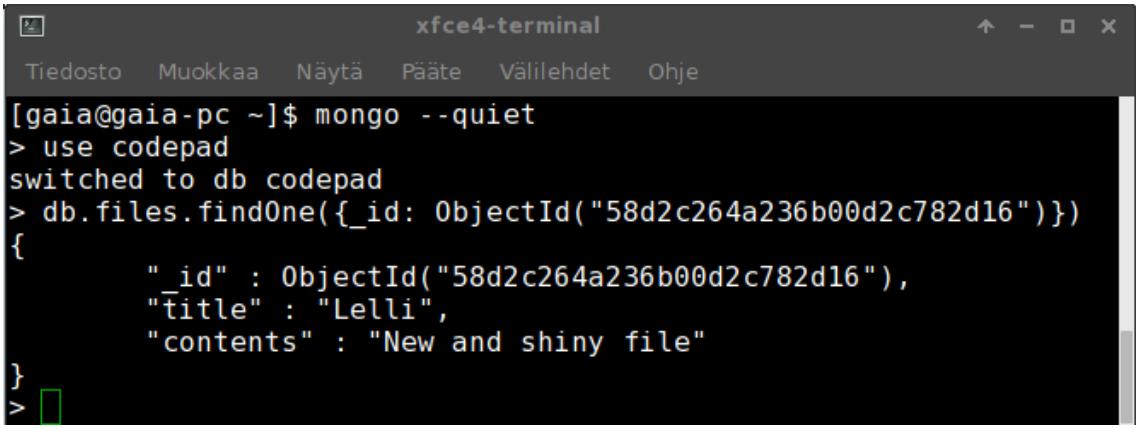
Asennetaan seuraavaksi MongoDB NoSQL-tietokanta. Asennuspaketin voi ladata osoitteesta https://www.mongodb.com/download-center. Asennuksen jälkeen MongoDB ei vielä suoraan toimi, vaan sille täytyy luoda asetustiedosto jossa määritellään datakansion sijainti.

```
#mongod.cfg
systemLog:
    destination: file
    path: c:\data\log\mongod.log
storage:
    dbPath: c:\data\log
```

Tämän jälkeen palvelin voidaan käynnistää ja asettaa käyttämään uutta asetustiedostoa.

```
mongod.exe –f mongod.cfg
```

Palvelimen toimintaa voi tarvittaessa testata mukana tulevan hallintakonsolin avulla (Kuva 9).



Kuva 9. MongoDB shell.

5.3 Uuden projektin luominen

Luodaan uusi DUB-projekti käyttäen vibe.d-mallipohjaa, jolloin projektiin tulee Vibe.d:n tarvitsemat riippuvuudet ja kansiorakenne (Kuva 10). Uusi projekti sisältää minimaalisen toimivan Vibe.d-sovelluksen. Vibe.d:n toimivuuden voi tässä vaiheessa varmistaa ”dub run”-komennolla, joka kääntää ja käynnistää sovelluksen.

Kuva 10. Uusi projekti.

Projektin kansiorakenne on seuraavanlainen:

```
              .gitignore
              dub.json
<DIR>         public
<DIR>         source
<DIR>         views
```

Public-kansioon tallennetaan staattinen data kuten kuvat, CSS ja JavaScript-koodi.
Source-kansiossa on D-lähdekoodi.
Views-kansioon tallennetaan näytettävät Diet-sivupohjat.

dub.json on DUBin asetustiedosto, joka voi olla SDLang tai JSON muotoinen. Käytännössä kannattaa aina valita JSON, sillä jotkin ohjelmat eivät tue kuin JSON-projekteja.

5.3.1 dub.json

Uuden projektin dub.json pitäisi olla seuraavanlainen:

```
{
    "name": "codepad",
    "authors": [
        "win8vm"
    ],
    "dependencies": {
        "vibe-d": "~>0.7.30"
    },
    "description": "Text editor.",
    "copyright": "Copyright © 2016, win8vm",
    "license": "proprietary"
}
```

Vibe-sovelluksen ominaisuuksia voi määritellä käännösaikaisesti versioiden avulla. Otetaan käyttöön Vibe.d:n määrittelemä main-funktio ja selkeämpi JSON-viestien tulostus.

```
"versions": [
    "VibeDefaultMain",
    "VibeJsonFieldNames",
    "JsonLineNumbers"
],
```

Oletuksena DUB etsii **source**-kansiosta D-moduuleja ja **views**-kansiosta muita tiedostoja. Polkuja voi lisätä tarpeen mukaan importPaths- ja stringImportPaths-asetusten avulla.

```
"stringImportPaths": [
    "views",
    "languages"
]
```

Projektille voi määritellä useita konfiguraatioita, jotka voi rajata esimerkiksi alustan mukaan. Esimerkkiprojekti toimii ilman erityisiä asetuksia Windows- ja Linux-alustoilla, mutta rajauksen voisi tehdä platforms-kentän avulla.

```
"configurations": [
    {
      "name": "codepad-win",
      "targetName": "CodePad",
      "targetType": "executable",
      "targetPath": "bin",
      "platforms": ["windows"]
    },
]
```

5.4 Vibe.d-palvelin

Vibe.d:ssä reitit on mahdollista lisätä URLRouter-luokkaa käyttämällä käsin, mutta tässä web-sovelluksessa käytetään Viben web-käyttöliittymägeneraattoria joka luo URL-reitit luokan pohjalta. Dokumenttien tallentamiseen käytetään sisäänrakennettua MongoDB-ajuria.

Uudessa projektissa source/app.d-tiedostossa on runko valmiina. this() toimii ohjelman alkukohtana main-funktion sijaan, koska VibeDefaultMain määriteltiin DUB-asetustiedostossa. Ohjelman alussa on version-lohko, jossa lisätään tuki muistivirheiden havaitsemiseen Linuxilla. **version**-avainsanalla voi D:ssä määritellä ehdollisen käännöksen tietylle alustalle, tai tietyn linkitystyypin.

```
version (linux)
{
    import etc.linux.memoryerror;
    static if (is(typeof(registerMemoryErrorHandler)))
        registerMemoryErrorHandler();
}
```

5.4.1 Web-rajapinta

Viben funktio web.web.registerWebInterface luo uuden web-rajapinnan annetun luokan pohjalta. Luokan julkiset funktiot reititetään URL-reiteiksi. Reitin tyyppi määräytyy funktion nimen ja parametrien perusteella. Reitille voi myös määritellä tietoja attribuuttien avulla. Esimerkkiprojektin kaikki reitit on määritelty **CodepadService**-luokassa. Reiteille on mahdollista antaa valinnaiset HTTPServerRequest- ja HTTPServerResponse-parametrit, joiden kautta HTTP-pyynnön tilaan pääsee suoraan käsiksi.

CodepadService sisältää listan käyttäjistä ja avoimista tiedostoista koko palvelimella, sekä vierailijan istuntokohtaiset muuttujat joihin säilötään avatun tiedoston tunnus ja käyttäjän nimi. Palvelinta avatessa yhdistetään MongoDB-palvelimeen. Jos yhdistäminen syystä tai toisesta epäonnistuu sovellus näyttää virheen.

```
class CodepadService
{
    EditorFile[string] openFiles;
    WebSocket[] clients;
    MongoClient db;

    private
    {
        SessionVar!(string, "cur_file") curFile;
        SessionVar!(string, "nick_name") nickname;
    }

    this()
    {
```

```
        try
        {
            db = connectMongoDB("localhost");
        }
        catch(MongoException e)
        {
            logError("Connect failed: %s", e);
        }
    }

    void index(string _error = null)
    {
        terminateSession();
        render!("index.dt");
    }

    void post()
    {
        redirect("/");
    }
```

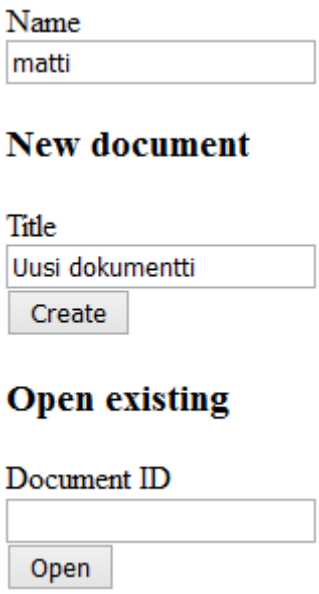
Uusi web-rajapinta luodaan **app.d**-tiedostossa ohjelmaa käynnistäessä, jonka jälkeen palvelin alkaa kuunnella yhteyksiä. Jos sovelluksella olisi käytössä REST-rajapinta, se rekisteröitäisiin registerRestInterface-funktion avulla. Istunnot tallennetaan esimerkkisovelluksessa muistiin käyttäen MemorySessionStorea, mutta Redis-sovelluksessa istunnot voisi säilöä myös RedisSessionStoreen.

```
auto router = new URLRouter;
router.get("*", serveStaticFiles("public/"));
router.registerWebInterface(new CodepadService);
auto settings = new HTTPServerSettings;
settings.port = 8080;
settings.bindAddresses = ["0.0.0.0"];
settings.sessionStore = new MemorySessionStore;
listenHTTP(settings, router);
```

HTTPServerSettings-luokan avulla voi säätää Vibe.d:n sisäisen HTTP-palvelimen asetuksia ja esimerkiksi ottaa HTTPS-tuki käyttöön, jos palvelimelle on luotu SSL-sertifikaatti. Staattiset tiedostot kuten kuvat asetetaan Vibe.d:n jaettavaksi **public**-kansiosta.

5.5 Etusivu

Etusivulla käyttäjä valitsee itselleen lomakkeella (Kuva 11) nimen ja joko luo uuden huoneen tai liittyy olemassa olevaan huoneeseen sen huonetunnuksella. Ennen lomakkeen varsinaista lähettämistä tehdään AJAX-pyyntö, joka asettaa käyttäjän nimen palvelimella istuntomuuttujaan ja tarvittaessa vaihtaa nimeä jos huoneessa on jo käyttäjä samalla nimellä. Tämän jälkeen lomake lähetetään ja käyttäjä liittyy huoneeseen.



Kuva 11. Etusivu.

Lomake käsitellään CodepadServicen **postNewDocument**-funktiossa, jos käyttäjä luo uuden tiedoston. Reitille on määritelty @path-attribuutti, jonka tehtävä on korvata reitin URL käyttäjän määrittelemällä polulla. Reitin tyyppi on POST, jonka registerWebInterface päätteli sen nimestä.

```
@path("/document/new")
void postNewDocument(string title)
{
    EditorFile f;
    f.id = BsonObjectID.generate();
    f.title = title;
    f.contents = "New and shiny file";

    try
    {
            db.getCollection("codepad.files").insert(f);
            curFile = f.id.toString;
            openFiles[curFile] = OpenFile(f, []);
```

```
        }
        catch(MongoException e)
        {
                logError("Insert failed");
                redirect("/");
        }

        redirect("/document/" ~ to!string(f.id));
}
```

Uusi tiedosto tehdään luomalla EditorFile-objekti generoidulla BSON-id:llä ja käyttäjän antamalla otsikolla. Generoitu id toimii huonetunnuksena, jota voi käyttää aloitussivulla huoneeseen liittyessä. Tämän jälkeen tiedosto lisätään MongoDB-kokoelmaan ja lisätään avointen tiedostojen listaan. Lopuksi käyttäjä ohjataan luotuun huoneeseen.

5.6 Huone

Uuden huoneen luomisen tai huoneeseen liittymisen jälkeen käyttäjä ohjataan etusivulta huonesivulla. Reitti käsitellään CodepadServicen getDocument-funktiossa.

```
@path("/document/:id")
void getDocument(string _id)
{
    Nullable!EditorFile file;
    if (_id !in openFiles)
    {
        file = db.getCollection("codepad.files")
               .findOne!EditorFile(
                   ["_id": BsonObjectID.fromString(_id)]
               );

        openFiles[_id] = OpenFile(file, []);

        if (file.isNull)
        {
            logError("not found: " ~ _id);
            curFile = "";
            redirect("/");
        }
    }
    ...
}
```

Tiedostoa yritetään hakea MongoDB:stä EditorFile-rakenteeseen, jos se ei vielä ole avoinna. Nullable(T) antaa arvolle erillisen tilan, joka kertoo onko muuttujalle asetettu arvoa. Tila on aluksi null ja vaihtuu kun arvo asete-

taan. Tiedoston löytyessä käyttäjän nimi tarkistetaan ja tarvittaessa vaihdetaan, jos huoneessa on jo samaa nimeä käyttävä toinen editoija. Tämän jälkeen käyttäjä lisätään huoneen käyttäjälistaan ja hänelle näytetään editview.dt-sivupohja.

Tarkastellaan seuraavaksi WebSocket-reittiä, jonka kautta editori kommunikoi palvelimen kanssa. Viben web-rajapintageneraattori päivittää HTTP-yhteyden käyttämään WebSocketia automaattisesti, jos reitille annetaan WebSocket parametriksi.

```
void getWS(scope WebSocket socket)
```

Uuden WebSocket-yhteyden saapuessa palvelimelle istuntomuuttujat asetetaan paikallisiin muuttujiin ja yhteys lisätään taulukkoon. Tämän jälkeen muille huoneessa oleville lähetetään tieto uudesta käyttäjästä:

```
Json joinmsg = Json.emptyObject;
joinmsg["cmd"] = 5;
joinmsg["data"] = serializeToJsonString(openFiles[_curF-
ile].users);
joinmsg["user"] = nickname.value;
sendTextToClients(joinmsg.toString);
```

Seuraavaksi funktio odottaa WebSocketilta saapuvaa dataa. Aina kun viestejä on saatavilla, viestin tunniste luetaan ja viesti välitetään muille samassa huoneessa oleville, paitsi jos kyseessä on tiedostopyyntö.

```
string recv = socket.receiveText;

JSONValue msg = parseJSON(recv);
long cmdId = msg["cmd"].integer;
if (cmdId != 3)
    sendTextToClients(recv);

switch (cmdId)
{
   default:
      break;
}
```

Yhteyden sulkeutuessa lähetetään viesti muille käyttäjille. Tämän jälkeen käyttäjä poistetaan taulukosta ja WebSocket-yhteys suljetaan. Vibe.d:n mukana tulevat funktiot **removeFromArray** ja **removeFromArrayIdx** ovat kätevä tapa poistaa elementtejä taulukoista.

```
clients.removeFromArray!WebSocket(socket);
socket.close();
```

5.6.1 Editoriviestit

Palvelimelle saapuvat viestit ovat JSON-muodossa. Valitsin sen, koska JavaScriptissä on hyvä tuki ja JSON on laajalti käytössä. Viestillä on data-

kenttä, jonka sisällä parametrit ovat. Kaikissa viesteissä on myös numeerinen cmd-kenttä, joka määrittää viestin tyypin. Viestit parametreineen on lueteltu alla (Taulukko 2).

Taulukko 2. Editoriviestit.

cmd	**viesti**	**parametrit**
1	muokkausoperaatio	index, delta
2	valintaoperaatio	ranges
3	tiedostopyyntö	file id
4	tiedosto	title, contents
5	uusi käyttäjä	users
6	yhteyden katkaisu	

Muokkausoperaatio on viesti, joka lähetetään kun käyttäjä muokkaa editorissa tekstiä. Viesti lähetetään aina kun editorin onTextChange-tapahtumankäsittelijää kutsutaan. Viestissä lähetettävä delta sisältää muuttuneet rivit ja operaation tyypin, joka voi olla lisäys tai poisto. Index on ensimmäisen muokattavan merkin indeksi dokumentissa, ja se haetaan Acen Document-luokan getPositionToIndex-funktiolla, joka muuttaa annetun {rivi, sarake}-sijainnin lineaariseksi indeksiksi. Viesti lähetetään vain kerran leikepöytää käytettäessä tai poistaessa useita merkkejä.

Palvelin lukee JSON-objektin muuttujiin ja päivittää EditorFilen sisällön muokkausoperaation tyypistä riippuen. Muutokset tekstiin tehdään Phoboksen **replaceInPlace**-funktion avulla. Se on funktio, joka korvaa annettujen indeksien välillä olevat elementit ja tarvittaessa samalla kasvattaa tai pienentää taulukon kokoa. Funktio toimii, koska merkkijonot ovat taulukkoja D:ssä.

```
replaceInPlace(
     openFiles[_curFile].file.contents,
     toUTFindex(openFiles[_curFile].file.contents,index),
     openFiles[_curFile].file.contents.toUTFindex(index),
     lines
);
db.getCollection("codepad.files").update(
  Bson(["_id": Bson(openFiles[_curFile].file.id)]),
  openFiles[_curFile].file
```

```
);
```

Valintaoperaatio-viesti lähetetään selaimesta kun tekstin valinta muuttuu ja onChangeSelection-tapahtumankäsittelijää kutsutaan. Data sisältää valinnan alku- ja loppusijainnit. Sijainteja voi myös olla useita, jos teksti valittiin Ctrl-näppäin pohjassa.

Selain lähettää tiedostopyyntö-viestin huonetta ladatessa käyttäen Dietmallille syötettyä tiedoston tunnusta. Vastauksena palvelin lähettää tiedoston sisällön.

Uuden käyttäjän yhdistäessä huoneeseen käyttäjä lisätään huoneen käyttäjälistaan, ja päivitetty lista lähetetään kaikille huoneessa oleville käyttäjille. Vastaavasti käyttäjän poistuessa lähetetään ensin viesti muille ja poistetaan tämän jälkeen käyttäjä listasta.

Viestien lähettäminen muille käyttäjille tapahtuu **sendTextToClients**-funktiossa. Ennen lähetystä tarkistetaan onko WebSocket yhdistettynä, sillä yhteys voi katketa koska tahansa.

5.6.2 Editorin alustaminen

Editorin toiminnallisuus on selaimen puolella editFuncs.js-tiedostossa, joka ladataan editview.dt-sivulla samalla kun Ace alustetaan. Editori alustetaan init-funktiossa, jolle annetaan parametreina sivupohjalle annetut muuttujat.

```
script(src="/src-noconflict/ace.js", type="text/javas-
cript", charset="utf-8")
script(src="/editFuncs.js", type="text/javascript", char-
set="utf-8")
script.
  var editor = ace.edit("editor");
  editor.setTheme("ace/theme/monokai");
  editor.getSession().setMode("ace/mode/javascript");
  init("#{_id}", "#{_nickname}");
```

editFuncs.js esittelee aluksi tarvittavia muuttujia. Tiedoston alku näyttää seuraavalta:

```
var aceRange = ace.require('ace/range').Range;

var marker = {};
var highlightMarkers = [];
var socket = undefined;

var updatingDelta = false;
var updatingHighlight = false;

var username = "";
var openFileId = undefined;
```

```
function init(initialFileId, usern) {
  openFileId = initialFileId;
  editor.$blockScrolling = Infinity;
```

Muuttujien updatingDelta ja updatingHighlight tarkoitus on estää uuden muutoksen tekeminen Acen tekstikenttään ennen vanhan operaation valmistumista.

Initin alussa luodaan **marker**-objekti, jota tarvitaan dynaamisten kursorien käyttämiseen. Marker on JavaScript-objekti, jossa on lista muiden editoijien sijaintia osoittavista kursoreista. Siinä on myös update-funktio, joka päivittää kursorit.

Tämän jälkeen initissä asetetaan tapahtumankäsittelijät editorille. Käsittelijöitä kutsutaan aina kun tekstikenttä tai valinta muuttuu.

```
editor.getSession().selection.on("changeSelection",
onChangeSelection);
editor.getSession().on("change", onTextChange);
```

Lopuksi initissä asetetaan käyttäjän nimi ja luodaan uusi WebSocket, joka lähettää palvelimelle tiedostopyynnön. Pyyntö lähetetään heti kun yhteys on auki käyttäen dokumentin BSON-tunnusta.

```
socket = new WebSocket("ws://" + serverUrl + "/ws");

socket.onopen = function() {
    console.log("Connection opened");
    socket.send("{\"cmd\": 3, \"id\": \""+openFileId+"\"}");
}

socket.onclose = function(event) {
    console.log("Disconnected (" + event.code + "): " +
event.reason);
}

socket.onerror = function() {
    console.log("WebSocket error");
}

socket.onmessage = handleSocketMessage;
```

Nyt editori on alustettu käyttövalmiiksi. Selain käsittelee palvelimelta tulevat WebSocket-viestit **handleSocketMessage**-funktiossa ja päivittää näkymän tarvittaessa.

5.6.3 Kursorien hallinta

Muiden käyttäjien sijainnin ja valintojen näyttäminen tapahtuu kahdenlaisilla kursoreilla. Tavalliset kursorit ovat ”edessä”, ja niiden uudelleenpiirto tehdään laukaisemalla Acen changeFrontMarker-signaali. Vastaavasti valintakursorit ovat ”takana”, ja ne käyttävät changeBackMarker-signaalia.

Muiden käyttäjien kursorit lisätään dynaamisina kursoreina, jolloin niiden sijaintia voi muokata nopeasti eikä kursoria tarvitse poistaa ja lisätä uudelleen jokaisen muutoksen jälkeen. Kursorit lisätään Acen addDynamicMarker-funktiolla, joka ottaa aiemmin luodun marker-objektin parametrina. Uusi kursori kutsuu markerin update-funktiota signaalin lauetessa. Dynaamisen kursorin sijaintia muokataan cursors-taulukon avulla, kun palvelimelta tulee muokkausoperaatio-viesti.

```
function insertUserCursor(row,col,name) {
    if (name == undefined) {
        console.log("Cannot add cursor - undefined name");
        return;
    }
    marker.session.addDynamicMarker(marker, true);
    marker.cursors.push({row: row, column: col, name:
name});
    marker.redraw();
}

function removeUserCursor(name) {
    var index = marker.cursors.map(function(e) {
        return e.name;
    }).indexOf(name);
    if (index == -1) {
        console.log("Cursor not found - " + name);
        return;
    }
    marker.session.removeMarker(index);
    marker.cursors.splice(index, 1);
    marker.redraw();
}
```

Valintakursorit näyttävät yhden tai useamman alueen, joka on maalattuna. Niiden lisääminen ja poisto tapahtuu Acen EditSession-luokasta löytyvien **addMarker**- ja **removeMarker**-funktioiden avulla. Valintojen CSS-tyyli on ace_selection, jolloin ne näkyvät kuten paikallisesti tehdyt valinnat. Jokaisella kursorilla on tunniste, jota käyttämällä kursori poistetaan.

```
function addHighlightRange(user, r) {
  var id = marker.session.addMarker(new aceRange(
                                        r.start.row
                                        r.start.column,
                                        r.end.row,
                                        r.end.column),
                                    "ace_selection",
                                    "text");
    highlightMarkers.push({"id": id, "user": user});

    return id;
}

function removeHighlightRange(markerId) {
    marker.session.removeMarker(markerId);
    highlightMarkers = $.grep(highlightMarkers, function(e)
{
        return e.id !== markerId;
    });
}

function removeUserHighlightRanges(user) {
    var selections = $.grep(highlightMarkers, function(e) {
        updatingHighlight = false;
        return e.user == user;
    });
    $.each(selections,function(k,v) {
        removeHighlightRange(v.id);
    });
}
```

Alla oleva kuva editorinäkymästä (Kuva 12) havainnollistaa kuinka valinnat ja kursorit näkyvät muille käyttäjille. Paikallisen käyttäjän kursori on harmaa ja vilkkuu jos editori on fokusoituna, kun taas muiden kursorit ovat värillisiä eivätkä vilku.

Kuva 12. Editorinäkymä.

6 YHTEENVETO

Kävin opinnäytetyössä aluksi läpi D-kieltä ja siihen liittyviä kehitystyökaluja kokonaisen luvun verran. Uskon, että D ei ole kaikille tuttu ja siksi tarvitsi tarkempaa esittelyä. Työn seuraavassa luvussa esittelin Vibe.d-kirjaston ja vertailin sitä muihin, kilpaileviin kirjastoihin. Neljännessä luvussa esittelin yleisiä tekniikoita, joita web-sovelluksissa käytetään. Työn viimeisessä luvussa käytiin läpi esimerkkiprojektin rakennetta.

Toteutin opinnäytetyötä varten esimerkkiprojektina selaimessa toimivan monen käyttäjän tekstieditorin. Aloitin projektin lisäämällä tyhjän sivupohjan ja tekstieditorikomponentin, jonka ympärille selaintoiminnallisuus rakentui. Palvelimen puolella tein web-rajapinnan ja lisäsin siihen kehityksen aikana reittejä aina tarvittaessa.

Esimerkkiprojektin lopputuloksena syntyi toimiva editori. Editorin toiminnallisuus on tällä hetkellä kohtalaisen suppea, mutta jatkokehityksen ei pitäisi olla hankalaa. Projektia tehdessä DUB-työkalun palvelin oli alhaalla pari viikkoa, jolloin työkalu ei toiminut. Käytin työtä tehdessä Git-versionhallintajärjestelmää.

Sain opinnäytetyötä tehdessäni käsityksen myös siitä, kuinka hyvin D soveltuu web-kehitykseen. Suosittuihin web-sovelluskehyksiin verrattuna Vibe.d:lle on paljon vähemmän dokumentaatiota tarjolla, mutta sama koskee muitakin D-kirjastoja. Käyttökokemus ei kaikilta osin tunnu yhtä viimeistellyltä kuin esimerkiksi Djangossa. Mielestäni Vibe.d soveltuu erityisesti REST-rajapintojen luomiseen tai tilanteisiin joissa tarvitaan integrointia työpöytäsovellukseen. Opinnäytetyön tekemisen aikana opin lisää käytetyistä tekniikoista ja työn toteuttaminen oli mielenkiintoista.

Lähteet

Alexandrescu, A. (2011). *D1 to be discontinued on December 31, 2012*. Haettu osoitteesta http://forum.dlang.org/post/jc0ic5$18bv$2@digitalmars.com

BlogGeek.me (2014). Defining AOT, JIT and Interpreted. Haettu 1.4.2017 osoitteesta https://bloggeek.me/aot-jit-interpreted/

Bright, W. (2014). *How I Came to Write D*. Haettu osoitteesta http://www.drdobbs.com/architecture-and-design/how-i-came-to-write-d/240165322

CaitieM (2013). A WebSocket Primer. Haettu 6.3.2017 osoitteesta https://caitiem.com/2013/12/02/a-websocket-primer/

caniuse.com (2017). Web Sockets. Haettu 20.3.2017 osoitteesta http://caniuse.com/#feat=websockets

Çehreli, A. (2016). Programming in D. Haettu 31.10.2016 osoitteesta http://ddili.org/ders/d.en/Programming_in_D.pdf

Cloud9 (n.d.). Ace API Reference. Haettu 10.2.2017 osoitteesta https://ace.c9.io/#nav=api

D Language Foundation. (n.d.). *Interfacing to C*. Haettu osoitteesta http://dlang.org/spec/interfaceToC.html

DB-Engines (2017). DB-Engines Ranking of Key-value Stores. Haettu 9.3.2017 osoitteesta http://db-engines.com/en/ranking/key-value+store

DConf2016 (2016). Quantum Break: AAA Gaming With Some D Code – Ethan Watson | DConf2016. Haettu 23.3.2017 osoitteesta https://www.youtube.com/watch?v=OLFBal4Qo_k

DrDobbs (2013). Facebook Adopts D Language. Haettu 23.3.2017 osoitteesta http://www.drdobbs.com/mobile/facebook-adopts-d-language/240162694

europa.eu (2016). Cookies. Haettu 2.4.2017 osoitteesta http://ec.europa.eu/ipg/basics/legal/cookies/index_en.htm

HotFrameworks (2017). Web framework rankings. Haettu 1.4.2017 osoitteesta https://hotframeworks.com/#rankings

rejectedsoftware e.K (n.d.). Haettu 2.11.2016 osoitteesta http://vibed.org/about

TechNet (2003). How TCP/IP Works. Haettu 23.3.2017 osoitteesta https://technet.microsoft.com/en-us/library/cc786128(v=ws.10).aspx

TIOBE (2017). TIOBE Index for February 2017. Haettu 28.2.2017 osoitteesta http://www.tiobe.com/tiobe-index/

Vibed.org (2017). Diet Templates. Haettu 20.3.2017 osoitteesta http://vibed.org/templates/diet

Visolve (2006). Implementing Reverse Proxy Using Squid. Haettu 30.3.2017 osoitteesta http://www.visolve.com/uploads/resources/reverseproxy.pdf

W3Techs (2017). Historical yearly trends in the usage of server-side programming languages for websites. Haettu 2.3.2017 osoitteesta https://w3techs.com/technologies/history_overview/programming_language/ms/y