

Tuomas Henriksson

# WEB-KEHYSTEN VERTAILUA

Informaatioteknologian ja viestinnän tiedekunta  
Kandidaattitutkielma  
Toukokuu 2020

# TIIVISTELMÄ

Tuomas Henriksson: Web-kehysten vertailua  
Kandidaattitutkielma  
Tampereen yliopisto  
Tietojenkäsittelytieteiden tutkinto-ohjelma  
Toukokuu 2020

---

Web-kehitys muodostaa suuren osan nykypäivän ohjelmistokehityksestä. Web-kehukset ovat yleisesti käytettyjä ohjelmistokehyyksiä, joita käytetään web-sovellusten kehittämisen helpottamiseksi ja ohjelmistojen laadun parantamiseksi. Tässä tutkielmassa keskitytään web-kehysten ominaisuuksien vertailuun ja pyritään selvittämään millaisiin käyttötarkoituksiin eri web-kehukset soveltuvat ja millainen on niiden suosio kehittäjien keskuudessa.

Tutkielman alussa käydään läpi aiempaa tutkimusta aiheesta sekä määritellään kriteerit web-kehysten vertailulle. Vertailuosiossa otetaan tarkasteluun kolme web-kehystä, jotka ovat Express.js, Django ja Ruby on Rails. Kehyksiä arvioitiin sen mukaan, miten hyvin niissä on toteutettu yleisimmät MVC-arkkitehtuurin (model-view-controller) vaatimat ominaisuudet, erityisesti tietokantatuki ja tiedon ohjelmistoon sitomisen helppous. Lisäksi tarkasteltiin web-kehysten käyttäjämääriä ja kehityksen aktiivisuutta käyttäen lähteenä näiden verkkosivustoja ja keskustelupalstoja.

Vertailu osoittaa, että vertailut web-kehukset ovat perusominaisuuksiltaan hyvin samankaltaisia. Web-kehukset eroavat perusarkkitehtuuriltaan ja ohjelmointityyliltään. Express.js eroaa minimalistisen tyylinsä vuoksi muista vertailuista, minimalistisuus mahdollistaa ohjelmiston räätälöinnin käyttötarkoituksen mukaan. Express.js:n etuna voidaan pitää sitä, että se pohjautuu Node.js-ympäristöön ja siten saatavilla on suuri määrä lisäosia ja laaja kehittäjäyhteisö tukena. Django ja Ruby on Rails sisältävät kattavan määrän ominaisuuksia ja riittävät useimpiin käyttötarkoituksiin sellaisenaan, molemmat sopivat erityisesti tietokantapohjaisiin web-sovelluksiin sillä tietokantojen käyttö ja tietojen sitominen ohjelmistoon on tehty helpoksi. Vaikka useimmat kehukset mahdollistavat melko vapaan ohjelmointityylin, Ruby on Rails pyrkii monin tavoin ohjaamaan käyttäjää yhteen tyyliin, joka voi olla rajoittavaa. Django ja Express.js kasvattavat molemmat suosiotaan, kun taas Ruby on Railsin suosio on ollut laskussa, tämä kannattaa ottaa huomioon uusien projekteja aloittaessa.

Avainsanat: Web-kehukset, Express.js, Django, Ruby on Rails

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck –ohjelmalla.

## Sisällysluettelo

|          |  |          |
|----------|--|----------|
| <b>1</b> | <b>Johdanto</b> .....                    | <b>1</b> |
| <b>2</b> | <b>Web-ohjelmistokehitys</b> .....       | <b>1</b> |
| <b>3</b> | <b>Kehykset</b> .....                    | <b>2</b> |
| <b>4</b> | <b>Web-kehysten laatukriteerit</b> ..... | <b>3</b> |
| <b>5</b> | <b>Vertailu</b> .....                    | <b>4</b> |
|          | 5.1 Express.js                           | 5        |
|          | 5.2 Django                               | 6        |
|          | 5.3 Ruby on Rails                        | 7        |
|          | 5.4 Yhteenveto                           | 8        |
| <b>6</b> | <b>Tulokset</b> .....                    | <b>8</b> |
|          | <b>Lähdeluettelo</b> .....               | <b>9</b> |

## 1 Johdanto

Web-kehitys on tällä hetkellä merkittävimpiä ohjelmistoalan osa-alueita, vaatimukset tarvittavan teknologian ja tietotaidon suhteen vaihtelevat merkittävästi projektien välillä. Erilaisia työkaluja ja teknologioita on runsaasti, ja niiden soveltuvuus erilaisiin tehtäviin vaihtelee. Oikeilla valinnoilla voidaan vaikuttaa projektien onnistumiseen, kustannuksiin ja nopeuteen.

*Web-kehukset* (web framework, web application framework) ovat ohjelmistokehyyksiä, joiden tarkoitus on helpottaa verkkopalveluiden toteuttamista. Web-kehys voidaan määritellä joukoksi ohjelmistokomponentteja, jotka helpottavat web-pohjaisten ohjelmistojen kehitystä ja toimintaa (Vosloo & Kourie 2008).

Kehyksiä on toteutettu useilla eri kielillä ja käyttäen erilaisia lähestymistapoja. Tässä tutkielmassa keskitytään palvelinpuolella toimiviin yleiskäyttöisiin kehyksiin. Työn tarkoituksena on kehysten vertailu ja sen selvittäminen millaisiin käyttötarkoituksiin eri vaihtoehdot sopivat. Vertailu tehdään tutkimalla kehysten teknisiä ominaisuuksia sekä laatua ja käytettävyyttä.

Lähempään tarkasteluun on valittu kolme kehystä Express.js, Django ja Ruby on Rails. Näistä Express.js käyttää ohjelmointikielenä JavaScriptiä ja Django Python-ohjelmointikieltä.

Toinen luku esittelee web-kehitystä yleisesti. Kolmannessa luvussa kerrotaan ohjelmistokehyyksistä ja web-kehyyksistä. Neljännessä luvussa esitellään erilaisia kriteereitä kehysten arviointiin ja tarkastellaan aiempia tutkimuksia, seuraavassa käydään läpi itse vertailu. Viimeisessä luvussa on vertailusta tehtävät johtopäätökset.

## 2 Web-ohjelmistokehitys

Tiedonsiirron pohjana www:ssä on *http* eli Hypertext Transfer Protocol. Web-sovellusten toiminta perustuu *http-pyyntöihin* (request) ja *vastauksiin* (response) *palvelimen* (server) ja *asiakasohjelman* (client) välillä. Sovelluksen tarvitsemat tiedostot sijaitsevat palvelimella, asiakasohjelman toimii useimmiten selain. Erilaisten resurssien löytämiseen www:stä käytetään URL-osoitteita (Fielding & Reschke, 2014).

Ohjelmakoodia voidaan ajaa sekä palvelimen että asiakasohjelman puolella. Palvelimella suoritetaan yleensä raskaita ja esimerkiksi turvallisuuden liittyviä tehtäviä, jotka halutaan piilottaa käyttäjiltä. Monet sivustot hyödyntävät myös tietokantoja, joiden käsittely ja varastointi kuuluu myös palvelimelle. Asiakasohjelma huolehtii tavallisesti käyttöliittymän esittämisestä ja syötteiden välittämisestä palvelimelle.

Alun perin web-sivustot ovat olleet staattisia. Yksinkertaiset web-sivustot koostuvat erillisistä sivuista, jotka ladataan kerralla palvelimelta ja kommunikaatiota palvelimen ja

asiakasohjelman välillä tapahtuu vain uutta sivua ladatessa. Toteutuksesta riippuen palvelin voi lähettää valmiit tiedostot suoraan asiakkaalle tai palvelinkoodi voi koostaa sivut pyynnön saatuaan.

Nykyaikaiset web-sovellukset hyödyntävät usein vuorovaikutteisuutta, sivut eivät ole staattisia vaan pyyntöjä voidaan lähettää palvelimille myös sivun sisältä ja sivun sisältöä muokata ilman koko sivun päivittämistä.

Web-sovellusten ulkoasun ja käyttöliittymän toteutukseen käytetään HTML-kuvauskieltä ja ulkoasun määrittelyyn useimmiten CSS-tyylejä. Interaktiivisuutta web-sivustoihin saadaan JavaScript kielellä, joka on selaimessa toimiva tulkittu skriptikieli.

### **3 Kehykset**

Ohjelmistokehysten tarkoitus on helpottaa ohjelmistojen kehittämistä. Ohjelmistokehyksillä voidaan nopeuttaa työtä, laskea kustannuksia ja parantaa ohjelmistojen laatua. Fayad & Schmidt (1997) määrittelee ohjelmistokehyksen uudelleenkäytettäväksi puolivalmiiksi ohjelmaksi.

Ohjelmistokehykset helpottavat kehitystyötä tarjoamalla uudelleenkäytettävyyttä ja vähentämällä tarvittavaa matalan tason työtä, jolloin voidaan keskittyä itse varsinaiseen ongelmaan. Oleellista on yksityiskohtien piilottaminen käyttäjältä ja selkeän rajapinnan tarjoaminen. Ohjelmiston uudelleenkäytettävyys parantaa tuottavuutta ja vaikuttaa myös ohjelmiston laatuun vähentämällä kehittäjille asetettuja vaatimuksia (Fayad & Schmidt 1997).

Toisaalta myös ohjelmistokehyksen käyttöönotto voi olla aikaa vievä prosessi, jolloin käyttö vain yhteen projektiin ei välttämättä ole kustannustehokasta. Lisäksi vaatimukset voivat vaihdella ohjelmistojen kehittyessä eikä valittu kehys välttämättä ole enää ihan-teellinen.

Ohjelmistokehysten suunnitteluperiaatteet eroavat toisistaan jossain määrin, erilaiset lähtökohdat vaikuttavat niiden ominaisuuksiin ja käyttökohteisiin. Joissakin kehyksissä lähtökohtana on pidetty mahdollisimman kattavia ominaisuuksia, kun taas toisissa panostetaan ydintoimintojen keveyteen ja laajennettavuuteen.

Web-sovellusten monimutkaistuttua ja yhä isomman osan ohjelmistokehityksestä siirryttyä verkkoon on myös tarve erilaisille kehitystä helpottaville ohjelmistoille ja muille apuvälineille kasvanut. Web-kehykset auttavat kehityksessä tarjoamalla pohjan kehitykselle sekä kokoamalla tärkeimpiä ominaisuuksia yhteen ja yhtenäistämällä kehitystä vähentämällä tarvetta eri osien yhteensovittamiseen.

Yleisellä tasolla useimmat ohjelmistokehyksiin pätevät asiat koskevat myös web-kehyskiä. Monesti web-kehys tarjoaa hyvän vaihtoehdon ohjelmiston luomiselle alusta alkaen itse. Web-kehityksessä vaatimukset eivät yleensä vaihtele ja käytössä ovat yleiset standardit.

Web-kehukset voidaan jakaa palvelin- ja asiakaspuolen kehyksiin. Asiakaan puolella toimivat kehykset suorittavat suurimman osan tehtävistä selaimessa ladaten vain sivun lähdekoodin palvelimelta. Palvelimella toimivat kehykset useimmiten lähettävät mahdollisimman valmiin sivun asiakkaalle ja tekevät ison osan vaativasta työstä.

Nykyaikaisia web-kehyyksiä on ollut käytössä 2000-luvun alkuvuosista asti. Erilaisilla ratkaisuilla voidaan ohjata kehysten käyttäjiä tietynlaisiin ohjelmointi- ja suunnittelutapoihin. Yleisimpiä käytössä olevia ohjelmistoarkkitehtuureja web-kehityksessä on MVC eli model-view-controller (malli-näkymä-käsittelijä.) MVC:ssä tieto ja käyttöliittymä on erotettu toisistaan. Malli sisältää tiedon kuten kuvat ja tietokannat, näkymä puolestaan hoitaa tiedon esittämisen käyttäjälle. Käsittelijä toimii mallin ja näkymän välissä ja käsittelee käyttäjän syötteet sekä hakee tarvittavat tiedot ja lähettää uuden tilan näkymälle (Wikipedia, 2020a).

Sille mitä osia tai ominaisuuksia web-kehyyksen tulisi sisältää ei ole yhtä yleispätevää määritelmää. Yleisiä ominaisuuksia web-kehyyksissä ovat muun muassa web template system, URL-ohjaus, erilaiset turvallisuusominaisuudet, tietokantatuki ja object-relational mapping eli ORM (Wikipedia, 2020b). Web template system on ohjelmiston osa, jolla voidaan luoda dynaamisesti uusia sivuja valmiiden pohjien perusteella. URL-ohjausta tarvitaan isompien sivustojen hallintaan, jotta käyttäjä saadaan ohjattua web-osoitteen osoittamaan paikkaan. ORM on olio-ohjelmoinnissa käytetty tekniikka, jolla tietokanta yhdistetään ohjelmointikielen tietojen helpomman saatavuuden vuoksi.

Web-kehyyksiä voidaan pitää osana laajempaa *ohjelmistopinoa* (software stack.) Ohjelmistopino on kokoelma erillisiä ohjelmistoja, jotka muodostavat yhdessä kokonaisuuden jonkin tehtävän hoitamiseen.

Monia web-kehyyksiä käytettäessä tärkeä työkalu on *pakettienhallinta* (package manager, package installer) eli ohjelmisto, joka auttaa lisäosien asentamisessa ja hallinnassa. Usein pakettienhallintaan käytetään ohjelmointikielen vakiintuneita työkaluja.

Fernández-Villamor et al. (2008) käyttää termiä *ketterä web-kehys* (Agile web framework) nykyaikaisista full stack ratkaisuista. Ketterät web-kehyykset eroavat perinteisemmistä ratkaisuista siten, että niissä yhdistyy koko ohjelmistopino.

#### **4 Web-kehysten laatukriteerit**

Kehyykset vaihtelevat ominaisuuksiltaan, kaikkia ominaisuuksia ei ole jokaisessa sisäinrakennettuna, myös käyttäjien tarpeet eroavat. Näin ollen ei ole myöskään yleispäteviä mittareita siihen mikä kehyy on paras.

Johtuen aiheen suhteellisesta uutuudesta, ei aiempaa vertailevaa tutkimusta ole erityisen paljon. Web-kehyyksiä käsittelevät vertailut keskittyvät usein yhteen osa-alueeseen tai tietynlaisiin kehyksiin.

Aiemmista tutkimuksista Fernández-Villamor et al. (2008) keskittyi ketteriin web-kehyyksiin. Artikkelissa vertaillaan eri kehyyksiä ja esitellään vertailumalli ketterille web-kehyyksille, tarkoituksena löytää eri tarkoituksiin parhaiten sopivat vaihtoehdot.

Vertailumallissa kiinnitettiin huomiota yleisiin web-sovellusten ominaisuuksiin ja ongelmiin. Kriteerit jaettiin kahdeksaan eri osa-alueeseen, kuten käytettävyys ja tunnettuus. Käytettävyyden arvioinnissa painotettiin automatisointia ja kehitysympäristön ominaisuuksia. Tunnettuudessa tärkeitä alueita olivat kehittäjäyhteisön koko, käyttäjien määrä ja teknologian kypsyys.

Pano et al. (2018) tutkimuksen aiheena oli JavaScript pohjaisten kehysten valintaan johtavat tekijät. Tutkimusta varten haastateltiin päätöksentekijöitä ohjelmistoalalta. Tutkimus keskittyi teknisen puolen sijaan kehittäjien havaintoihin ja kokemuksiin, tarkoituksena oli selvittää miksi kehittäjät valitsevat tietyn vaihtoehdon. Tutkimuksesta selviää, että lähes kaikki osanottajat käyttivät web-kehyyksiä työssään. Osallistujat pitivät tärkeinä ominaisuuksina asioita kuten mahdollisimman tiivistä koodia ja hyvää dokumentaatiota. Laajaa aktiivista yhteisöä ja pitkään jatkunutta kehitystä arvostettiin. Tärkeänä pidettiin myös sitä, että kehys mahdollistaa selkeän koodin kirjoittamisen.

Salas-Zárate et al. (2015) tutkimuksessa vertailtiin web-kehityksen parhaita käytäntöjä web-kehysten yhteydessä Scala-pohjaisen Lift-kehyyksen pohjalta. Tutkimuksessa otettiin tarkasteluun useita kehyyksiä ja mukana olivat myös Django sekä Ruby on Rails. Tutkimuksen mukaan vaihtelevat vaatimukset ja eri kieliä käyttävien web-kehysten suuri määrä tekevät oikean kehyyksen valinnasta vaikeaa. Vääränlaisen kehyyksen valinta voi johtaa kustannuksiin ja kuluttaa aikaa. Vertailukriteereinä pidettiin kehyyksen teknistä ajanmukaisuutta ja ominaisuuksien määrää.

Aiemmissä tutkimuksissa korostui selkeyden ja käytettävyyden painottaminen kehyyksiä arvioidessa. Tärkeänä pidettiin myös kehittäjäyhteisön kokoa, käyttäjien määrä helpottaa vastausten saantia ongelmiin ja takaa työvoiman riittävyyden. Tutkimuksissa oli päädytty melko samankaltaisiin päätelmiin siitä, minkälaiset tekniset ja muut ominaisuudet ovat oleellisia. Ominaisuuksien ajanmukaisuutta pidetään tärkeänä, web-sovelluksissa turvallisuus on oleellinen ominaisuus, joten säännölliset päivitykset ovat tärkeitä.

## 5 Vertailu

Vertailussa keskitytään yleisimpiin nykyaikaisissa web-sovelluksissa tarvittaviin ominaisuuksiin. Ohjelmistoja voidaan vertailla eri tavoin, tässä tutkielmassa keskitytään käytettävyyteen, yleisyyteen sekä ominaisuuksien määrään ja ajanmukaisuuteen. Nykyaikaisissa MVC-arkkitehtuuriin pohjautuvissa web-sovelluksissa tärkeimpiä ominaisuuksia ovat tietokantatuki, näkymien esittäminen ja näiden mahdollisimman helppo yhteensovittaminen.

Yleisyys ja käyttötarkoitustiedot on haettu eri lähteistä internetistä. Käytön yleisyyttä voidaan arvioida tutkimalla ohjelmistojen GitHub-sivustoja, joilta löytyy tiedot julkaisuista ja päivitysaktiivisuudesta. Lisäksi nähdään moniko muu projekti käyttää ohjelmistoa.

Vertailuun valikoituivat Express.js, Django ja Ruby on Rails. Kehykset valittiin niiden suosion ja erilaisen suunnittelun takia, toteutukseen on käytetty eri ohjelmointikieliä ja ominaisuudet vaihtelevat. Ruby on Rails ja Django ovat vanhempia kun taas Express.js edustaa uudempaa suunnittelua.

Ensimmäinen julkinen Express.js versio on julkaistu vuonna 2010, Django sekä Ruby on Rails 2005. Kaikki vertailtavat kehykset ovat vakaita ja ovat olleet yleisessä käytössä useita vuosia. Kehysten käyttämät ohjelmointikielet ovat suosituimpien joukossa. GitHubin The State of the Octoverse -katsaus vuodelle 2019 listaa JavaScriptin suosituimmaksi ohjelmointikieleksi. Python on listalla sijalla 2. ja Ruby sijalla 10 (GitHub, 2020e).

Stack Overflow -sivuston keskustelujen aiheita mittaavien tilastojen mukaan Djangoa koskevat viesti muodostivat noin 2 % sivuston viesteistä, Express.js ja Ruby on Rails viestien määrä oli selvästi alhaisempi. Django ja Express.js ovat myös kasvattaneet suosiotaan viime vuosina, kun taas Ruby on Railsin suosio on laskenut useita vuosia (Stack Overflow, 2020).

## 5.1 Express.js

Express.js on vertailun laajimmin käytetty ja siihen on saatavilla suuri määrä lisäosia. Kehys toimii Node.js runtime-ympäristön päällä ja kehityksestä vastaa Node.js Foundation, joka on vastuussa myös Node.js:n kehityksestä. Sitä pidetään alustan oletuskehysenä. Express.js on vertailtavista kehyksistä uusin ja tällä hetkellä käytetyin.

GitHubissa Express.js:llä on 235 osanottajaa. Commiteja on 5584 eli noin 507 vuodessa ja julkaisuja 235 eli noin 21 vuodessa (GitHub, 2020b).

Kehykseen liittyy apuohjelmia kuten Express application generator, joka helpottaa projektien aloittamista luomalla valmiiksi ohjelmiston perusrungon. Asennuksessa ja lisäosien käyttöönotossa käytetään yleisesti suosittua npm-pakettienhallintaohjelmistoa. Node.js ekosysteemin osana kehys hyötyy saatavilla olevasta laajasta valikoimasta erilaisia lisäosia ja suuresta käyttäjämäärästä.

Express.js kehityksessä on pyritty nopeuteen ja minimalistisuuteen. Kehyksen ydin on pieni ja iso osa ominaisuuksista saatavilla lisäosina (Express.js, 2020). Näin ollen kehys ei myöskään pakota käyttäjää tiettyyn ohjelmointi- tai suunnittelutyyliin. Vaikka kehys itse on kevyt, saattaa käyttöönotto olla aikaa vievää koska se vaatii toimiakseen Node.js -ympäristön. Se toimii myös pohjana useille muille kehyksille.

Express.js käytetään usein osana MEAN-ohjelmistopinoa. MEAN koostuu MongoDB, Express.js, Angular ja Node.js ohjelmistoista, kyseessä on täysin JavaScript-pohjainen ratkaisu, jolloin kehityksessä ei tarvitse hallita useita kieliä.

Node.js ei sisällä tietokantatukea, mutta vaihtoehtoja löytyy runsaasti. Mitä tahansa tietokantaa, jolla on Node.js tuki voidaan käyttää myös Express.js:n kanssa, sama pätee myös muihin lisäosiin. Monet template engineet tukevat Express.js:ää suoraan.

Arkkitehtuurin pohjana on router, joka hoitaa URL-ohjauksen. Toinen perusominaisuus on *väliohjelmisto* (middleware). Koska Express.js omat ominaisuudet ovat hyvin rajatut, suurin osa toiminnallisuudesta on toteutettu väliohjelmiston avulla. Väliohjelmisto toimii ohjelmiston eri tasojen välillä, Express.js:n avulla toteutettu sivusto on sarja http-pyyntöjä, jotka väitetään väliohjelmistolle ja lopputulos saadaan takaisin vastauksena.

## 5.2 Django

Djangon kehityksessä on keskitytty kehityksen nopeuteen ja perusominaisuuksien saatavuuteen (Django, 2020). Ohjelmiston perusversio sisältää työkalut moniin web-sivustojen perustoimintoihin. Kehystä käytetään laajasti sen helppokäyttöisyyden ja kattavien perusominaisuuksien vuoksi (Li, 2014). Kehityksestä vastaa Django Software Foundation -säätiö.

Djangon mottona on ”for perfectionists with deadlines”, suunnittelussa on pyritty yhdistämään kattavat ominaisuudet käyttöönoton nopeuteen. Vaikka perusominaisuudet ovatkin kattavat on saatavilla myös runsaasti lisäosia, joiden asentaminen onnistuu usein Pythonin pip-pakettienhallintaohjelmistolla (djangopackages.org, 2020).

Kehystä käytetään monilla suurilla sivustoilla ja kehittäjäyhteisö on aktiivinen. Projektilla on GitHubissa 1886 osanottajaa ja kehitys jatkuu aktiivisena. Commiteja projektilla on 28362 eli noin 1890 vuodessa ja julkaisuja 282 (noin 18 vuodessa) (GitHub, 2020c). Yhteisöä tukevat myös virallinen foorumi ja postituslista.

Djangossa on hyvä tietokantatuki ja se suosittu tietokantapohjaisten sivustojen pohjana (Liawatimena et al. 2019). Virallinen tuki löytyy useille relaatiotietokannoille. Djangon models-moduuli mahdollistaa tietokantojen liittämisen suoraan Python-luokkiin ja toimii ORM-järjestelmänä.

Django ei pakota käyttäjää yhteen ohjelmointiarkkitehtuuriin. Yleisesti käytössä on *model-view-template* (MVT) joka muistuttaa MVC-arkkitehtuuria. MVT-arkkitehtuurissa Django hoitaa interaktion mallin ja näkymän välillä, jolloin käsittelijää ei tarvita. Kehittäjän tulee lisäksi luoda sivulle pohja näkymän luomiseksi. Näkymiä varten Djangossa on sisäänrakennettu tuki omalle Django Template Language -muotoilukielelle sekä Jinja2-muotoilukielelle.

URL-ohjaus on Djangossa URLconf-moduulilla. Näkymät on toteutettu Python-funktioina, jotka palauttavat näytettävän sivun, URLconf yhdistää näkymäfunktiot URL-osoitteisiin. URLconf mahdollistaa URL-osoitteiden käsittelyn ilman rajoituksia, osoitteet voivat olla dynaamisesti luotuja tai staattisia.

Liawatimena et al. (2019). arvioivat Django lähdekoodia erilaisten metriikoiden avulla. Koodin oikeellisuutta mittaava pylint score 6,69/10 eli kohtalainen tulos. Pylint on ohjelma, joka arvioi Python-ohjelmien koodia tarkastelemalla, kuinka hyvin noudatetaan hyviä ohjelmointikäytäntöjä ja paljonko koodissa on virheitä. Koodin päivitettävyyttä ja kompleksisuutta mittaavalla Radon-ohjelmistolla mitattu maintainability index on pääsääntöisesti hyvä.

### 5.3 Ruby on Rails

Ruby on Rails on yksi ensimmäisistä nykyaikaisista web kehyksistä ja sitä voi pitää ensimmäisenä ketteränä web-kehiksenä, joka oli käytettävyydeltään huomattavasti edeltäjiään parempi (Fernández-Villamor et al. 2008; Ruby on Rails, 2020). Kehityksessä on pyritty yksinkertaisuuteen ja toiston välttämiseen ja uudelleenkäytettävyyteen (Bächle & Kirchberg 2007). Perusohjelmisto sisältää työkalut monien perustoimintojen automaattiseen luomiseen sekä kehitysympäristön vaatimat toiminnot.

Suunnittelun lähtökohtana on pidetty sitä, että on oikea tapa tehdä asioita, jota pyritään tukemaan. Ohjelmistojen tulisi noudattaa MVC paradigmaa. Suunnittelulla on pyritty ohjaamaan käyttäjiä tiettyyn ohjelmointityyliin ja tästä poikkeaminen voi johtaa heikosti toimiviin tuloksiin, eikä ohjelmointityyli välttämättä ole intuitiivinen.

Ruby on Railsin kehitti alun perin David Heinemeier Hansson, joka on edelleen aktiivisesti mukana kehitystyössä. Projektiin on osallistunut GitHubissa 4015 osallistujaa ja kehitys jatkuu edelleen. Commiteja on kertynyt 76879 (n.4804/vuosi) ja julkaisuja 425 (n.28/vuosi) (GitHub, 2020d).

Ohjelmistossa on sisäänrakennettu tuki SQLite tietokannoille, kehystä pidetäänkin erityisen sopivana tehtäviin, joihin liittyy tiedon noutamista tietokannoista (Geer, 2006). Ruby on Rails käyttää näkymien renderointiin ERB template engineä joka mahdollistaa Ruby koodin sisällyttämisen html-tiedostoihin.

Kehyksen suunnittelussa on pyritty tekemään tarkka määrittely ja määritelmien uudelleenkäyttö mahdollisimman helpoksi, näin myös tietokantatuki on luonnollisesti integroitu kehykseen (Bächle & Kirchberg 2007). ORM-toiminnallisuus on toteutettu omalla Active Record -järjestelmällä (Geer, 2006). Suunnittelun lähtökohtana on *convention over configuration* paradigma. Paradigmalla pyritään välttämään turhaa konfigurointia, mallit ja käsittelijät yhdistetään oletuksena automaattisesti Active Recordia käyttäen.

Ohjelmistoissa suositellaan käyttämään DRY eli Don't Repeat Yourself -periaatetta. Jokaisen ohjelmiston osan tulisi olla määritelty yksiselitteisesti ja vain yhdessä paikassa. DRY-tyylin tarkoitus on vähentää työtä, toisaalta se voi johtaa vaikeaselkoiseen koodiin.

## 5.4 Yhteenveto

Django ja Ruby on Rails ovat ominaisuuksiltaan hyvin samanlaisia ja sisältävät kattavat perusominaisuudet, mutta eroavat muutoin tyyliltään Django tarjotessa enemmän valinnanvapautta. Express.js suunnittelun lähtökohta eroaa selkeästi muista vertailun kehyksistä minimalistisuudellaan, kehystä voidaan käyttää kevyisiin sovelluksiin sellaisenaan tai laajentaa tarvittavilla lisäosilla. Express.js myös hyötyy suuresta suosiosta ja Node.js-ekosysteemin laajuudesta.

Vaikka GitHubin luvut eivät annakaan tarkkaa tietoa aktiivisuudesta saadaan niistä suuntaa antavaa tietoa kehityksen aktiivisuudesta. sekä Django että Ruby on Rails ovat selvästi aktiivisemmin päivitettyjä kuin Express.js ja niillä on isompi kehitysyhteisö, tiedot eivät välttämättä kerro todellisesta kehityksestä vaan erot voivat johtua erilaisesta kehittämiskulttuurista. Toisaalta sekä Ruby-ohjelmointikielen että Ruby on Railsin suosio vaikuttaa olevan laskussa.

Koodin laatua on hankala arvioida saatavilla olevan tutkimuksen vähäisyyden vuoksi. Kaikkien kehysten taustalla on hyvin organisoidut tahot ja oletettavasti suunnitteluun ja laatuun panostetaan.

Taulukko 1. Kehysten ominaisuuksia (GitHub, 2020b; GitHub, 2020c; GitHub, 2020d; Django, 2020; Express.js, 2020; Ruby on Rails, 2020).

|                                  | Express.js                      | Django   | Ruby on Rails |
|----------------------------------|---------------------------------|--|---------------|
| Julkaisuvuosi                    | 2010                            | 2005   | 2005          |
| Käyttävät projektit <sup>1</sup> | 5,9 milj.                       | 364000   | 1,3 milj.     |
| Ohjelmointikieli                 | JavaScript                      | Python   | Ruby          |
| Tietokantatuki                   | Saatavana useita lisäosia       | PostgreSQL<br>MariaDB<br>MySQL<br>Oracle<br>SQLite | SQLite        |
| ORM-tuki                         | Ei                              | Kyllä  | Kyllä         |
| Template engine                  | Useissa natiivi Express.js tuki | DTL ja Jinja2.                                     | ERB           |

## 6 Tulokset

Kaikki vertailun web-kehykset soveltuvat yleisimpien tehtävien toteuttamiseen. Käytetyimmät ominaisuudet ovat saatavilla joko suoraan tai pienellä lisävaivalla. Koska ominaisuudet ovat melko samanlaisia jonkin kehysten ollessa ennalta tuttu ei normaalisti kannata käyttää aikaa uuden käyttöönottoon.

Ruby on Rails on vertailuista kehyksistä eniten ohjelmointityyliä rajoittava ja käyttö voi olla hankalaa, jos kehittäjä ei halua käyttää oletettuja käytäntöjä. Selkeät ohjelmointikäytännöt voivat kuitenkin vaikuttaa positiivisesti ohjelmiston laatuun, jos käyttäjä nou-

dattaa ohjeita. Express.js mahdollistaa monenlaisten sovellusten kehittämisen ja on vertailun monipuolisin, toisaalta myös käyttöönotto saattaa vaatia aikaa ja esimerkiksi lisäosien etsimistä.

Yleensä web-kehyksillä voidaan säästää reilusti aikaa ja muita resursseja, jatkuvasti kehittyvässä ympäristössä kaikkien tekniikoiden hallinta voi olla vaikeata tai mahdotonta. Myös web-kehyskäytössä on tärkeää, että saatavilla on hyvä dokumentaatio ja yhteisö. Aina kehysten käyttö ei kuitenkaan ole tarpeen ja pienissä projekteissa niistä saatava hyöty ei ole välttämättä käyttöönoton vaatiman vaivan arvoista.

Vaikka vertailun kehykset kuuluvatkin suosituimpien joukkoon, muodostaa niiden osuus kuitenkin vain pienen osan käytetyistä web-kehyksistä. Ala myös kehittyä jatkuvasti, joten lyhyestä vertailusta ei voi tehdä kovin merkittäviä johtopäätöksiä.

Kaikki vertailun web-kehukset ovat avointa lähdekoodia kuten suurin osa web-kehysistä yleensäkin. Avoimen lähdekoodin mallin etuna on suuri käyttäjä- ja kehittäjä-määrä, jonka ansiosta ohjelmistoja testataan ja kehitetään aktiivisesti.

## Lähdeluettelo

- Bächle, M., Kirchberg, P. (2007). Ruby on Rails. *IEEE Software*. 24(6), 105-108. <https://doi.org/10.1109/MS.2007.176>
- Django. (2020). *Djangon kotisivu*. <https://www.djangoproject.com/> (Haettu 26.5.2020)
- djangopackages.org. (2020). *Django packages*. <https://djangopackages.org/> (Haettu 26.5.2020)
- Express.js. (2020). *Express.js kotisivu*. <https://www.expressjs.com/> (Haettu 26.5.2020)
- Fayad, M., Schmidt, D., (1997). Object-oriented application frameworks. *Communications of the ACM*. 40(10). <https://doi.org/10.1145/262793.262798>
- Fernández-Villamor, J., Díaz-Casillas, L., Iglesias, C. (2008). A comparison model for agile web frameworks. *EATIS '08: Proceedings of the 2008 Euro American Conference on Telematics and Information Systems: 1-8*. <https://doi.org/10.1145/1621087.1621101>
- Fielding, R., Reschke J., (2014). Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing, *RFC 7230*, <https://doi.org/10.17487/RFC7230>.
- Geer, D. (2006). Will software developers ride Ruby on Rails to success? *Computer*. vol. 39(2), 18-20. IEEE Computer Society. <https://doi.org/10.1109/MC.2006.74>
- GitHub. (2020a). *GitHub kotisivu*. [www.github.com](http://www.github.com) (Haettu 26.5.2020).
- GitHub. (2020b). *Express.js GitHub sivu*. [www.github.com/expressjs/express](http://www.github.com/expressjs/express) (Haettu 26.5.2020)

- GitHub. (2020c). *Django GitHub sivu*. [www.github.com/django/django](http://www.github.com/django/django) (Haettu 26.5.2020)
- GitHub. (2020d). *Ruby on Rails GitHub sivu*. [www.github.com/rails/rails](http://www.github.com/rails/rails) (Haettu 26.5.2020)
- GitHub. (2020e). *The State of the Octoverse*. [octoverse.github.com](http://octoverse.github.com) (Haettu 26.5.2020)
- Liawatimena, S., Warnars, H., Trisetjarso, A., Abdurahman, E., Soewito, B., Wibowo, A., Gaol, F., Abbas, B. (2019). Django Web Framework Software Metrics Measurement Using Radon and Pylint. *2018 Indonesian Association for Pattern Recognition International Conference (INAPR)*, 218-222. <https://doi.org/10.1109/IN-APR.2018.8627009>
- Li, Zhi. (2014). Design and Implementation of the Software Testing Management System Based on Django. *Applied Mechanics and Materials, Feb 2014*, 525, 707-710. <https://doi.org/10.4028/www.scientific.net/AMM.525.707>
- Pano, A., Graziotin, D. & Abrahamsson, P. (2018). Factors and actors leading to the adoption of a JavaScript framework. *Empirical Software Engineering* 23(6), 3503–3534. <https://doi.org/10.1007/s10664-018-9613-x>
- Ruby on Rails. (2020). *Ruby on Rails kotisivu*. <https://rubyonrails.org/>
- Salas-Zárate, M., Alor-Hernández, G., Valencia-García, R., Rodríguez-Mazahua, L., Rodríguez-González, A., López Cuadrado, J., (2015). Analyzing best practices on Web development frameworks: The lift approach. *Science of Computer Programming*, 102, 1-19. <https://doi.org/10.1016/j.scico.2014.12.004>.
- Stack Overflow. (2020). *Stack Overflow -sivuston tilastot*. <https://insights.stackoverflow.com/trends> (Haettu 26.5.2020)
- Vosloo, I., Kourie, D. G. (2008) Server-centric web frameworks: An overview. *ACM Comput. Surv.* 40(2). <https://doi.org/10.1145/1348246.1348247>
- Wikipedia. (2020a). *Wikipedian sivu MVC-arkkitehtuurista*. [fi.wikipedia.org/wiki/MVC-arkkitehtuuri](http://fi.wikipedia.org/wiki/MVC-arkkitehtuuri) (Haettu 26.5.2020)
- Wikipedia. (2020b). *Wikipedian sivu web-kehysistä*. [https://en.wikipedia.org/wiki/Web\\_framework](https://en.wikipedia.org/wiki/Web_framework) (Haettu 26.5.2020)