Julkaisuhakusivun suunnittelu ja toteutus

Ville Törnblom

Opinnäytetyö, AMK
Toukokuu 2024
Insinööri (AMK), Tieto- ja viestintätekniikka

jamk | Jyväskylän ammattikorkeakoulu
University of Applied Sciences

jamk

Kuvailulehti

Törnblom, Ville

Opinnäytetyön nimi. Julkaisuhakusivun suunnittelu ja toteutus

Jyväskylä: Jyväskylän ammattikorkeakoulu. Toukokuu 2024, 64 sivua.

Tietoliikenne- ja viestintäteknologian tutkinto-ohjelma. Opinnäytetyö AMK.

Julkaisun kieli: suomi

Julkaisulupa avoimessa verkossa: kyllä

Tiivistelmä

Opinnäytetyö tehtiin Jyväskylän Yliopiston Digipalveluille. Digipalvelut tuottavat verkkopalveluita ja ohjelmistoja Jyväskylän Yliopiston opiskelijoiden ja työntekijöiden käytettäväksi.

Opinnäytetyön pääaiheina oli julkaisuhakusivun suunnitteluprosessi, sen toteuttaminen ja ohjelmistokehyksien vertailu palvelua varten. Työssä perehdyttiin websuunnittelun teoriaan ja periaatteisiin. Ohjelmistokehyksien teoriaa, ominaisuuksia ja vertailtiin niiden suorituskykyä. Opinnäytetyössä perehdyttiin FastAPI:lla rakennetun projektin rakenteeseen, ominaisuuksiin ja komponentteihin. Opinnäytetyössä luotiin vaatimusmäärittely, käytiin läpi projektin arkkitehtuuri, käytetyt työkalut ja sovelluksen toteutus.

Opinnäytetyö toteutettiin soveltavana tutkimus- ja kehitystyönä.
Työn lopputuloksena saatiin aikaan toimiva prototyyppi, joka todistaa konseptin toimivuuden. Prototyyppi käytti sille toivottuja teknologioita, mutta prototyypin voi kehittää täysin toimivaksi palveluksi. Työn lopussa esitellään mahdollisia jatkokehitysideoita.

Avainsanat (asiasanat)

API, FastAPI, Figma, kehitys, Python, ohjelmointi

Muut tiedot (salassa pidettävät liitteet)

jamk

Description

Törnblom, Ville

Design and implementation of a search page for releases

Jyväskylä: JAMK University of Applied Sciences, May 2024, 64 pages.

Degree Program in Information and Communication Technology. Bachelor´s thesis.

Permission for open access publication: Yes

Language of publication: Finnish

Abstract

Thesis was done for Digital Services in Jyväskylä University. Digital Services produce online services and software for Jyväskylä University students and employees.

The main topics of the thesis were the planning process, implementation, and comparison between different frameworks for search page for scientific research papers. The thesis studied web developments theory and principals. The thesis studied the theory behind frameworks, their properties and compared the benchmarks for their performance. The thesis accessed the properties, components and the structure of a project build with FastAPI. During the project the requirement specifications were created, went through the project's architecture, the used technologies, and the development of the application.

The thesis was implemented as a research-based development assignment. As the result of the project was a functioning prototype, that proves that the concept is functional. The prototype uses the set technologies, though the prototype can be developed into a full working service. At the end of the thesis further development ideas were presented.

Keywords/tags (subjects)

API, developing, FastAPI, Figma, programming, Python

Miscellaneous (Confidential information)

Kuviot

Taulukot

Lyhenteet ja termit

APA	American Psychological Association, kirjoitusmuoto akateemisille asiakirjoille.
API	Application Programming Interface, ohjelmointirajapinta.
ASGI	Asynchronous Server Gateway Interface, kutsukäytäntö web-palvelimille välittämään pyyntöjä asynkroniseen Python-ohjelmointikielikehykseen ja -sovelluksiin.
CPR	Co-Citation Percentile Rank, sijoitus, joka annetaan julkaisulle riippuen julkaisun viittausmäärästä sen jälkeen, kun julkaisuun voi viitata.
CSV	Comma Separated Values, tiedostomuoto, jota käytetään tiedoille taulukkomuodossa.
Debuggaus	Ohjelmoinnissa käytettävä termi, testataan ohjelmaa tai koodia virheiden etsimisen varalta.
Deserilisaatio	Deserialization, prosessi. jolloin saatu data muutetaan objektiksi.
Dimensions	Dimensions Analytics, kansainvälinen laaja-alainen viitetietokanta.
ENV	Environmental, Unix-tyyppisten käyttöjärjestelmien komento.
Footer	Sivun alatunniste, termi, jota käytetään web-kehityksessä.
GraphQL	Kysely- ja käsittelykieli API-liittymille.

Header	Sivun ylätunniste. termi, jotka käytetään web-kehityksessä.
HTTP	Hypertext Transfer Protocol, tietokoneiden välinen tiedonsiirtoprotokolla.
JSON	JavaScript Object Notation, tiedonsiirtomuoto, joka perustuu Javascript-ohjelmointikieleen.
Jufo	Julkaisufoorumi.
Käyttöliittymä	Rajapinta, jonka avulla sovelluksen käyttäjät kykenevät käyttämään sovellusta.
Microsoft Excel	Microsoftin kehittämä laskentataulukko-ohjelmisto.
OA	Open Access, julkaisujen kategoria, joka on yleisesti avoin
OKM	Opetus- ja kulttuuriministeriö
Placeholder	Tilapäinen tekstikenttä tai objekti, joka näyttää esimerkkiä tai ohjeita tietojen syöttämistä tai esimerkkinäkymää.
Python	Yksinkertainen ja selkeä rakenteinen tulkattava ohjelmointikieli.
Query	SQL kielellä käytettävä tietokantakysely.
REST	Representational State Transfer, arkkitehtuurinen tyyli ohjelmointirajapintojen toteuttamiseen.
Sarjallistaminen	Serialization, prosessi, jolloin objektin tilaa muutetaan muotoon, jolloin sen voi siirtää.
Seurantakohde	Follow-up group, Jyväskylän Yliopiston raportointia varten keksitty termi. Käytetään julkaisujen ja hankkeiden kirjauksessa sekä tutkijaprofiileissa merkitsemään tärkeää luokittelutietoa.

SQL	Tietokantakyselykieli, jonka avulla suoritetaan toimintoja tietokannassa.
Visual Studio	Microsoftin kehittämä ohjelmointityökalu.
XML	Extensible Markup Language, merkintäkielien standardi.
WSGI	Web Server Gateway Interface, kutsukäytäntö web-palvelimille välittämään pyyntöjä synkroniseen Python-ohjelmointikielikehykseen ja -sovelluksiin.
YSO	Yleinen suomalainen ontologia, yleiskäsitteistä koostuva ontologia. Käytetään kuvailtavien aineistojen aihealueiden ollessa monipuolisia.

1 Johdanto

1.1 Taustaa ja toimeksiantaja

Jyväskylän yliopiston Digipalvelut ovat yliopiston tarjoamia digitaalisia palveluita, jotka tukevat yliopiston opetusta, tutkimusta ja hallintoa. Digipalveluihin kuuluvat verkkotyökalut ja ohjelmistot, joiden avulla opiskelijat voivat osallistua etäopetukseen, tehdä verkkotehtäviä ja kommunikoida opettajien kanssa. (Jyväskylän yliopiston digipalveluiden verkkosivut n.d.)

Tämän opinnäytetyön aiheena on Jyväskylän yliopiston Digipalveluiden tilaama projekti julkaisuhakusivun suunnittelusta ja toteutuksesta. Digipalveluilla on entuudestaan tutkimustietojärjestelmiä, käytöstä poistuva TUTKA sekä kirjoittamisen aikaan käytössä oleva Converis. Kehitettävän julkisen julkaisuhakusivun tulee hakea tietoa Converiksesta ja tulostaa tätä tietoa tulostettavan raportin tavoin.

Opinnäytetyö tehdään tekijän harjoittelujakson jatkeena Digipalveluille. Opinnäytetyö on tehty työskennellessä Digipalveluiden T-Tiimissä, johon kuuluu 4 ihmistä ja tiiminvetäjä. Opinnäytetyö perustuu toteutettuun projektiin.

1.2 Tehtävä ja tavoitteet

Työn tavoitteena on suunnitella ja toteuttaa hakusivu, joka mahdollistaa julkaisujen hakemisen, tarkastelun ja suodattamisen. Hakusivun tulee pystyä luoda APA-viitteitä tehdyistä julkaisuista ja tarjota käyttäjälle yksinkertainen käyttöliittymä julkaisulistoja vastaavien tiedon hakuun ja kevyeen integrointiin. Hakusivun kehittämistä varten tulee löytää sille sopivat ohjelmistokehykset, jotka sopivat digipalveluiden kehitysympäristöön. Projektin kanssa tulee myös käyttää virtuaalista kehitysympäristöä ja arvioida sen jatkokäyttöä digipalveluiden kehitystyössä.

Työn toteuttamista varten laadittiin vaatimusmäärittely, joka sisältää yleisiä tavoitteita, mutta työn tarpeita ja ominaisuuksia lisätään ja mukautetaan ajan kanssa kehitystyön ohella. Projekti toteutetaan Python-ohjelmointikielellä. Ohjelmointikieli on valittu digipalveluiden alustojen mukaan, jotka käyttävät valmiiksi Pythonia.

1.3 Tutkimuksellinen kehittämistyö

Tutkimusmenetelmällisyys tässä opinnäytetyössä pohjautui tutkimukselliseen kehittämistyöhön, jonka tarkoituksena oli luoda julkaisuhakusivu ja selvittää siihen parhaiten soveltuva ohjelmistokehys sekä löytää käyttäjälle soveltuvin esitettävä sivurakenne.

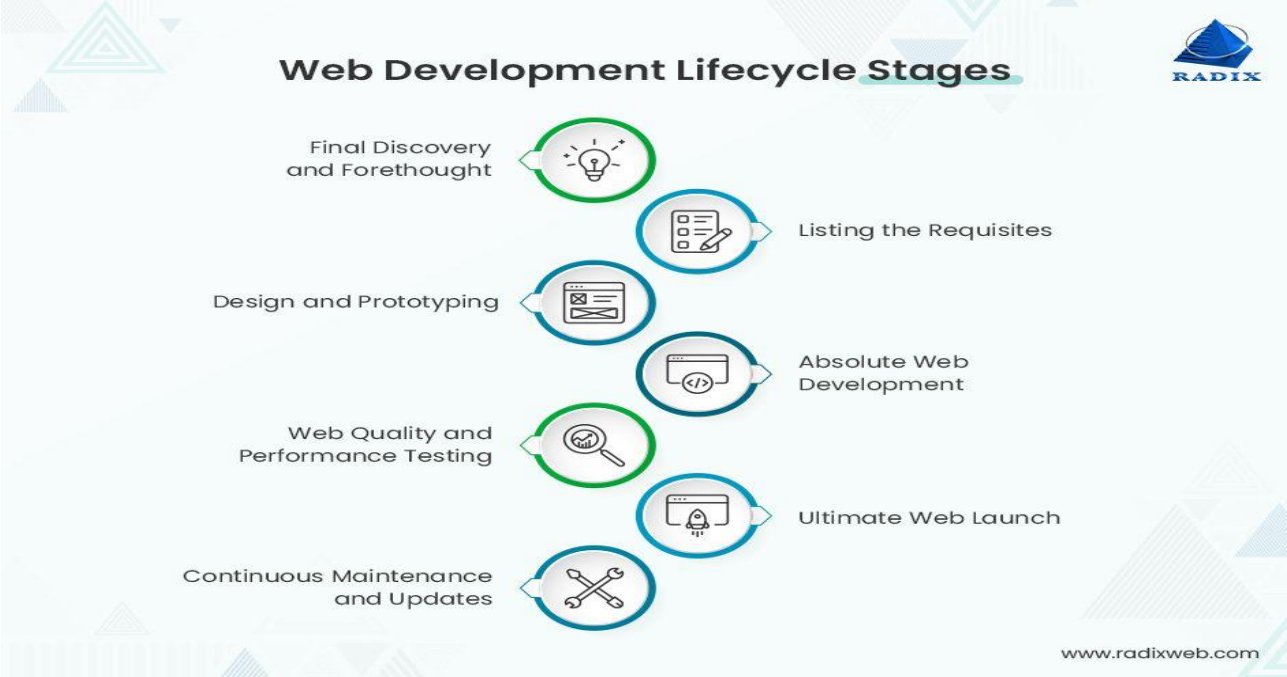
Kehityksen tarkoituksena on luoda prototyyppi, joka todistaa valitun ohjelmistokehyksen toiminnan digipalveluiden kehitysympäristössä, tuoden käyttäjälle haettua tietoa. Tämä edellyttää tietoa tutkijoiden tarpeista, web-kehityksen käytännöistä ja digipalveluiden olemassa olevien kehitysympäristöjen ymmärrystä.

Opinnäytetyötä varten kerättiin tietoa tietokannoista, e-kirjoista, internetistä löytyvistä artikkeleista ja digipalveluiden sisäisestä dokumentaatiosta. Toteutuksessa hyödynnettiin kerättyä tietoa ja hyviä kehitysmenetelmiä yhdistelemällä, testaamalla ja soveltamalla, jonka tulosten pohjalta syntyi työn tulos.

2 Web-suunnittelu ja -kehitys

Web-sivujen rakentamiseen kuuluu kaksi pääosaa, web-suunnittelu ja web-kehitys. Nämä rakennuspalat koostuvat vielä useammasta eri askeleesta, mutta molemmat ovat yhtä tärkeitä kokonaisuuden rakentamiseksi. Web-suunnittelu keskittyy web-sivujen ulkonäköön ja web-kehitys sivujen toiminallisuuteen. (Mailchimp n.d.)

Web-kehitys on systemaattinen prosessi, joka seuraa selkeästi merkattua polkua, joka sisältää useita askelia onnistuneen websivun tai -sovelluksen kehittämiseen. Vatsal Parmar on kehittänyt systemaattisen polun (Ks. Kuvio 1), joka perustuu 7 askeleeseen webkehityksen elinkaaressa. (Parmar 2023.)



Kuvio 1. Webkehityksen elinkaari.

Parmarin ehdottaman elinkaaren 7 askelta ovat:

- ”Final discovery” ja ennakointi
- Vaatimusten listaaminen
- Suunnittelu ja prototyypit
- Absoluuttinen webkehitys
- Laadun ja suorituksen testaaminen
- Websivujen julkaisu
- Jatkuva ylläpito ja päivittäminen.

”Final discovery” ja ennakoinnin askel sisältää suunnittelua ja vaatimusten tarkastelua. Tässä askeleessa mitataan projektin vaatimat resurssit, tavoitteet ja aikataulut. Tämän avulla varmistetaan kehityksen sujuvuus tulevaisuudessa. Suunnittelun mukana havainnoidaan kohdeyleisö, kartoitetaan kilpailijoiden tarjoamia palveluita ja asetetaan projektin aikataulu. (Parmar 2023.)

Vaatimusten listaamisen kanssa aloitetaan prosessi vaatimusten keräämisestä, jossa käydään läpi yksityiskohtaisesti projektin teknisistä tiedoista, suunnittelusta ja käyttäjien odotuksista.

Tavoitteena on taata lopullisen tuotteen vastaavan asiakkaan toiveita ja sopivan kohdeyleisön käyttöön. (Parmar 2023.)

Seuraavana askeleena on web-suunnittelijoilla luoda projektista visuaalista näyttöä lankamallien ja yksityiskohtaisten prototyyppien muodossa. Näiden avulla hahmotetaan web-sivujen rakennetta ja käyttöliittymää. Prototyyppien avulla kartoitetaan tehokkaasti suunnitteluvaiheessa projektin interaktiivisia elementtejä ennen varsinaisen kehitystyön alkamista. (Parmar 2023.)

Kokeellinen vaihe on olennainen osa web-kehitys prosessia. Tässä tilassa kehitetään placeholder-käyttöliittymä ja luodaan olennaisia ominaisuuksia ja konfiguroidaan tietokantoja. Suuremman projektin luonnollisen etenemisen takaamiseksi hyödynnetään suunnitelmallisia käytäntöjä ja git-ympäristöjä aikatauluttamisen ohella. Projektin käyttöön suunniteltujen teknologioita hyödyntäen sivut optimoidaan, jotta sivut olisivat mahdollisimman responsiiviset loppukäyttäjälle. (Parmar 2023.)

Kun sivun toiminnallisuudet ja UI/UX-elementit ovat valmiina, testausprosessi voi alkaa. Jokainen sivu tulisi tulla testatuksi, jotta kaikki elementit, linkit ja ominaisuudet toimivat ja että sivu näytetään oikein eri selaimilla. Tässä vaiheessa on parasta tutkia miten web-sivun sanoitukset ja sanajärjestykset vaikuttavat hakukoneissa websivun löydettävyyteen. (Parmar 2023.)

Testauksen jälkeen web-sivut voidaan julkaista. Web-sivujen tulee olla palvelimella, jotta ne olisivat löydettävissä. Kun websivut ovat julkisia ja pyörivät palvelimella, niiden toiminallisuus tulisi testata vielä, varmistukseksi että julkaisu tuottanut ongelmia. (Parmar 2023.)

Web-sivujen julkaisu ei pysäytä web-kehityksen tarpeita. Julkaisun jälkeen web-sivut vaativat tasa-aikaisia tarkastuksia sekä uusien päivitysten ja sisällön tuottamista. Säännölliset päivitykset ovat myös tärkeitä web-sivujen turvallisuuden takaamiseksi. Ylläpitäminen sisältää teknologioiden päivittämistä, datan varmuuskopioimista, web-liikenteen valvomista ja haavoittuvuuksien kartoittamista. Ylläpidon kanssa varmistetaan web-sivujen pysyvän ajan tasalla ja saavutettavana kaikille käyttäjille. (Chawre n.d.)

2.1 Vaatimusmäärittely

Vaatimusten hallinta pyrkii takaamaan, että tuotekehityspäämäärät täyttyvät onnistuneesti käyttäen hyväksi erilaisia tekniikoita kuten vaatimusten dokumentointia, analysointia ja priorisointia. Tämä auttaa suunnittelijoita pysymään ajan tasalla sen hetkisistä ja hyväksytyistä vaatimuksista. Yleensä tuotepäällikkö on vastuussa vaatimusten määrittelystä, mutta projektin luonteesta riippuen vaatimukset voi määrittää joku sidosryhmään kuuluva. (IBM n.d.)

Vaatimusmäärittely on keskeinen askel organisaatioiden tiimin työssä, sillä se auttaa ymmärtämään asiakkaan tarpeet ja odotukset ennen kuin aloitetaan suunnittelu tai toteutus. Tämä prosessi varmistaa, että kaikkien osapuolten välillä on yhteisymmärrys siitä, mitä järjestelmästä voidaan odottaa. Vaatimusmäärittelydokumentti sisältää yksityiskohtaisesti kuvauksen siitä, mitkä toiminnallisuudet ja ominaisuudet järjestelmällä on oltava, sekä määrittelee myös mahdolliset rajoitteet. (Le Vier 2010.)

2.2 Käyttöliittymän suunnittelu

Hyvän käyttöliittymän tulee olla tehokas, luotettava ja mukava käyttää. Käyttöliittymäsuunnittelussa pyritään suunnittelemaan käyttöliittymä niin, että se täyttää nämä kriteerit käyttäjän näkökulmasta. Suunnittelussa on tärkeää ottaa huomioon tuotteen kohdeyleisö, jotta käyttöliittymä vastaa käyttäjien odotuksia ja tarpeita. Tämän ymmärtämiseksi suunnittelijat hyödyntävät käyttäjäkeskeisiä suunnittelumenetelmiä, kuten käyttäjähaastatteluja. (Adobe n.d.)

Käyttöliittymäsuunnittelun voi jakaa kolmeen kategoriaan: graafiseen, ääni- ja elepohjaiseen käyttöliittymän suunnitteluun. Graafinen käyttöliittymä tarjoaa visuaalisen esityksen ohjelman toiminnoista, muistuttaen esimerkiksi tietokoneen työpöytää. Äänikäyttöliittymässä käyttäjä ohjaa palvelua puheella, kuten tapahtuu esimerkiksi iPhonen Sirin ja Amazonin Alexan puheohjattavien virtuaaliavustajien avulla. Elepohjainen käyttöliittymä hyödyntää kehon liikkeitä toimintojen ohjaamiseen. Tätä käytetään esimerkiksi virtuaalitodellisuuspeleissä. (Interaction Design Foundation n.d.)

Joel Spolsky tuo esille web-käyttöliittymille tärkeän elementin olevan käytön sujuvuus, sillä käyttöliittymät, jotka saavat käyttäjän odottamaan asioiden tapahtumista, tuntuvat kömpelöiltä.

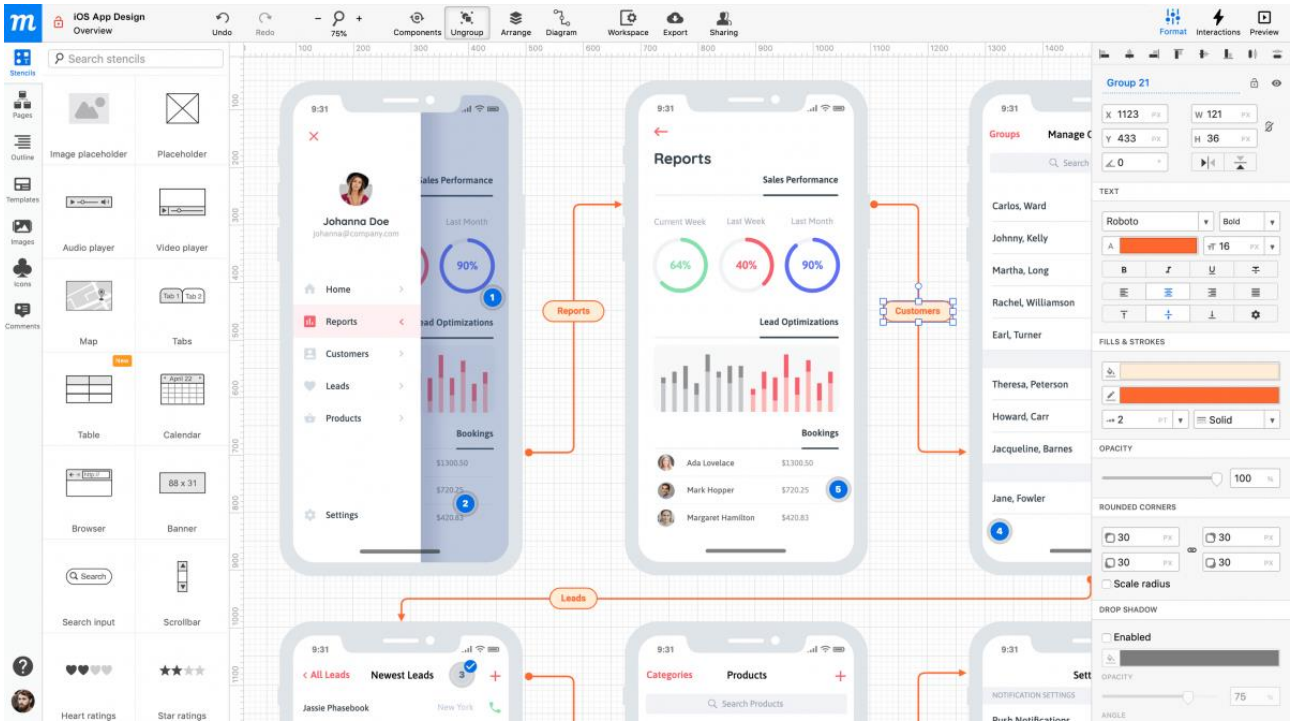
(Joel Spolsky 2001). Graafista käyttöliittymää voidaan parantaa tarjoamalla käyttäjilleen miellyttävämpi ja sujuvampi kokemus antamalla heille enemmän hallintaa. Mahdollistamalla virheiden helposti peruuttaminen käyttäjille, vähennetään heidän pelkoansa tehdä valintoja. Täten toimintojen tulisi olla selkeitä ja niiden merkityksen pitää olla ilmeinen, jotta käyttäjien ei tarvitse arvailla niiden tarkoitusta. Käyttöliittymä tulisi pitää mahdollisimman yksinkertaisena, poistamalla kaikki tarpeettomat toiminnot, jotka eivät lisää käyttäjäkokemuksen arvoa. Liialliset toiminnot vain lisäävät sekavuutta ja vaikeuttavat käyttöä. Painikkeiden tulisi olla riittävän suuria ja näkyviä, jotta niitä on helppo käyttää. Visuaaliset vihjeet toimintojen yhteydessä auttavat käyttäjiä tunnistamaan niitä helpommin. Käyttöliittymän tulee aina noudattaa johdonmukaista linjaa sekä toiminnallisesti että visuaalisesti (Babich 2019.) sekä tehdä käytöstä helppoa ja miellyttävää, joka johtaa parempaan käyttäjäkokemukseen. (Haltu 2023.)

2.3 Prototyypin laadinta

Prototyyppi on käytännöllinen malli, jota käytetään suunnitteluratkaisun testaamiseen ja joka yleensä rakennetaan tai luodaan suunnittelu- ja rakennushaasteen aikana. Todellisessa elämässä prototyypit auttavat arvioimaan suunnitteluratkaisujen tehokkuutta ennen lopullisen tuotteen tai järjestelmän valmistamista. (Lets talk science N.d.)

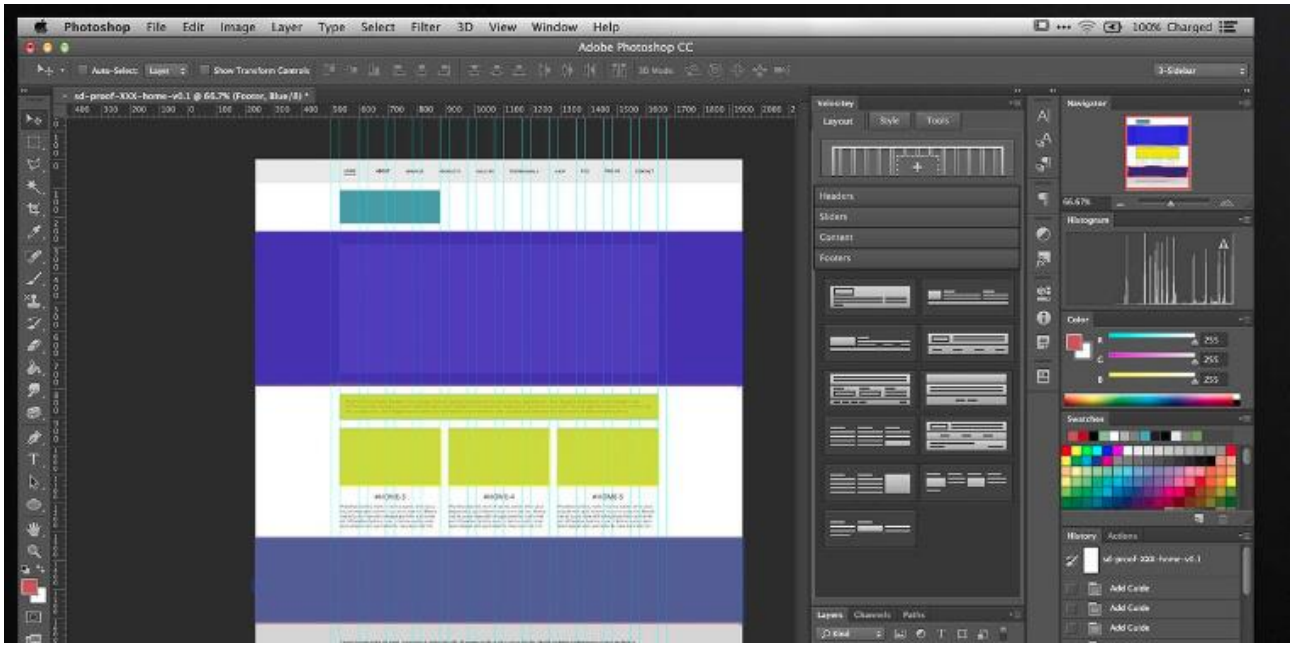
Prototyypin laatiminen kehitysvaiheessa on monella tasolla hyödyllistä, sillä se mahdollistaa realistisen aikataulun suunnittelun resurssien mukaisesti, voit helpommin suunnitella miltä sivustosi tulee näyttämään ja toimimaan jo varhaisessa vaiheessa. Prototyypin avulla saat myös selvän vision kehittämisen kannalta, säästät aikaa tulevaisuudessa suunnittelulta ja voit tunnistaa sivuston tarpeet ajoissa. (Internetdevels 2020.)

Prototyyppejä voi rakentaa usealla eri tavalla, kuten internetistä löytyvillä työkaluilla (ks.Kuvio 2.), jotka soveltuvat parhaiten tiimin kanssa työskentelyyn ja ovat yleisesti helppoja käyttää. (Internetdevels 2020.)



Kuvio 2. Online-työkalut.

Prototyyppejä voi rakentaa myös ammattilaisille rakennetuilla sovelluksilla. (Ks. Kuvio 3.) Näiden etuna on kyky rakentaa suurempia prototyyppejä isoille projekteille ja luoda interaktiivisia elementtejä, joiden avulla voidaan testata sivun suunniteltuja ominaisuuksia. (Internetdevels 2020.)



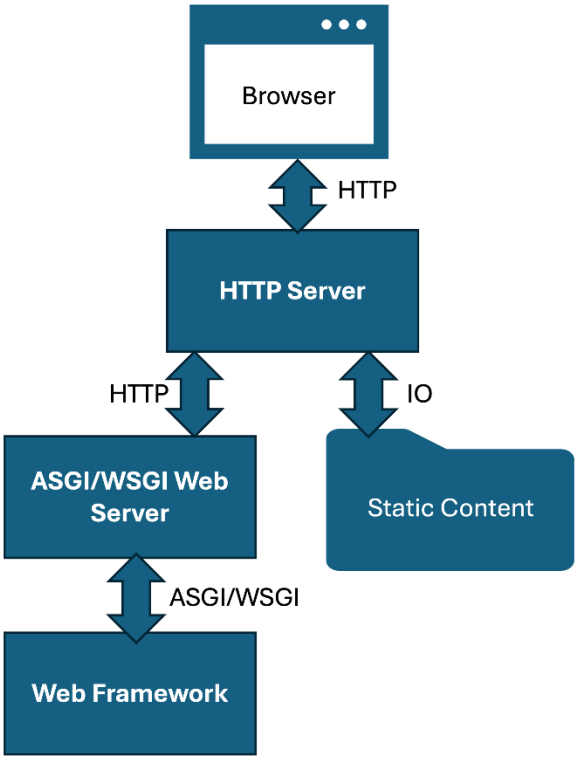
Kuvio 3. Ammattilaisen sovellus.

2.4 Web-sovellusten arkkitehtuuri

Web-sovellusten arkkitehtuuri koostuu tyypillisesti kolmesta komponentista. Web-selaimesta, verkkopalvelimesta ja tietokantapalvelimesta. Web-selain on ainoa asia mikä näytetään loppukäyttäjälle, tämä vastaanottaa syötteen ja hallinnoi esityksen logiikan sekä ohjailee käyttäjän interaktioita sovelluksen kanssa. Verkkopalvelin, taustalla tai palvelimen puolella olevana komponenttina, hoitaa liiketoiminallisen logiikan ja käsittelee käyttäjätiedostot ohjaamalla pyynnöt oikeaan komponenttiin ja hallinnoimalla koko sovellusoperaatiot. Tietokantapalvelin tarjoaa sovellukselle tarvittavat tiedot. Se hoitaa tieto-operaatiot. Useita tasoja sisältävässä arkkitehtuurissa tietokannan palvelimet voivat hallita liiketoiminnallista logiikkaa avullaan tallennetuilla proseduureilla. (William 2024.)

API-päätepisteet web-sovellusten arkkitehtuurissa ovat API-muutoksen hallintatyökalu, joka mahdollistaa HTTP-, WebSocket- ja REST-API:en luomisen, julkaisemisen, turvallisuuden varmistamisen ja niiden hallinnan. Toimien palvelun vastaanottajana, API-päätepiste saa erilaisia API-kutsuja, yhdistää tarvittavat palvelut kutsujen täyttämiseksi ja palauttaa tulokset. API-päätepisteet auttavat suojelemaan API:a, käynnistämään analytiikkatoiminnot API:lle ja hallinnoimaan vanhoja ja poistettuja API:a. (William 2024.)

Web-sovelluksia kehittäessä ohjelmistokehyksien kanssa, arkkitehtuuri tarvitsee kaksi ylimääräistä ohjelmistoa, jotta käyttäjät voivat käyttää palvelua selaimessa. ASGI/WSGI web-palvelimen, joka käynnistää web-sovelluksen ja kuuntelee käyttäjien tekemiä HTTP-pyyntöjä. (Ks. Kuvio 4.) Vaihtoehtoisesti arkkitehtuuri voi hyödyntää HTTP-palvelinta, joka tuottaa staattiset elementit sovellukselle, kuten CSS tyylitiedostot, kuvat ja Javascriptin ohjelmointitiedostot. (Shaw 2023.)



Kuvio 4. Web-sovellusten arkkitehtuuri.

3 Web-suunnittelussa käytettävät teknologiat ja ympäristöt

3.1 Ohjelmistokehys

Ohjelmistokehys on joukko lähdekoodia tai kirjastoja, jotka tarjoavat toimintoja, jotka ovat yhteisiä koko sovellusluokalle. Yhden kirjaston tarjotessa yhden tietyn toiminnallisuuden, ohjelmistokehykset tarjoavat laajemman valikoiman toimintoja, joita käytetään saman ohjelmistopaketin sisällä. Ohjelmistokehyksiä hyödyntämällä vähennetään uudelleenkirjoittamista, sen sijaan että ohjelmoija kirjoittaisi saman logiikan koodiin uudelleen, vähentäen sovelluksen rakentamiseen kuluvaa aikaa ja vähentäen mahdollisuuksia uusille virheille. (Framework 2013.)

Ohjelmistokehysten valitseminen projektia varten vaatii aikaa, testausta ja tutkimusta olemassa olevien ohjelmistokehysten välillä. (Salas-Zárate, Alor-Hernández, Valencia-García, Rodríguez-Mazahua, Rodríguez-González & Cuadrado, 2015.)

3.2 Virtual Environment

Virtual Environment eli virtuaalinen kehitysympäristö, on Pythonin kehitysympäristö, jossa siihen asennetut Pythonin kääntäjät, kirjastot ja skriptit ovat eristetty muista, jotka ovat asennettu muille virtuaalisille ympäristöille. Täten ne ovat myös eristetty niistä, jotka ovat asennettu osana käyttöjärjestelmää. Laajemmissa projekteissa työskennellään useiden tiedostojen, pakettien sekä suhteiden kanssa, jonka vuoksi projekti eristetään erilliseen virtuaaliseen kehitysympäristöön. (Creation of virtual environments n.d.)

Virtuaaliset kehitysympäristöt ovat hyödyllisiä tilanteissa missä on useampi projekti, jotka käyttävät eri ohjelmistokehyksiä tai eri ohjelmistokehysversioita. Näissä tapauksissa virtuaalinen kehitysympäristö pitää nämä projektit erillään ja pitää huolen molempien projektien riippuvuuksien yhdenmukaisuudesta. (Agrawal 2023.)

3.3 WSGI ja ASGI

WSGI (Web Server Gateway Interface) ja ASGI (Asynchronous Server Gateway Interface) ovat määritelmiä, jotka kuvaavat miten verkkoserverit viestivät Python-käyttöliittymien kanssa. WSGI toimii synkronisesti, mikä tarkoittaa, että se käsittelee yhtä pyyntöä kerrallaan ja estää muun työn jatkamisen, kunnes nykyinen pyyntö on valmis. (Python 2010.) ASGI on suunniteltu käsittelemään asynkronisia operaatioita, mikä mahdollistaa monipuolisen pyyntöjen samanaikaisen hallinnan ilman esteitä. (ASGI n.d.)

3.4 Mod_wsgi

Mod_wsgi on paketti, joka implementoi helposti käyttöön otettavan Apache-moduulin, joka voi isännöidä (host) mitä tahansa Pythonin web-sovellusta, joka tukee Pythonin WSGI määrittelyitä. Mod_wsgi paketti mahdollistaa Pythonin web-sovellusten ajamisen, jotka on rakennettu ohjelmistokehyksillä Apache web-palvelimen sisällä. Se tarjoaa yhteyden Apachen ja Pythonin

välillä, joka antaa Apachen käsitellä tulevia HTTP-pyyntöjä ja välittää ne Python-projektille prosessoitavaksi. (mod_wsgi n.d.)

3.5 Middleware

Middlewarella tarkoitetaan ohjelmaa, joka varmistaa, että kaikki teknologisessa järjestelmässä käytettävät osa-alueet kommunikoivat toistensa kanssa vaihtaen informaatiota. Middleware on ohjelmistokomponentti, joka sijoittuu web-sovelluksen ja web-palvelimen väliin, mahdollistaen erilaisten tehtävien suorittamisen, kuten pyyntöjen käsittelyn, vastausten muokkaamisen tai lisätoiminnallisuuksien tarjoamisen. (OPC router n.d.) WSGIMiddleware on osa WSGI-standardia, joka määrittelee protokollan Python-sovellusten ja web-palvelimen välille. WSGIMiddleware on komponentti, joka mahdollistaa ASGI-palvelimien toimia WSGI-sovellusten kanssa. Se yhdistää WSGI-sovellusten synkroniset ja ASGI-palvelimien asynkroniset palvelut. (Shaw 2023.)

3.6 Visual Studio Code

Visual Studio Code on kevyt ja suorituskykyinen tekstieditori, joka noudattaa avoimen lähdekoodin periaatteita. Microsoft on kehittänyt Visual Studio Code -ohjelman ja ohjelma on saatavilla Windows-, macOS- ja Linux-käyttöjärjestelmille. Se tukee monia ohjelmointikieliä, kuten JavaScriptiä, Pythonia ja HTML:ää. Lisäksi Visual Studio Codeen on saatavilla laajasti laajennuksia, jotka mahdollistavat ohjelmointi- ja merkintäkielen käytön, joita käyttäjät voivat asentaa tarpeensa mukaan. Ohjelmassa on monipuoliset ominaisuudet koodin kirjoittamiseen, kuten virheiden korjaus, automaattinen koodin täydennys, syntaksin korostus ja integroitu terminaali. (Visual Studio Code: Getting Started n.d.).

3.7 Python

Python on monipuolinen ja interaktiivinen ohjelmointikieli, joka soveltuu monipuolisesti erilaisiin ohjelmointiparadigmoihin mikä mahdollistaa monimutkaisemman tietokoneohjelmien toteuttamisen. Erityisesti Pythonia käytetään oliopohjaisessa ohjelmoinnissa, ja sen rakenne on suunniteltu yksinkertaiseksi ja selkeäksi. Kieli sisältää moduuleja, poikkeuksia, dynaamista kirjoittamista sekä korkean tason dynaamisia tietotyyppejä ja luokkia. (General Python FAQ 2022.)

3.8 Converis

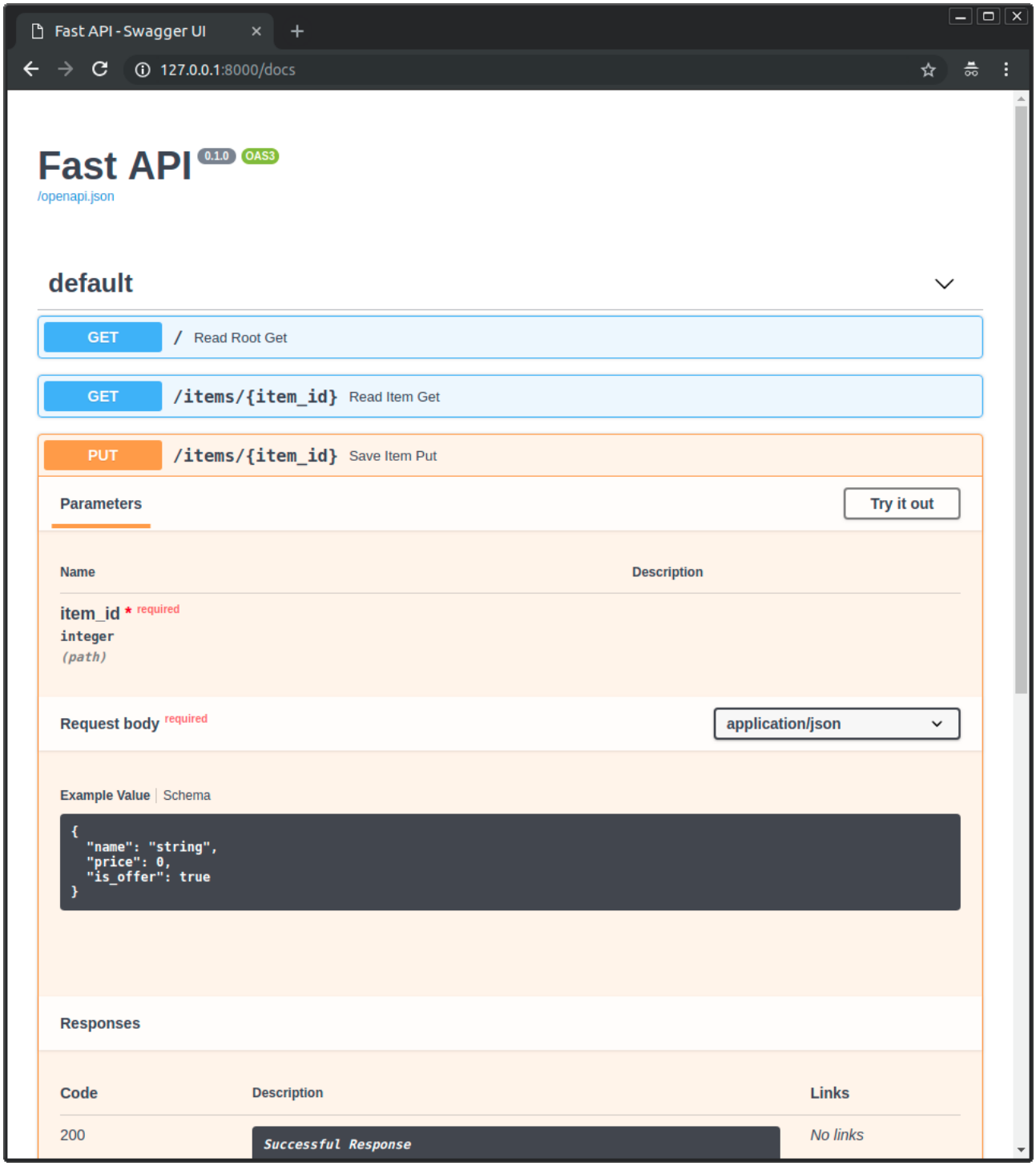
Converis on tutkimuksen hallinta- ja tietojärjestelmä, joka on suunniteltu erityisesti tutkimusorganisaatioille, kuten yliopistoille ja tutkimuslaitoksille. Se tarjoaa monipuolisia työkaluja tutkimuksen hallintaan, kuten tutkimustiedon tallennukseen, seurantaan, raportointiin ja analysointiin sekä tutkimusprosessin eri vaiheiden ja tuotosten seuraamiseen. (Clarivate.)

Yksi Converiksen keskeisistä ominaisuuksista on sen kyky hallinnoida laajaa kirjoa tutkimustietoa, kuten tutkimusprojekteja, julkaisuja, rahoitusta, patentteja, konferenssiesityksiä ja tutkijoiden tietoja. Järjestelmä auttaa tutkimusorganisaatioita tehokkaasti hallinnoimaan ja hyödyntämään tutkimustietoa rahoituksen hankkimiseen, yhteistyökumppanien löytämiseen, ja tutkimustulosten levittämiseen. Converis mahdollistaa näiden tietojen helpon ja keskitetyn hallinnan, mikä auttaa tutkimusorganisaatioita pitämään kirjaa omasta tutkimustoiminnastaan ja resursseistaan. Lisäksi Converis tarjoaa useita rajapintoja muihin tutkimushallinnon järjestelmiin ja tietokantoihin, mikä mahdollistaa sen integroinnin osaksi laajempia tietojärjestelmäkokonaisuuksia. (Clarivate.)

3.9 Fastapi

FastAPI on Python-kirjasto, joka on suunniteltu rajapintojen, eli API:en kehittämistä varten. Se on rakennettu Starlette- ja Pydantic-kirjastojen kanssa ja se tukee REST-rajapintamallia ja GraphQL-arkkitehtuuria. FastAPI vaatii Python 3.8 tai uudemman version. (FastAPI n.d.)

OpenAPI- ja JSON Schema-teknologiat ovat FastAPI-kirjaston standardien perusteet. OpenAPI-teknologialla kehitetään rajapinta ja data mallinnetaan automaattisesti JSON-kaavioiden avulla. FastAPI tarjoaa interaktiivisen rajapinnan dokumentoinnin ja testauskäyttöliittymän suoraan selaimeen. Interaktiivista dokumentoinnin käyttöliittymää varten kehittäjä voi valita haluamansa teknologian, kuten ReDoc:in tai Swagger UI:n (ks Kuvio.5). Rajapintateknologia pystyy validoimaan Python-ohjelmointikielen datatyyppejä. (FastAPI n.d.)



Kuvio 5. Swagger.

Rajapinta on erinomaisesti dokumentoitu valmiilla esimerkeillä, joten FastAPI:a on helppo opetella käyttämään. Dokumentaation ansiota oman rajapinnan pystyy kehittämään nopeasti. (FastAPI n.d.)

3.10 Figma

Figma on pilvipohjainen suunnittelutyökalu, joka on käytettävässä joko web-selaimella tai sovellus pohjalla. Figman ominaisuuksiin kuuluu versiohallinta, pilvipohjainen automaattinen tallentaminen, vektorien suunnittelu, prototyyppien luominen sekä palvelun sisäinen kommentointi tiimien ja osallistujien kesken. (Kopf 2018)

4 Julkaisuhakusivun suunnittelu

Projektin lopullinen tuote on julkinen hakusivu, jolla yksittäiset tutkijat voivat hakea julkaisujen tietoa Converis-palvelimelta. Haetut julkaisut esitetään joustavasti käyttäjälle. Vastaava toiminto on ollut aiemmin käytössä Converista edeltäneessä TUTKA-tutkimustietojärjestelmässä.

Projektin tilaajan käytiin yhdessä vaatimusmäärittely, missä määriteltiin miltä lopullisen tuotteen pitäisi näyttää ja miten sen pitäisi teknisesti toimia. Alkuperäisessä vaatimusmäärittelyssä todettiin, ettei projektille ole tarkkaa loppuvisiota, mutta tiettyjä toiveita on ja lopputuote pitää sisällään paljon samoja ominaisuuksia aikaisemmasta TUTKA alustan julkaisuhakusivusta.

4.1 Vaatimusmäärittelyn tulokset

4.1.1 Yleiset tavoitteet

Projektin yleisenä tavoitteena on toteuttaa Converis-dataan pohjautuen aikaisemman käytössä olevan TUTKA-tutkimustietojärjestelmän luomia raportteja vastaava hakusivu, samalla mahdollistaen projektin jatkokehitykselle. Hakusivun tulee tarjota loppukäyttäjälle yksinkertainen käyttöliittymä julkaisujen tietojen hakuun ja kevyeen integrointiin, esimerkkinä vakioidusta url:stä hakutulosten vienti exceliin tai toiseen web-sovellukseen. Hakusivun tulee tarjota nopea pääsy Converiksen-verkkolevyraportteja vastaaviin tietoihin julkaisujen osalta. Hakusivun tulee korvata seurantakohteiden pohjalta luotu APA-viitelista.

4.1.2 Rakenne

Hakutoiminto tulee koostumaan aloitussivusta, jossa määritellään hakuehdot julkaisuja hakiessa, selaussivusta, joka näyttää hakutulokset kompaktissa muodossa, yksittäisen julkaisun sivu, jossa näytetään yleistä tietoa käyttäjälle mahdollisimman geneerisessä muodossa ja tulostussivu, jolta tulostetaan hakutulokset HTML-taulukkona, vuosittain jaoteltuna APA-viitelistana, CSV-datana tai JSON-datana.

Sivut toteutetaan teknisesti yhtenä tai useampana päätteenä. Kaikille sivuille toteutetaan yhtenäiset tyylit JYU:n brändisääntöjä noudattaen. Sivut tulisi toteuttaa niin, että ne olisivat valmiina monikieliseen toteutukseen parametreilla toteuttamalla.

4.1.3 Aloitus

Aloitussivulle tulee pystyä antamaan HTTP-parametreilla hakuehdot, jolloin siirryttäessä selaussivulta takaisin aloitukseen hakua voidaan muuttaa.

4.1.4 Keskeisiä hakukenttiä

Julkaisuhakusivun keskeisimmät hakukentät sisältävät mahdollisuuden tiedekunnan tai yksikön hakulistan näyttämiseen.

Tuloksissa tulee näkyä:

- Nimi valitulla kielellä sekä yksikön lyhenne
- Haku tiedekunnan mukaan sisällyttää mukaan sekä tiedekunnan että tämän alaiset laitokset, jos niitä on. Erillislaitokset näytetään myös omana tiedekuntana.
- Haku yksikön mukaan sisällyttää vain valitut yksiköt
- Tuloksia pitää voida valita useita tai jättää tyhjäksi

Hakukentällä haetaan julkaisuvuosi tai vuosirajaus, jossa yksittäinen vuosi näytetään oletuksena:

- Oletuksena valittu kuluva vuosi, joka voidaan myös jättää tyhjäksi

- Parametrien käytön logiikka kuten vuosiparametri, joka syrjäyttää *startYear/endYear*-parametrit ja tyhjään *startYear/endYear*-parametriin täydennetään automaattisesti vuodet 1900/2100.

Tuloksien kanssa näytetään myös julkaisujen seurantakohteet. Seurantakohteita tuloksissa tulisi näyttää:

- Hakuelementissä näytetään seurantakohteen nimi, lyhenne sekä suluissa liittyvien yksiköiden lyhenteet
- Pohjatiedot voidaan hakea käyttämällä SQL-haulla. Mukaan otetaan vain aktiiviset ja valittavissa olevat varsinaiset seurantakohteet tai profilointiala.
- Voidaan valita useita tai jättää tyhjäksi.

Julkaisuilla on myös useita eri julkaisutyyppejä. Tuloksissa tulisi näyttää listassa vain karkeat julkaisutyypit. Julkaisutyyppejä voidaan valita useita tai jättää tyhjäksi. Jos käyttäjä ei valitse mitään, tulisi hakea koko yliopiston kaikki julkiset julkaisut kuluvan vuoden ajalta.

4.1.5 Hakua tarkentavia kenttiä

Julkaisuhakusivun ominaisuuksiin tarvitaan hakua tarkentavia kenttiä, jotka toimivat rastitettavilla laatikoilla, joilla voidaan ohjata tietyn kentän näyttämiseen:

- Julkaisukanavan tiedot, joka on valittu oletuksena tai Julkaisun tekijät tai Julkaisun tiedot viitemuodossa
- Jufo-tiedot, joka on rastitettu oletuksena
- Julkaisuun liittyvät yksiköt
- Julkaisuun liittyvät seurantakohteet
- Laajat kuvailutiedot, johon kuuluu tiivistelmä ja asiasanat
- Open Access -tiedot
- OKM-raportointitiedot

Hakutulosten esitystapaa tarkentavia kenttiä implementoidaan seuraavasti:

- Tulokset voidaan järjestää julkaisuvuoden mukaan oletuksena, tekijälistan mukaan, julkaisun otsikon mukaan. Järjestettäessä julkaisuvuoden mukaan järjestetään

toissijaisesti tekijän ja julkaisun otsikon mukaan, jotta tulosten järjestys esitetään luontevammin, vaikka haettaisiin vain yhden vuoden julkaisut
- Ryhmittele tulokset. Oletuksena ei ryhmittelyä, julkaisutyypin mukaan, Jufo-luokan mukaan, ja OKM-hyväksyntästatuksen mukaan.

4.1.6 Selaus

Selausnäkymässä perusnäkymässä näytetään hakutulos tarkastelua varten. Sivulla voi myös muokata hakuehtoja samoilla HTML-elementeillä kuin aloitussivulla, vaikka hakuelementit esitettäisiin toisella tavalla asemoituna, kuten listan kulmaan ryhmittely- tai järjestämisehtoja varten tai vasempaan reunaan tulosten rajausta varten.

SQL-haku muodostetaan osittain dynaamisesti, hakuehdot ja tietyt valinnaisesti näytettävät tiedot eritellään useamman taulun yli menevien tietojen osalta. Toteutuksessa voidaan hakea kaikki tiedot. SQL:n tietojen määrää voidaan rajata käyttäjän valitsemaan määrää 1–10000 julkaisun väliltä. Jos hakutulosten määrä ylittää tämän, tästä ilmoitetaan käyttäjälle.

Selaussivulla näytetään huomattavasti rajatumpi määrä julkaisuja kerrallaan, maksimissaan 100 sivulla, joka on myös käyttäjän muokattavissa oleva määrä.

Jos hakuehdoissa on valittu ryhmittely, ryhmistä tehdään väliotsikoita kuten Jufo-luokat 3, 2, 1, ja tyhjä.

Listassa jokaisesta julkaisusta näytettävät tiedot:

- Julkaisutyyppi julkaisukohtaisessa header-elementissä
- JY-tekijät. JY:n org. yksiköt jos rastitettu hakuehdoissa. Seurantakohteet (jos rastitettu).
- Riippuen valinnasta näytetään aluksi julkaisun tekijät, otsikko ja julkaisuvuosi tai otsikko, julkaisuvuosi ja kokoomateoksen tai lehden/sarjan nimi tai APA-viite. Näiden jälkeen, jos rastitettuna niin näytetään Jufo-luokka, avoin julkaisukanava, avoin julkaisu ja rinnakkaisjulkaisu-tiedot. Laajenna-linkki näyttää ponnahdusikkunassa yksittäisen julkaisun tarkemmat tiedot
- Julkaisukohtaisessa footer-elementissä näytetään linkki julkaisun sivulle Converiksessa, CPR ja linkki Dimensionsiin avulla. Linkki julkaisun kustantajaan ja linkki julkaisun OA-versioon. Viittaus-linkki, joka palauttaa APA-viitteen.

4.1.7 Yksittäinen julkaisu

Yksittäisen julkaisun näyttävällä sivulle ei tule tehdä suuren julkaisumassan hakua tietokannasta. Toteutusjärjestyksessä yksittäisen julkaisun sivun voi priorisoida muun toiminnallisuuden jälkeen. Yksittäisen julkaisun sivu toteutetaan mahdollisimman yleisenä ja ilman linkityksiä ennen jatkokehitystä.

Näytettävät tiedot esitetään listoissa ja taulukoissa, jotka ryhmitellään kohtiin perustiedot, kuvailutiedot ja luokittelutiedot. Näytettävät kentät riippuvat hakuehdoista.

Julkaisun perustiedoissa tulee näkyä:

- Julkaisutyyppiryhmä
- Julkaisun otsikko
- Julkaisuvuosi
- Julkaisupäivämäärä, jos on tiedossa
- Julkaisuun liittyvät JYU-tekijät
- Julkaisuun liittyvät yksiköt tiedekunta ja erillislaitokset-tasolla, jos liittyvät yksiköt rastitettu
- Julkaisuun liittyvät yksiköt, jos ne on rastitettu
- Verkko-osoite
- Avoimen julkaisun osoite
- Dimensions-linkki, jos on tiedossa
- APA-viite
- Converis-tunniste, jossa on linkki julkaisun Converis-portaalisivulle.

Julkaisun kuvailutiedoissa tulee näkyä:

- Laajat kuvailutiedot: Tiivistelmä
- Laajat kuvailutiedot: Kaikki tekijät
- Emojulkaisun nimi
- Laajat kuvailutiedot: Emojulkaisun toimittajat
- Lehden tai sarjan nimi
- Laajat kuvailutiedot: Konferenssin vakiintunut nimi
- Laajat kuvailutiedot: Kustantajan nimi
- Laajat kuvailutiedot: Vapaat asiasanat
- Laajat kuvailutiedot: YSO-asiasanat
- Laajat kuvailutiedot: liittyvät hankkeet, vain julkiset, lyhyellä kuvauksella
- Laajat kuvailutiedot: liittyvät tutkimusaineistot, vain julkiset, lyhyellä kuvauksella.

Julkaisun luokittelutiedoissa tulee näkyä:

- Julkaisutyyppi
- Laajat kuvailutiedot: Onko vertaisarvioitu

- Laajat kuvailutiedot: Kansainvälisyys
- Laajat kuvailutiedot: Tieteenala
- OA-tiedot: Julkaisukanavan avoimuus
- OA-tiedot: Julkaisun avoimuus kustantajan palvelussa
- OA-tiedot: Onko rinnakkaistallennettu
- Jufo-tiedot: Jufo-taso
- Jufo-tiedot: Julkaisukanavan tunniste
- Jufo-tiedot: Julkaisukanavan nimi
- Seurantakohteet ja profilointialat, jos rastitettu
- CPR, jos on tiedossa
- OKM-raportointitiedot: OKM-raportointi
- OKM-raportointitiedot: Ministeriöraportoinnin tila
- OKM-raportointitiedot: Raportointivuosi.

4.1.8 Tulostukset

Tulostussivulta tulee pystyä tulostamaan hakutulokset HTML-taulukkona, vuosittain jaoteltuna APA-viitelistana, CSV-datana tai JSON-datana.

HTML-taulukossa yksittäisen tuloksen tiedot vastaavat tulossivun tuottamia tuloksia. Sivun alkuun tunnistettava teksti ”Julkaisuhaku JYU-yksikön mukaan (pp.kk.vvvv)”, hakuehdot joissa näkee valitut yksiköt, seurantakohteet ja vuodet. HTML-viitelistaan tulee vuosittain jaoteltu APA-viitelista.

CSV-datan taulukossa hakutulokset näytetään CSV:nä joka aukeaa Microsoft Excelissä sellaisenaan. Näytettävät kentät ovat yksittäisen tuloksen näytössä koottujen pohjalta, ja niissä näkyy myös julkaisun tila.

JSON-datan taulukossa on sama tieto kuin CSV-taulukossa, mutta data tulostetaan JSON:ina. JSON-viitelistassa on jaoteltu APA-viitelista JSON-datana.

4.1.9 Ohjelmistokehykset

Huomioon ottaen ohjelmistokehysten monipuolisuuden sekä niiden tarjoaman laajan valikoiman, projektiin sopivan ohjelmistokehyksen vaatimukset tulee määrittää. Digipalveluille tärkeimmät periaatteet ohjelmistokehyksen valinnassa ovat tietoturvallisuus, helppo käyttöönotettavuus, kehityksen helpottaminen ja nopeuttaminen, ja että ohjelmistokehys ei hidasta palvelun tarjontaa.

4.2 Ohjelmistokehykset

Projektin toteuttamisen aloitusvaiheessa pidettiin palaveri toiveista projektin kehitykseen käytettävästi ohjelmistokehyksestä. Projektin toteutusta varten tavoitteena oli löytää sopiva ohjelmistokehys, joka tukee kehitystä ja jonka käyttöönottoa voidaan hyödyntää muissa Digipalveluiden palveluiden jalostamisessa. Aloituskohteena oli toimeksiantajalle valmiiksi tutut ohjelmistokehykset Flask ja FastAPI. Tässä kappaleessa käymme läpi ominaisuuksia näiden ohjelmistokehyksien välillä ja niiden sopivuudesta tähän projektiin.

4.2.1 FastAPI

FastAPI tarjoaa sisäänrakennetun interaktiivisen API-dokumentaation, joka perustuu OpenAPI-standardiin. Tämä interaktiivinen dokumentaatio luodaan automaattisesti perustuen FastAPI-sovelluksen määriteltyihin reitteihin ja niiden metatietoihin. Ketterämmän kehityksen kannalta FastAPI:n voi yhdistää Visual Studio Code:n tai Pycharmin debuggeriin. (FastAPI n.d.)

FastAPI käyttää Pydantic-malleja reittien sisään- ja ulostulodatan määrittelyyn. Näitä malleja käytetään myös API-dokumentaation luomiseen. Kun FastAPI-sovelluksesi käynnistetään ja käytetään projektin osoitetta /docs, FastAPI generoi interaktiivisen dokumentaation, joka sisältää:

1. Kaikki määrittellyt reitit näkyvät dokumentaatiossa. Tämä sisältää reittien polut, HTTP-menetelmät ja polun parametrit.
2. Pydantic-mallit: Dokumentaatiossa näytetään Pydantic-mallit, jotka olet määritellyt sisään- ja ulostulodatan validointiin. Tämä auttaa ymmärtämään, millaisia tietoja reitit hyväksyvät ja palauttavat.
3. Esimerkkipyyntöjä ja -vasteita: Dokumentaatiossa näytetään esimerkkipyyntöjä ja -vasteita, jotka perustuvat määriteltyihin Pydantic-malleihin. Tämä mahdollistaa selkeästi näkemään, millaista dataa API hyväksyy ja palauttaa.
4. Mahdollisuus testata APIa: Interaktiivisessa dokumentaatiossa on sisäänrakennettu API-testaaja, jonka avulla voit lähettää pyyntöjä API:lle suoraan dokumentaation kautta ja nähdä vastaukset reaaliajassa.

Tätä interaktiivista dokumentaatiota voidaan käyttää kommunikointiin ja API:n käyttämisen helpottamiseen. Se myös toimii hyvänä itseopiskelumateriaalina ja auttaa näyttämään miten API toimii ja miten sitä käytetään. (Troy 2021.)

FastAPI käyttää Pydantic-kirjastoa, joka tarjoaa FastAPI:in sisäänrakennettuna datavalidaatiota ja asetusten hallinnointia käyttämällä Pythonin annotaatioita. Datavalidaatio prosessina varmistaa että ohjelmaan syötetty data on yhdenmukaista ja oikein sille tarkoitetulle ohjelmalle. Tämä varmistaa että vääräntyyppistä dataa ei pääse palvelimelle. (Sahai 2024.)

FastAPI käyttää myös Jinja2 mallipakettia. Jinja2 käytetään dynaamisen sisällön luomiseen websovelluksissa. Jinja2:n nimi ”template model” viittaa sen rakenteeseen ja ominaisuuksiin, joita käytetään mallien luomiseen, jotka ovat tiedostoja, jotka pitävät sisällään valmiita koodipaketteja ja logiikkaa, jotka korvataan websovelluksen oikealla sisällöllä, kun se renderöidään. (Jinja n.d.)

FastAPI perustuu ASGI pohjaiseen Starlette ohjelmistokehykseen, kehittämisessä voidaan käyttää ASGI middlewarea. FastAPI ASGI tukee asynkronoituja taskeja. FastAPI:n virhe(error) viestit tulevat JSON formaatissa. FastAPI:in on sisällytetty HTTP-pyynnöt ja XML/JSON-vastaukset. (FastAPI n.d.)

FastAPI käyttää SSL/TLS-salausmetodeja. (FastAPI n.d.) Metodissa käytetään sertifikaattia, joka on digitaalinen objekti, joka mahdollistaa systeemien vahvistaa käyttäjän henkilöllisyys ja luoda salattu yhteys toiseen systeemiin käyttämällä SSL/TLS-protokollaa. SSL/TLS tulee sanoista Secure Socket Layer/Transport Layer Security. SSL/TLS-salauksen menetelmä koostuu useista osista, jotka yhdessä muodostavat turvallisen tietoliikenteen verkon yli. (F5 n.d.)

Yhteyden muodostamiseen käyttäjän ja palvelimen välillä kuuluu seuraavat askeleet:

1. Käyttäjä ottaa yhteyden palvelimeen käyttäen turvallista URL-osoitetta, kuten HTTPS.
2. Palvelin lähettää käyttäjälle sertifikaatin ja julkisen avaimen.
3. Käyttäjä varmistaa TRCA:n(Trusted Root Certification Authority) kanssa että sertifikaatti on luotettava.
4. Käyttäjä ja palvelin keskustelevat keskenään vahvimmasta salausmetodista, jota molemmat osapuolet voivat käyttää.

5. Käyttäjä salaa session salaisella avaimella käyttäen palvelimen antamaa julkista avainta ja lähettää sen takaisin palvelimelle.
6. Palvelin avaa yhteyden käyttäjän kanssa käyttäen käyttäjän yksityistä avainta.
7. Yhteyden sessioavainta käytetään datan kryptaamiseen ja purkamiseen käyttäjän ja palvelimen välillä.

FastAPI käyttää OAuth-autentikointiprosessia. (FastAPI n.d.) OAuth on avoimella lähdekoodilla oleva standardi, jota käytetään valtuuttamaan sovellusten pääsy resursseihin toisen palvelun käyttäjän tilillä ilman salasanan jakamista. Se tarjoaa yksinkertaisen ja turvallisen tavan antaa sovelluksille pääsy käyttäjän tietoihin eri verkkopalveluissa, kuten sosiaalisen median alustoilla tai pilvipalveluissa.

OAuth autentikointiprosessi toimii seuraavasti:

1. Sovellus rekisteröidään palveluntarjoajan kehittäjäportaalissa ja sille annetaan tunnisteet, joita se käyttää kommunikoidessaan palveluntarjoajan kanssa.
2. Kun käyttäjä yrittää käyttää sovellusta ensimmäistä kertaa, hänet ohjataan palveluntarjoajan kirjautumissivulle, jossa hän voi antaa suostumuksensa sovelluksen käyttöön.
3. Käyttäjä kirjautuu sisään palveluntarjoajan tilille ja antaa sovellukselle luvan käyttää käyttäjälle tarkoitettuja resursseja.
4. Palveluntarjoaja antaa sovellukselle väliaikaisen autentikointiavaimen, jota sovellus voi käyttää lähettämään pyyntöjä palveluntarjoajan API:in.
5. Sovellus käyttää saamaansa autentikointiavainta hakeakseen käyttäjän sallimat resurssit palveluntarjoajan API:sta.

4.2.2 Flask

Flask on yhteensopiva mod_wsgi paketin kanssa hyödyntäen sisäänrakennettua mod_wsgi-express komentoa, joka konfiguroi ja käynnistää palvelimen ilman tarvetta kirjoittaa Apachen web-palvelimen konfiguraatioita. (Flask n.d.)

Flaskilla on myös sisäänrakennettu debugger sekä kehityspalvelin. Flaskilla ei ole sisäänrakennettua datavalidointia, validointi on koodattava itse tai käytettävä erillisiä lisäosia, kuten Flask-Marshmallow tai Flask-Inputs. Näiden lisäosien ollessa erikseen kehitettyjä ne voivat tuoda yhteensopivuusongelmia. (Flask n.d.)

Flask-Security tarjoaa laajan valikoiman turvallisuus ja kirjautumismenetelmiä moduulipakettien muodossa, näitä paketteja ovat esimerkiksi Flask-Login, Flask-Principal, Flask-WTF, passlib, authlib ja webauthn. Nämä tarjoavat laajan valikoiman turvallisuuteen liittyviä ominaisuuksia kuten istuntokohtaisen autentikoinnin, roolien ja oikeuksien hallinnoinnin, salasanojen salauksen, HTTP ja merkki (token) pohjainen autentikoinnin, kaksiosaisen tunnistautumisen, käyttäjien rekisteröinnin ja kirjautuminen sekä OAuth-autentikoinnin. (Flask-security-too n.d.)

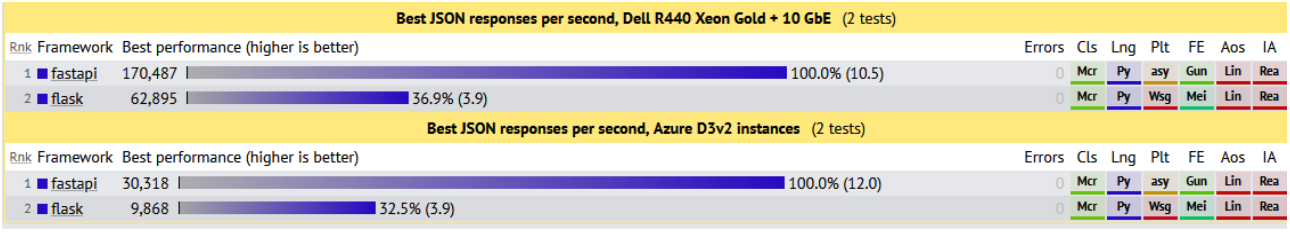
Flask-Marshmallow käytetään kompleksien datatyyppien, kuten JSON-datan, validoimiseen ja muuttamiseen ohjelmistokäyttöön. Flask Marshmallow kirjastona tuo monia ominaisuuksia, kuten objektien sarjallistaminen ja deserilisaatio, Python-objektien muuttamisen JSON-dataksi ja JSON-datan muuttamisen Python-objekteiksi. Flask-Marshmallow tuo mukanaan myös virheenkäsittelypaketin. Flask-Marshmallow mahdollistaa tietorakenteiden määrittämisen käyttäen Marshmallowin kaavioluokkia. (Flask-Marshmallow n.d.)

Flask-Inputs kirjasto tuo mukanaan datavalidointimetodeja. Tällä varmistetaan, että ohjelmaan syötetty data on yhdenmukaista ja oikein sille tarkoitetulle ohjelmalle. Flask-Inputs tukee myös JSON-kaavioiden validointia. (Flask-Inputs n.d.)

4.2.3 Vertailu

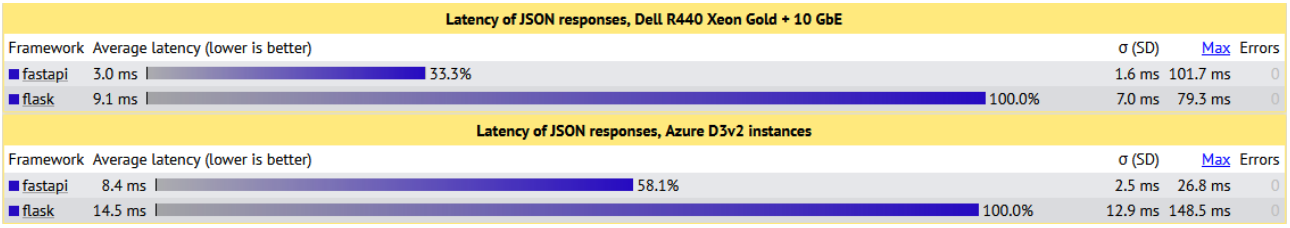
Ohjelmistokehysten vertailu perustuu Techempowerin tekemiin vertailuarvoihin. Techempower on testannut ohjelmistokehysten suorituskykyä eri kategorioissa. Testit on suoritettu 2 ympäristössä, fyysisessä sekä virtuaalisessa. Fyysiset testit on suoritettu Dell R440 Xeon Gold-palvelimella ja 1 gigabitin internetyhteydellä. Virtuaaliset testit on suoritettu Azure D3v2 virtuaalisella koneella. Jokaisen kategorian kanssa on tehty myös viivetesti, jotka on myös suoritettu sekä fyysisessä ja virtuaalisessa ympäristössä. Fyysiset ja visuaaliset ympäristöt ovat samoja kuin suorituskykyä mitattaessa. Testit on suoritettu päivämääränä 02. elokuuta 2021. (Techempower 2021.)

Kuviossa 6 nähdään testitulokset, jossa jokainen vastaus on JSON-sarjoitus juuri käynnistetystä objektista, joka yhdistää avainviestin arvoon ”Hello, World!” Testi harjoittaa ohjelmistokehyksen perusteita, kuten palvelupyyntöjen reititystä, olioiden instanssien luomista, header-elementin luomista ja palvelupyyntöjen laskemista. Testi on suoritettu fyysisellä palvelimella, Dell R440 Xeon Gold palvelimella ja 1 gigabitin internetyhteydellä. (Techempower 2021.)



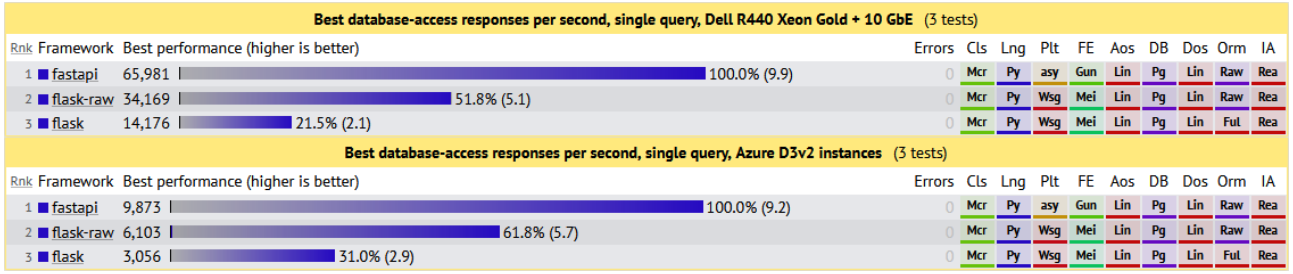
Kuvio 6. JSON.

Kuviossa 7 nähdään JSON-sarjoitustestin viiveajat mitattavien ohjelmistokehysten välillä. (Techempower 2021.)



Kuvio 7. JSON-viive.

Kuviossa 8 nähdään testitulokset, jossa jokainen pyyntö on prosessoitu hakemalla rivi dataa tietokannasta ja se rivi on muutettu JSON-tyyppiseksi. Testi harjoittaa ohjelmistokehyksen objekti relaatiokartoitusta (ORM), satunnaisten numeroiden luomista, tietokannan ajureita ja tietokannan yhteyksiä. (Techempower 2021.)



Best database-access responses per second, single query, Dell R440 Xeon Gold + 10 GbE (3 tests)

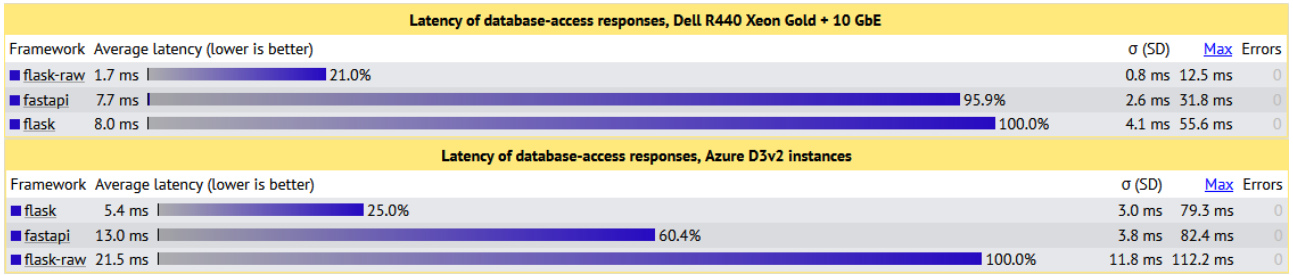
Rnk	Framework	Best performance (higher is better)	Errors	Cls	Lng	Plt	FE	Aos	DB	Dos	Orm	IA
1	fastapi	65,981 100.0% (9.9)	0	Mcr	Py	asy	Gun	Lin	Pg	Lin	Raw	Rea
2	flask-raw	34,169 51.8% (5.1)	0	Mcr	Py	Wsg	Mei	Lin	Pg	Lin	Raw	Rea
3	flask	14,176 21.5% (2.1)	0	Mcr	Py	Wsg	Mei	Lin	Pg	Lin	Ful	Rea

Best database-access responses per second, single query, Azure D3v2 instances (3 tests)

Rnk	Framework	Best performance (higher is better)	Errors	Cls	Lng	Plt	FE	Aos	DB	Dos	Orm	IA
1	fastapi	9,873 100.0% (9.2)	0	Mcr	Py	asy	Gun	Lin	Pg	Lin	Raw	Rea
2	flask-raw	6,103 61.8% (5.7)	0	Mcr	Py	Wsg	Mei	Lin	Pg	Lin	Raw	Rea
3	flask	3,056 31.0% (2.9)	0	Mcr	Py	Wsg	Mei	Lin	Pg	Lin	Ful	Rea

Kuvio 8. Single query.

Kuviossa 9 nähdään yksittäisen kyselyn testin viiveajat mitattavien ohjelmistokehysten välillä. (Techempower 2021.)



Latency of database-access responses, Dell R440 Xeon Gold + 10 GbE

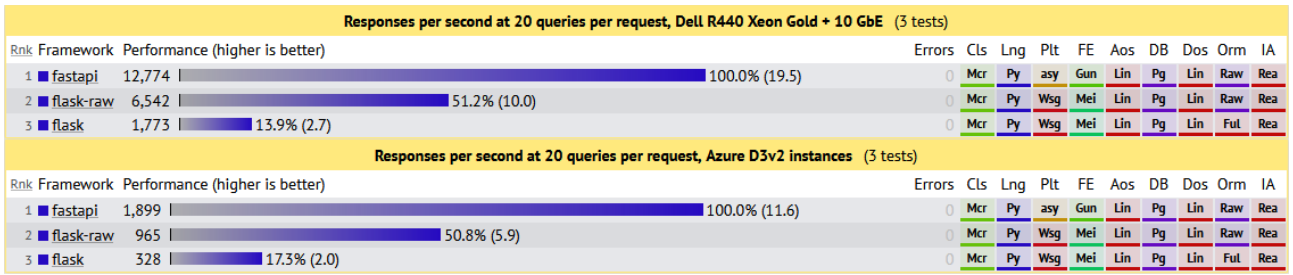
Framework	Average latency (lower is better)	σ (SD)	Max	Errors
flask-raw	1.7 ms 21.0%	0.8 ms	12.5 ms	0
fastapi	7.7 ms 95.9%	2.6 ms	31.8 ms	0
flask	8.0 ms 100.0%	4.1 ms	55.6 ms	0

Latency of database-access responses, Azure D3v2 instances

Framework	Average latency (lower is better)	σ (SD)	Max	Errors
flask	5.4 ms 25.0%	3.0 ms	79.3 ms	0
fastapi	13.0 ms 60.4%	3.8 ms	82.4 ms	0
flask-raw	21.5 ms 100.0%	11.8 ms	112.2 ms	0

Kuvio 9. Single query-viive.

Kuviossa 10 jokainen pyyntö on prosessoitu hakemalla rivi dataa tietokannasta ja se rivi on muutettu JSON-tyyppiseksi. Tämä testi on toteutettu usean kerran, kerroin 1, 5, 10, 15 ja 20 riviä dataa per pyyntö. Testissä haetaan useita rivejä dataa testatakseen ohjelmistokehyksen rajoja tietokannan ajureiden sekä yhteyksien kanssa. (Techempower 2021.)



Responses per second at 20 queries per request, Dell R440 Xeon Gold + 10 GbE (3 tests)

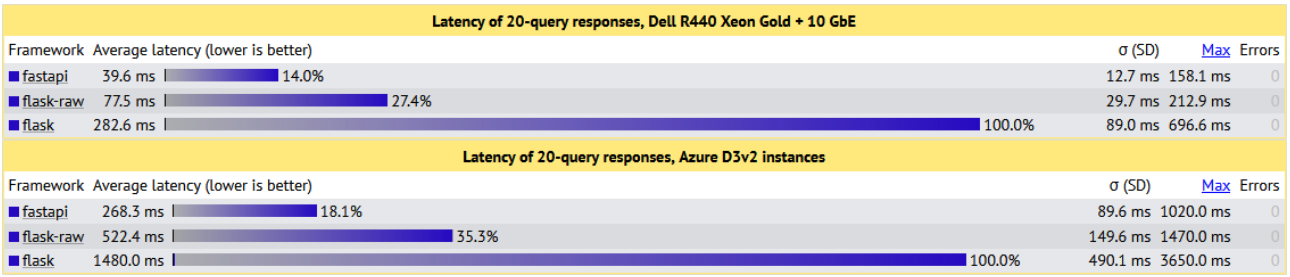
Rnk	Framework	Performance (higher is better)	Errors	Cls	Lng	Plt	FE	Aos	DB	Dos	Orm	IA
1	fastapi	12,774 100.0% (19.5)	0	Mcr	Py	asy	Gun	Lin	Pg	Lin	Raw	Rea
2	flask-raw	6,542 51.2% (10.0)	0	Mcr	Py	Wsg	Mei	Lin	Pg	Lin	Raw	Rea
3	flask	1,773 13.9% (2.7)	0	Mcr	Py	Wsg	Mei	Lin	Pg	Lin	Ful	Rea

Responses per second at 20 queries per request, Azure D3v2 instances (3 tests)

Rnk	Framework	Performance (higher is better)	Errors	Cls	Lng	Plt	FE	Aos	DB	Dos	Orm	IA
1	fastapi	1,899 100.0% (11.6)	0	Mcr	Py	asy	Gun	Lin	Pg	Lin	Raw	Rea
2	flask-raw	965 50.8% (5.9)	0	Mcr	Py	Wsg	Mei	Lin	Pg	Lin	Raw	Rea
3	flask	328 17.3% (2.0)	0	Mcr	Py	Wsg	Mei	Lin	Pg	Lin	Ful	Rea

Kuvio 10. Multiple query.

Kuviossa 11 nähdään useamman kyselyn testin viiveajat mitattavien ohjelmistokehysten välillä. (Techempower 2021.)



Latency of 20-query responses, Dell R440 Xeon Gold + 10 GbE

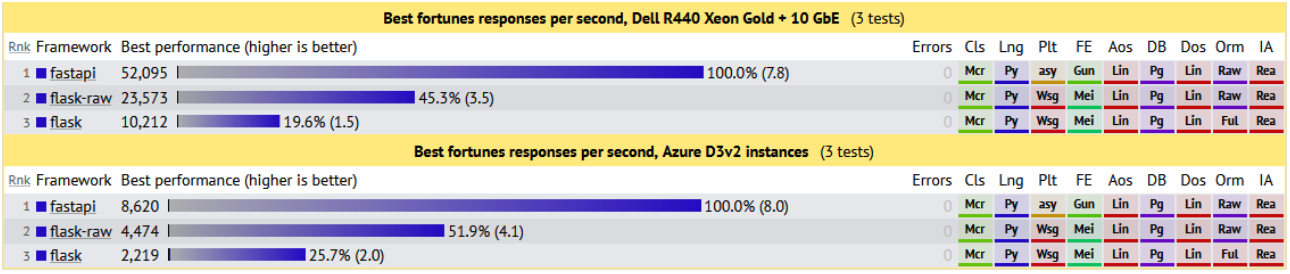
Framework	Average latency (lower is better)		σ (SD)	Max	Errors
fastapi	39.6 ms	14.0%	12.7 ms	158.1 ms	0
flask-raw	77.5 ms	27.4%	29.7 ms	212.9 ms	0
flask	282.6 ms	100.0%	89.0 ms	696.6 ms	0

Latency of 20-query responses, Azure D3v2 instances

Framework	Average latency (lower is better)		σ (SD)	Max	Errors
fastapi	268.3 ms	18.1%	89.6 ms	1020.0 ms	0
flask-raw	522.4 ms	35.3%	149.6 ms	1470.0 ms	0
flask	1480.0 ms	100.0%	490.1 ms	3650.0 ms	0

Kuvio 11. Multiple query-viive.

Fortune-testissä ohjelmistokehyksen ORM käytetään hakemaan kaikki rivit tietokannasta, jotka sisältävät tuntemattoman määrän Unix:in evästeiden viestejä. (KS. Kuvio 12.) Ylimääräinen evästeviesti lisätään listaan ajamisen aikana ja lista lajitellaan viestin tekstin mukaan. Lista toimitetaan käyttäjälle käyttämällä palvelimen HTML-mallia. Tällä testillä testataan objekti relaatiokartoitusta (ORM), tietokannan yhteyksiä, lajittelua, palvelimen puolen malleja, XSS-vastatoimia ja merkkien koodaamista. (Techempower 2021.)



Best fortunes responses per second, Dell R440 Xeon Gold + 10 GbE (3 tests)

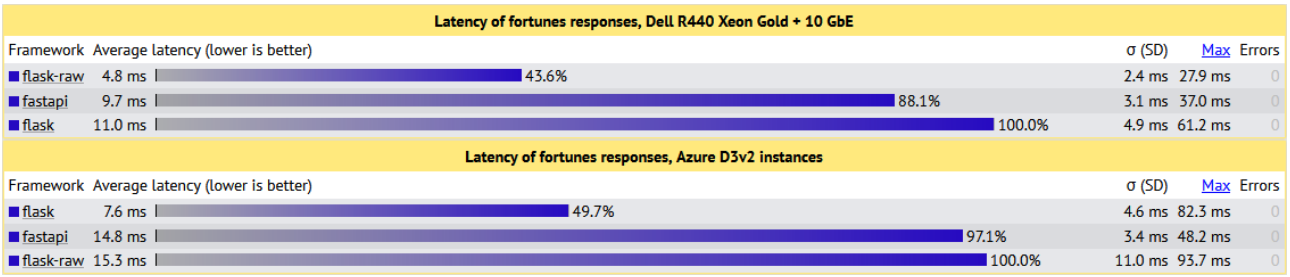
Rnk	Framework	Best performance (higher is better)		Errors	Cls	Lng	Plt	FE	Aos	DB	Dos	Orm	IA
1	fastapi	52,095	100.0% (7.8)	0	Mcr	Py	asy	Gun	Lin	Pg	Lin	Raw	Rea
2	flask-raw	23,573	45.3% (3.5)	0	Mcr	Py	Wsg	Mei	Lin	Pg	Lin	Raw	Rea
3	flask	10,212	19.6% (1.5)	0	Mcr	Py	Wsg	Mei	Lin	Pg	Lin	Ful	Rea

Best fortunes responses per second, Azure D3v2 instances (3 tests)

Rnk	Framework	Best performance (higher is better)		Errors	Cls	Lng	Plt	FE	Aos	DB	Dos	Orm	IA
1	fastapi	8,620	100.0% (8.0)	0	Mcr	Py	asy	Gun	Lin	Pg	Lin	Raw	Rea
2	flask-raw	4,474	51.9% (4.1)	0	Mcr	Py	Wsg	Mei	Lin	Pg	Lin	Raw	Rea
3	flask	2,219	25.7% (2.0)	0	Mcr	Py	Wsg	Mei	Lin	Pg	Lin	Ful	Rea

Kuvio 12. Fortune.

Kuviossa 13 nähdään Fortune-testin viiveajat mitattavien ohjelmistokehysten välillä.



Latency of fortunes responses, Dell R440 Xeon Gold + 10 GbE

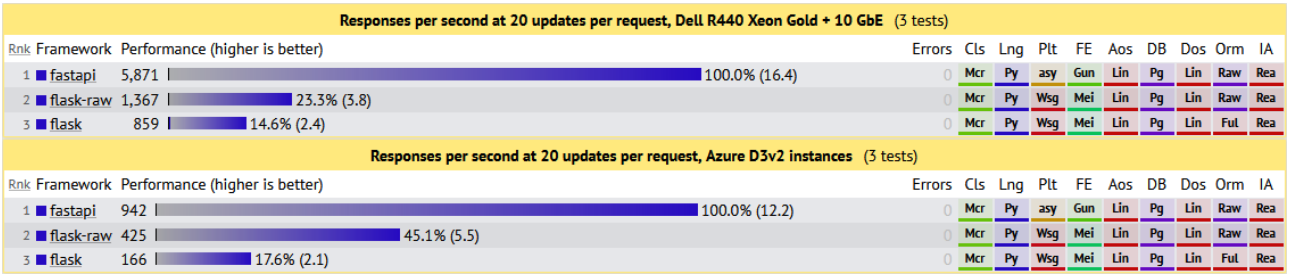
Framework	Average latency (lower is better)	σ (SD)	Max	Errors
flask-raw	4.8 ms 43.6%	2.4 ms	27.9 ms	0
fastapi	9.7 ms 88.1%	3.1 ms	37.0 ms	0
flask	11.0 ms 100.0%	4.9 ms	61.2 ms	0

Latency of fortunes responses, Azure D3v2 instances

Framework	Average latency (lower is better)	σ (SD)	Max	Errors
flask	7.6 ms 49.7%	4.6 ms	82.3 ms	0
fastapi	14.8 ms 97.1%	3.4 ms	48.2 ms	0
flask-raw	15.3 ms 100.0%	11.0 ms	93.7 ms	0

Kuvio 13. Fortune-viive.

Kuviossa 14 nähdään tulokset testille, jossa testataan ohjelmistokehyksen kykyä datan päivittämiselle. Testissä jokainen palvelupyyntö prosessoidaan hakemalla usea rivi dataa yksittäisestä tietokannan taulusta, muuttamalla nuo rivit muistissa oleviksi objekteiksi, muuttamalla attribuutin jokaisesta objektista, päivittämällä erikseen jokaisen rivin dataa tietokannassa ja sarjoittamalla listan objekteista JSON-tyyppiseksi vastaukseksi. Tämä testi ajetaan useamman kerran, 1, 5, 10, 15 ja 20 kertaa jokaista palvelupyyntöä kohden. Palvelupyynnöillä testataan ohjelmistokehyksen kykyä luku- ja kirjoitustyylisille tietokantaoperaatioille. (Techempower 2021.)



Responses per second at 20 updates per request, Dell R440 Xeon Gold + 10 GbE (3 tests)

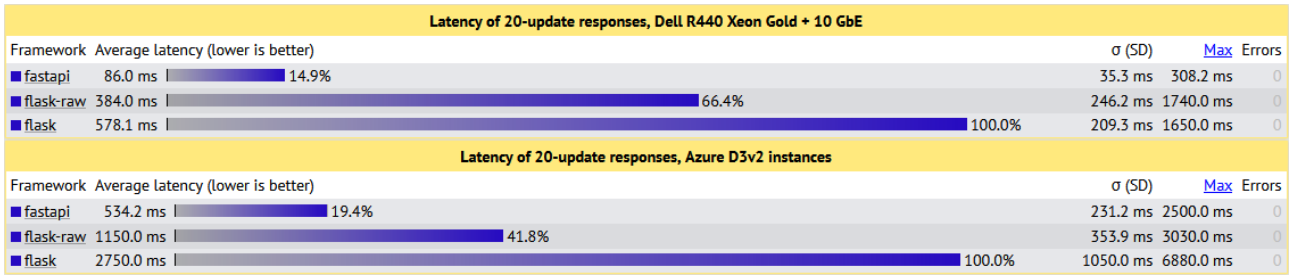
Rnk	Framework	Performance (higher is better)	Errors	Cls	Lng	Plt	FE	Aos	DB	Dos	Orm	IA
1	fastapi	5,871 100.0% (16.4)	0	Mcr	Py	asy	Gun	Lin	Pg	Lin	Raw	Rea
2	flask-raw	1,367 23.3% (3.8)	0	Mcr	Py	Wsg	Mei	Lin	Pg	Lin	Raw	Rea
3	flask	859 14.6% (2.4)	0	Mcr	Py	Wsg	Mei	Lin	Pg	Lin	Ful	Rea

Responses per second at 20 updates per request, Azure D3v2 instances (3 tests)

Rnk	Framework	Performance (higher is better)	Errors	Cls	Lng	Plt	FE	Aos	DB	Dos	Orm	IA
1	fastapi	942 100.0% (12.2)	0	Mcr	Py	asy	Gun	Lin	Pg	Lin	Raw	Rea
2	flask-raw	425 45.1% (5.5)	0	Mcr	Py	Wsg	Mei	Lin	Pg	Lin	Raw	Rea
3	flask	166 17.6% (2.1)	0	Mcr	Py	Wsg	Mei	Lin	Pg	Lin	Ful	Rea

Kuvio 14. Data update.

Kuviossa 15 nähdään datan päivitystestin viiveajat mitattavien ohjelmistokehysten välillä. (Techempower 2021.)



Latency of 20-update responses, Dell R440 Xeon Gold + 10 GbE

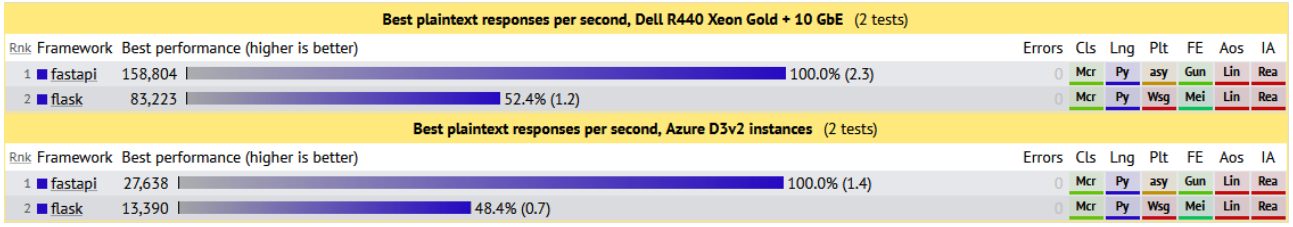
Framework	Average latency (lower is better)	σ (SD)	Max	Errors
fastapi	86.0 ms 14.9%	35.3 ms	308.2 ms	0
flask-raw	384.0 ms 66.4%	246.2 ms	1740.0 ms	0
flask	578.1 ms 100.0%	209.3 ms	1650.0 ms	0

Latency of 20-update responses, Azure D3v2 instances

Framework	Average latency (lower is better)	σ (SD)	Max	Errors
fastapi	534.2 ms 19.4%	231.2 ms	2500.0 ms	0
flask-raw	1150.0 ms 41.8%	353.9 ms	3030.0 ms	0
flask	2750.0 ms 100.0%	1050.0 ms	6880.0 ms	0

Kuvio 15. Data update-viive.

Kuviossa 16 nähdään viimeinen Techempowerin testi ohjelmistokehysten vertailukohteille. Ohjelmistokehys vastaa yksinkertaisimpiin palvelupyyntöihin, joka on ”Hello, World” -viestin renderöiminen raakatekstiksi. Palvelupyynnön vastaus pidetään pienenä, jotta internetyhteys ei ole vaikuttava tekijä testissä. Testin tarkoituksena on testata ohjelmistokehyksen palvelupyyntöjen reitittämisen perusteita. (Techempower 2021.)



Best plaintext responses per second, Dell R440 Xeon Gold + 10 GbE (2 tests)

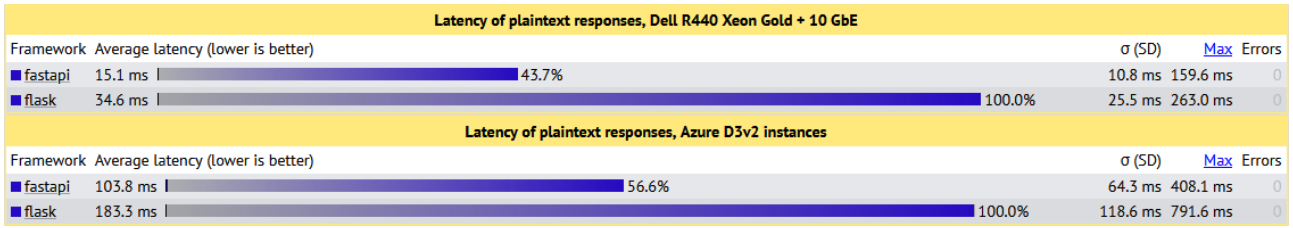
Rnk	Framework	Best performance (higher is better)	Errors	Cls	Lng	Plt	FE	Aos	IA
1	fastapi	158,804 100.0% (2.3)	0	Mcr	Py	asy	Gun	Lin	Rea
2	flask	83,223 52.4% (1.2)	0	Mcr	Py	Wsg	Mei	Lin	Rea

Best plaintext responses per second, Azure D3v2 instances (2 tests)

Rnk	Framework	Best performance (higher is better)	Errors	Cls	Lng	Plt	FE	Aos	IA
1	fastapi	27,638 100.0% (1.4)	0	Mcr	Py	asy	Gun	Lin	Rea
2	flask	13,390 48.4% (0.7)	0	Mcr	Py	Wsg	Mei	Lin	Rea

Kuvio 16. Plaintext.

Kuviossa 17 nähdään raakatekstin testin viiveajat mitattavien ohjelmistokehysten välillä. (Techempower 2021.)



Latency of plaintext responses, Dell R440 Xeon Gold + 10 GbE

Framework	Average latency (lower is better)	σ (SD)	Max	Errors
fastapi	15.1 ms 43.7%	10.8 ms	159.6 ms	0
flask	34.6 ms 100.0%	25.5 ms	263.0 ms	0

Latency of plaintext responses, Azure D3v2 instances

Framework	Average latency (lower is better)	σ (SD)	Max	Errors
fastapi	103.8 ms 56.6%	64.3 ms	408.1 ms	0
flask	183.3 ms 100.0%	118.6 ms	791.6 ms	0

Kuvio 17. Plaintext-viive.

Taulukossa 1 näkyy ohjelmistokehysten ominaispiirteiden vertailu.

Taulukko 1. Ohjelmistokehysten vertailu.

FastAPI	**Flask**
Automaattinen sisäänrakennettu dokumentointi	Manuaalinen dokumentointi
ASGI asynkronoidut toiminnot	Tukee synkronoituja ja asynkronoituja toimintoja
Sisäänrakennettu datavalidointi	Ei sisäänrakennettua datavalidointia, erilisiä moduuleja
SSL/TLS-salaus	Flask-Security moduulit
Sisäänrakennettu OAuth-autentikointi	Kirjautumishallintaan erilaisia moduuleja, kuten Flask-Login
Ei sisäänrakennettua kehityspalvelinta	Sisäänrakennettu kehityspalvelin
Ei yhteensopiva mod_wsgi:n kanssa ilman ylimääräisiä teknologioita	Yhteensopiva mod_wsgi:n kanssa

Vaatimusmäärittelyllä oli neljä kategoriaa, jotka olivat tietoturvallisuus, helppo käyttöönotettavuus, kehityksen helpottaminen ja nopeuttaminen, ja että ohjelmistokehys ei hidasta palvelun tarjontaa. FastAPI suoriutui Flaskia paremmin kaikissa kategorioissa ja piti pienemmän viiveajan testien tuloksena, jolloin se ei tulisi hidastamaan palvelun tarjontaa. Vaatimusmäärittelyiden mukaisesti ohjelmistokehyksen tavoite kehityksen helpottamisesta ja nopeuttamisesta toteutuu FastAPI:n automaattisen sisäänrakennetun dokumentaation ja datavalidoinnin avulla. FastAPI:n sisäänrakennetut ominaisuudet varmistavat sen, että ylimääräisten kirjastojen käyttämiseltä voidaan välttyä näiden toiminallisuuksien tavoittamiseksi. Tällä vältytään yhteensopivuusongelmilta.

Flaskin turvallisuusmoduulien ollessa kolmannen osapuolen moduuleja lisää turvallisuusriskejä, mutta molemmat ohjelmistokehykset vaativat määrätietoisia tietoturvallisuuden käytäntöjä, etteivät ne olisi haavoittuvaisia hyökkäyksille. (Escape tech 2024.)

FastAPI:n tehokkuuden, sisäänrakennettujen ominaisuuksien ja projektin tilaajan toiveista FastAPI:n käyttöönottamisesta projekti lähdettiin toteuttamaan FastAPI:a käyttämällä ja sen käyttöönoton dokumentoinnista.

FastAPI:lla on yhteensopivuusongelma sen ollessa ASGI-ohjelmistokehys ja Digipalveluiden ympäristön pyöriessä mod_WSGI moduulilla. Tämä yhteensopivuusongelma tullaan ratkaisemaan implementoimalla middleware projektiympäristöön.

4.3 Kehityspalvelin

FastAPI:lla ei ole sisäänrakennettua kehityspalvelinta, mutta sitä voi ylläpitää erilaisilla etäpalvelimilla, kuten Uvicorn, Hypercorn ja Daphne. Palvelimella viitataan joko fyysiseen tietokoneeseen tai virtuaalikoneeseen, joka pyörittää palvelua. Sillä viitataan myös ohjelmaan, joka pyörii kyseisellä laitteella. Kehityspalvelimet ovat ohjelmia, jotka mahdollistavat websovellusten tai API:en isännöimisen. Palvelimet käsittelevät niille saapuvat HTTP-palvelupyynnöt, prosessoivat ne ja palauttavat asianmukaisen vastauksen. (FastAPI n.d.)

4.3.1 Uvicorn

Uvicorn on ASGI-palvelin, jota käytetään web-sovellusten ja API:en ajamiseen, jotka on kehitetty käyttäen ASGI-ohjelmistokehyksiä. Se on suunniteltu käsittelemään asynkronoitua Python-koodia, joka mahdollistaa sen tehokkaasti voimassa olevien yhteyksien ja pitkäkestoisten tehtävien hallinnan ja käsittelyn estämättä muita palvelupyyntöjä. (Uvicorn n.d.)

Uvicorn tuo myös sisäänrakennetut monitorointi ja lokitiedostojen kirjausominaisuudet, joka mahdollistaa palvelimen toiminnan seuraamisen, palvelupyyntöjen ja vastauksien vasteajan monitoroimisen. Univorn tarjoaa moduulipaketteja, jotka mahdollistavat tapoja toistaa tapahtumia säännöllisesti (loop), kuten *uvloop*. Uvicorn tukee WebSocket protokollaa, joka mahdollistaa sen käsitellä kaksisuuntaista kommunikointia käyttäjän ja palvelimen välillä reaaliajassa valmiiksi olemassa olevan TCP-yhteyden lisäksi. Uvicorn tukee HTTP/1 ja ASGI/1, ASGI/2 ja ASGI/3-standardeja. (Uvicorn n.d.)

4.3.2 Hypercorn

Hypercorn on ASGI-verkkopalvelin, joka perustuu sans-io hyper-, h11-, h2- ja wsprotoko-kirjastoihin. Hypercorn tukee HTTP/1 ja HTTP/2 -standardeja, joka mahdollistaa yhteyksien nopeuden ja tehokkuuden lisäämisen web-palvelimien välisessä kommunikoinnissa. Se tukee myös ASGI/2 ja ASGI/3 standardeja, joka mahdollistaa asynkronisten palveluiden skaalautuvuuden web-sovelluksissa ja palvelimilla. Se tukee myös WebSocket-teknologiaa. Hypercorn on alun perin suunniteltu osaksi Quart-ohjelmistokehystä, kunnes se erotettiin itsenäiseksi ASGI-palvelimeksi. Hypercornin lähdekoodi on avoin. (Hypercorn n.d.)

Hypercorn tarjoaa erilaisia moduuleja, kuten *asyncio*, *uvloop* ja *trio*, jotka mahdollistavat asynkronisen ohjelmoinnin hyödyntämisen. *Uvloop* ja *trio* ovat sisäänrakennettuja tapoja toistaa tapahtuma säännöllisesti (loop), joiden joustavuus mahdollistaa Hypercornin käyttämisen monissa erilaisissa sovellusasetelmissa. (Hypercorn n.d.)

4.3.3 Daphne

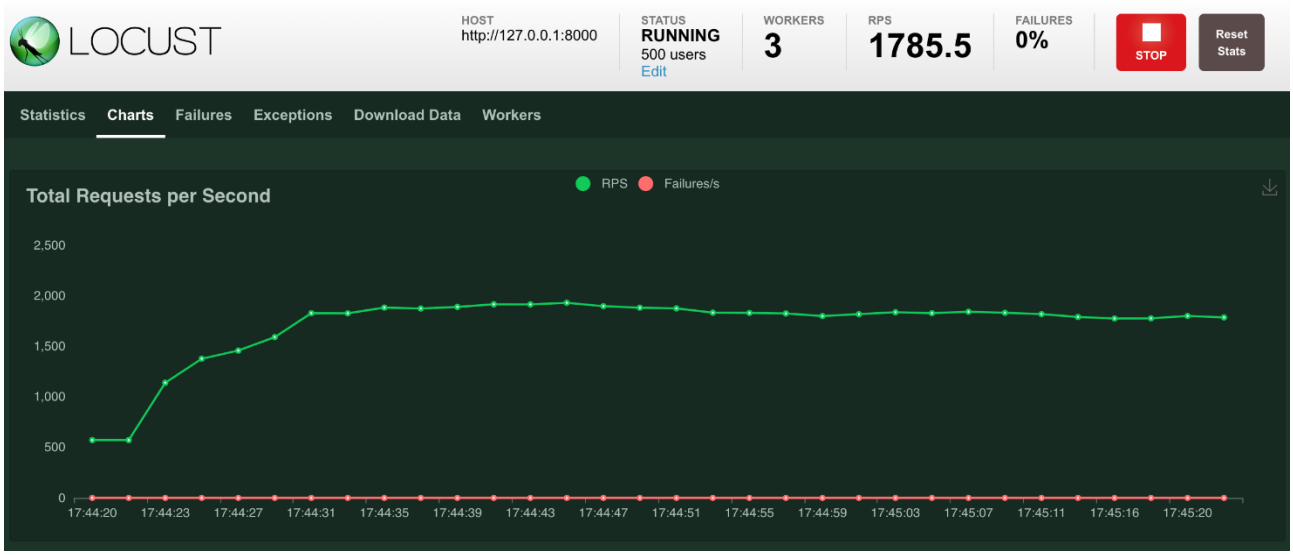
Daphne on ASGI-verkkopalvelin, joka on suunniteltu Django-Channels-projektin kanssa. Django Channels on Python-kirjasto, joka mahdollistaa WebSocket-, chat-, reaaliaikaisia- ja muita interaktiivisia sovelluksia. Daphne on erityisesti suunniteltu tukemaan Django Channels -projektin tarpeita, mutta se on myös yhteensopiva minkä tahansa ASGI-sovelluksen kanssa, kuten FastAPI-sovelluksilla. Daphne tukee HTTP/1 ja HTTP/2 standardeja ja se on suunniteltu asynkroniseen ohjelmointiin. Daphnea voi käyttää FastAPI-sovellusten ajamiseen, kun tarvitaan WebSocket-protokollan tukemista tai asynkronisen ohjelmoinnin tukemista. Daphne vaatii Python 3.8 tai myöhemmän version. (Django n.d.)

4.4 Vertailu

Piccolo on tehnyt vertailuarvoja kehityspalvelimille käyttäen Locust-testausohjelmistoa. Piccolo on moderni, asynkroninen kyselyiden rakentaja ja ORM Pythonille. Locust on avoimen lähdekoodin suorituskyvyn ja rasituksen testaustyökalu HTTP-protokollille. Testit on suoritettu 2.7GHz i7 kuuden ytimen prosessorilla ja 16GB RAM-muistin kanssa. Testit paljastavat kuinka monta vastausta kehityspalvelimet antavat sekunnissa. Kuviossa näkyy Uvicornin antamat vastaukset, joka on keskimäärin 2437,2. (Townsend.) (Ks.Kuvio 18.)

Kuvio 18. Uvicorn RPS.

Kuviossa 19 nähdään Hypercornin antamat arvot, joka on keskimäärin 1785,5.



Kuvio 19. Hypercorn RPS.

Kuviossa 20 nähdään Daphnen antamat arvot, joka on keskimäärin 1716,7.

Kuvio 20. Daphne RPS.

Uvicorn suoriutui muita palvelimia paremmin suorituskyvyltään. Prototyypin toteutusta varten käyttöönotetaan Uvicorn sen helppokäyttöisyyden, skaalautuvuuden sekä suorituskyvyn vuoksi kehitystä varten.

5 Julkaisuhakusivun toteutus

5.1 Ympäristöt

5.1.1 Virtuaalinen kehitysympäristö

Virtuaalinen kehitysympäristö *virtualenv* varmistaa, että projektia varten asennetut paketit eivät vaikuta kehitysympäristön ulkopuolisiin projekteihin. Tämä estää versionkonfliktien syntymisen. (Pierre 2023). Virtuaalinen kehitysympäristö *virtualenv* asennetaan kohdekansioon terminaalissa. (Ks. Kuvio 21.)

```
python -m pip install --user virtualenv
```

Kuvio 21. Install virtualEnv.

Kun virtuaalinen kehitysympäristö on asennettu, kehitysympäristöä kutsutaan luomaan kansio nimeltä *env.* (Ks. Kuvio 22.)

```
virtualenv env
```

Kuvio 22. VirtualEnv.

Virtuaalisen ympäristö aktivoidaan kutsumalla *activate.sh* tiedostoa *env/bin* kansiorakenteesta. (Ks. Kuvio 23.) *Activate.sh* tiedosto on shell tiedosto, joka aktivoi virtuaalisen kehitysympäristön Python-projektille. Suoritettuna se muuttaa komentotulkin ympäristömuuttujia varmistaakseen, että virtuaaliympäristössä suoritettavat komennot käyttävät kyseiseen ympäristöön asennettuja Python- ja muita pakettiversioita eikä järjestelmässä yleisesti saatavilla olevia versioita. Tämä eristäminen auttaa hallitsemaan eri projektien riippuvuuksia erikseen, jolloin vältetään ristiriidat eri sovellusten tarvitsemien pakettiversioiden välillä. (Python. n.d.)

```
source env/bin/activate
```

Kuvio 23. Source activate.

Kun virtuaalinen kehitysympäristö on aktivoitu ja pystytetty, sinne voi asentaa kaikki projektia varten tarvittavat paketit ja teknologiat.

5.1.2 FastAPI

FastAPI vaatii toimiakseen Python-version 3.8 tai uudemman version. (FastAPI. n.d.) FastAPIn voi asentaa pip-pakettihallintatyökalun avulla käyttäen komentoa terminaalin kautta Kuvion 24 mukaisesti.

```
pip install fastapi[all]
```

Kuvio 24. FastAPI All.

FastAPI tarvitsee etäpalvelinkoneen toimiakseen. Tätä varten on käytetty FastAPI:in asennuspaketissa tullutta Uvicorn ASGI-palvelinta. Uvicorn on valittu tähän tarkoitukseen sen tehokkuuden ja pienen vasteviiveen vuoksi. (FastAPI n.d.) Uvicornin avulla FastAPilla pyörivä main.py käynnistetään komennolla terminaalin kautta kuvion mukaisesti. (Ks. Kuvio 25.)

```
uvicorn main:app -reload
```

Kuvio 25. Uvicorn reload.

Kuvion Vaaditut moduulit mukaan main.py-tiedostoon asetetaan moduulit. Moduulit sisältävät Python-koodia, joka lisätään haluttuun ohjelmaan, vähentäen uudelleenkirjoittamista ja pitäen halutut paketit helposti kasassa. (python.org) Ensisijaisissa moduuleissa on lisätty FastAPI:n hyödyntämiä moduuleja erilaisten vastauksien saamiseksi, kuten *HTMLResponse*, *JSONResponse* ja *PlainTextResponse*. Moduulipaketeissa lisätään myös reititysten, pyyntöjen, lomakkeiden, kyselyiden, asetusten ja mallien moduulit. Nämä moduulit mahdollistavat FastAPI:n, Starletten ja Pydanticin toimivan oikein. Moduulien mukana tuodaan myös ominaisuudet koodaamiselle ja kirjoittamiselle.

```
# Import necessary modules and libraries
from fastapi import FastAPI, Request, Form, Query
from fastapi.responses import HTMLResponse
from fastapi.responses import JSONResponse
from fastapi.responses import PlainTextResponse
from fastapi.staticfiles import StaticFiles
from fastapi.templating import Jinja2Templates
from core.config import settings
from backend.apis.general_pages.route_homepage import general_pages_router
from pathlib import Path
from starlette.responses import FileResponse
from pydantic import BaseModel
from typing import Union
from fastapi.encoders import jsonable_encoder
```

Kuvio 26. Vaaditut moduulit.

Kuvion Mukautetut moduulit ja Olennaismoduulit lisätään moduuleja, jotka mahdollistavat Converiksen tietokantaan yhdistämisen ja tietokannassa olevan datan tulostamisen.

```
#Import custom modules and functions
import ConverisDbUtils
import ConverisDbConnector
import httpUtils
import converisUtils
import pythonUtils
import publicationUtils
import fileutils
```

Kuvio 27. Mukautetut moduulit.

Kuviossa 28 nähdään olennaisia moduuleja sovelluksen toimiseksi. Moduulit tarjoavat laajasti erilaisia ohjelmoinnin kannalta oleellisia ominaisuuksia oleellisia ohjelmoinnille, kuten *CSV*, *JSON* ja *unicodeCsvHandlers*, jotka takaavat erilaisten tietotyyppien oikean käsittelyn ja vastaanottamisen.

```
import stringlib
import csv
import json
import unicodeCsvHandlers
import urllib.parse
import types
import decimal
import os
import sys
import envconfig
import traceback
import time
import datetime
import io
```

Kuvio 28. Olennaismoduulit.

App-muuttuja toimii FastAPI-luokan ilmentymänä. (Ks. Kuvio 29.) Tämä mahdollistaa luokan tarjoamien funktioiden kutsumisen.

```
# Initialize the FastAPI application
app = FastAPI()
```

Kuvio 29. App-muuttujan ilmentymä.

FastAPI:n luomien sivujen muokkaamista varten asetetaan kansio *mount* komennolla, jonka sisällä säilytetään ja muokataan Javascriptin kirjastot kuten *bootstrap*, CSS tyylittelytiedostot ja sivulla esiintyvät kuvat. (Ks.Kuvio 30.)

```
# Mount static files directory
app.mount(
    "/static",
    StaticFiles(directory=Path(__file__).parent.parent.absolute() / "env/static"),
    name="static",
)
```

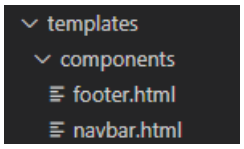
Kuvio 30. Staattisten tiedostojen asettaminen mount-komennolla.

Uudelleenkirjoittamisen vähentämiseksi hyödynnetään Jinja2-malleja, joiden kautta muokataan sivun eri palasia, kuten footer- ja header-elementtiä sekä kokonaisia sivuja. Jinja2-lisäosa hyödyntää myös *shared*-kansiota, johon linkitetään kaikki tarvittavat kirjastot ja Jinja2 ottaa nämä käyttöön kaikissa ympäristön sisäisillä sivuilla. (Ks.Kuvio 31.)

```
# Initialize Jinja2 templates
templates = Jinja2Templates(directory="templates")
```

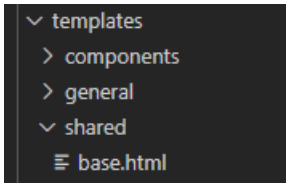
Kuvio 31. Jinja2-mallien ilmentyminen.

Kuvion 32 mukaisesti *components*-kansioon lisätään Jinja2 käyttämiä malleja, tässä tapauksessa footer.html ja navbar.html.



Kuvio 32. Komponentit jinjan template-mallissa.

Kuvion 33 mukaisesti kansiorakenteesta löytää base.html tiedoston. Tämä tiedosto toimii pohjana kaikille sivuille, johon on liitetty kaikki skriptit, tyylitiedostot ja komponentit.



Kuvio 33. Jaetut tiedostot shared kansiossa.

Kuvion 34 mukaisesti base.html tiedostoon lisätään *include* elementillä komponentin tiedostosijainti, jotta haluttu komponentti näkyy halutussa osassa sivua. Rakennetussa prototyypissä sisälletyt komponentit ovat navigointipalkki *navbar* ja footer-elementti.

```
{% include "components/navbar.html" %}
<body>···
</body>
{% include "components/footer.html" %}
```

Kuvio 34. Include-elementti base.html-tiedostossa.

Kuvion 35 mukaisesti projektin *SCRIPT_BASE* on määritetty, joka ajaa projektin skriptejä määritellystä polusta. Samalla asetetaan muuttuja *defaultYear*, jota käytetään määrittämään oletusvuosi. Oletusvuotena toimii nykyvuosi. Muuttujaa kutsutaan ryhmittelyfunktioissa, pääasiassa kun haetaan julkaisuja määritettyjen vuosilukujen väliltä. Nämä sijaitsevat main.py-tiedostossa.

```
# Define common variables
SCRIPT_BASE = os.path.abspath(sys.path[0]+'/..')
defaultYear = datetime.datetime.today().year
```

Kuvio 35. Yleiset muuttujat main.py-tiedostossa.

5.2 Middleware

Digipalveluiden palvelin pyörii mod_wsgi-ympäristössä. Valittu ohjelmistokehys Fastapi tukee ASGI-palvelimia, joten yhteensopivuuden takaamiseksi kokonaisuudessa käytetään välikätenä

WSGIMIDDLEWARE sovellusta. Middleware mahdollistaa WSGI-sovellusten liittämisen FastAPI-sovellukseen ja FastAPI:n ajamisen Apachella mod_wsgi:n kanssa. Middleware vaatii toimiakseen Python 2.6 tai 3.3 version tai uudemman toimiakseen, sekä Pythonin *dev*-paketin. Pythonin voi asentaa jaettuna kirjastona mod_wsgi:n muistinkäytön vähentämiseksi. (FastAPI n.d.)

Apache mod_wsgi:n kanssa vaatii Apachen konfiguroimista, jotta se toimii FastAPI:n kanssa. Tämä sisältää välttämättömien direktiivien asettamisen mod_wsgi:n lataamiseksi ja sovelluksen polun asettamiseksi, jotta sovellus latautuu oikein. Esimerkki Apache-konfiguraatiosta. (FastAPI n.d.) (Ks.Kuvio 36.)

```
LoadModule wsgi_module modules/mod_wsgi.so

<VirtualHost *:80>
    ServerName yourdomain.com
    DocumentRoot "crisapps/webscripts/manageui/fastapi/env"

    <Directory "crisapps/webscripts/manageui/fastapi/env">
        Require all granted
    </Directory>

    WSGIScriptAlias / crisapps/webscripts/manageui/fastapi/env/main.py
    WSGIDaemonProcess yourapp python-path=/path/to/your/app:/usr/local/lib/python3.8/site-packages
    WSGIProcessGroup yourapp
</VirtualHost>
```

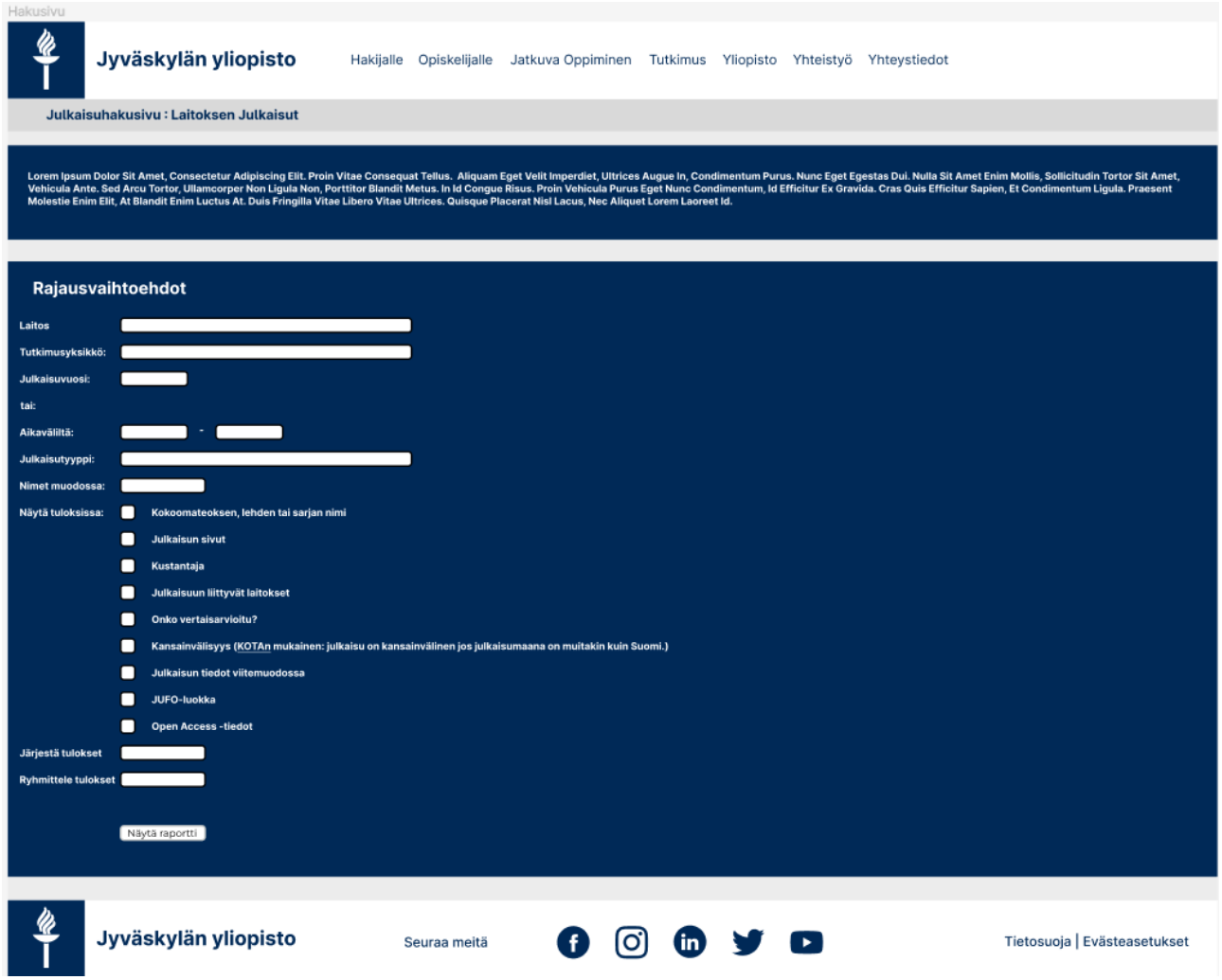
Kuvio 36. Apache-konfiguraatio.

5.3 Figma Prototyyppi

Projektin alkuvaiheessa luotiin Figma-työkalulla prototyyppi, jonka mallia käytetään käyttöliittymää suunnitellessa. Suunnitteluvaiheessa suunniteltiin 5 prototyyppisivua, jotka sisältäisivät vaatimusmäärittelyn mukaiset ominaisuudet. Nämä sivut ovat hakusivu, hakutulokset, yksittäisen julkaisun sivu, julkaisun metatietojen ponnahdusikkuna ja APA-viitteen ponnahdusikkuna.

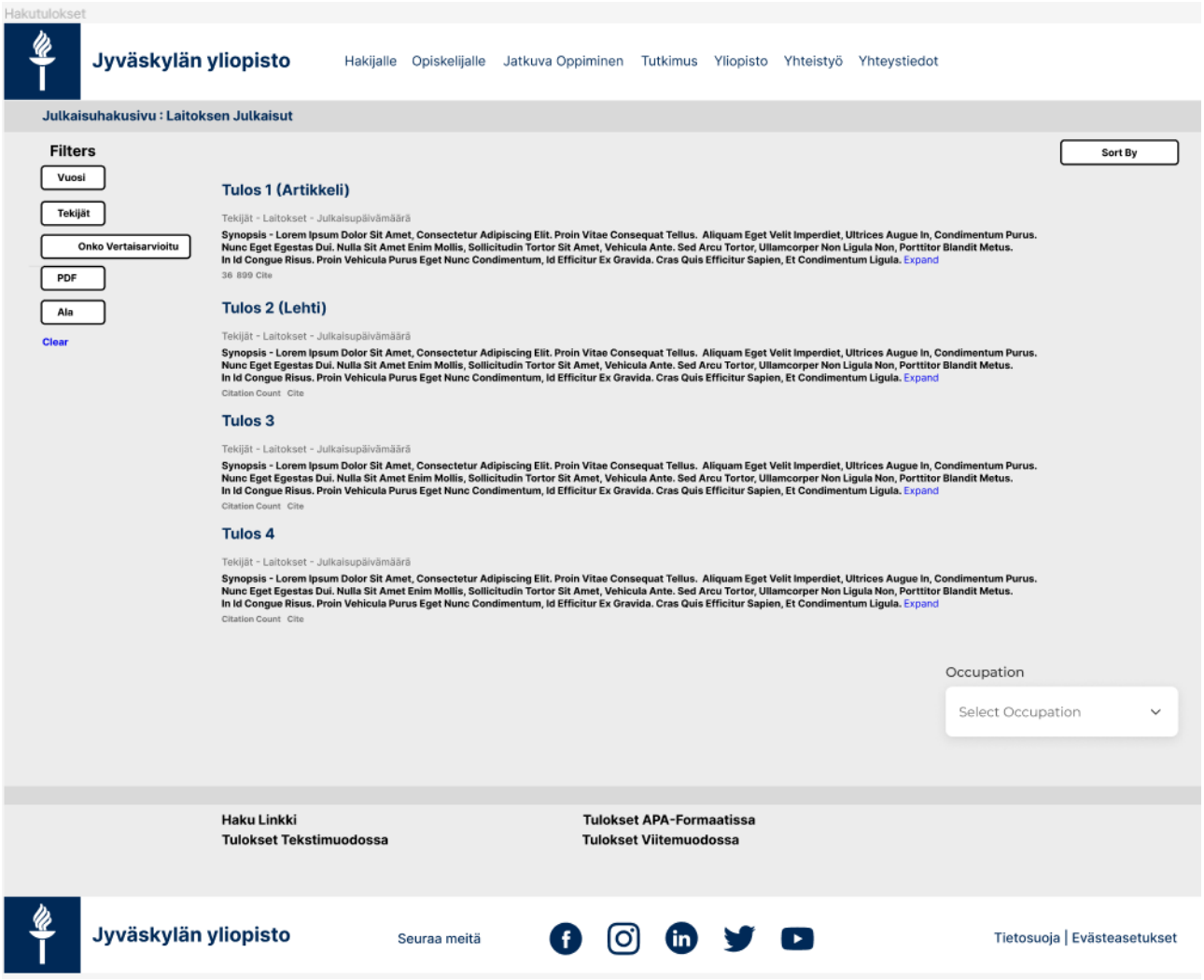
Käyttäjän avatessa sivun hän päätyy ensimmäiseksi hakusivulle. Sivu sisältää footer- ja header-elementin lisäksi hakujen rajausvaihtoehdot, jotka sisältävät mahdollisuuden rajata näytettävän raportin laitos, tutkimusyksikkö, julkaisuvuosi tai aikaväliltä, julkaisutyyppi, nimimuodon mukaan,

järjestää tai ryhmittää tiedot tulosten mukaan ja mahdollisuuden näyttää eri tiedot tuloksissa, joihin tietoihin kuuluu kokoomateoksen, lehden tai sarjan nimi, julkaisun sivut, kustantaja, julkaisuun liittyvät laitokset, onko julkaisu vertaisarvioitu, julkaisun kansainvälisyyden tila, julkaisun tiedot viitemuodossa, Jufo-luokka ja Open Access-tiedot. (Ks.Kuvio 37.)



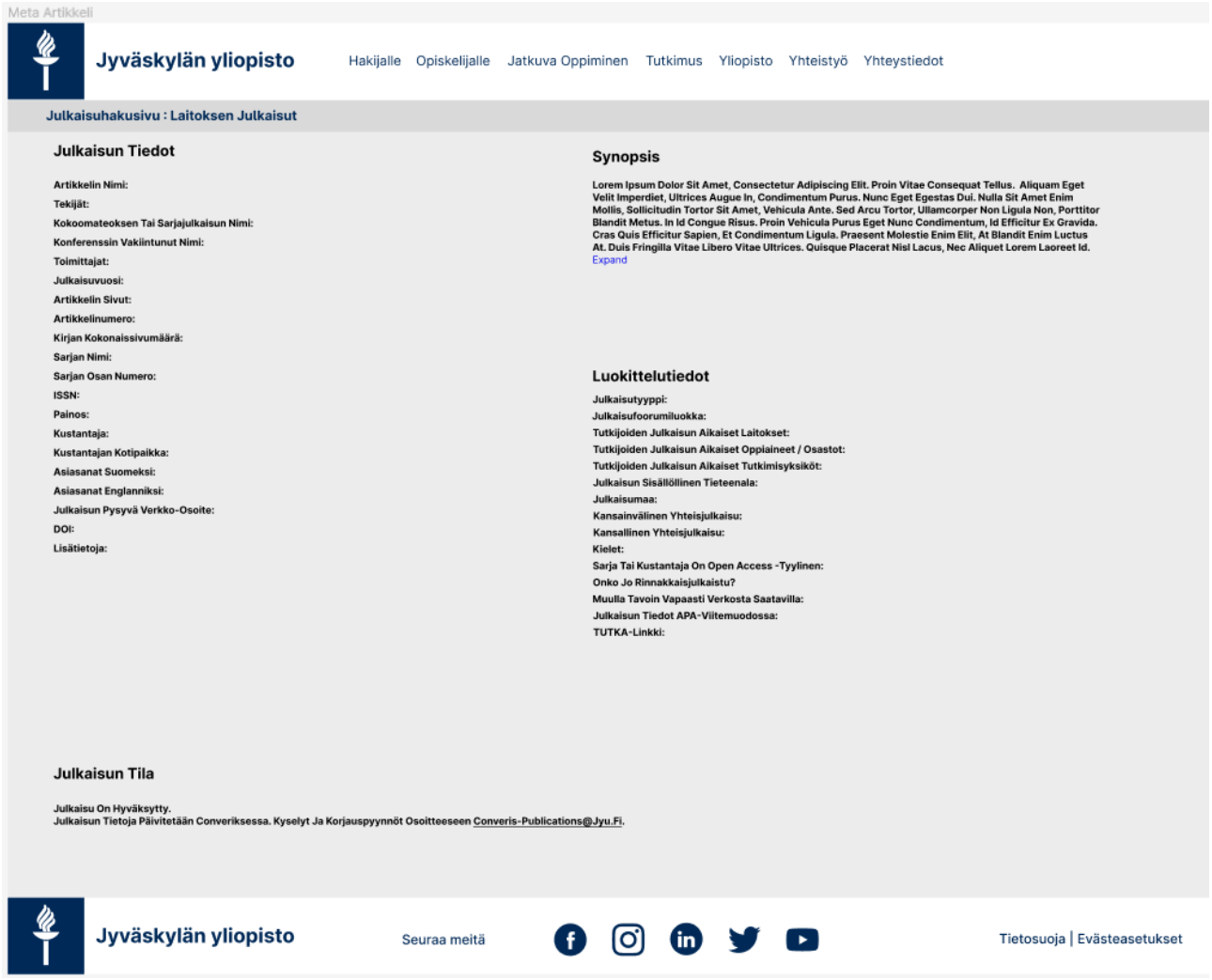
Kuvio 37. Hakusivun landing page.

Käyttäjän suoritettua haun, hän pääsee hakusivulle, jolla on filtterit julkaisujen vuosien, tekijöiden, vertaisarviointistatuksen ja alan mukaan. Yksittäisissä tuloksissa löytyy julkaisun nimi, julkaisun tyyppi, julkaisun tekijät, julkaisun latauskerrat sekä julkaisun päivämäärä. (Ks.Kuvio 38.)



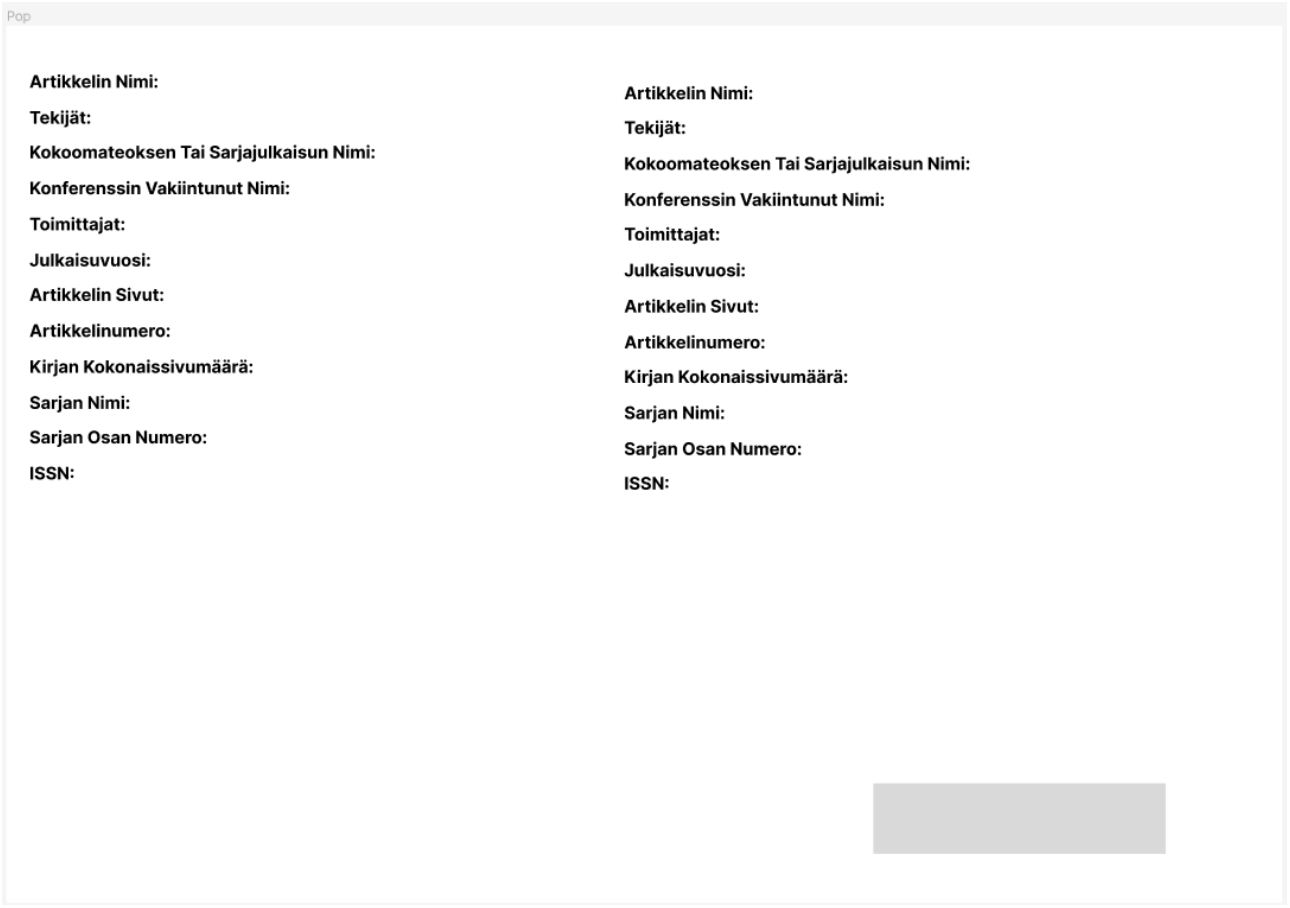
Kuvio 38. Hakutulokset.

Käyttäjän avatessa julkaisun kuvion 38 valintojen mukaan sivulla näkyy julkaisun tiedot, joihin sisältyy artikkelin nimi, tekijät, kokoomateoksen tai sarjajulkaisun nimi, konferenssin vakiintunut nimi, toimittajat, julkaisuvuosi, artikkelin sivut ja numero, kirjan kokonaissivumäärä, sarjan nimi, sarjan osan numero, ISSN (International Standard Serial Number), painos, kustantaja, kustantajan kotipaikka, asiasanat suomeksi ja englanniksi, julkaisun pysyvä verkko-osoite, DOI (Digital Object Identifier) ja julkaisun lisätiedot. Sivulla näkyy myös kirjan yhteenveto sekä luokittelutiedot, joihin kuuluu julkaisutyyppi, julkaisufoorumiluokka, julkaisun tutkijoiden aikaiset laitokset, oppiaineet/osastot ja tutkimusyksiköt, julkaisun sisällöllinen tieteenala, julkaisumaa, kansainvälinen ja kansallinen yhteisjulkaisu, kielet, julkaisun Open Access-tila, rinnakkaisjulkaisu tila, muilla tavoin vapaasti verkosta saatavuuden tila, julkaisun tiedot APA-viitemuodossa ja linkki tutkimustietojärjestelmään.



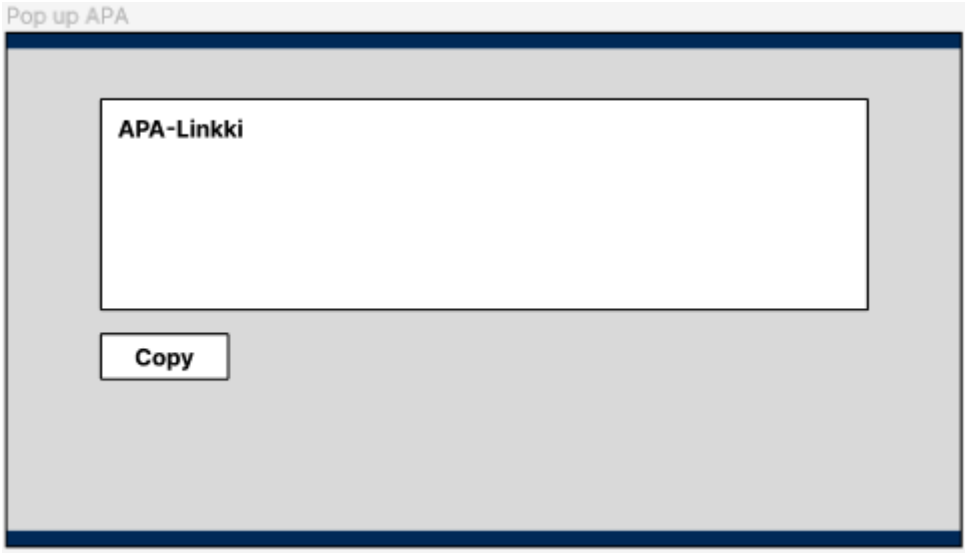
Kuvio 39. Julkaisun metatiedot.

Hakutulossivulta on mahdollisuus siirtyä ponnahdusikkunaan julkaisun metatiedoista, joka sisältää samaa tietoa kuin julkaisun metatiedoissa. (Ks.Kuvio 40.)



Kuvio 40. Ponnahdusikkunametatiedot.

Hakutulossivulta on myös mahdollisuus hakea julkaisun APA-viite, jonka saa APA-viite-ponnahdusikkunasta. (Ks.Kuvio 41.)



Kuvio 41. APA-viite-ponnahdusikkuna.

5.4 Tiedonhaku

Parametrit-kuvion mukaisesti luodaan viite */pubs,* jonka sisältö tulostetaan tulosivulle. Haun suorituksen jälkeen hakutuloksetlöytyvät */pubs* viitteen takaa. Jinja2 mahdollistaa tämän toiminnon yksinkertaistamisen. Asynkronoidulla funktiolla haetaan tietokannasta tietoa, jonka perusteella julkaisuja voidaan ryhmitellä vuosien väliltä, ryhmä ID:n, seurantakohteen ID:n ja organisaation mukaan. Nämä tulokset voidaan tulostaa halutussa laajuudessa, tässä tapauksessa maksimissaan 100 kerrallaan. (Ks.Kuvio 42.)

```
@app.get("/pubs")
async def get_publications(
    request: Request,
    publyear_start: int = Query(..., description="Start year for publication filter"),
    publyear_end: int = Query(..., description="End year for publication filter"),
    groupcgv_values: list = Query([], description="List of groupcgv values to filter"),
    classgroup_id: int = Query(..., description="Classgroup ID to filter"),
    org_ids: list = Query([], description="List of organization IDs to filter"),
    limit: int = Query(100, description="Maximum number of results to return")

):
```

Kuvio 42. Parametrit /pubs-viitteessä.

Kuvion 43 mukaisesti määritetään julkaisuille tilat millä julkaisut luokitellaan, tässä esimerkissä rinnakkaistallenteen versio. Julkaisut voidaan näyttää käyttäjälle tilassa hyväksytty versio, julkaistu versio, luonnos, palautettu versio, päivitetty versio sekä muut. Julkaisut voidaan näyttää myös pelkästään julkaisujen metadataa esittämällä.

```
versionMap = {
'acceptedVersion':'Final Draft',
'publishedVersion':"Publisher's PDF",
'metadataOnly':'',
'draft':'Preprint',
'submittedVersion':'Preprint',
'updatedVersion':"Publisher's PDF",
'other':''}
```

Kuvio 43. Rinnakkaistallenteen versioiden luokittelu.

Sisäänrakennetuilla lajitteluvaihtoehdoilla voidaan tuoda halutuloksia halutulla ryhmittelyllä. Parametriraportti-kuvion mukaisesti määritellään tietoa raportti parametreissa, tässä tapauksessa haetaan kaikki julkaisut tietyltä yksilöltä. SQL-hakulausekkeen pohjana toimii getPubInfoForCsv ja parametrien tietona haetaan julkaisuja tiettyjen vuosilukujen väliltä, yksikön nimen ja yksikön Id:n mukaan. (Ks.Kuvio 44.)

```
repInfo = {
    'allPublicationsByUnit':{
        'sql':'getPubInfoForCsv',
        'paramNames':['startYear','endYear','unitName','unitFedId'],
        'paramInfo':{
            'startYear':['int',defaultYear,None],
            'endYear':['int',defaultYear,None],
            'unitName':['str','%',None],
            'unitFedId':['str','%',None]
        }
```

Kuvio 44. Parametrien määritys julkaisuraportille laitoksen mukaan.

Kuvion 45 mukaisesti haetaan vain julkaisut, joissa on DOI-tunniste. SQL-hakulauseke haetaan *getPubDoiInfo*-tiedostosta, jonka jälkeen parametrien nimiksi määritetään *startYear* ja *endYear*. Julkaisuja voidaan sitten hakea tiettyjen julkaisuvuosien väliltä.

```
'allPublicationsWithDoi':{
    'sql':'getPubDoiInfo',
    'paramNames':['startYear','endYear'],
    'paramInfo':{
        'startYear':['int',defaultYear,None],
        'endYear':['int',defaultYear,None],
    }
```

Kuvio 45. DOI-julkaisut.

Kuvion 46 mukaan lajitellaan julkaisut tietyn tutkijan mukaan. SQL hakulauseke haetaan *getAuthoPubInfo*-tiedostosta, jonka jälkeen parametrien nimet määritellään nimillä *startYear*, *endYear*, *perfonFedId* ja *personFedIdExt*. Näiden parametrien avulla haetaan tietoa tutkijan mukaan, hakuparametrit rajoittavat hakutulokset käyttäjän määrittämien julkaisuvuosien väliin ja tutkijan käyttäjätunnuksen mukaisesti. (Ks.Kuvio 46.)

```
'publicationsByResearcher':{
    'sql':'getAuthorPubInfo',
    'paramNames':['startYear','endYear','personFedId','personFedIdExt'],
    'paramInfo':{
        'startYear':['int',defaultYear,None],
        'endYear':['int',defaultYear,None],
        'personFedId':['str','%',None],
        'personFedIdExt':['str','%',None]
    }
}
```

Kuvio 46. Tutkijan julkaisut.

Kuvion 47 mukaan result.html-tiedostoon määritellään rivit p-elementtiin, johon sisältyy *Publication ID*, *Publication Year* ja *Groupcgv* ja näille riveille tulostetaan haettujen julkaisujen ID, julkaisuvuosi ja ryhmätunniste.

```
{% for publication in publications %}
    <li>
        <h2>{{ publication["title"] }}</h2>
        <p>Publication ID: {{ publication["id"] }}</p>
        <p>Publication Year: {{ publication["publyear"] }}</p>
        <p>Groupcgv: {{ publication["groupcgv.value_1"] }}</p>
        <!-- Add more publication details here as needed -->
    </li>
{% endfor %}
```

Kuvio 47. Julkaisut result.html-tiedostossa.

Kuvion 48 mukaan asetetaan SQL-kysely, joka lähetetään tietokannalle. Kyselyssä haetaan kaikki julkaisut taulusta *reporting.iot_publication* annettujen julkaisuvuosien väliltä, joissa julkaisujen *publyear_start*-arvo on yhtä suuri tai suurempi kuin annettu arvo ja julkaisujen *publyear_end*-arvo on yhtä suuri tai pienempi kuin annettu arvo. Haettujen esineiden raja on 10.

```
db = ConverisDbConnector.ObjDbConnector(connect=True)
cursor = db.getCursor()
# Your SQL query with parameters
sql_query = """
SELECT * FROM reporting.iot_publication AS pub
    JOIN reporting.rel_clas_has_publ AS pub_gr ON pub.id = pub_gr.iot_publication
    WHERE pub.publyear >= %(publyear_start)s
    AND pub.publyear <= %(publyear_end)s
    Limit 10
"""
```

Kuvio 48. Yksinkertainen SQL-kysely.

Cursor.fetchall komennolla haetaan kaikki rivit, joita SQL-kysely koskee ja palauttaa rivit listana. Palautetut rivit lisätään *results*-luokkaan. *Cursor.execute*-komennolla suoritetaan *sql_query* ja *query_params* komennot, joilla suoritetaan SQL-kysely ja asetetaan kyselyn parametrit. (Ks.Kuvio 49.)

```
cursor.execute(sql_query, query_params)
results = cursor.fetchall()
```

Kuvio 49. Cursor-komennot.

Kuvion 50 mukaan asetetaan jokainen row-elementti *results*-luokassa. Luodaan lista (dictionary) jokaiselle julkaisulle, jossa parametrit ovat *title*, *id* ja *publyear* jotka asetetaan riveille 0, 1 ja 2. Nämä rivit asetetaan *append*-komennolla ja palautetaan result.html lomakkeelle.

```
        publications = []
        # Process the result and return it as JSON or in the desired format
        for row in results:
            #publication = {}  # Create an empty dictionary for each publication
            publication = {
                "title": row[0],  # Replace with the actual column names from your query
                "id": row[1],     # Replace with the actual column names from your query
                "publyear": row[2]  # Replace with the actual column names from your query
            }
            publications.append(publication)
        cursor.close()
        db.closeConnection()  # Close the database connection
        print("publications:", publications)
        # Render the 'result.html' template and pass the 'publications' data to it
        return templates.TemplateResponse("general/result.html", {"request": request, "publications": publications})
```

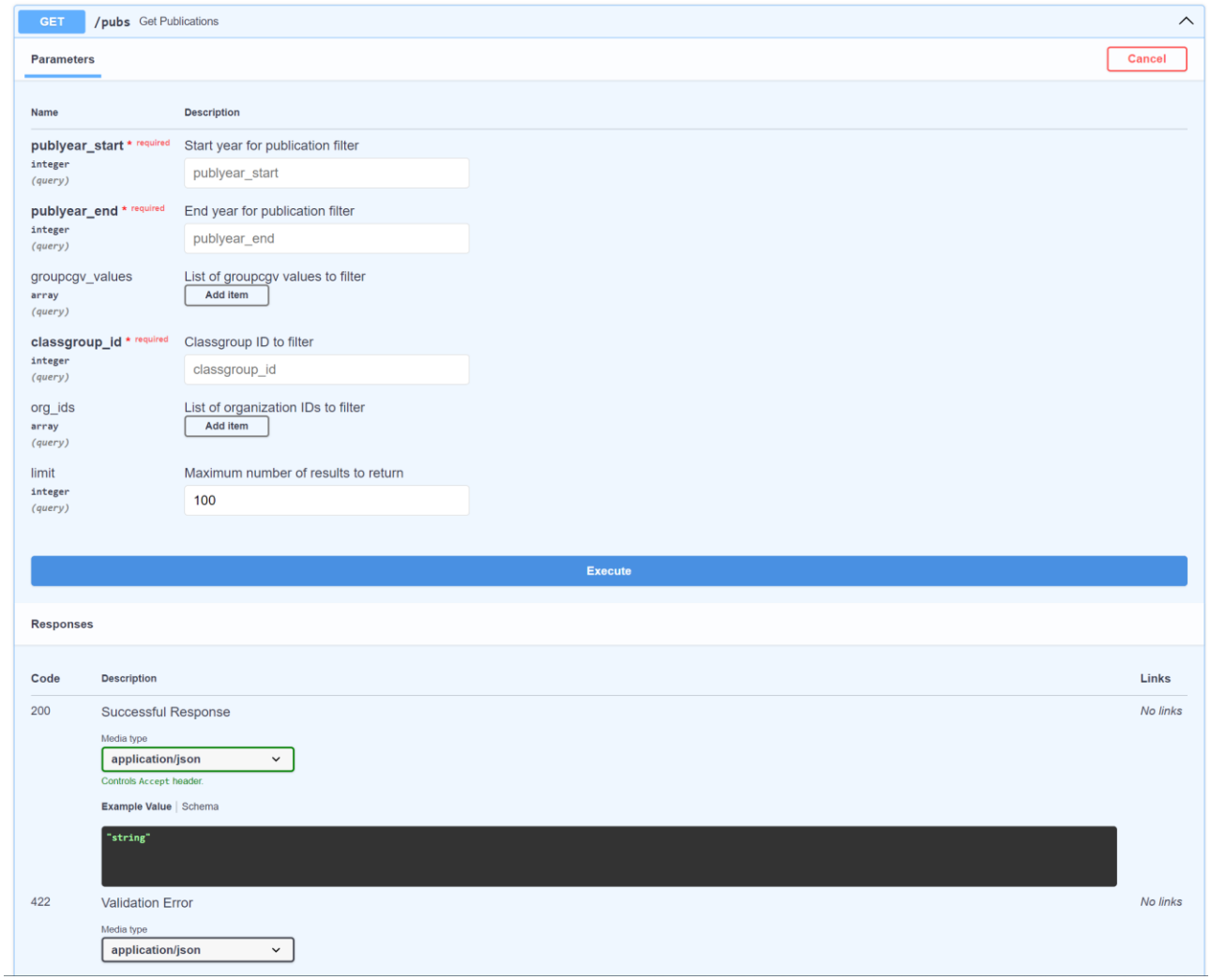
Kuvio 50. Julkaisujen asettaminen publications-listaan.

Haetut tiedot tulostetaan results.html-lomakkeella. Kuviossa 51 näkyvät elementit, joille haetut tiedot tulostetaan.

```
<h1>Publications</h1>
<ul>
    {% for publication in publications %}
        <li>
            <h2>{{ publication["title"] }}</h2>
            <p>Publication ID: {{ publication["id"] }}</p>
            <p>Publication Year: {{ publication["publyear"] }}</p>
            <p>Groupcgv: {{ publication["groupcgv.value_1"] }}</p>
        </li>
    {% endfor %}
</ul>
```

Kuvio 51. Tulokset Results.html lomakkeella.

FastAPI:n interaktiivisen dokumentoinnin kautta pyydetyt tiedot voidaan hakea tietokannasta, joka myös palauttaa saadut tiedot JSON-datana. (KS.Kuvio 52.)



Kuvio 52. Tulostettavat parametrit FastAPI:n dokumentaationäkymässä.

6 Yhteenveto ja pohdinta

Projektin päättyessä lopullinen tuote oli toiminallinen prototyyppi, jonka avulla käyttäjä pystyy hakemaan rajatusti tietoa julkaisujen tietokannasta. Projektia varten on asetettu valmis työskentelyalusta ohjelmistokehyksien hyödyntämiselle jatkokehitykselle, uusien parametrien ja ominaisuuksien lisäämiseksi. Prototyypin pystyisi rakentamaan täysin toimivaksi palveluksi ylimääräisen ajan kanssa.

Projektin alkuun lähdettiin selvittämään uuden julkaisuhakusivun kehittämisvaiheita, sen suunnittelua ja toteutusta sekä siihen tarvittavia työkaluja ja teknologioita. Nämä vaiheet dokumentoitiin tekijän opinnäytetyötä sekä toimeksiantajan sisäistä käyttöä varten.

Opinnäytetyössä käytetyt lähteet ovat suurimmalta osin työkalujen kehittäjien omilta sivuilta, manuaaleista lainattuja ohjeita tai alan ammattilaisten tekemiä artikkeleita useiden vuosien kokemuksella sivuilta, joilla ei ole tunnettuja puolueellista asettumista tiettyjä teknologioita kohden, tehden opinnäytetyössä tehdyt dokumentaatiot ja testitulokset luotettaviksi.

Opinnäytetyön aihe oli laaja, jonka jokaisesta osa-alueesta voisi tehdä oman aiheensa. Tässä opinnäytetyössä käytettiin tekijän tehtyä työtä projektin eteen, johon kuului sekä suunnittelu ja toteutus. Aihe tuo uutta alalle FastAPI:n ollessa vielä uudempi ohjelmistokehys, jota ei olla hyödynnetty projektikehityksessä laajasti eikä sen dokumentaatio ole yhtä kattava kuin vanhemmilla ohjelmistokehyksillä.

Toteutusmenetelmien valintaan vaikutti opinnäytetyön tekijän oma osaaminen. Valitut toteutusmenetelmät eivät ole ainoita tapoja projektin toiminallisuutta ajatellen. Opinnäytetyön tekijän oma osaaminen vaikutti myös siihen, että paljonko projektin suunnitellusta työmäärästä saatiin tehdyksi määräaikaan mennessä. Vaikeuksia tuli vastaan suurempien kokonaisuuksien hallinnassa ja monimutkaisempien funktioiden luomisessa, kun oma ohjelmointiosaaminen ei ollut riittävää. Ohjelmistokehyksien vertailu ja käyttöönotto vei odotettua enemmän aikaa ja middlewaren implementoiminen projektin toiminallisuuden takaamiseksi osoittautui haastavaksi, jolloin palvelua ei saatu pyörimään sisäisessä verkossa, se toimii määräajan päättyessä vain paikallisessa verkossa.

Toimeksiantajalle tehtiin loppuesitys, jossa esiteltiin prototyypin toiminallisuutta, siinä olevien teknologioiden osa, projektissa olevat ongelmat ja jatkokehityksen tarpeet.

Opinnäytetyön alussa lähdettiin selvittämään tutkimuskysymyksiä, joihin kuuluvat sopivan ohjelmistokehyksen löytäminen, julkaisuhakusivun kehittäminen ja siihen sopivan sivurakenteen löytäminen. Näihin kysymyksiin tutustuttiin opinnäytetyön aikana, niiden takana olevan teoriaan ja niihin tarvittavaan teknologiaan. Tutkimuskysymykseen ohjelmistokehyksiin liittyen onnistuttiin vertailemalla kehyksiä toisiinsa, tämän dokumentointi ja soveltuvan ohjelmistokehyksen etsintä projektin tarpeisiin. Julkaisuhakusivun toteutus jäi valmiista tuotteesta prototyyppiin, toteutusta varten kerättiin tarvittava teoria ja menetelmät, mutta tekijän osaaminen jarrutti projektin etenemistä määräaikaan mennessä. Julkaisuhakusivulle tehtiin sopiva sivurakenne Figma-

prototyypin muodossa, josta konsultoitiin toimeksiantajan kanssa usein. Toimeksiantaja oli näkymään tyytyväinen, mutta yksittäisiä tutkijoita sivun ulkonäöstä ei konsultoitu projektin prioriteetin ollessa tuotannon ja toteutuksen puolella. Kyselyn välittäminen tutkijoille ja muulle henkilöstölle toivotusta sivunrakenteesta olisi mahdollisesti muuttanut tulevan sivujen rakennetta ja ulkonäköä.

Lähteet

A Complete Guide to Web Development Process. N.d. Huzefa Chawre. Viitattu 16.3.2024. HTTPs://www.turing.com/resources/web-development-process .

A Guide to Website Development. N.d. Artikkeli Mailchimp www-sivuilla. Viitattu 17.3.2024 HTTPs://mailchimp.com/resources/guide-to-website-development/ .

Advanced Middleware. N.d. Ohjesivu FastAPI:n www-sivuilla. Viitattu 22.5.2024. https://fastapi.tiangolo.com/advanced/middleware/

Agrawal, M. Python Virtual Environment | Introduction. 2023. Viitattu HTTPs://www.geeksforgeeks.org/python-virtual-environment/ .

Babich, N. 2019. The 4 Golden Rules of UI Design. Artikkeli Adoben www-sivuilla. Viitattu 11.3.2024. HTTPs://xd.adobe.com/ideas/process/ui-design/4-golden-rules-ui-design/ .

Converis. N.d. Artikkeli tietotekniikka yrityksen Clarivaten sivuilta. Viivattu 27.2.2024 HTTPs://clarivate.com/products/scientific-and-academic-research/research-funding-and-analytics/converis/ .

Creation of Virtual Environments. N.d. Ohjedokumentti Pythonin www-sivuilla. Viitattu 23.2.2024 HTTPs://docs.python.org/3/library/venv.html .

Debugging Application Errors. N.d. Ohjedokumentti Flaskin www-sivuilla. Viitattu 27.2.2024. https://flask.palletsprojects.com/en/3.0.x/debugging/ .

Demystyifying the Web Development Process: A Step-by-Step Guide. Vatsal Parmar. 2023 Viitattu 15.32024. HTTPs://radixweb.com/blog/web-development-process .

Deploying. N.d. Artikkeli Django Channels projektin dokumentaatio sivuilta. Viitattu 17.5.2024. HTTPs://channels.readthedocs.io/en/stable/deploying.html .

FastAPI. N.d. Ohjedokumentti FastAPI www-sivuilla. Viitattu 27.2.2024 HTTPs://fastapi.tiangolo.com/ .

Flask-Security. N.d. Ohjedokumentti Flaskin www-sivuilla. Viitattu 22.5.2024. https://flask-security-too.readthedocs.io/en/stable/ .

Framework, 2013. N.d. Artikkeli DocForgen www-sivuilla. HTTPs://web.archive.org/web/20150717020405/HTTP://docforge.com/wiki/Framework .

General Python FAQ. 2022. Dokumentaatio Python -ohjelmointikielen omilla www-sivuilla. Viitattu 23.2.2024. HTTPs://docs.python.org/3/faq/general.html .

Haltu. UX-suunnittelu – mitä se on ja mistä asioista se koostuu? 2023. Viitattu 11.3.2024. HTTPs://www.haltu.fi/blogi/ux-suunnittelu .

Tang, A. How to secure APIs built with FastAPI: a complete guide. 2024. Viitattu 22.5.2024. https://escape.tech/blog/how-to-secure-fastapi-api/ .

Hypercorn. 2024. Artikkeli Hypercornin dokumentaatio sivuilta. Viitattu 17.5.2024. HTTPs://pypi.org/project/Hypercorn/ .

Jinja.N.d. Ohjedokumentti Jinjan www-sivuilla. Viitattu 21.5.2024. HTTPs://jinja.palletsprojects.com/en/3.1.x/templates/ .

IBM. N.d. Artikkeli IBM:n www-sivuilla. Viitattu 18.5.2024. HTTPs://www.ibm.com/topics/what-is-requirements-management .

Internetdevels. Website prototype: How to make a prototype? 2020. Viitattu 13.3.2024. HTTPs://internetdevels.com/blog/what-is-website-prototype-how-build-website-prototype .

Introduction. N.d. Ohjesivu ASGI:n dokumentaatio sivulla. Viitattu 22.05.2024. https://asgi.readthedocs.io/en/latest/introduction.html .

Kopf, B. N.d. The Power of Figma as a Design Tool. Viitattu 11.3.2024. HTTPs://www.toptal.com/designers/ui/figma-design-tool .

Le Vier, D. 2010. Writing Software Requirements Specifications. Viitattu 28.3.2024. HTTP://techwhirl.com/articles/writingsoftwarerequirementsspecs/ .

mod_wsgi. N.d. Ohjedokumentti Flaskin www-sivuilla. Viitattu 27.2.2024. https://flask.palletsprojects.com/en/2.3.x/deploying/mod_wsgi/ .

mod_wsgi. N.d. Ohjedokumentti mod_wsgi www-sivuilla. Viitattu 23.2.2024 HTTPs://modwsgi.readthedocs.io/en/master/ .

Muranen, A & Harmainen, L. Käyttöliittymä- & käyttäjäkokemussuunnittelu (UI & UX Design.) Itewiki. Viitattu 11.3.2024. HTTPs://www.itewiki.fi/opas/kayttoliittymasuunnittelu-ux-user-experience-design-eli-kayttajakokemus/ .

PEP 3333 – Python Web Server Gateway Interface v1.0.1. 2010. Ohjesivu Pythonin dokumentaatio sivulla. Viitattu 22.5.2024. https://peps.python.org/pep-3333/

Piccolo. Which is the fastest ASGI server? 2021. Viitattu 16.5.2024. HTTPs://piccolo-orm.com/blog/which-is-the-fastest-asgi-server/ .

Prototyping. N.d. Artikkeli Lets Talk Science www-sivuilla. Viitattu 17.3.2024. HTTPs://letstalkscience.ca/educational-resources/learning-strategies/prototyping .

Salas-Zárate, M., Alor-Hernández, G. Valencia-García, R,. Rodríguez-Mazahua, L,. Rodríguez-González, A. & Cuadrado, J. 2015. Analyzing best practices on Web development frameworks: The lift approach. HTTPs://www.sciencedirect.com/science/article/pii/S0167642314005735?via%3Dihub .

Sadrach, P. A Guide to Python Virtual Environments. 2023. Viitattu 19.4.2024. HTTPs://builtin.com/data-science/python-virtual-environment .

Sahai, S. What is Data Validation in Python? 2024. Viitattu 14.4.2024. HTTPs://www.projectpro.io/recipes/perform-data-validation-python-by-processing-only-matched-columns .

Shaw, A. Fine Tuning Python WSGI and ASGI applications for Flask, Django and FastAPI. 2023. Viitattu 21.5.2024. https://tonybaloney.github.io/posts/fine-tuning-wsgi-and-asgi-applications.html .

Spolsky, J. User Interface Design for Programmers. 2001. Viitattu 23.2.2024.

Tang, A. Best practices to protect your Flask applications. 2024. Viitattu 22.5.2024. https://escape.tech/blog/best-practices-protect-flask-applications/ .

Troy, C. Document a FastAPI App with OpenAPI. 2021. Viitattu 14.4.2024. HTTPs://www.linode.com/docs/guides/documenting-a-fastapi-app-with-openapi/ .

UI Design. Viitattu 11.3.2024. HTTPs://xd.adobe.com/ideas/process/ui-design/ .

User Interface Design. N.d. Viitattu 11.3.2024. HTTPs://www.interaction-design.org/literature/topics/ui-design .

Uvicorn. N.d. Viitattu 21.5.2024. https://www.uvicorn.org/

Uxacademy. Käyttöliittymä- ja käyttäjäkokemussuunnittelu. Viitattu 11.3.2024. HTTPs://www.uxacademy.fi/kayttoliittyma-ja-kayttajakokemussuunnittelu-ui-ux/ .

Visual Studio Code: Getting Started. N.d. Ohjedokumentti Visual Studio Code www-sivuilla. Viitattu 23.2.2024. HTTPs://code.visualstudio.com/docs .

William. Web Application Architecture: The Latest Guide 2024. 2024. Viitattu 22.5.2024. https://www.clickittech.com/devops/web-application-architecture/

What is Middleware and how does it work? N.d. Artikkeli OPC Routerin www-sivuilta. Viitattu 21.05.2024. https://www.opc-router.com/what-is-middleware/ .

What is SSL/TLS Encryption? N.d. Artikkeli kyberturvallisuus yrityksen F5 sivuilta. Viitattu 16.4.2024. HTTPs://www.f5.com/glossary/ssl-tls-encryption .