KARELIA-AMMATTIKORKEAKOULU
Tietojenkäsittelyn koulutus

Lari Strand

INFONÄYTTÖTOTEUTUS RASPBERRY PI:LLÄ GOOGLE DRIVE API:A HYÖDYNTÄEN

Opinnäytetyö
Elokuu 2021

Karelia AMMATTIKORKEAKOULU	**OPINNÄYTETYÖ** **Elokuu 2021** **Tietojenkäsittelyn koulutus** Tikkarinne 9 80200 JOENSUU +358 13 260 600 (vaihde)

Tekijä(t)
Lari Strand

Nimeke
INFONÄYTTÖTOTEUTUS RASPBERRY PI:LLÄ GOOGLE DRIVE API:A HYÖDYNTÄEN

Toimeksiantaja
Pyhäselän kirjasto

Tiivistelmä

Opinnäytetyössä suunniteltiin ja kehitettiin toimeksiannon pohjalta infonäyttötoteutus digitaalisen markkinoinnin tehostamiseen Pyhäselän kirjastolle. Toteutuksen pyrkimyksenä oli luoda aiemman käytössä olevan infonäyttöratkaisun korvaava helppokäyttöinen infonäyttökokonaisuus, joka vastaisi vaatimusmäärittelyn tavoitteisiin. Tavoitteena oli saada luotua toteutuksesta laitteistoriippumaton ja kaiken tasoisille käyttäjille helppokäyttöinen. Lisäksi tavoitteena oli, että laitteella näytettävän sisällön päivittäminen olisi mahdollista etäkäyttöisesti.

Työn teoreettisessa osassa käsitellään infonäyttöjärjestelmiä ja Raspberry Pi -korttitietokoneen taustaa ja teknisiä ominaisuuksia. Toteutukseen valittu laitteisto, käyttöjärjestelmä, ohjelmointikielet ja ohjelmistokehykset perustellaan ja taustoitetaan erillisessä teknisessä osiossa.

Opinnäytetyöprosessin teknisen toteutuksen etenemistä esitellään korttitietokonekokonaisuuden kokoamisesta alkaen. Teknisen prosessin kuvauksessa kerrotaan lyhyesti ohjelmointiskriptien toimintalogiikasta ja siitä, miten Google Drive API:a hyödynnetään toteutuksen kokonaisuudessa. Teknisessä osassa kerrotaan myös paikallisen palvelimen sekä näytettävän infosisällön esittämiseen käytettävän kuvakarusellin luomisesta.

Infonäyttötoteutus onnistui suunnitellusti ja se otettiin Pyhäselän kirjastossa päivittäiskäyttöön. Käyttäjiltä saadun suullisen palautteen perusteella se on helppokäyttöisempi, modernimpi ja käyttäjäystävällisempi kuin aiemmin käytössä ollut ratkaisu.

Kieli suomi	Sivuja 51 Liitteet 1 Liitesivumäärä 2

Asiasanat
Infonäyttö, infotelevisio, InfoTV-järjestelmä, Google Drive API, Python, Node.js, HTML, Bootstrap, Express.js, JavaScript, Firefox, Raspberry Pi, Raspberry Pi OS, Linux.

THESIS
August 2021
Degree Programme in Business Information Technology
Tikkarinne 9
80200 JOENSUU
FINLAND
+ 358 13 260 600 (switchboard)

Author
Lari Strand

Title
Digital Signage Display System with Raspberry Pi and Google Drive API

Commissioned by
Pyhäselkä Library

Abstract

The purpose of this thesis was to design and develop a digital signage display system for Pyhäselkä Library to improve its digital marketing. The digital signage display system was commissioned by Pyhäselkä Library. The aim of the design was to replace the existing digital signage solution with an easy-to-use system which would fulfil the requirements specification. The goal was to create a digital signage system which is simple to administrate for all user skillsets and which would be device independent. In addition, the management of the material to be viewed by the solution should be remotely accessed.

The theoretical framework of the thesis studies existing digital signage display systems and presents the history of Raspberry Pi computers' development and the technical aspects of the device. The technical part of the thesis covers why the selected hardware, software, programming languages and software frameworks were chosen for the design. They are also briefly introduced.

The progress of the technical implementation of the thesis process is presented starting from the construction of the Raspberry Pi 4 computer kit. The description of the technical process provides a brief overview of the operating logic of the programming scripts and how the Google Drive API is utilised in the implementation. The technical section also describes how a local server is created and how an image carousel is used to display the material to render on the digital signage display system.

The project succeeded as planned, and the digital signage display system is in day-to-day use in Pyhäselkä library. Based on feedback from users, the new system is more modern, more user-friendly, and easier to operate than the previous system.

Language
Finnish

Pages 51
Appendices 1
Pages of Appendices 2

Keywords
Digital signage, digital signage display, digital signage display system, Google Drive API, Python, Node.js, HTML, Bootstrap, Express.js, JavaScript, Firefox, Raspberry Pi, Raspberry Pi OS, Linux.

Sisältö

Käsitteet ja lyhenteet

API	”Ohjelmointirajapinta (Application programming interface, API) määrittelee, miten ohjelmisto tarjoaa tietoja tai palveluita sovelluksille tai muille tietojärjestelmille” (Open Knowledge Finland 2021).
Bootstrap	Bootstrap on kehitysympäristö, jonka avulla voidaan luoda responsiivisia websivustoja (Kuivanen 2014).
Cron	Cron on ohjelmien tai skriptien ajastettuun ajamiseen tarkoitettu Linux-järjestelmän palvelu (Kuutti 2011, 135).
Crontab	Crontab-sovellus sisältää Linux-järjestelmän ajastetut palvelut ja sillä muokataan ajastuksia (Kuutti 2011, 135).
Infotelevisio/Infonäyttö	Infotelevisioilla ja -näytöillä tarkoitetaan erilaisissa julkisissa tiloissa käytössä olevia sisäiseen ja ulkoiseen markkinointiin, tiedottamiseen ja opastamiseen käytettäviä digitaalisia näyttöjä (Wikipedia 2021a).
JavaScript	JavaScript-ohjelmointikieltä käytetään pääosin Web-selainpuolella ajettavien sovellusten kirjoittamiseen. Sitä ajetaan joko Web-selaimessa tai muussa ympäristössä (kuten Node.js). (Peltomäki 2017, 1,12.)
Journaloiva tiedostojärjestelmä	Tiedostojärjestelmä, joka ylläpitää erityistä tiedostoa, jota kutsutaan journaliksi. Tämän tiedoston hyödyntämistä kutsutaan journaloinniksi. Sitä käytetään korjaamaan tiedostojärjestelmän epäjohdonmukaisuudet, jotka johtuvat muun muassa tietokoneen virheellisestä sammutuksesta. Tällaiset sammutukset johtuvat yleensä virransyötön keskeytymisestä tai ohjelmisto-ongelmasta, jota ei voida ratkaista ilman uudelleenkäynnistystä. (Linux Information Project 2021.)
Korttitietokone	Korttitietokone on pienikokoinen tietokone, jonka laitteiston komponentit on mahdutettu yhdelle piirilevylle.

Moduuli	Moduuli on tiedosto, joka sisältää jonkin erikoistuneen tehtävän. Moduuleja käytetään modulaarisessa ohjelmoinnissa ja ne helpottavat koodin luettavuutta ja uudelleenkäytettävyyttä. (Kuuppelomäki 2009.)
Node.js	Node.js on asynkroninen eli estämätön avoimen lähdekoodin sovelluskehys. Se on suunniteltu skaalautuvien verkkosovellusten rakentamiseen. Sitä käytetään usein interaktiivisten, reaaliaikaisten verkkosovellusten sovelluskehyksenä. (Nodejs.org 2021; Peltomäki 2020, 7.)
OAuth 2.0	Valtuutuskehys, jonka avulla sovellukselle voi myöntää rajoitetun pääsyn HTTP-palvelun, kuten Facebook, GitHub ja DigitalOcean, käyttäjätileihin. Se toimii delegoimalla käyttäjän todennus palvelulle, joka isännöi käyttäjätiliä, ja valtuuttamalla kolmannen osapuolen sovelluksia käyttämään käyttäjätiliä. (Anicas 2014.)
Raspberry Pi	Raspberry Pi on pienikokoinen tietokone, korttitietokone, joka on rakennettu luottokortin kokoiselle piirilevylle. (Raspberry Pi Foundation 2021a).

1 Johdanto

Opinnäytetyön tavoitteena on luoda infonäyttötoteutus helpottamaan tilaviestintää ja korvaamaan tällä hetkellä toimeksiantajan tilaviestinnässä käytössä oleva tekninen ratkaisu. Opinnäytetyön toimeksiantaja on Joensuun seutukirjaston Pyhäselän kirjasto. Infonäyttötoteutus tulee jatkossa käyttöön myös Joensuun kaupungin Pyhäselän palvelupisteen tilaviestintää varten.

Opinnäytetyössä pyritään luomaan helppokäyttöinen, laitteisto- ja alustariippumaton infonäyttösovelluskokonaisuus. Käytännön toteutus tapahtuu Raspberry Pi 4 Model B -korttitietokoneelle. Työssä selvitetään, millainen kuvatiedostojen siirtoon ja synkronointiin tarkoitettu ohjelmistoratkaisu on sopiva työhön valitun Google Drive API:n kanssa käytettäväksi.

Opinnäytetyössä kerrotaan lyhyesti erilaisista infonäyttötoteutuksista. Siinä esitellään lyhyesti Raspberry Pi -korttitietokoneen tekniikkaa, historiaa ja käyttöönotto sekä ohjelmistotototeutuksessa käytetyt koodikielet ja koodausympäristöt Python, JavaScript, HMTL ja Node.js.

Opinnäytetyön toteutusosiossa kerrotaan tarkemmin valituista menetelmistä ja siitä, miten niihin on päädytty sekä siitä, millä tavalla lopullinen tekninen toteutus on rakennettu. Opinnäytetyön lopuksi pohditaan työskentelymetodeja, ongelmatilanteita ja niiden ratkaisuja sekä sitä, miten työskentelyprosessi onnistui. Opinnäytetyöprosessin aikana valmistuneen infonäyttötoteutuksen toiminta testataan ja esitellään käyttöympäristössään.

2 Toimeksianto ja vaatimusten määrittely

2.1 Toimeksiantajan esittely ja alkutilanne

Pyhäselän kirjasto on pieni Joensuun maaseutualueella, Hammaslahden kylässä sijaitseva kirjasto. Kirjasto on yksi alueen kokoontumispaikoista, minkä vuoksi alueen kyläyhdistys toivoi muutama vuosi sitten, että tiloihin hankittaisiin

infonäyttö, jossa voisi mainostaa paikallisia tapahtumia. Kirjasto vastasi toiveeseen. Infonäyttöratkaisu toteutettiin hankkimalla jo olemassa olevalle 32-tuumaiselle televisiolle korkea jalallinen teline ja tekemällä mainostettavista julkaisuista muistitikulle valmis video, jonka henkilökunta laittaa television kaukosäätimellä päälle aina kirjaston auetessa. Näyttönä tässä ratkaisussa toimii nimenomaan televisio, jonka USB-paikkaan muistitikku kytketään. Infonäyttö on käytössä kirjaston omien sekä paikkakunnalla järjestettävien tapahtumien mainostukseen.

Infonäyttöön tehtävää videota varten on käytetty muistitikulla olevaa Movavi Video Editor -ohjelmaa, sillä se ei vaadi erillistä asennusta (ns. stand alone -sovellus) eli ohjelmaa voi käyttää siltä koneelta, johon muistitikku on kiinnitetty. Kirjaston henkilökunta ei voi asentaa ohjelmistoja koneisiinsa ilman Meitan (Meidän IT ja Talous) järjestelmänvalvojien apua, joten stand alone -mallinen sovellus on ollut tarpeellinen.

Pyhäselän kirjaston nykyinen muistitikulta toimiva infonäyttöratkaisu on ollut alkeellinen mutta toimiva. Se kuormittaa kuitenkin yhtä henkilöä, sillä jos jokin tapahtumamainoksista tai ilmoituksista halutaan poistaa videosta tai siihen lisätään uutta materiaalia, tämän henkilön tulee tehdä uusi infonäytöllä esitettävä video. Henkilökunnasta vain yksi henkilö osaa tehdä JPG-kuvista infonäytöllä esitettävän videon ja hän myös tarvittaessa muuntaa PDF-mainokset ensin JPG-muotoon, jotta video voidaan luoda Movavi-sovelluksessa. Vastaavanlainen ratkaisu on käytössä myös Joensuun pääkirjastolla, jossa muistitikulta esitetään mainoksista luotu PowerPoint-esitys.

Toimeksiantajan kanssa oli puhe, että sovellus olisi hyvä olla käytössä, kun Joensuun kaupungin Pyhäselän palvelupiste muuttaa kirjastolle tammikuussa 2021, sillä tuolloin ilmoitusten määrä potentiaalisesti lisääntyisi. Aikataulu oli silti joustava, sillä koronarajoitustilanteessa tapahtumia ja muuta tiedotettavaa oli niukasti.

2.2 Vaatimusmäärittely

Keskustelin vaatimusmäärittelyä varten Pyhäselän kirjaston henkilökunnan ja johtajan kanssa kirjaston infonäyttöratkaisusta. Ilmeni, että kirjaston työntekijät haluaisivat helppokäyttöisemmän infonäytön, jossa näytettävää materiaalia koko henkilökunta voisi taitotasoista riippumatta lisätä ja päivittää. Sisällön päivittämisen tulisi olla mahdollista ilman tiedostomuotojen muuntamista, videoitten luomista, muistitikkujen siirtelyä tai vaikeita ja monimutkaisia kirjautumisia. Teimme yhdessä kirjastonjohtajan kanssa vaatimusmäärittelyn infonäyttöratkaisua varten. Taulukossa 1 on nähtävillä asiakkaan vaatimukset infonäyttötoteutukselle. Tein vaatimusten pohjalta toteutusta koskevia valintoja.

Nro	**Vaatimus, syksy 2020**	**Pakollisuus**
1.	Infonäyttötoteutuksessa pitää pystyä näyttämään kuvia (jpg, png).	x
2.	Tuki videoille.	
3.	Infonäyttösovelluksen tulee toimia nykyisen television kautta.	x
4.	Infonäytön päivityksen tulee olla helppokäyttöinen.	x
5.	Kuvien/tiedostojen poiston pitää olla helppoa ja nopeaa.	x
6.	Infonäytön materiaalin päivityksen olisi hyvä toimia selaimen kautta, mieluiten Firefoxissa, sillä kirjastojärjestelmää käytetään sillä.	x
7.	Kustannukset minimoitava. Saatu lupa noin 100–150 euron hankintaan.	x
	Vaatimus, talvi 2021	
8.	Suurin osa mainoksista tulee PDF-tiedostoina, tuki niille.	
	Vaatimus, kevät 2021	
9.	Ohje kuvatiedostojen päivittämiseen.	x

Taulukko 1. Vaatimusmäärittely, Pyhäselän kirjaston infonäyttötoteutus.

Käyn seuraavaksi läpi ratkaisutapoja, joita olen käyttänyt infonäytön ja opinnäytetyöni toteutusprosessissa täyttääkseni taulukossa 1 esitetyt vaatimukset. Käyn vaatimukset läpi taulukon mukaisessa numerojärjestyksessä. Ratkaisuja pohtiessani olen pyrkinyt siihen, että ratkaisumallit olisivat sen tasoisia, että ne voisi viedä lopputoteutukseen saakka.

Vaatimus numero 1

Selaimen, jota käytetään kuvatiedostojen näyttämiseen, on tuettava vähintään yleisimpiä kuvatiedostoformaatteja (.jpg, .png ja .gif).

Vaatimus numero 2

Selaimen, jota käytetään kuvatiedostojen näyttämiseen, on tuettava jotain selainten tukemaa videotiedostoformaattia.

Vaatimus numero 3

Ohjelmiston/koodin sisältämässä laitteessa tulee olla HDMI-ulostulo.

Vaatimukset numero 4, 5 ja 6

Infonäytön kuvamateriaalin lisäykseen ja poistoon käytettävän selainpohjaisen palvelun käyttöliittymän tulee olla helppopääsyinen (helppo kirjautuminen millä tahansa tietokoneella tai älylaitteella) ja selkeä käyttää.

Vaatimus numero 7

Vaatimusmäärittelyssä sovimme myös kustannuksista. Koko toteutukseen laitteistohankintoineen oli käytettävissä 100–150 euroa. Tämän vuoksi käytettävän tiedostonjakopalvelun tuli olla ilmainen.

Harkitsin yleisimmistä pilvitallennuspalveluista Dropboxia ja Google Drivea. Kirjastolla oli käytössä myös Office 365 -pilvitallennustila, mutta tietosuojasyistä suljin tämän vaihtoehdon pois. Tietoturvan kannalta (laitteen varkaustapauksessa) Google Drivea käytettäessä ei menetettäisi edes laitteen käyttöön luotua tiliä, sillä Drive API -sovellusten käyttöoikeudet todennetaan OAuth 2.0:lla. OAuth 2.0 on valtuutuskehys, jonka avulla sovellukselle voi myöntää rajoitetun pääsyn HTTP-palvelun, kuten Facebook, GitHub ja DigitalOcean, käyttäjätileihin. OAuth 2.0 on Internet Engineering Task Forcen (IETF) ylläpitämä internet-

standardi (rfc6749). Se toimii delegoimalla käyttäjän todennus palvelulle, joka isännöi käyttäjätiliä, ja valtuuttamalla kolmannen osapuolen sovelluksia käyttämään käyttäjätiliä. (Anicas 2014; Internet Engineering Task Force 2012.) OAuth 2.0:lla käyttäjän todentaminen ei siis missään vaiheessa paljasta, kuka tilin käyttäjä todellisuudessa on, ja vaikka infonäyttölaitteen sisältöön pääsisi käsiksi, ei siihen liitettyä Google-tiliä voi varastaa. Laite ei siis sisällä infonäytössä esitettävän materiaalin lisäksi mitään tietoturvan vaarantavaa dataa. Infonäyttösovellukseen käytettävällä Google-tilillä ei olisi muutenkaan tarkoitus säilyttää mitään arkaluontoista materiaalia.

Vaatimus numero 8

Vaatimuksiin tuli jälkikäteen lisäys, jossa toivottiin, että toteutus tukisi PDF-tiedostojen lisäystä. Tämä sovittiin toteuttavaksi, jos aika riittää, ja ominaisuus on teknisesti helppo toteuttaa. Teknisen toteutuksen helppoudella tarkoitettiin tässä sitä, että PDF-tiedostojen konvertointiin löytyisi valmiita moduuleja.

Vaatimus numero 9

Kun infonäyttötoteutusta testattiin toukokuussa 2021, kirjastonjohtaja toivoi vielä, että tekisin helpon käyttöohjeen infonäyttöön tiedostoja päivittävälle henkilökunnalle. Ohje on liitteenä (liite 1).

3 Infonäyttöjärjestelmät

Digital signage -termi viittaa infotelevisio- ja näyttöteknologiaan. Infotelevisioilla ja -näytöillä tarkoitetaan erilaisissa julkisissa tiloissa käytössä olevia sisäiseen ja ulkoiseen markkinointiin, tiedottamiseen ja opastamiseen käytettäviä digitaalisia näyttöjä. (Wikipedia 2021a.) Niissä voi tiloista riippuen esittää mainoksia, tuote- ja hintatietoja, opasteita, ohjeita, pohjakarttoja, ajankohtaisia tiedotteita, livekuvaa, tekstiä, animaatioita tai muuta vastaavaa yleistä informaatiota. Infonäyttöjä on käytössä monenlaisissa tiloissa taloyhtiöistä ja kouluista tehtaisiin, sairaaloihin, julkiseen liikenteeseen, ravintoloihin ja muunlaiseen yritystoimintaan. Kun tarkkailee ympäristöään, huomaa, että infonäyttöjen ja erilaisten videoseinien käyttö on jo hyvin tavallinen osa markkinointia ja myös tuotannon

seurantaa. Infonäyttöjen avulla voidaan myös osallistaa asiakasta ilmoittautumaan tarjottuihin palveluihin, antamaan palautetta, tekemään tilauksia tai vaikkapa tutustumaan tarkemmin yritykseen.

Infonäyttötoteutuksilla nähdään olleen kolme sukupolvea (generation). Ensimmäisen sukupolven infonäytöt toimivat pistematriisiteknologialla, kuten LED- tai OLED-näyttöjen avulla. Näytettävä infosisältö on staattista, päivittäjä lisää viestit näytölle ja multimediatukea ei ole. Tällaisia toteutuksia on ollut esimerkiksi rautatie- ja lentoasemien lähtevien junien ja lentojen näyttötauluissa. Toisen sukupolven infonäyttötoteutukset käsittävät ne järjestelmät, joissa on käytössä erillinen ohjaava keskusyksikkö, mahdollisuus multimedian toistoon ja jotka voidaan päivittää esimerkiksi USB-muistitikulta, SD-kortilta tai CD-romilta. Median lisäys voi tapahtua myös internetin tai muiden verkkoyhteyksien avulla. Kolmannen sukupolven infonäyttöversiot ovat interaktiivisia. Käyttäjä voi kosketusnäytön avulla esimerkiksi selata tai etsiä tuotetietoja tai vaikkapa jakaa informaatiota tai kuvia verkkoon ja sosiaaliseen mediaan. Näitten kolmen sukupolven infonäyttötoteutusten edeltäjäksi nähdään sähköpaperi. (Wikipedia 2021a.)

Toisen sukupolven infonäyttötoteutuksen voi tehdä helposti kuvakaruselliesitystä tai esitysgrafiikkatiedostoa pyörittävän tietokoneen tai tablettitietokoneen avulla. Omatekoinen infonäyttöjärjestelmä syntyy vaivattomasti myös Pyhäselän kirjastossa tällä hetkellä käytössä olevalla mallilla. Tässä toteutuksessa televisioon liitetyltä muistitikulta näytetään kuva- tai videoesitystä television käyttöjärjestelmän kuvan- tai videonkatseluominaisuutta hyödyntäen. Joensuun pääkirjastolla on käytössä Raspberry Pi:llä tehty infonäyttötoteutus, joka toistaa muistitikulle tallennetun PowerPoint-esityksen.

Infonäyttöjärjestelmän voi luoda myös All in one -PC:n avulla. All in one -PC:ssä PC:n komponentit on upotettu näytön sisään, jolloin infonäyttötoteutus voidaan saada aikaan perustason sovelluksella tai ostetulla ohjelmistolla. Tekemäni havainnoinnin perusteella näitä ratkaisuja on käytössä esimerkiksi ruokakaupoissa, muun muassa joissakin K-ryhmän liikkeissä. Olen nähnyt marketeissa myös infonäyttökokonaisuuksia, joissa näytön taakse on rakennettu kehys, jonka sisälle on mitä ilmeisimmin upotettu minitietokone, sillä kokonaisuus on yhdistetty verkkoon verkkojohdolla. Tällaisia toteutuksia on käytössä esimerkiksi S-ryhmän liikkeissä.

Markkinoilla on myös monia erilaisia kaupallisia infonäyttötoteutuksia. Yksinkertaisimpia toteutuksia tarjotaan yritysten muiden tuotteiden, kuten kameravalvontatekniikan, antennitekniikan, mittalaitteistojen ja muiden vastaavien laitteistojen myynnin ohessa. Eräs tällaisia monenlaisia palvelukokonaisuuksia tarjoava yritys on Finnsat. Finnsat tarjoaa taloyhtiöille omaa Rapputaulu-toteutustaan, joka on ”digitaalinen etähallittava tiedotuskanava”. Finnsatin toteutukseen kuuluu 43-tuumainen LED-näyttö, info-tv-päätelaite, lukittava kotelo ja tarvittavat konfiguroinnit näytölle ja päätelaitteelle. Palvelu on kuukausimaksullinen. Toteutus toimii sekä langallisessa että langattomassa verkossa. Toteutuksessa on myös ajastustoiminto, jolloin vain ajankohtainen sisältö näkyy infoesityksessä, eikä sisällön päivittämisestä tarvitse tällöin huolehtia. Yrityksille Finnsatilla on tarjolla Ruutu-infonäyttöjärjestelmiä, joissa yhden päätelaitteen sisältöä voi jakaa KanexPro-lähettimen avulla useammalle näytölle. Tässä hyödynnetään CAT-6- eli ethernet-kaapelointia ja 8-paikkaista HDMI-jakajaa. Finnsatin kautta yrityskäyttöön tarjolla olevat infonäyttötoteutukset ovat Rapputaulu-toteutusta suurempia (65-tuumaisia). Niihin tarjotaan oheislaitteina muun muassa langattomia kuvansiirtojärjestelmiä (Barco Clickshare). Ruutu-järjestelmien päätelaitteena Finnsat käyttää Raspberry Pi:tä. Finnsatin sivuston mukaan päätelaite hakee sisällöt Finnsatin omasta Ruutu-pilvipalvelusta, johon sisällöt syötetään kirjautumisen jälkeen selaimella. Käyttäjätunnuksia voi olla useita eritasoisin käyttöoikeuksin. Finnsatin mukaan ”näyttö voidaan jakaa kolmeen osaan ja jokaiseen osaan voidaan sisällöt ajastaa ja priorisoida erikseen”. (Finnsat 2021.)

AVconcept-yritys edustaa Suomessa Valotalive- ja Signagelive-infonäyttötoteutuksia. Yritys keskittyy palveluissaan niihin sekä erilaisiin esitystekniikan toteutuksiin. (AVconcept 2021.) Valotalive on suomalainen yritys, jonka tarjoama palvelu on myös pilvipohjainen, ja siihen kytkettyjä näyttöjä voi olla yhdestä satoihin. Valotaliven infonäyttöpalvelu on käytössä esimerkiksi Nesteellä, Roviolla ja Vismalla. Valotaliven referenssikuvissa ja -videoissa on esimerkkejä siitä, miten infonäyttötoteutuksia voi käyttää hyödyksi yritysmaailmassa. Yhdessä esimerkeistä voi nähdä esimerkiksi MS Power BI -pohjaisen liveseurannan yrityksen KPI-toteutumasta. ”Key Performance Indicator eli KPI on suomeksi käännettynä mitattava arvo, joka osoittaa, kuinka tehokkaasti yritys saavuttaa liiketoiminnan keskeiset tavoitteet. Organisaatiot käyttävät KPI-mittareita useilla tasoilla arvioidakseen yritystä tavoitteiden saavuttamisessa.” (Turunen 2020.)

Infonäyttöjen avulla voidaan siis käytännössä toteuttaa myös tuotannon seurannan visualisointia esimerkiksi tehdasympäristöissä.

Pohjois-Karjalan alueen Vaara-kirjastoissa on infonäyttöjä käytössä tämänhetkisen tietoni mukaan vain Joensuussa ja Pyhäselässä. Infonäyttöjen käyttöönotto kirjastoympäristössä onkin vähäisempää kuin kaupallisessa toiminnassa. Julkisella sektorilla infonäytöt ovat vasta hiljalleen löytämässä käyttötarkoituksensa. Kirjastoja useammin infonäyttö löytyy julkisen terveydenhuollon toimipisteen tai museon tiloista. Suuremmilla paikkakunnilla infonäytöt ovat kuitenkin jo yleistyneet myös kirjastoissa. Helsingin, Espoon ja Vantaan alueella miltei jokaisessa kirjastossa on käytössä infonäyttö, joiden sisältöjä kaikki viestinnästä vastaavat henkilöt päivittävät. Helsingin keskustakirjasto Oodissa infonäyttöjä on useampia jokaisessa kerroksessa. Vantaalla sijaitsevan Pointin kirjaston infonäyttömateriaali löytyy osoitteesta: https://kirjasto.one/info/point/. (Kärkkäinen 2021). Sivuston lähdekoodia tarkastelemalla voi havaita, että Vantaan Pointissa on käytössä hiukan vastaavanlainen toteutus kuin Raspberry Pi:llä toteuttamani infonäyttökokoonpano. Sisältö lisätään palvelimelle, josta WordPress-pohjainen HTML-sivu tarjoaa sen infonäyttöön käyttämällä visualisointiefektejä, jotka sisältyvät erilaista mediasisältöä renderöivään Fancybox-jquery-lisäosaan. Toteutus ei ole responsiivinen, eli se ei skaalaa materiaalia käytettävän näyttöalueen mukaan, vaan olettaa näyttöalueen olevan pystysuuntaan asennetun 16:9-kuvasuhdetta käyttävän näyttölaitteen muotoinen. Tästä syystä myös kuvasisällön, jota toteutuksella halutaan näyttää, täytyy olla samassa kuvasuhteessa ja sopivan kokoinen.

Infonäytöt eivät kirjastoissa palvele ainoastaan mainontaa ja tiedottamista. Osassa kirjastoista on jo käytössä RFID-viivakoodeja, jotka hyödyntävät radiotaajuuksista etätunnistusta. RFID-tekniikkaa apuna käyttäen niteen tiedot haettuaan voi saada esille sen ajantasaisen sijainnin kirjastossa, vaikka teos olisi väärässä sijainnissakin. RFID-tekniikkaa käytetään myös avuksi lainatessa ja palauttaessa, jolloin käsilukijaa ei enää tarvita, vaan RFID:n avulla teokset lainautuvat tai palautuvat automaattisesti tietylle alustalle asetettaessa. Tekniikkaa voidaan käyttää myös varkauksien estoon. (Suomen 3M Oy 2021.) Infonäyttöjä voisi käyttää RFID-tunnisteiden kanssa apuna esimerkiksi havainnollistamaan pohjakartalta hyllysijaintia tai kulkureittiä oikealle hyllylle. Tästä olisi apua sekä asiakkaille että kirjastohenkilökunnalle.

4 Raspberry Pi

4.1 Historiatausta

Raspberry Pi Foundation perustettiin vuonna 2009 tukemaan kouluissa tapahtuvaa tietotekniikan opetusta. Tarkoituksena oli myös luoda edullinen tietokone, jolla parantaa opiskelijoiden hiipunutta kiinnostusta tietotekniikan opiskeluun. Alun perin säätiön toiminta perustui Cambridgen yliopistoalueelle, Isoon-Britanniaan. (Hitaltech 2021.) Raspberry Pi on Raspberry Pi Foundationin ja puolijohdevalmistaja Broadcomin yhteistyössä kehittämä korttitietokone, jonka kaikki komponentit on sijoitettu yhdelle piirilevylle. Laitteesta tuli odotettua suositumpi ja sen käyttö levisi muun muassa robotiikkaan. Pienen kokonsa ansiosta se sopii moniin eri tehtäviin. Nykyisin se on suosittu tietokone- ja elektroniikkaharrastajien keskuudessa kätevien USB- ja HDMI-liitäntöjensä ansiosta. (Wikipedia 2021b.)

Laitetta on julkaistu neljässä sukupolvessa (generation) erilaisilla piirilevyillä (taulukko 2). Versioiden mallinumeroiden välillä on eroavaisuuksia etenkin suoritintehojen ja liitäntöjen osalta. Ensimmäinen malli, Raspberry Pi 1 Model B, julkaistiin helmikuussa 2012. Siinä oli Broadcom BCM2835 -piirisarja, 700 MHz:n prosessori ja 512 Mt:n keskusmuisti. Opinnäytetyöni teknisessä toteutuksessa on käytetty kesäkuussa 2019 julkaistua mallia Raspberry Pi 4 Model B. Sen suorituskyky kasvoi edeltävistä Raspberry Pi 3 -malleista noin kolminkertaiseksi, ja se kykenee tuottamaan kuvaa 4K-resoluutiolla. (Hitaltech 2021; Wikipedia 2021b; Embedded Linux Wiki 2021.)

	Raspberry Pi 1 (Model B+)	**Raspberry Pi 2 (Model B)**	**Raspberry Pi Zero**	**Raspberry Pi 3 (Model B v1.2)**	**Raspberry Pi 4 Model B**
Julkaisupäivä:	14.7.2014	2.2.2015	26.11.2015	29.2.2016	24.6.2019
Prosessori:	700 MHz, yksiytiminen, ARM11, 32-bittinen, ARMv6	900 MHz, moniydinprosessori, ARM Cortex-A7, 32-bittinen, ARMv7-A	1 GHz, yksiytiminen, ARM11, 32-bittinen, ARMv6	1.2 GHz, moniydinprosessori, ARM Cortex-A53, 64-bittinen, ARMv8-A	1.5 GHz, moniydinprosessori, ARM Cortex-A72, 64-bittinen, ARMv8-A

	Raspberry Pi 1 (Model B+)	**Raspberry Pi 2 (Model B)**	**Raspberry Pi Zero**	**Raspberry Pi 3 (Model B v1.2)**	**Raspberry Pi 4 Model B**
Näytönohjain:	Broadcom VideoCore IV, OpenGL ES 2.0, 1080p, h.264, 250MHz			Broadcom VideoCore IV, OpenGL ES 2.0, 1080p, h.264, 400MHz	Broadcom VideoCore VI, OpenGL ES 3.0, 1080p, h.265, 500MHz
Piirisarja:	Broadcom BCM2835 (CPU, GPU, DSP, SDRAM)	Broadcom BCM2836 (CPU, GPU, DSP, SDRAM)	Broadcom BCM2835 (CPU, GPU, DSP, SDRAM)	Broadcom BCM2837 (CPU, GPU, DSP, SDRAM)	Broadcom BCM2711 (CPU, GPU, DSP, SDRAM)
Muisti:	512 Mt (jaettu näytönohjaimen kanssa)	1 Gt (jaettu näytönohjaimen kanssa)	512 Mt (jaettu näytönohjaimen kanssa)	1 Gt (jaettu näytönohjaimen kanssa)	1/2/4/8 Gt LPDDR4-2400 (jaettu näytönohjaimen kanssa)
USB-portit:	4 kpl	4 kpl	2 kpl (microUSB)	4 kpl	4 kpl, 2*USB 3.0 ja 2*USB 2.0
Videoulostulot:	HDMI (rev 1.4), komposiitti RCA (PAL & NTSC), Model A+/B+ jakkiliitin korvaa RCA:n		miniHDMI, GPIO (komposiitti)	HDMI (rev 1.4), komposiitti RCA (PAL & NTSC), Model A+/B+ jakkiliitin korvaa RCA:n	HDMI mikro, 2kpl, tukee 4kp60 saakka komposiitti RCA (PAL & NTSC), Model A+/B+ jakkiliitin korvaa RCA:n
Ääni:	HDMI, 3,5 mm jakki		miniHDMI, GPIO	HDMI, 3,5 mm jakki	
Massamuisti:	SD / MMC / SDIO-muistikortti, Model A+/B+ vain Secure Digital				
Verkkosovitin:	10/100 Ethernet (RJ45) (USB-Ethernet-silta)	10/100 Ethernet (RJ45) (USB-Ethernet-silta)	Ei ole	10/100 Ethernet (RJ45) (USB-Ethernet-silta)	10/100/1000 Ethernet (RJ45)
Langattomat yhteydet:	Ei ole			802.11n ja Bluetooth 4.1	802.11ac ja Bluetooth 5, BLE
Muut liittimet:	40-pinninen GPIO-liitin				
Koko:	85,60 × 53,98 × 17 mm		65 mm × 30 mm × 5 mm	85,60 × 53,98 × 17 mm	88 x 58 x 19.5 mm
Paino:	45 g	45 g	9 g	45 g	46 g

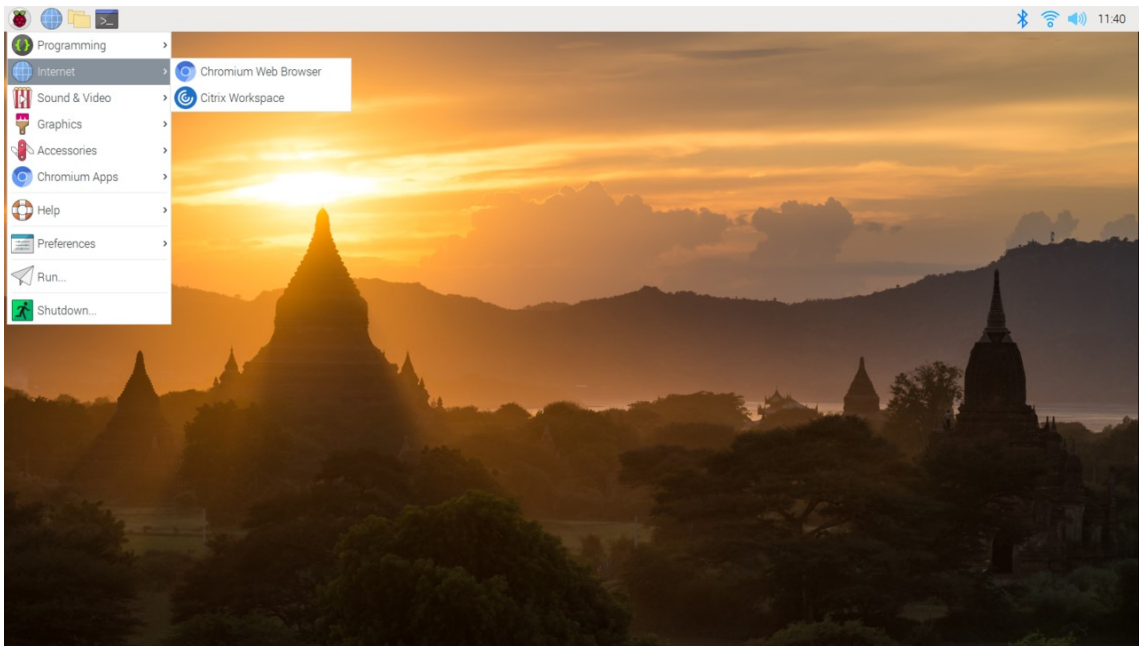
Taulukko 2. Raspberry Pi -korttitietokoneiden teknisiä ominaisuuksia, mukailtu versio (Wikipedia 2021b; Embedded Linux Wiki 2021).

Raspberry Pi 4 Model B:n julkaisun jälkeen Raspberry-tuotteista on tullut saataville lisäksi näppäimistön sisään integroitu malli Raspberry Pi 400 ja yhdellä mikrokontrollerilla varustettu pienikokoinen Raspberry Pi Pico (Raspberry Pi Foundation 2021b).

4.2 Raspberry Pi OS -käyttöjärjestelmä

Raspberry Pi OS -käyttöjärjestelmä on suunniteltu ja optimoitu toimimaan erityisesti kaikissa Raspberry Pi -laitteissa paitsi Raspberry Pi Picossa (Raspberry Pi Foundation 2021a). Se on kustomoitu versio Debian-käyttöjärjestelmästä. Raspberry Pi OS on ollut vuodesta 2015 oletuskäyttöjärjestelmänä Raspberry Pi -aloituspakkauksissa. (Wikipedia 2021c.)

Raspberry Pi OS -käyttöjärjestelmän voi asentaa esimerkiksi virallisten Raspberry Pi -aloituspakettien mukana tulevalta MicroSD-muistikortilta, joissa käyttöjärjestelmän asennusohjelma on esiasennettuna. Käyttöjärjestelmän voi myös ladata verkosta levykuvana. Käyttöjärjestelmästä on saatavilla versiot, jotka toimivat vain Raspberry-laitteilla, sekä työpöytäversiot muille prosessorityypeille. Valitusta levykuvasta riippuen käyttöjärjestelmän mukana asentuu myös eri määrä sovelluksia ja apuohjelmia. Kevyimmässä Lite-versiossakin käyttöjärjestelmään asentuu Chromium-web-selain ja PCMan -tiedostoselain (kuva 1). (Raspberry Pi Foundation 2021c; 2021d.)



Kuva 1. Raspberry Pi OS -käyttöjärjestelmän työpöytänäkymä ja Chromium-oletusselain (Wikipedia 2021c).

Raspberry Pi OS -käyttöjärjestelmä käyttää työpöytäympäristönään kevyttä, monikielistä LXDE-työpöytäympäristöä (Lightweight X11 Desktop Environment)

ja ikkunointiohjelmana OpenBoxia. LXDE:tä käytetään useiden Linux-jakeluiden kevytversioissa alhaisten järjestelmävaatimustensa takia. Openbox on kevyt ikkunamanageri, jota voidaan käyttää X-ikkunointia tukevien käyttöjärjestelmien kanssa. (Raspberry Pi Foundation 2021c; LXDE Wiki 2021a; LXDE Wiki 2021b.)

5 Laitteiston ja käyttöjärjestelmän valinta

5.1 Korttitietokoneen valinta ja perusteet

Ehdotin heti aluksi toimeksiantajalle, että infonäyttötoteutuksen materiaalia näyttäväksi laitteistoksi voisi hankkia korttitietokoneen, ja päätin, että toteutuksen siirrettävyyden vuoksi infomateriaalin esittäminen tapahtuisi selaimella. Korttitietokone olisi helppo piilottaa kirjastoasiakkaitten näkyvistä eikä veisi tilaa. Selaimen käyttäminen jaettavan materiaalin renderöintiin lisäisi toteutuksen kannettavuutta ja eri järjestelmien välistä yhteensopivuutta. Infonäyttötoteutus toimisi myös siinä tapauksessa, jos sitä haluttaisiin käyttää useammassakin kohteessa eri laitteilla, käyttöjärjestelmissä ja erityyppisten näyttölaitteiden kanssa.

Otin selvää erilaisten saatavilla olevien korttitietokoneiden suorituskyvystä ja tarkastelin niiden kykyä renderöidä täyden resoluution (FullHD) videoita selaimessa. Tämä oli yksi esille tulleista vaatimuksista (taulukko 1), joskaan ei pakollinen. Käytin sitä silti rajaavana tekijänä laitetta valittaessa. Vuoden 2020 syksyllä saatavilla olevista laitevaihtoehdoista kriteerit täytti Raspberry Pi 4 Model B -korttitietokone. Se oli sillä hetkellä uusin Raspberry Pi -malli. Päätin valita kyseisen mallin toteutukseni pohjalle. Käytän jatkossa tästä valitsemastani korttitietokonemallista nimeä Raspberry Pi 4.

Info-tv-toteutukseni yksinkertaisuuden vuoksi arvioin Raspberry Pi 4:n keskusmuistin kooksi riittävän hyvin 4 gigatavua. Suunnittelin, että korttitietokoneen massamuistina voisi toimia muistikortti, sillä tallennustilaa tultaisiin tarvitsemaan toteutukseeni hyvin vähän ja massamuistin suorituskyvyllä ei sinänsä olisi

myöskään väliä. Toinen vaihtoehto olisi ollut ulkoinen USB-asema, jonka käyttäminen olisi kuitenkin kasvattanut laitekokonaisuuden kokoa, eikä lisääntyneestä massamuistin määrästä ja suorituskyvystä olisi ollut toteutuksen kannalta hyötyä. Ulkoinen levyasema olisi vaikuttanut korkeintaan muutaman sekunnin verran laitteen käyttöjärjestelmän ja sovelluksen käynnistymisnopeuteen.

Laitteen virransyötön tavaksi suunnittelin näyttölaitteena toimivan television USB-liitäntää. Asiaan perehdyttyäni ja virallisen Raspberry Pi -foorumin (Raspberry Pi Foundation 2021e) keskusteluja tutkiessani selvisi, että television tai näytön USB-liitinten tuottama vähäinen virta ei riitä Raspberry Pi 4:n virtalähteeksi. Sen kanssa suositeltiin käytettäväksi virallista ulkoista virtalähdettä. Toisekseen, jos korttitietokoneen virta katkeaisi aina TV:n tai näytön virran sammuessa, se saattaisi aiheuttaa muistikortin tiedostojen korruptoitumista. Näistä syistä päätin hankkia laitteen, jolla on oma virtalähde.

Virallisen kotelon lämmönsiirto oli Zdnet-sivuston (Watson 2019) mukaan riittämätön, joten päätin etsiä paremman jäähdytystehon vuoksi konepaketin, jossa jäähdytys oli otettu huomioon. Korttitietokoneen mahdollinen sijoittaminen lämpöä tuottavan television tai monitorin taakse voisi kasvattaa korttitietokoneen jo entisestään korkeaa lämpötilaa, enkä halunnut ottaa tietoista riskiä ja kokeilla onneani mahdollisten lämpötilaongelmien suhteen.

Korttitietokonetta valitessani kävin läpi suorituskykyä, keskusmuistia, lämpötilaa ja virransyöttöä koskevat seikat. Ne huomioiden päädyin hankkimaan OKdo-tuotemerkillä myydyn Raspberry Pi 4 Model B -valmispaketin, jossa jäähdytys oli otettu huomioon ja joka sisälsi valmiina haluamani lisälaitteet (kuva 2). Paketissa oli korttitietokoneen lisäksi muistikortti, muistikortin lukija, ulkoinen virtalähde, HDMI-kaapeli, suojakotelo, jäähdytyssiilit kortin komponenteille, kotelon ulkopuolelle asennettavat kumiset jalkatassut sekä tuuletin. Tämän pakkauksen mukana tullut suojakotelo oli jäähdytysratkaisuna parempi kuin virallinen kotelomalli. Laitehankinta tuli tehdä toimeksiantajan kautta tiettyjen kilpailutettujen hankintapaikkojen valikoimista, joten tämä paketti oli virallisen Raspberry Pi 4 -aloituspakkauksen lisäksi ainut varteenotettava vaihtoehto.

Kuva 2. OKdo-tuotemerkillä myyty Raspberry Pi 4 Model B -aloituspakkaus (OKdo 2021.)

Tämän opinnäytetyön toteutuksessa käytetyn OKdon Raspberry Pi 4 Model B -mallin laitteistokokoonpano on seuraava:

- 1.5GHz neliytiminen 64-bittinen ARM-prosessori,
- USB-C-virransyöttö,
- 2 x Micro HDMI -liitin,
- 4Gt RAM-muisti,
- 2 x USB 2.0 -liitin,
- 2 x USB 3.0 -liitin.

5.2 Käyttöjärjestelmän valinta ja perusteet

Raspberry Pi 4:lle asennettavaa käyttöjärjestelmää valitessani tutkin eri vaihtoehtoja lukemalla Raspberry Pi:n viralliselta foorumilta käyttäjien havaintoja eri Linux-jakeluista. Keskusteluissa ilmeni, että Raspberry Pi 4:lle soveltuvia epävirallisia käyttöjärjestelmävaihtoehtoja oli useampi, mutta kaikkien kanssa esiintyi suorituskykyongelmia tai laitteiston toimintaan liittyviä puutteita ja ongelmia. Nämä ongelmat saattaisivat heikentää infonäyttötoteutuksen vakautta tai aiheuttaa muita ongelmia toteutuksen kanssa. Hylkäsin muut vaihtoehdot ja päädyin Raspberry Pi Foundationin ylläpitämään ja virallisesti tukemaan Raspberry Pi OS -käyttöjärjestelmään. Raspberry Pi OS on Debian-käyttöjärjestelmän

muunnos, joka on optimoitu toimimaan kaikissa Raspberry-laitteissa (Raspberry Pi Foundation 2021e).

Käyttöjärjestelmävalintaani vaikutti myös Raspberry Pi OS:n käyttämä ext4-tiedostojärjestelmä, joka tukee journalointia. Sähkövikojen aiheuttama tiedostojen korruptoituminen on journaloinnin ansiosta hyvin epätodennäköistä, sillä se takaa korkean vikasietoisuuden tiedostojärjestelmän vikatilanteissa (Vähimaa 2021, 55–56).

> Journalointi perustuu tyypillisesti erilliseen tiedoston tai levyn alueeseen, journaliin, johon levylle tehtävät muutokset kirjoitetaan ennen kuin ne konkreettisesti suoritetaan. Tämä tekee muutoksista jakamattomia: mikäli virhetilanne tapahtuu ennen kuin muutokset on kirjoitettu journaliin kokonaisuudessaan, tiedostojärjestelmä jättää ne järjestelmän palauduttua huomiotta ja levykirjanpito palautuu alkuperäiseen tilaansa ennen muutoksia. Jos taas muutokset on ehditty kirjoittaa journaliin kokonaisuudessaan, järjestelmä suorittaa kaikki nämä muutokset myös levylle. Molemmissa tapauksissa levykirjanpito säilyy eheänä, ja pahimmassakin tapauksessa vain levylle juuri ongelmatilanteen aikana kirjoitettavat tiedot menetetään. (Vähimaa 2021, 55–56).

Raspberry Pi OS:sta on tarjolla myös työpöytä-PC-versio, jota voi ajaa esimerkiksi virtuaalikoneella (Raspberry Pi Foundation 2021d). Käyttöjärjestelmän ajamisen mahdollisuus virtuaalikoneella hyödyttäisi kehitystyötäni, sillä minun ei tarvitsisi odotella Raspberry Pi 4:n hankkimista ja toimitusta ennen kuin pystyisin tutustumaan käyttöjärjestelmään ja aloittamaan kehitystyön. Virtuaalikoneella ajetun ympäristön suorituskyky on myös parempi kuin Raspberry Pi 4:lla.

Harkitsin Raspberry Pi 4:ään asennettavan ARM-käyttöjärjestelmän emulointia, mutta Virtualbox ei tukenut sitä. Olisin joutunut käyttämään QEmu-sovellusta emulointiin. En nähnyt, että tällaisesta emuloinnista olisi sovelluskehityksen kannalta hyötyä. Sovelluksestani tulisi joka tapauksessa helposti siirrettävä. Kehittämisen aikana käytetty käyttöjärjestelmä ei tulisi suorituskykyä lukuun ottamatta olemaan suuressa roolissa kehityksessä, joten laiteympäristö ja käyttöjärjestelmä saisi kehitysvaiheessa olla mielestäni mikä tahansa. Mikäli työpöytäversio käyttöjärjestelmästä poikkeaisi esimerkiksi käynnistysmääritysten tai muiden omaan sovellukseeni liittymättömien seikkojen suhteen, saisin ne

määritettyä Raspberry Pi 4:ssä myöhemmässä vaiheessa. Päätin siis tyytyä sovellukseni testaamisalustana työpöytäversioon Raspberry Pi OS -käyttöjärjestelmästä. Muun sovelluksen ympärille tarvittavan käyttöjärjestelmäympäristön muokkaamisen, esimerkiksi käynnistysasetusten, laitteistoasetusten ja ajastettavien tehtävien muokkaamisen päätin jättää tehtäväksi Raspberry Pi 4:ssä ja sen ARM-käyttöjärjestelmässä. Kerron tästä työvaiheesta tarkemmin Toteutusprosessi-luvussa (luku 7).

6 Ohjelmointityökalujen valinta

6.1 Koodaustyökalut

Sovellukseni koodia työstin niin Windows 10 -ympäristössä kuin virtuaalikoneella testiympäristössä Raspberry Pi OS:n x86 -käyttöjärjestelmässä. Windows 10:ssä käytössäni oli Visual Studio Code -ohjelmointityökalu, joka oli minulle aiemmista projekteista tuttu. Raspberry Pi OS -käyttöjärjestelmässä esiasennettuna ollut Geany-sovellus oli minulle uusi tuttavuus. Se ei sisältänyt lainkaan kattavia ohjelmointityökaluja, jotka löytyivät Visual Studio Code:sta, mutta se ajoi silti asiansa monia tiedostomuotoja tukevana yksinkertaisena ohjelmointityökaluna.

6.2 Google Drive API

Päädyin käyttämään Googlen Drive -palvelua infonäytön materiaalin jakopalveluksi, sillä Googlen tarjoama sähköpostipalvelu ja Drive-palvelu oli kirjaston henkilökunnalle jo jokseenkin tuttu. Google Driven yksinkertainen käyttöliittymä on helppo käyttää, ja Drive-kansionäkymään pääsee kirjautumaan helposti Google-tunnuksilla aivan kuten sähköpostiinkin. Drive-kansioon voi luoda selaimen kirjanmerkkeihin suoran linkin, joka helpottaa ja nopeuttaa Drive-kansioon pääsyä.

Google-tilin luominen ja Drive-palvelun käyttö on ilmaista. Kuvamateriaalia sisältämän pilvipalvelun täytyi olla maksuttomasti käytettävä, sillä toteutusbudjetti

käsitti vain laitehankintakustannukset. Google Driven käyttö on tietoturvan kannalta hyvä ratkaisu Google Drive API-sovellusten käyttämän token-järjestelmän vuoksi. Kerron token-järjestelmästä tarkemmin luvussa Google Drive -tiedostojen käsittelyskripti (luku 7.3) ja sen alaluvuissa.

Tutkin aluksi Google Drive API:n Node.js-toteutusta, sillä suunnittelin toteuttavani myös paikallisen tiedostopalvelimeni Node.js:llä. Kokeilin luoda yksinkertaisen tiedostolataajan Googlen esimerkkisovelluksesta. Huomasin kuitenkin jo Drive-lataussovellukseni kehityksen alkuvaiheessa, että Node.js-ympäristössä minun tulisi jatkuvasti ottaa huomioon se, että Node.js:ssä kaikki tapahtuu asynkronisesti ja se on estämätön järjestelmä. Tästä syystä esimerkiksi HTTP-pyyntöjen käsittely ja tiedostoon kirjoittaminen ei pysäytä ohjelman suoritusta eikä estä toisia operaatioita ennen datan palauttamista. (Peltomäki 2020, 15.) Tämä seikka monimutkaistaisi huomattavasti koodin laatimista, sillä sovellukseni tulisi odottaa tiedostojen lataamisen, tiedostotarkistuksen ja useiden muiden tapahtumien valmistumista ennen kuin koodissa saataisiin siirtyä eteenpäin ja päivittää kuvamateriaali vasta onnistuneen tiedostojenkäsittelyn jälkeen. Päädyin vaihtamaan Drive-tiedostolataajani toteutuksen Google Drive API:n Python-versiota käyttäväksi Python-sovellukseksi, sillä se olisi loogisempi, yksinkertaisempi ja suoraviivaisempi toteuttaa.

6.3 Node.js

Node.js on opintojeni aikana tutuksi tullut Javascript-kielen laajennus. Se on suunniteltu skaalautuvien verkkosovelluksien laatimiseen. Node.js on asynkroninen ja tapahtumapohjainen ohjelmointiympäristö. (Nodejs.org 2021.) Node.js:llä voi luoda verkko-ohjelmia, jotka voivat olla palvelinohjelmia, palveluita tai asiakasohjelmia, jotka tukevat esimerkiksi HTTP-, TCP-, UDP-, DNS- sekä SSL-protokollia. Node.js:llä voi luoda myös ohjelmia, jotka voivat kirjoittaa I/O-kanaviin ja lukea I/O-kanavia (tiedostojärjestelmät, tietokannat, prosessit, muisti sekä erilaiset muut tietovirrat). Node.js:n I/O-tuki on monipuolinen ja nopea. (Peltomäki 2020, 14.)

Raspberry Pi 4:n käyttöjärjestelmään ja virtuaalikoneen Raspberry Pi OS -käyttöjärjestelmään oli esiasennettu Node.js:stä versio 14. Asennuksen mukana tulevalla npm-paketinhallintasovelluksella asensin palvelinsovellukseni toteutukseen tarvitsemani moduulit. Npm on paketinhallintasovellus Node.js:n moduuleille (Npm Docs 2021). Näiden moduulien avulla toteutin Node.js-ympäristöön paikallisen palvelimen ja palvelimen sisällön muutoksia seuraavan skriptin.

6.4 Python

Python on avoimen lähdekoodin ohjelmointikieli, johon olen tutustunut myös opintojeni kautta. Python toimii useilla alustoilla eri käyttöjärjestelmissä (muun muassa Windows, Mac, Linux ja Raspberry Pi). Python on tulkkaava, interaktiivinen olio-ohjelmointikieli. Python-kielessä hyödynnetään moduuleja, se sisältää poikkeusten käsittelyn ja se tukee dynaamista kirjoittamista (muuttujien tyyppejä ei tarvitse erikseen määrittää, ne päätetään ajon aikana). Se sopii oliomallisen ohjelmoinnin lisäksi esimerkiksi proseduraaliseen ohjelmointiin. Proseduraalisessa ohjelmoinnissa tietokoneelle luodaan luettelo ohjeista, mitä sen tulisi askel askeleelta suorittaa käsillä olevan tehtävän loppuun saattamiseksi. Tällainen ohjelmointitapa oli hyödyksi Drive-tiedostoja käsittelevän skriptini luomisessa. (Python.org 2021a; Bhatia 2021.)

Raspberry Pi OS -käyttöjärjestelmään oli esiasennettu Pythonin versio 3.7, jota päädyin käyttämään. Python-asennuksen mukana asentuvaa pip-paketinhallintajärjestelmää hyödynsin Python-laajennusmoduulien, kuten PDF-konvertoijamoduulin asentamiseen.

Virtuaalikoneen Python-versioksi valitsin uusimman vakaan julkaisun Python-kirjastosta, joka oli asennushetkellä versio 3.9.4. Google Drive API:n järjestelmävaatimuksena oli Pythonin versio 2.6 tai uudempi. Ohjelmoin tiedostolataajan pääosin Windows 10:ssä. Pythonin Windows-asennus oli helppoa ja tapahtui asentamalla Python suoraan asennuspaketista oletusasetuksin. Visual Studio Code ymmärsi suositella Pythonille lisättäväksi tarvittavan apukirjaston oikeinkirjoituksen ja funktioiden tarkistukseen, minkä asensin Visual Studio Coden

sitä ehdottaessa. Etsin myös muita Python-koodaamista hyödyttäviä laajennuksia ja päädyin lisäämään Visual Studio Codeen myös RainBow-sovelluslaajennuksen. Se auttaa visualisoimaan Python-koodin sisennyksiä eri värein ja näin helpottaa koodin tulkitsemista ja sen sisäkkäisten rakenteiden tarkastelua ja laatimista.

6.5 Firefox-selain

Valitsin infonäyttötoteutuksen kuvakarusellin esittäväksi selaimeksi Firefox-selaimen ESR-version, joka ei omatoimisesti lataa tai pyydä asentamaan päivityksiä. Se oli helppo valinta projektiini, sillä tiesin jo entuudestaan sen sisältävän niin sanotun Kiosk-moodin, jossa selaimen käyttöliittymä piiloutuu ja selainnäkymän web-sivusto laajentuu täyttämään koko näyttölaitteen alueen.

Otin myös selvää Firefoxin kustomointimahdollisuuksista, sillä halusin poistaa esimerkiksi sivustojen latauksen yhteydessä näytettävän lataus-/informaatiopalkin, joka näkyy Firefox-selaimessa vasemmassa alalaidassa, kun selain lataa sivustoa tai kun selain ottaa yhteyttä palvelimeen. Tämän ja myös monien muiden ominaisuuksien muokkaaminen paljastui mahdolliseksi. (Mozilla 2021.) Hyödynsin myös Firefoxin kehittäjätyökaluja[1] HTML-sivuani ja kuvakarusellitoteutusta luodessa.

7 Toteutusprosessi

7.1 Sovellustoteutuksen rakenne

Toteutin infonäyttöjärjestelmäni ohjelmiston luomalla Python-skriptin käsittelemään (synkronoimaan) Raspberry Pi 4:n paikallisia tiedostoja Drive-tiedostojen kanssa (drivelogged.py). Paikallisen palvelimen, jonka sisältöä Firefox-selain näyttäisi, toteutin luomalla Node.js-palvelinskriptin (nodeServer.js). Sama Node.js-skripti huolehtii myös selainnäkymän sisällön päivittämisestä tarpeen

[1] Kehittäjätyökaluihin pääsee selaimen asetusvalikosta kohdasta Web Developer Tools.

tullen. Drive-tiedostonkäsittelijäskriptin ja Firefox-selaimen käynnistämistä varten loin molemmille shell-skriptit (runDrivePy.sh, runBrowser.sh).

Alla tree-komennolla tulostettu hakemistopuu tilanteesta, jossa Drive-käsittelyskripti on jo ladannut Google Drivestä materiaalia Raspberry Pi 4:n muistikortille selaimen näytettäväksi. Tiedostot voivat sijaita tiedostojärjestelmässä missä vain, kunhan Drive-tiedostonlataajaskriptin muuttuja *dir* on määritetty osoittamaan skriptitiedostojen juurihakemistoon ja hakemistoon on kirjoitusoikeus. Omassa toteutuksessani skriptien ja muun sisällön juurihakemisto oli */home/pi/infotv/server*.

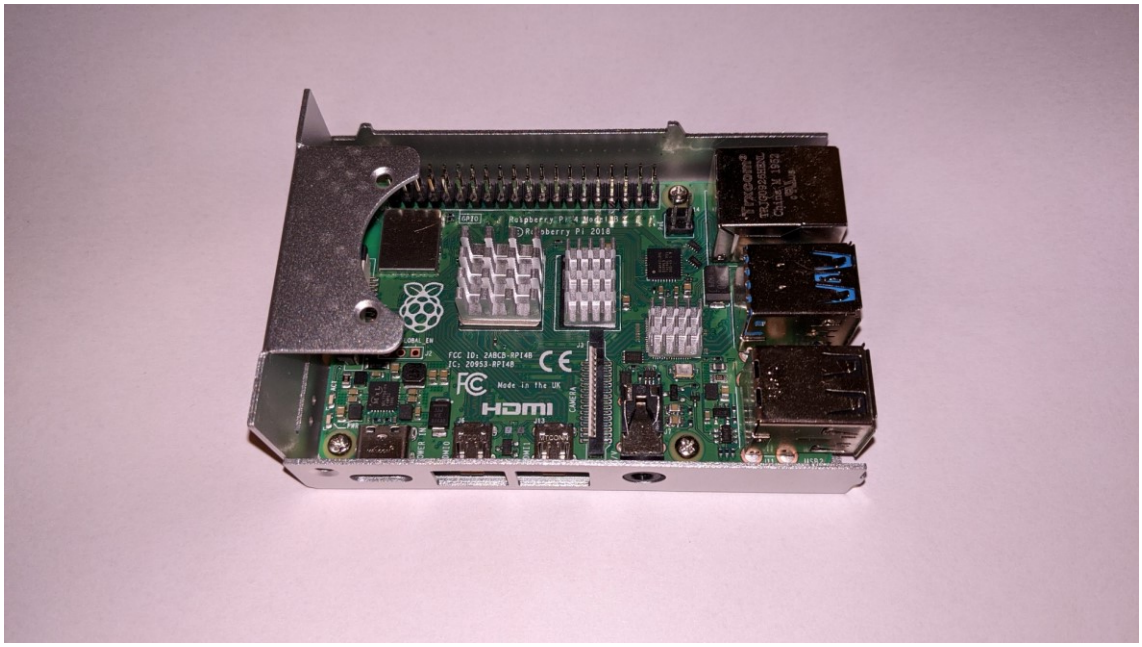
```
|-- drivelogged.py
|-- log.log
|-- nodeServer.js
|-- package.json
|-- package-lock.json
|-- public
|   |-- buttonScript.js
|   |-- config.json
|   |-- filelist.txt
|   |-- index.html
|   `-- media
|       |-- 138733.jpg
|       |-- ö.jpg
|       |-- converted
|       |   `-- Hammaslahti-mainos.pdf.png
|       |-- e-kirjoja ja e-äänikirjoja ilmaiseksi 365 päivää vuodessa.png
|       `-- Hammaslahti-mainos.pdf
|-- runBrowser.sh
|-- runDrivePy.sh
`-- token.pickle
```

Hakemistopuun rakenteesta on poistettu node-moduulien hakemisto *node-modules* ja web-sivua varten paikallisesti asennetut apukirjastohakemistot (jquery- ja bootstrap-hakemiston sisältö) selkeyden vuoksi.

7.2 Raspberry Pi 4:n käyttöönotto

Raspberry Pi 4:n käyttöönotto oli helppoa. OKdon aloituspaketin mukana sain korttitietokoneen lisäksi kaikki tarvittavat kaapelit ja muistikortin, johon oli esiasennettu käyttöjärjestelmän asennustiedostot. Paketissa oli mukana myös jäähdytyssiilit lämpöä tuottaville komponenteille ja pieni kotelotuuletin.

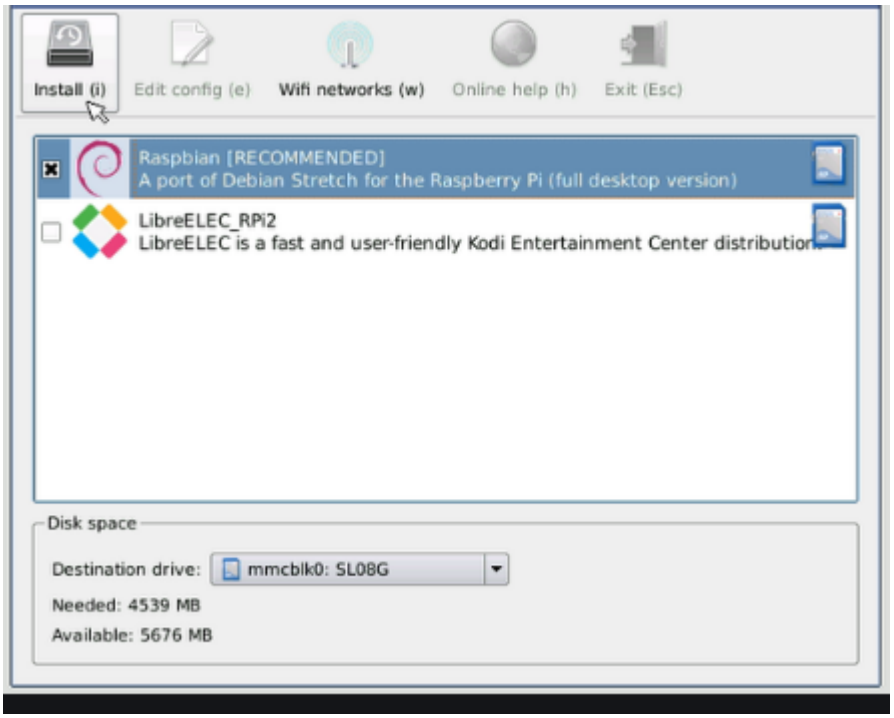
OKdo-aloituspaketissa neuvottiin tekemään laitteen asennus OKdo-sivustolla olevan ohjeen mukaisesti[2]. Noudatin ohjetta ja kiinnitin korttitietokoneeseen jäähdytyssiilit, minkä jälkeen asetin kokonaisuuden paketin mukana tulleeseen koteloon (kuva 3). Kotelotuulettimen jätin kokonaisuudesta pois, sillä en halunnut toteutuksen pitävän ääntä. En myöskään halunnut jättää mahdollisesti vikaantuvaa tuuletinta aiheuttamaan tulevaisuudessa päänvaivaa. Jäähdytyssiilit ja lämpöä siirtävä suojakotelo tarjoaisivat mielestäni riittävän jäähdytyksen.



Kuva 3. Raspberry Pi 4 B jäähdytyssiilit kiinnitettynä ja asennettuna OKdon suojakoteloon.

Käyttöjärjestelmän asennus oli vaivatonta. Asetin Raspberry Pi:n muistikorttipaikkaan konepaketin mukana tulleen MicroSD-muistikortin, jossa oli esiasennettuna käyttöjärjestelmän asennusohjelma NOOBS (kuva 4). Liitin laitteeseen virtalähteen konepaketin mukana tulevalla USB-liitännällä. Tämän jälkeenliitin siihen näytön käyttäen konepaketissa mukana ollutta HDMI-johtoa. Kun oheislaitteet oli liitetty, käynnistin Raspberry Pi:n. Käyttöjärjestelmän asennus alkoi saman tien.

[2] https://www.okdo.com/getstarted/



Kuva 4. Esimerkkikuva NOOBS-asennussovelluksen näkymästä, jossa valitaan asennettava käyttöjärjestelmä (Imgur 2021).

Näytön tunnistumisen tai sen resoluution kanssa ei ollut ongelmia. Näyttö tunnistui heti laitteen käynnistyttyä ja resoluutio oli valmiiksi laitteen oletustarkkuudella (FullHD). Käyttämäni Logitechin langaton medianäppäimistö, jossa on integroitu kosketuslevy/hiiri toimi laitteen kanssa jo asennusvaiheessa, kun olin liittänyt näppäimistön langattoman vastaanottimen yhteen Raspberry Pi:n vapaista USB-paikoista. Asennusvaiheessa jouduin valitsemaan asennettavan käyttöjärjestelmän lisäksi asennuskielen ja näppäimistön kielen. Käytin asennussovelluksen suosittelemaa oletusvalintaa asennussijaintina. Asennuksen valmistuttua ja laitteen uudelleenkäynnistyttyä oli asennus tehty ja pääsin Raspberry Pi OS:n työpöytänäkymään. Laite oli valmis ohjelmoitavaksi.

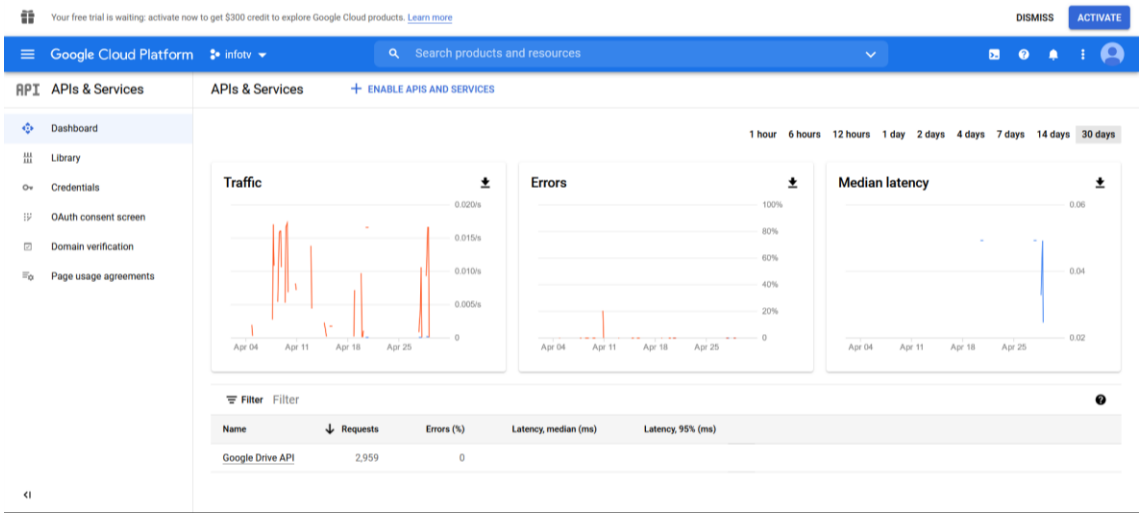
7.3 Google Drive -tiedostojen käsittelyskripti

7.3.1 Google Drive API:n integrointi Drive-tiedostojen käsittelyskriptiin

Toteutin Google Drive -tiedostojen käsittelyskriptin Python-kielellä. Tämän Python-sovelluksen tarkoituksena oli synkronoida sovellukseen liitetyn Google-tilin Drive-palveluun luodun tiedostokansion sisältö Raspberry Pi 4:n muistikortilla sijaitsevan tiedostokansion sisällön kanssa. Google tarjosi sivustollaan valmiin

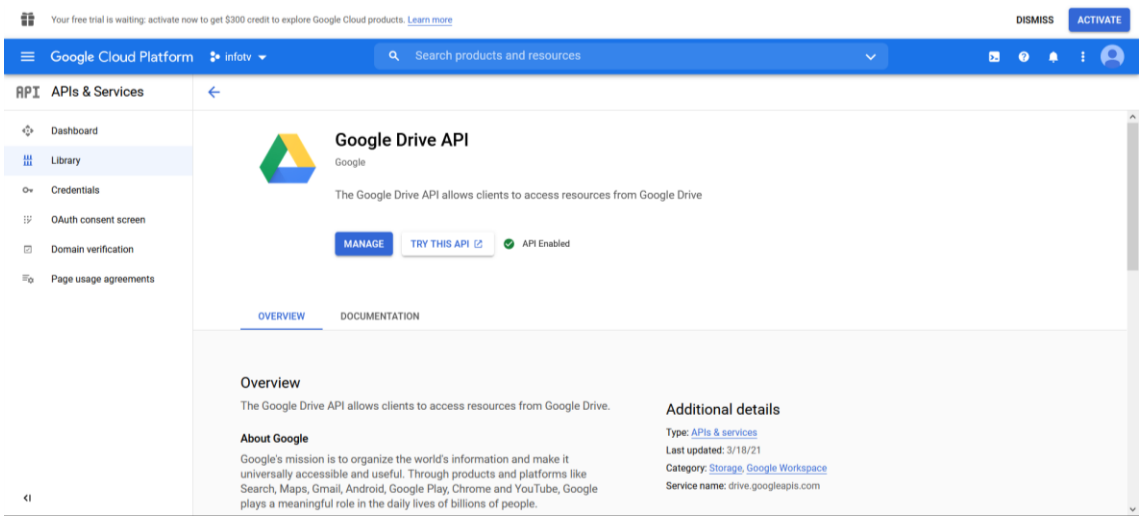
esimerkkikoodin oman Drive-sovellusprojektin aloittamisen helpottamiseksi ja selkeät ohjeet Google-tilien kytkemiseen omaan sovellukseen. Googlen esimerkkisovellus Python Quickstart listasi käyttäjän Drive-palveluun luodut tiedostot ja kansiot niiden nimellä ja tulosti ne konsoliin. Tätä esimerkkiä käytin oman toteutukseni pohjana. (Google Developers 2021a.)

Googlen Drive API:n kanssa keskustelevan Python-toteutuksen laatimisen aloitin noudattamalla Googlen ohjeita ja toteuttamalla yksinkertaisen Google Driven sisällön listaajan Googlen oman esimerkin mukaan (Google Developers 2021a). Ensimmäisenä askeleena käsittelyskriptin toteutusprosessissa loin Googlen Cloud Platform -sivustolla uuden projektin. Google-tili, jolla Cloud Platform -alustalle kirjaudutaan, on kehittäjätili, jolla voi hallinnoida tulevaa sovellusta ja tarkkailla esimerkiksi sen dataliikennettä (kuva 5).



Kuva 5. Infonäyttöprojektini pääsivu Google Cloud Platform -projektinhallintanäkymässä (Lari Strand).

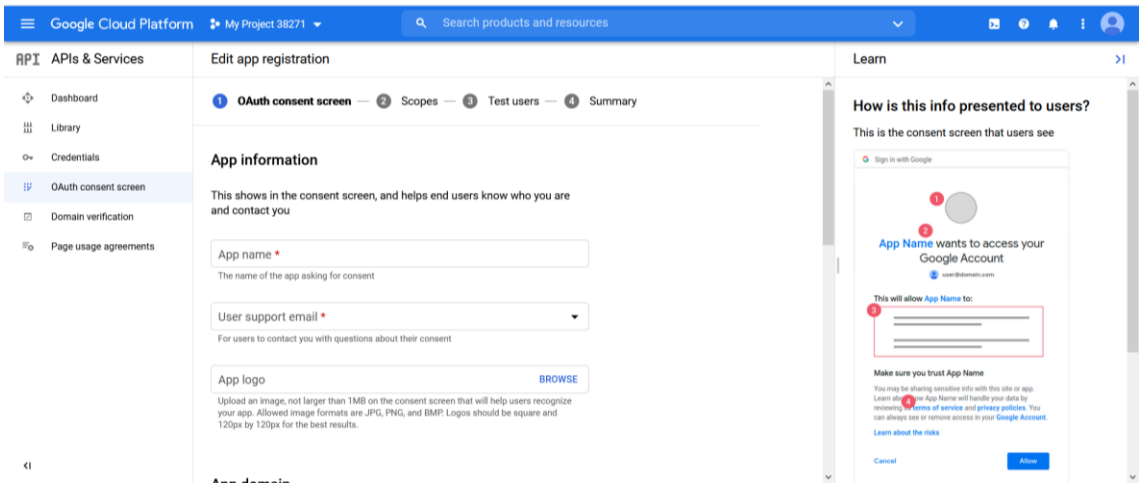
Koska sovellukseni käyttää Google Drive API:a, siihen piti liittää Googlen APIs and Services -sivustolla Drive API:n käyttöoikeus (kuva 6).



Kuva 6. Google Drive API:n Google API -konsolinäkymä, jossa projektiin voi liittää Drive API:n käyttöoikeuden (Lari Strand).

Google tarjoaa myös muunlaisille sovelluksille kattavasti ohjelmointirajapintoja (API) ja palveluja. Esimerkkinä voisin mainita ”Gmail API”:n Googlen sähköpostia käyttäville sovelluksille ja kehitystyökalun ”Maps SDK for Android” Android-sovelluksille, jotka käyttävät Googlen Maps-palvelua. (Google Cloud Platform 2021.)

Seuraavaksi sovellukselleni piti luoda valtuustiedot (credentials). Sovellukselle piti lisätä OAuth 2.0 -tunnistustapa, jota sovellukseni tulisi käyttämään sen ensimmäisen ajokerran yhteydessä (kuva 7). Noudatin tässä vaiheessa Googlen ohjeita, jotka löytyivät Google Developers -ohjeympäristöstä (Google Developers 2021b).



Kuva 7. Google-sovellusten hallintasivuston OAuth-määritysnäkymä (Lari Strand).

Valtuustiedoilla sovellus saa jatkossa oikeuden käyttää sovellukseen liitettyä Google-tiliä, kunhan Google Drive -sovellustilin tilinhaltija on myöntänyt valtuudet sovelluksen käyttöön selaimessa. Sovelluksen Drive-moduuleja hyödynnetään kommunikointiin Googlen valtuustietopalvelimen kanssa oikeuksien tarkistamiseksi ja paikallisen token-tiedoston luomiseksi. (Google Developers 2021b).

Kun suoritin Drive-sovellukseni ensimmäisen kerran, sovellus pyysi konsoli-ikkunassa seuraamaan web-linkkiä (Googlen esimerkkitoteutuksessa oli ohjelmoitu valmiiksi tämä todennusvaihe) ja kirjautumaan selaimessa Google-tilille, jonka Drive-palvelua sovellus käyttää. Kirjauduttuani sisään sovellusta käyttävälle Google-tilille sivusto pyysi sovellukselle lupaa käyttää Drive-kansioiden luku- ja kirjoitusoikeuksia (nämä tarvittavat oikeudet olin sisällyttänyt sovellukseni koodiin). Ne myönnettyäni sovellus loi token.PICKLE -nimisen tiedoston sovellustiedostoni rinnalle. Tämän tiedoston avulla myönnetyt valtuustiedot pysyvät tallessa ja ne voi aina lukea token.PICKLE-tiedostosta sovelluksen suorituksen yhteydessä uudelleen.

Drive API -sovelluksen tarvitsemat oikeudet määritetään sovelluksen koodissa SCOPES-muuttujilla. Jos sovelluksen oikeudet halutaan rajata esimerkiksi tiedostojen metatiedon lukuoikeuteen, voidaan määrittää Scopeksi:
SCOPES = ['https://www.googleapis.com/auth/drive.metadata.readonly']

Täydet oikeudet sovellus saa määrityksellä:
SCOPES = ['https://www.googleapis.com/auth/drive']

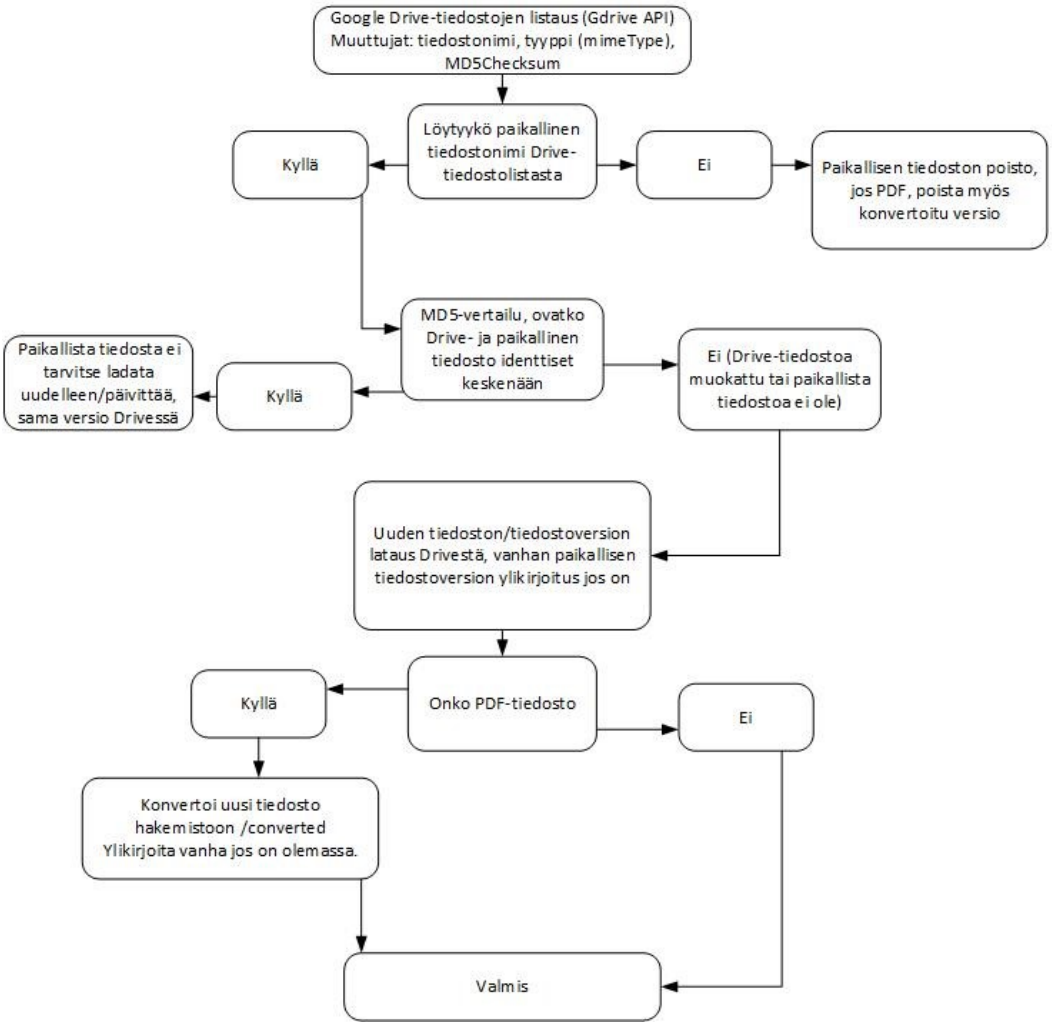
Mikäli tarvittavia oikeuksia muuttaa SCOPES-muuttujalla tai token.PICKLE-tiedoston poistaa, on oikeudet käytävä myöntämässä tilille uudelleen selainnäkymässä.

Vaikka toteutukseni ei tarvitse täysiä oikeuksia, sillä sovellus lukee ainoastaan Drive-tiedostojen nimet ja metadataa ja lataa ne, päätin silti määrittää sovellukselle kaikki oikeudet. Periaatteessa ohjelmassa voisi näillä oikeuksilla esimerkiksi myös poistaa Drive-tilin tiedostoja ja luoda uusia tiedostoja Drive-kansioon.

7.3.2 Google Drive -tiedostojen käsittelyskriptin laatiminen ja rakenne

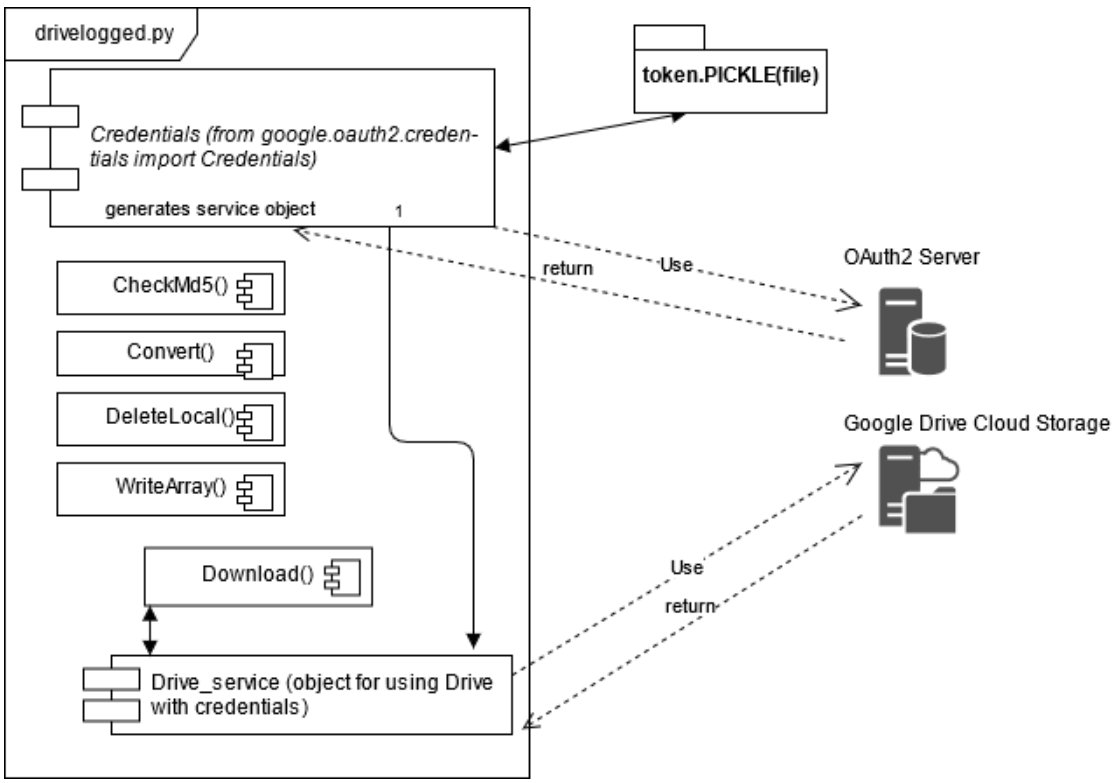
Toteuttamani Googlen Drive API:n kanssa keskustelevan Python-skriptin laadin suurimmaksi osaksi Windows 10:ssä Visual Studio Codella. Huomasin sovelluksen luonnin aikana, että joudun asentamaan sovellukseni käyttöön tarvittavat Python-moduulit eri tavalla Raspberry Pi OS -käyttöjärjestelmässä kuin Windowsiin asentamani moduulit. Keskityn opinnäytetyössäni kuitenkin vain Raspberry Pi OS -käyttöjärjestelmäympäristöön ja sovellukseni toimintaan tarvittuihin asennuksiin ja asetuksiin Raspberry Pi OS -ympäristössä.

Drive API -sovellukseni suorituksenaikainen toimintalogiikka noudattaa pitkälti if–else-rakennetta (kuvio 1). Drive API:lla haetaan jokaiselle Google Drivessä olevalle tiedostolle tiedostonimi, tiedostotyyppi ja MD5-tarkistussumma, joka on jokaiselle tiedostolle uniikki ja yksilöivä tieto. Ne tallennetaan muuttujien arvoiksi ja jokainen tiedosto lisätään muuttujineen taulukkoon/listaan.



Kuvio 1. Drive-tiedostojen synkronointiskriptin toimintalogiikka, vuokaavio (Lari Strand).

Google Drive -tiedostojen synkronointiskripti *drivelogged.py* rakentuu pääohjelman lisäksi seuraavista funktioista: *CheckMd5(), Download(), Convert(), DeleteLocal()* ja *WriteArray()*. Olen kuvannut skriptin rakenteen ja sen toimintalogiikkaa kuviossa 2.



Kuvio 2. Google Drive -tiedostojen synkronointiskriptin rakenne ja toimintalogiikka (Lari Strand).

Kun Drive-tiedostojen synkronointiskripti ajetaan, siirrytään (Drive Apin tarkastettua valtuustiedot service) pääohjelmasta funktioon *CheckMd5()*. Tämä funktio tarkastaa, ovatko Drivestä löytyvät tiedostot muuttuneet paikallisiin tiedostoihin verrattuna. Mikäli tarkistussummat ovat yhtenäiset, ei Drive-tiedostoon ole tullut muutoksia paikalliseen tiedostoon verrattuna eikä uudelleenlatausta tarvita. Jos tarkistussummat eivät ole identtiset, ladataan uusi tiedosto ja se tallennetaan ylikirjoittamalla vanha versio, jos sellainen on olemassa. *CheckMd5()* siis ilmoittaa latausfunktiolle myös siitä, että ellei paikallista tiedostoa ole vielä olemassa, se tulee ladata Drivestä. *CheckMd5()*-funktio palauttaa pääohjelmalle totuusarvon (boolean), jonka perusteella päätetään lataustarpeesta pääohjelmassa.

Drive-tiedostojen synkronointiskriptin *Convert()*-funktio konvertoi sitä kutsuessa PDF-tiedoston PNG-tiedostoksi hakemistoon */converted*. Se vaatii muuttujaksi

Drivestä ladatun PDF-tiedoston nimen. Funktio konvertoi ainoastaan PDF-tiedoston ensimmäisen sivun tai yksisivuisen PDF-tiedoston. Koska kirjastossa esitettäviä tapahtumamainoksia ja muita ilmoituksia ei henkilökunnan mukaan lähes koskaan toteuteta monisivuisina, on tämä konvertointikäytäntö toteutukseen sopiva.

Drive-tiedostojen synkronointiskriptin *Download()*-funktio suorittaa tiedoston lataamisen Drive-palvelusta paikalliselle järjestelmälle. Se tarvitsee muuttujakseen pääohjelmassa luodun *Drive_service* -instanssin, joka on tarkistanut sovelluksen oikeuden käyttää sovellukselle määrättyä Google-tiliä ja että tälle sovellukselle myönnetyt valtuudet ovat kunnossa. Funktio tarvitsee lisäksi muuttujaksi ladattavan tiedoston tunnuksen (id). Tunnus on identtinen Google Driven luoman jaettavan tiedostonlatauslinkin loppuosan kanssa.

Jokaisen tiedoston latauksen tai päivittämisen yhteydessä uudet tai muuttuneet tiedostot lisätään globaaliin muuttujaan *locFileArray*, joka pitää listaa selaimen kuvakarusellissa näytettävistä kuvatiedostoista. Kun kaikki tiedostot on käsitelty onnistuneesti, tämä lista kirjoitetaan aakkostettuna HTML-sivun kuvakarusellia varten tekstitiedostoon funktiolla *writeArray()*. Nyt päivittynyt kuvasisältö on selaimen saatavilla, kunhan HTML-sivun JavaScript-funktio lukee ja parsii tiedostonimien listan sisältämän tekstitiedoston taulukkoon sivun päivityksen yhteydessä. Aakkostus mahdollistaa kuvamateriaalin esitysjärjestyksenmäärittämisen nimeämällä Driveen lisättävät tiedostot aakkosittain haluttuun järjestykseen. Järjestystä voi muokata uudelleennimeämällä tiedostot myös Drivessä.

Jokainen paikallinen tiedosto käsitellään vielä viimeiseksi Drive-tiedostojen synkronointiskriptin funktiossa *deleteLocal()*. Tämä funktio tarkastaa, löytyykö paikallisia tiedostoja Driven tiedostolistalta ja poistaa tarpeettomat tiedostot. Se siis vertaa paikallisen hakemiston sisältämiä tiedostoja Drive-tiedostoihin. Jos paikallista tiedostonimeä ei löydy Drive-tiedostolistasta, se poistetaan ja mikäli tiedoston pääte on PDF, poistetaan myös sen konvertoitu versio. Konvertoidut versiot PDF-tiedostoista päätyvät hakemiston */converted* alle. Konvertoitu tiedosto nimetään konvertointivaiheessa saman nimiseksi kuin alkuperäinen, mutta siihen liitetään vielä .png-liite. Jos poistettava tiedosto olisi esimerkiksi *näyttely.pdf*, poistettaisiin tiedoston *näyttely.pdf* alihakemistosta */converted* lisäksi myös tiedosto *näyttely.pdf.png*. Ylimääräisten tiedostojen poisto tapahtuu

skriptissä viimeisenä siitä syystä, että jos skriptin suorittaminen jostain syystä keskeytyy, edellisen kerran onnistunut ajokerta ja sen synkronoima materiaali ovat edelleen tallessa ja kuvakarusellin käytettävissä.

7.3.3 Python-moduulit

Drive-tiedostolataaja käyttää toimiakseen seuraavia Python-moduuleja, jotka asensin pip-paketinhallintasovelluksella:

- *google-api-python-client*: Google Drive API:n käyttöön vaadittu moduuli.
- *google-auth-httplib2*: Google Drive API:n käyttöön vaadittu moduuli.
- *google-auth-oauthlib*: Google Drive API:n käyttöön vaadittu moduuli.
- *pdf2image*: moduuli PDF-tiedostojen konvertointiin.
- *pickle*: pdf2image-moduulin ja tiedostolataajan tarvitsema lisäkirjasto.
- *logger*: Sovelluksen lokit luodaan tämän moduulin avulla.

Lisäksi toteutus käyttää tiedostojen käsittelyyn Pythonin io-, shutil- ja os-moduuleja.

7.3.4 Lokitus

Drive-tiedostojen synkronointiskriptin lokit muodostuvat skriptitiedoston juurihakemistoon tiedostoon log.log. Lokit muodostetaan logger-moduulin avulla. Drive-skriptiin on määritetty virhetilanteita varten try-catch -lohkoja ja niiden sisään catch-lohkoon määritettyjä, virhetilanteita kuvaavia ERROR-tason lokihuomautuksia. Muu lokitulostus tapahtuu INFO-tasolla tai DEBUG-tasolla.

Logger-moduuli käyttää seuraavia lokitasoja:

CRITICAL
ERROR
WARNING
INFO
DEBUG
NOTSET

Jos lokituksen tasoksi määrittää esimerkiksi WARNING, ainoastaan WARNING- ja sitä ylemmät lokitasot eli esimerkin tapauksessa myös ERROR- ja CRITICAL-tasoiset lokimerkinnät käsitellään (Python.org 2021b). Lokituksen tasoksi asetin ERROR, joten jos Drive-skripti ei kohtaa vakavia ongelmia, ei lokiin muodostu muita merkintöjä. Tällä tavoin estin lokitiedoston koon kasvun liian nopeasti suureksi, mutta pääsisin silti käsiksi mahdollisissa virhetilanteissa syntyvään hyödylliseen dataan niiden syyn selvittämiseksi. Jos kaikki lokiin menevä informaatio tulostuisi lokitiedostoon, tiedoston koko kasvaisi tarpeettomasti suureksi. Lokittamalla vain virhetilanteet ei lokitiedosto ollut kasvanut lainkaan viikon testiajon aikana. Päätin jättää Drive-tiedostojen synkronointitiedoston lokituksen tähän malliin, sillä levytilaa muistikortilla oli runsaasti ja sen täyttyminen lokidatasta veisi luultavasti vuosia. Toinen ratkaisu lokitiedoston koon hallitsemiseksi olisi ollut lokirotaatio, jossa lokitiedostoja kierrätetään ja uudelleenkäytetään. Kasvaessaan määritetyn koon yli, luodaan uusi lokitiedosto vanhimman päälle ja edellinen pakataan levytilan säästämiseksi ja vanhimmat poistetaan. Hyvin vähäisen lokidatan karttumisen vuoksi päätin, että lokirotaatio ei ole tarpeen Drive-tiedostojen synkronointiskriptin tuottaman lokin kohdalla. Jätin myös käyttöjärjestelmän ja ajastettujen cron-tehtävien muodostaman lokisisällön käsittelyn käyttöjärjestelmän oletuksiin.

7.4 HTML- ja Javascript-koodilla toteutettu kuvakaruselli ja Firefox

7.4.1 HTML-sivu

HTML-sivu, joka esittää infonäytön kuvakarusellia on hyvin yksinkertainen. Se sisältää responsiivisen Bootstrap-kuvaelementin, joka skaalautuu leveys- ja korkeussuunnassa selainikkunaan täyttäen selainnäkymän. Kuvatiedostot skaalautuvat responsiivisen kuvaelementin sisään säilyttäen kuvasuhteensa, jos kuvat ovat isompia kuin itse selainnäkymä. Responsiivista kuvaelementtiä isompien kuvatiedostojen kuvasuhde ei muutu alkuperäisestä, ja ne täyttävät selainnäkymän ensin joko pysty- tai vaakasuunnassa koon mukaan. Mikäli kuvat ovat pienempiä kuin selainnäkymä, ne keskitetään kuvaelementtiin keskelle selainnäkymää.

HTML-sivun JavaScript-osiossa luetaan ensiksi tekstitiedosto, joka sisältää esitettävien kuvatiedostojen nimet ja sijainnit. Sen jälkeen responsiiviseen kuvaelementtiin syötetään silmukassa JavaScript-koodilla uusi kuvatiedostosijainti seitsemän sekunnin välein käyttäen viivettä (timeout). Se, että kuva näkyy juuri seitsemän sekunnin ajan, sovittiin yhdessä kirjastohenkilökunnan kanssa edellisen infonäyttötoteutuksen käytännön pohjalta. Kun skripti pääsee tekstitiedostosta muodostetun listan loppuun, toisto aloitetaan alusta. Tähän JavaScript-silmukkaan on siis kovakoodattu kesto, jonka ajan jokainen kuva näkyy.

7.4.2 HTML-sivun päivitysskripti

HTML-sivusto päivittyy aina, jos paikallisen palvelimen kuvasisältö muuttuu. Tarkkailtavia tiedostoja ovat kaikki infonäyttötoteutuksen tukemat tiedostomuodot. Sisältö päivittyy myös, mikäli tarkkailtavasta hakemistosta poistuu materiaalia. Tämä toiminto toteutuu yhdessä Node.js:llä toteutetun paikallisen palvelimen *Livereload*-moduulin (Npm 2021) ja HTML-tiedostoon liitetyn JavaScript-elementin avulla.

7.4.3 Firefox-selaimen kustomointi

Jotta sain Firefox-selaimen kaikki käyttöliittymäkomponentit piiloon, piti Firefox käynnistää käynnistysskriptissä Kiosk-moodissa. Se onnistui käyttämällä skriptissä vipua ”—kiosk”. Tämä ei silti riittänyt poistamaan Firefoxin alalaidassa näkyvää tilapalkkia, joka näyttää selaimessa käynnissä olevia prosesseja. Omassa toteutuksessani tilapalkki ilmestyi aina silloin, kun seuraavaa kuvaa ladattiin selaimen näkymään, joten halusin piilottaa sen. Löysin vastauksen tilapalkin piilottamiseen Firefoxin tukisivustolta ja noudatin siellä annettuja ohjeita. (Mozilla 2020).

7.5 Paikallinen palvelin ja sen sisällön päivittäminen

7.5.1 Palvelin: JavaScript- ja Node.js-moduulit

Paikallinen tiedostopalvelin ja palvelimessa tapahtuvien muutosten päivitys selainnäkymään on toteutettu JavaScriptillä Node.js-ajoympäristössä. Itse web-sivun sisällön näyttämiseen tarvittava materiaali jaetaan *node-static*-moduulin avulla (Npm 2018). Moduuli helpottaa ja yksinkertaistaa tiedostohakemistojen sisällön jakamista Node.js-palvelimella. Asensin moduulin npm:llä ja otin sen käyttöön määrittämällä: *var nStatic = require('node-static');*

Yksinkertaisuudessaan palvelimen pystytys tapahtuu määrittämällä tiedostoserverimuuttuja ja jaettava kansio komennolla *var fileServer = new nStatic.Server('./public');*

Tämän jälkeen määritin kuuntelijan kuuntelemaan liikennettä porttiin 3000.

```
http.createServer(function (req, res) {

   fileServer.serve(req, res);

}).listen(3000);
```

Komennossa on hyödynnetty Node.js:n sisäänrakennettua moduulia http, jolla voidaan siirtää dataa Hyper Text Transfer Protocol eli HTTP:ta käyttäen.

7.5.2 Cronjobit

Infonäyttötoteutuksen toiminta perustuu ajoitettuihin tehtäviin, cronjobeihin, joita Raspberry Pi OS -käyttöjärjestelmä ajaa määritetysti käyttäen ohjelmien ajastettuun ajamiseen tarkoitettua Cron-ajastuspalvelua. Käyttäjien ajastetut toiminnot sijoitetaan */var/spool/cron/crontabs* -hakemistoon. Niitä voi muokata käyttämällä Crontab-ohjelmaa, joka sisältää järjestelmän ajastetut palvelut. (Kuutti 2011, 135.)

Ajastin Crontabilla Google Drive -tiedostolataajaskriptin suoriutuvaksi ensimmäisen kerran, kun laite ja käyttöjärjestelmä käynnistyy eli boottaa. Viivästin suoritusta 5 sekuntia bootin jälkeen, jotta langaton verkkoyhteys ehtisi varmasti muodostua Google Drive -tiedostojen tarkastamista varten. Näin laitteessa olisi aina käynnistyksen jälkeen viimeisin ja tuore Google Drivesta löytyvä mediasisältö.

Toinen järjestelmän käynnistymisenaikainen skripti, jonka ajastin, on Node.js:llä toteuttamani palvelimen käynnistysskripti. Skripti alkaa kuuntelemaan localhost-osoitetta portissa 3000 ja paljastaa infonäyttösivustolle kaiken tarpeellisen materiaalin eli kuvakarusellin sisältävän HTML:n tiedoston lisäksi kuvat ja tiedostolistan näytettävistä kuvista. Skripti jatkaa toimintaansa prosessina, joka myös tarkkailee tiedostoissa tapahtuvia muutoksia ja raportoi niistä HTML-sivulle sen päivitystä varten. Skriptin ajastin käynnistymään heti bootin jälkeen.

Kolmas cronjob-tehtävä on ajaa komentokehoteskripti, joka käynnistää Firefox-selaimen Kiosk-moodissa eli koko näyttöalueen kattavassa työkalupalkittomassa selainnäkymässä. Määritin selaimen oletusosoitteeksi localhost ja portiksi 3000, jota paikallinen palvelin kuuntelee. Firefoxin auetessa Kiosk-näkymässä, alkaa kuvakaruselli pyöriä saman tien. Tämä cronjob on viivästetty niin, että Google Drive -lataaja ehtii saada ensimmäisen bootin jälkeisen ajonsa valmiiksi ja paikallinen palvelin on jo käynnistetty. Ajastin tämän Firefox-selaimen käynnistystehtävän suoriutuvaksi kymmenen sekuntia järjestelmän onnistuneen boottauksen jälkeen.

Neljäs cronjob-tehtävä määrittää Google Drive -tiedostolataajaskriptin ajautuvaksi viiden minuutin välein aamuseitsemän ja iltaseitsemän välillä, jolloin kirjastossa on useimmiten työntekijä paikalla. Muutoksien ja esimerkiksi uuden lisätyn materiaalin tarkastelu infonäytössä on mahdollista tällöin vain hetken päästä muutosten teon jälkeen.

7.5.3 Palvelimen sisällön reaaliaikainen päivittäminen web-sivulle

Toteutuksessani staattisen serverin jakamissa tiedostoissa tapahtuvat muutokset päivittävät selainnäkymän, jotta uusi kuvamateriaali tulisi saataville heti si-

sällön muututtua. Tämän toiminnallisuuden toteutin käyttämällä Livereload-moduulia. Sen tehtävä on seurata muutoksia asetuksissa määritettyihin tiedostoihin, ja jos muutoksia tapahtuu, raportoida siitä selaimelle ja päivittää selainnäkymä. Livereload-moduuli vaatii toimiakseen muutoksia myös HTML-sivun koodiin, jotta Node.js-moduuli ja siihen liitetty paikallisen palvelimen web-sivu keskustelisivat toistensa kanssa (Npm 2021).

Livereload on tarkoitettu ensisijaisesti web-kehittäjien työkaluksi ja poistamaan jatkuvan manuaalisen selainnäkymän päivitystarpeen aina sivuston koodiin tai rakenteeseen muutoksia tehdessä. Node.js-toteutus Livereloadista eli Livereload-moduuli oli juuri sopiva tarkoitukseeni. Livereload-moduulia käyttämällä selainnäkymä päivittyy vain tarvittaessa. Ilman sitä joutuisin ajastamaan selainnäkymän päivityksen tapahtumaan aina Drive-tiedostolataajan suorituksen jälkeen, vaikkei kuvasisältö välttämättä olisi muuttunutkaan. Tällainen tarpeeton sivustopäivitys aiheuttaisi turhia katkoksia kuvien rotaatioon, ja pahimmillaan ajastus saattaisi tapahtua niin, että jotakin kuvaa olisi ehditty esittää vain silmänräpäyksen ajan, kun sivustonpäivitys keskeyttäisi sen ja kuvakarusellin toisto alkaisi alusta.

Livereload siis poisti tarpeen sivuston turhille päivitystapahtumille. Livereload päivittää kuvamateriaalin selaimeen, jos Raspberry Pi 4:n paikallisissa tiedostoissa havaitaan muutoksia. Mahdolliset muutokset syntyvät siis silloin, kun Drive-skripti suorittuu crontabin ajastuksessa määritetysti. Kuvien esittäminen saattaa edelleen ”katketa” työntekijän lisätessä tai poistaessa tiedostoja Google Drivessä. Sallin tämän hyvin satunnaisen kauneusvirheen toteutuksessani. Livereload mahdollistaa siis myös melkein reaaliaikaisen sisällön päivityksen, joten Drive-skriptiä voisi käytännössä ajaa melkein reaaliajassa, esimerkiksi minuutin välein, eli huomattavasti tiuhemmin kuin alun perin teoriatasolla alustavasti miettimälläni tunnin välein -aikataululla. Livereload-moduulin päivitystapahtuma on viivästetty, jotta tiedostojen synkronointi ehtii valmistua ennen kuin selainnäkymä päivitetään. Ensimmäisen ja viimeisen tiedostomuutoksen välillä saattaa kulua pidemmänkin aikaa, jos uutta ladattavaa materiaalia on runsaasti. Tällä estetään tilanne, jossa selain yrittää näyttää edelleen vanhaa kuvamateriaalia, vaikka kuvamateriaaliin on tullut muutoksia.

7.6 Mahdolliset virhetilanteet

Harkitsin toteuttavani laitteen vikatilanteita varten jonkinlaisen etähallintamahdollisuuden, mutta sovellustoteutuksen yksinkertaisen luonteen takia päätin unohtaa etähallinnan. Laitteen toiminnan estäviksi ongelmaksi voivat muodostua ainoastaan sähköverkon aiheuttamat sähköviat ja niiden seurauksena mahdolliset massiiviset tiedostojärjestelmävirheet, laitteen komponenteissa esiintyvät viat tai levytilan loppuminen. Levytilan loppuminen voisi käytännössä tapahtua ainoastaan, jos sovellukseni muodostamat lokitiedostot täyttäisivät muistikortin. Laitteelle ladattavat kuvatiedostot eivät hyvin todennäköisesti tulisi täyttämään muistikorttia tiedostojen keskimääräisen koon ollessa korkeintaan muutamia megatavuja.

Päivitetyn sisällön hankkiminen voi estyä tilapäisen verkko-ongelman takia. Tällöin sovellus lisää lokiin huomautuksen verkko-ongelmasta ladatessa/varmistaessa Google API:n käyttämää credentials-tiedostoa (jolloin sovelluksen suorittaminen lopetetaan) tai jos itse tiedoston lataaminen Googlen Drive-palvelusta epäonnistuu. Mikäli yksittäisen tiedoston tarkistus tai lataus epäonnistuu Drive-tiedostolataajaskriptissä, sitä ei lisätä selaimessa näytettävään listaan. Kaikkien tiedostojen synkronoinnin pitää onnistua ja skriptin ajautua onnistuneesti loppuun, jotta muutokset näkyvät selaimessa. Muutoin käytetään edellisen onnistuneen Drive-skriptin ajon päivittämää kuvasisältöä. Mikäli credentials-tiedostoa ei voida vahvistaa, tiedostolataajan toiminta keskeytetään ja infonäyttötoteutus jää käyttämään aikaisempaa materiaalia, kunnes seuraava ajoitettu tiedostonlatausajo onnistuu.

Jos siis tiedostojen lataamisessa Drive-palvelimelta muistiin tai muistissa olevan tiedoston kirjoittamisessa Raspberry Pi:n muistikortille tapahtuu virhe, tiedostoa ei lisätä selaimessa näytettävään tiedostolistaukseen ja skriptin suorittaminen keskeytyy. Tällöin käytetään sisältöä, jonka skriptin edellinen onnistunut ajokerta on luonut.

7.7 Infonäyttötoteutuksen käyttöönotto toimipisteessä

Kun infonäyttötoteutuksen koodi ja ajastetut cron-palvelut oli asennettu Raspberry Pi 4:lle ja olin testannut toteutuksen toiminnan oman näyttöni kanssa, oli aika kokeilla toteutusta toimeksiantajan kanssa. Laite asennettiin kirjaston infonäyttönä toimivan television yhteyteen (kuva 8). Television jalustasta löytyi jatkojohto, johon sain liitettyä Raspberry Pi 4:n virtalähteen. Korttitietokoneen koteloineen sain piilotettua television taakse jalustasta löytyneeseen räkkityyppiseen telineeseen. Myös HDMI-johdon sai lopullisessa asennuksessa piilotettua niin, ettei se näy television kyljestä ja piiloutui siististi television taakse. Kuvassa 8 laite on testauskäytössä eikä lopullisessa sijoituspaikassaan.

Kuva 8. Raspberry Pi 4 kokeilutilanteessa television jalustalla ja liitettynä televisioon (Lari Strand).

Testasimme infonäyttöä Pyhäselän kirjastossa television näytön ollessa vaakasuunnassa (kuva 9). Toteutus toimi hyvin, mutta vaakasuuntainen näyttö ei hyödyntänyt näyttöaluetta parhaalla mahdollisella tavalla.

Kuva 9. Infonäyttö testikäytössä Pyhäselän kirjastossa (Lari Strand).

Ehdotin television kääntämistä jalustassaan pystysuuntaan, jotta television kuva-alaa voisi hyödyntää tehokkaammin. Suurin osa näytettävästä materiaalista oli pystysuuntaisia, A4-paperia vastaavassa kuvasuhteessa olevia kuvatiedostoja ja television kääntäminen 90 astetta sallisi kuvien skaalautumisen isommaksi pystysuuntaisella kuva-alueella. Television kääntämisestä pystysuuntaan sovittiin ja se toteutettiin lopullisen asennuksen yhteydessä (kuva 10).

Kuva 10. Näyttöaluetta hyödynnettiin paremmin kääntämällä infonäyttö pysty-asentoon (Lari Strand).

Television suunnan kääntämisestä ei tullut ongelmia toteutuksen kannalta. Mikäli näyttölaitetta ei käytetä oletusasennossa (vaaka), Raspberry Pi OS:n näyttöasetuksista täytyy ainoastaan kiertää näyttöalue näyttölaitteen asennusta vastaavaksi. Infonäyttötoteutus skaalaa kuvasisällön automaattisesti uuden näyttöalueen sisään.

Toteutin kirjastohenkilökunnan toiveesta myös yksinkertaisen ohjeen (liite 1) infonäytöllä esitettävän sisällön muokkaamiseen. Ohjeessa kerrotaan, miten Google Driveen kirjaudutaan, miten sinne lisätään ja miten sieltä poistetaan sisältöä sekä miten lisättyjen tiedostojen nimiä muutetaan.

8 Ongelmatilanteet

Ongelmatilanteeni infonäyttötoteutusta toimivaksi kokonaisuudeksi rakentaessa koskivat lähinnä Raspberry Pi:n ja sen käyttöjärjestelmän ominaisuuksia ja Python-skriptin ajoa ajastettuna cron-tehtävänä. Pythonilla toteuttamani Driven tiedostolataaja ei pystynyt renderöimään kuvatiedostoja, jotka sisälsivät ääkkösiä. Tiedostot latautuivat silti omalla nimellään ääkkösten kera ja näkyivät selaimessa normaalisti, kun tiedoston avasi selaimessa paikallisesti. Paikallisen

palvelimen kautta näytettävä tiedosto ei ilmestynyt näkyviin ja kuvan tilalla näkyi rikkinäisen kuvalinkin ikoni, jonka Firefox näyttää oletuksena, kun kuvalinkki ei johda mihinkään tai kuvatiedosto on vahingoittunut.

Aluksi etsin ongelman syytä tiedostolistan tekstitiedostoksi laativasta skriptistä ja sen lukevasta HTML-tiedostoon sisällytetystä skriptistä. Kokeilin muuttaa tiedostosijainnin suoraan koodissa prosenttikoodatuksi, mutta se ei auttanut. Myöskään internetistä ei löytynyt kohtaamaani ongelmaa käsitteleviä ja suoraan ongelman selittäviä keskusteluketjuja tai sivustoja. Kirjoitin Node.js:llä toteutetun yksinkertaisen tiedostolataajan ja sen lataamat tiedostot renderöityivät normaalisti, joten rajasin ongelman Pythoniin ja siihen liittyviin enkoodausongelmiin. Löysin Stackoverflow-sivustolta keskusteluketjun, jossa eräs koodaaja oli kysynyt Pythonin oletus-enkoodauksesta, ympäristömuuttujista ja siitä, miten niitä muutetaan (Stackoverflow 2018). Keskusteluketju toi minulle ajatuksen, että kyseessä saattaisi olla jonkinlainen ongelma johtuen englanninkielisestä Raspberry Pi OS -käyttöjärjestelmästäni tai lokaatioasetuksista, jotka saattaisivat virheellisesti raportoida Pythonille olla käyttämättä UTF-8-koodausta oletusympäristömuuttujana. Löysin myös Googlen Cloud Storage -oppaasta mainitsinnan:

> To reduce the chance for filename encoding interoperability problems gsutil uses UTF-8 character encoding when uploading and downloading files. Because UTF-8 is in widespread (and growing) use, for most users nothing needs to be done to use UTF-8. Users with files stored in other encodings (such as Latin 1) must convert those filenames to UTF-8 before attempting to upload the files. (Google Cloud Storage 2021.)

Teksti ei koskenut suoraan ongelmaani, mutta se sai minut pohtimaan ongelmaa Pythonin io-operaatioissa, mikä puolestaan johti minut yllä mainitun ympäristömuuttujan jäljille.

Ongelma tiedostojen omituisen käyttäytymisen kanssa ratkesi käyttämällä ympäristömuuttujaa *PYTHONUTF8=1*, jonka tallensin Raspberry Pi OS:n Bash-komentotulkin ympäristömuuttujatiedostoon */etc/profile*.

Jotta tämä Bash-komentotulkin ympäristömuuttuja tulisi huomioiduksi, ajoin Drive-lataajaskriptin cron-tehtävänä niin, että Bash-komentotulkki suorittaa sen

komennolla *bash -l -c '/usr/local/bin/python3.9 /home/pi/infotv/server/drivelogged.py'* (Linux Manual 2021).

Tutkiessani asiaa lisää, huomasin, että ainakin uudempien Python 3 -versioiden oletuskoodaus olisi vanhan ASCII:n sijasta juuri UTF8 (PyBites 2021). Ongelma ei siis voinut johtua Python-versiostani. Etsin verkosta tietoa, miksi cron-tehtäväni ei ymmärtänyt käyttää ympäristömuuttujaa ja löysin siihen ratkaisun. Ongelma siis oli cron-tehtävässäni. Cron-tehtävät ajetaan minimaalisessa ajoympäristössä, ja kaikki ympäristömuuttujat pitää tuoda manuaalisesti komentoon mukaan. Tämä paljastui StackExchange- keskustelupalstalta, jossa käsiteltiin samaa ongelmaa eri tapauksessa (StackExchange 2021).

Kutsuessani bash-komentotulkkia cron-tehtävässäni, bashiin asetettu ympäristömuuttuja huomioitiin. Toinen ratkaisu olisi ollut tuoda ympäristön oletusprofiilin sisältämä ympäristömuuttuja mukaan cron-tehtävääni keskustelupalstan esimerkin mukaisesti, jonka jälkeen cron ajaisi ympäristömuuttujan tarvitseman skriptin:

0 5 * * * . $HOME/.profile; /path/to/command/to/run

Myös Firefoxin avaamisessa oli ongelmia. Kun graafisen käyttöliittymän sisältävä ohjelma ajetaan cronina, on sille ensiksi määritettävä ikkuna/näyttö, jota sovellus käyttää. Ilman tätä vaihetta konsoliin tulostui virhe:
Error: no DISPLAY environment variable specified

Toteutuksessani tämä ongelma oli kierrettävissä määrittämällä ajastettuna cronina käynnistämääni shell-skriptiin ikkuna, jota graafinen ohjelma, eli tässä tapauksessa Firefox-selain, käyttää. Ongelma poistui seuraavalla komennolla: *export DISPLAY=:0.0* (StackExchange 2015), minkä jälkeen Firefox käynnistyi normaalisti komennolla: */usr/bin/firefox*.

Tässä vaiheessa aloin pelätä, toimiiko toteutukseni lainkaan, jos Raspberryyn liitettävä näyttölaite on kokonaan kiinni tai jopa irti laitteesta. Pohdin, voiko DISPLAY= -muuttujaa määrittää ilman, että laitteessa on näyttölaitetta liitettynä. Tätä testatessani huomasin, että laitteen oleminen suljettuna tai Raspberrysta

irrotettuna ei vaikuttanut Firefox-selaimen avautumiseen taustalla, eikä virheilmoitusta tullut. Kun näyttölaitteen laittoi jälleen päälle tai liitti kiinni korttitietokoneeseen, näkyi näyttölaitteen ruudussa Firefoxin Kiosk-moodi. Tämä on toteutuksen kannalta oleellista, sillä näyttölaitteena toimiva televisio sammuu automaattisesti kahdeksan tunnin päällä olon jälkeen. Käyttötestauksessa television sammuminen ja uudelleenkäynnistys ei aiheuttanut infonäyttötoteutuksen suhteen ongelmatilanteita.

9 Pohdinta

Olen tyytyväinen siihen, että sain opinnäytetyöprosessin tuloksena aikaiseksi toimivan ja vaatimusmäärittelyä vastaavan infonäyttöjärjestelmän. Myös toimeksiantaja oli tyytyväinen lopputulokseen ja toteutuksen helppokäyttöisyyteen. Olen itse myös tyytyväinen kommunikointiini toimeksiantajan kanssa. Lopullinen tuote oli hyvässä yhteisymmärryksessä rakentunut tuote, eikä yllätyksiä tullut kummallekaan.

Opinnäytetyön teknisen työskentelyn aloitusaikataulu venyi suunnitellusta laitteiden tilausprosessin hitauden ja pitkän toimitusajan vuoksi. Kirjallisen raportin työstämiseen ei kulunut kuin muutaman viikon verran, mutta hioin kieliasua ja tein työhön pieniä rakenteellisia muutoksia vielä pitkään infonäyttötoteutuksen testauksen ja luovutuksen jälkeen.

Työskentelytavoissani olisi voinut olla enemmän aikataulutusta, järjestelmällisyyttä ja olisin voinut kirjata muistiin selkeämmin toteutuksenaikaista toimintaani. Opinnäytetyöni tekstiosiota työstäessä jouduin palauttamaan mieleeni toteutusvaiheita vaikeaselkoisista muistiinpanoistani. Myös Git-versionhallintatyökalusta olisi voinut olla hyötyä, sillä työstin koodia jopa kolmessa eri käyttöjärjestelmässä (Windows 10, Raspberry Pi OS ARM, Raspberry Pi OS x86). Toteutuksen kannalta versiointi ei olisi ollut tarpeellista suhteellisen vähäisen koodimäärän ja hyvän pohjasuunnitelman takia, mutta ilman pilvipohjaista Git-palvelua jouduin siirtelemään koodia käyttöjärjestelmästä toiseen sähköpostia avuksi käyttäen.

Infonäyttötoteutukseni on hyvin skaalautuva isompiinkin toimitiloihin jo tällaisenaan. Infonäyttölaitteita voisi asentaa useampia eri puolille kohdetta. Esimerkiksi Joensuun pääkirjastolla voisi olla osastokohtaisia ilmoitustauluja, joiden käyttölaitteet lukisivat niille luoduista ja määritetyistä Drive-kansioistaan kullekin osastolle kohdennetun infonäyttömateriaalin. Kaikkea materiaalia voi edelleen hallinnoida yhdellä Google-tilillä, jolloin kaikkien eri infonäyttölaitteiden materiaali on keskitetysti hallittavissa samassa käyttöliittymässä. Näyttölaitteet saavat myös olla resoluutioltaan ja kuvasuhteeltaan erilaisia.

Toteutusta voisi jatkokehittää vaikkapa tehdas- ja erilaisiin tuotantoympäristöihin liittämällä toteutukseen esimerkiksi Googlen Data Studio API:n tai Microsoftin Power BI -datanvisualisointityökalun näyttämään reaaliaikaista dataa esimerkiksi tuotantoprosessien sujuvuudesta ja tavoitteista. Modifioimalla nykyistä toteutusta se voisi myös kierrättää infonäytössä monipuolisempaa materiaalia, vaikkapa erilaisia nettisivuja. Omalla sisältöpalvelimella varustettuna ja omalla graafisella käyttöliittymällä toteutus voisi olla jo tutkimieni kaupallisten toteutusten veroinen.

Olin yllättynyt Googlen tarjoamien ilmaisten työkalujen ja API:en määrästä. Niillä voi luoda sovelluksia, jotka käyttävät Googlen palvelujen ominaisuuksia lähes rajattomasti. Omien Google-sovellusten luomisessa tuntuu olevan vain mielikuvitus rajana. Haluaisin tulevaisuudessa palata tutkimaan lisää Googlen tarjoamia ja sovelluskehittäjille avattuja mahdollisuuksia ja tuottaa lisää omiin tai muiden tarpeisiin soveltuvia sovelluksia.

Lähteet

Anicas, M. 2014. An Introduction to OAuth 2. Tutorial. https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2. 5.5.2021.

AVconcept 2021. Infojärjestelmä ja digital signage. https://avconcept.fi/info-jarjestelma-ja-digital-signage/. 4.5.2021.

Bhatia, S. 2021. Procedural Programming [Definition]. https://hackr.io/blog/procedural-programming. 1.6.2021.

Embedded Linux Wiki. 2021. RPi Hardware. https://elinux.org/RPi_Hardware. 1.6.2021.

Finnsat. 2021. AV-ratkaisut. https://www.finnsat.fi/category/174/av---ratkaisut 4.5.2021.

Google Cloud Platform. 2021. https://cloud.google.com. 16.5.2021.

Google Cloud Storage. 2021. Filename encoding and interoperability problems. https://cloud.google.com/storage/docs/gsutil/addlhelp/FilenameencodingandinteroperabilityproblemsGoogle Developers. 2021a. Google Drive for Developers. Python Quickstart. https://developers.google.com/drive/api/v3/quickstart/python. 16.5.2021.

Google Developers. 2021b. Google Workspace for Developers. Create credentials. https://developers.google.com/workspace/guides/create-credentials. 16.5.2021.

Hitaltech. 2021. The History of Raspberry Pi. https://hitaltech.co.uk/the-history-of-the-raspberry-pi/. 1.6.2021.

Imgur. 2021. NOOBS-näkymä. https://imgur.com/sNmsHbj. 16.5.2021.

Internet Engineering Task Force. 2012. https://datatracker.ietf.org/doc/html/rfc6749. 13.6.2021.

Kasurinen, J. 2009. Python 3 -ohjelmointi. Jyväskylä: Docendo.

Kuivanen, I. 2014. Responsiiviset sivustot. http://users.metropolia.fi/~kuivi/bfish/bootstrap.php. 16.5.2021.

Kuuppelomäki, Päivi. 2009. Modulaarinen ohjelmointi. Luentomoniste. https://www.cs.helsinki.fi/u/kuuppelo/C/S2009/luento9.pdf. 1.6.2021.

Kuutti, W. 2011. Linux-käsikirja. Jyväskylä: Docendo.

Kärkkäinen, E. 2021. Erikoiskirjastovirkailija. Pointin kirjasto. Puhelinkeskustelu. 1.6.2021.

Linux Information Project. 2021. Journaling Filesystem Definition. http://www.linfo.org/journaling_filesystem.html. 15.5.2021.

Linux Manual. 2021. Bash. https://linux.die.net/man/1/bash. 15.5.2021.

LXDE Wiki 2021a. LXDE. https://wiki.lxde.org/en/Main_Page. 1.6.2021.

LXDE Wiki 2021b. Openbox. https://wiki.lxde.org/en/Openbox. 1.6.2021.

Mozilla. 2020. Mozilla Support. https://support.mozilla.org/en-US/questions/1285060#answer-1307487. 16.5.2021.

Mozilla. 2021. Mozilla Support. https://support.mozilla.org/en-US/. 16.5.2021.

Npm. 2018. Node-static. https://www.npmjs.com/package/node-static. 16.5.2021.

Npm. 2021. Livereload. https://www.npmjs.com/package/livereload. 16.5.2021.

Npm Docs. 2021. https://docs.npmjs.com/about-npm. 1.6.2021.

Nodejs.org. 2021. About Node.js. https://nodejs.org/en/about/. 1.6.2021.

OKdo 2021. OKdo Raspberry Pi 4 4Gb Model B Starter Kit. https://www.okdo.com/p/okdo-raspberry-pi-4-4gb-model-b-starter-kit/. 16.5.2021.

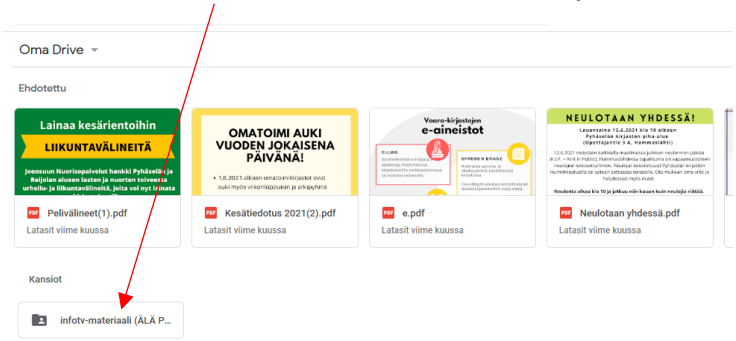
Open Knowledge Finland. 2021. Avoimen rajapinnan määritelmä. http://avoinrajapinta.fi/. 1.6.2021.
Peltomäki, J. 2017. JavaScript-kieli : uudet ominaisuudet. Helsinki: Books on Demand.
Peltomäki, J. 2020. Node.js : web-palveluiden ohjelmointi. Helsinki: Books on Demand.
PyBites. 2021. How Encoding Works in Python. https://pybit.es/python-encodings.html. 15.5.2021.
Python.org. 2021a. Python Frequently Asked Questions. https://docs.python.org/3/faq/. 1.6.2021.
Python.org. 2021b. Logging facility for Python. https://docs.python.org/3/library/logging.html. 1.6.2021.
Raspberry Pi Foundation. 2021a. Raspberry Pi FAQs. https://www.raspberrypi.org/documentation/faqs/. 4.5.2021.
Raspberry Pi Foundation. 2021b. Products. https://www.raspberrypi.org/products/. 6.5.2021.
Raspberry Pi Foundation. 2021c. Software. https://www.raspberrypi.org/software/. 5.5.2021.
Raspberry Pi Foundation. 2021d. Operating systems. https://www.raspberrypi.org/software/operating-systems/. 5.5.2021.
Raspberry Pi Foundation. 2021e. Forums. https://www.raspberrypi.org/forums/. 5.5.2021.
Raspberry Pi Foundation. 2021f. Documentation. https://www.raspberrypi.org/documentation/raspbian/. 5.5.2021.
Signagelive. 2021. https://signagelive.com/signagelive-launches-support-for-google-chrome-os/. 4.5.2021.
StackExchange. 2015. What is DISPLAY=:0? https://unix.stackexchange.com/questions/193827/what-is-display-0. 15.5.2021.
StackExchange. 2011. How can I run a cron command with existing environmental variables? https://unix.stackexchange.com/questions/27289/how-can-i-run-a-cron-command-with-existing-environmental-variables. 25.5.2021.
Stackoverflow. 2018. https://stackoverflow.com/questions/50933194/how-do-i-set-the-pythonutf8-environment-variable-to-enable-utf-8-encoding-by-def. 15.5.2021.
Suomen 3M Oy. 2021. RFID 201: Yleistä. https://multimedia.3m.com/mws/media/608528O/rfid-201.pdf. 1.6.2021.
Turunen, T. 2020. Myynnin tärkeimmät KPI-mittarit. Vainun blogi B2B-ammattilaisille, 26.2.2020. https://www.vainu.com/fi/blogi/kpi-mittari/. 5.5.2021.
W3Schools. 2021. Node.js NPM. https://www.w3schools.com/nodejs/nodejs_npm.asp. 16.5.2021.
Valotalive. 2021. https://valota.live/. 4.5.2021.
Watson, J. 2019. Raspberry Pi 4 getting hot? A closer look. https://www.zdnet.com/article/raspberry-pi-4-getting-hot-a-closer-look/. 5.5.2021.
Wikipedia. 2021a. Digital signage. https://en.wikipedia.org/wiki/Digital_signage. 5.5.2021.
Wikipedia. 2021b. Raspberry Pi. https://en.wikipedia.org/wiki/Raspberry_Pi. 4.5.2021.
Wikipedia. 2021c. Raspberry Pi OS. https://en.wikipedia.org/wiki/Raspberry_Pi_OS. 4.5.2021.

Vähimaa, A. 2021. Bitit poikki ja pinoon. Opas: tiedostojärjestelmät. Mikrobitti 2021 (3), 54–63.

Infonäytön esityssisällön päivitysohje henkilökunnalle

Infonäytön mainosten lisääminen/poistaminen

1. Mene osoitteeseen: https://drive.google.com/
2. Kirjaudu sisään tunnuksilla: [xxx]
3. Avaa kansio: ”Infotv-materiaali” tuplaklikkaamalla sitä

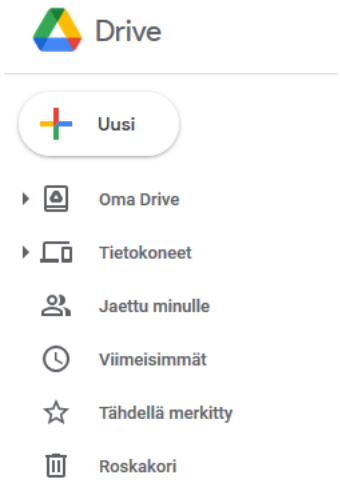


Tiedoston lisääminen kansioon

Lisää Infotv-materiaali-kansioon sisältöä raahaamalla tiedosto(t) hiirellä haluamastasi sijainnista ja tiputtamalla tiedosto(t) kansioon. Voit myös lisätä tiedoston valitsemalla vasemmasta sivuvalikosta Uusi -> Lataa tiedosto Driveen ja valitsemalla avautuvasta valikosta omalta tietokoneeltasi haluamasi tiedoston ja valitsemalla sen jälkeen Avaa, jolloin tiedosto latautuu kansioon. Voit lisätä Infotv-materiaali-kansioon yksisivuisia PDF-tiedostoja ja Firefoxin tukemia kuvatiedostotyyppejä (APNG, AVIF, GIF, JPEG eli JPG, PNG, SVG, WebP, BMP, ICO, TIFF). Huomioi, että infonäytössä esitettävä sisältö päivittyy viiden minuutin välein, eli Driveen lisätty tiedosto ilmestyy esitykseen pienellä viiveellä.

Tiedostojärjestyksen muuttaminen kansiossa

Tiedostot järjestyvät tiedostonimen mukaan. Jos haluat muuttaa niiden järjestystä, voit nimetä tiedostot uudelleen. Se tapahtuu painamalla hiiren oikeaa painiketta halutun tiedoston päällä ja valitsemalla ”Nimeä uudelleen” ja antamalla tiedostolle uuden nimen. Vinkki: tiedostojen numerointi voi auttaa selkiyttämään ne haluttuun järjestykseen. Tiedostojen nimissä voi olla kirjaimia, numeroita, ääkkösiä ja erikoismerkkejä.



Tiedoston poistaminen kansiosta ja palauttaminen

Vanhentuneet mainokset voit poistaa valitsemalla mainoksen hiirellä aktiiviseksi ja painamalla näppäimistöltä Delete. Poiston voi tehdä myös valitsemalla tiedoston aktiiviseksi, painamalla hiiren oikeaa painiketta ja valitsemalla avautuvasta valikosta Poista. Poistetut mainokset siirtyvät vasemman sivuvalikon roskakoriin, josta ne voi tarvittaessa palauttaa. Jos haluat palauttaa mainoksen, valitse vasemmasta sivuvalikosta Roskakori, klikkaa hiiren oikealla painikkeella haluttua tiedostoa ja valitse Palauta.

Huomioitavaa. Älä poista kansiota ”Infotv-materiali”. Jos poistat sen vahingossa, sen voi palauttaa roskakorista samalla tavalla kuin tiedostot.