Prototyypin toteutusdokumentti

Lassi Aaltonen, Jami Jokinen, Kasper Kouhia ja Tomi Mauno

Toteutusdokumentaatio

Lassi Aaltonen, Jami Jokinen, Kasper Kouhia ja Tomi Mauno

1. Johdanto & Määrittely

Tämä dokumentti käsittelee aurinkopaneeli- ja säädatan tulostusohjelma projektin toteutusta.

Ohjelman tulee olla toiminnallinen, tehokas, sekä intuitiivinen käyttää minkä vaan tasoiselle käyttäjälle.

Ohjelman kautta tulee käyttäjän pystyä muuntamaan aurinkoenergian viikko- ja kuukausiraportin Excel-tiedostot PDF-tiedostoiksi, joissa aurinkopaneelien datasta on tehty kaaviot.

Tämä totetusdokumentaatio tulee käymään läpi seuraavat osat projektista:

1. Ohjelman yleiskuvaus
2. Projektin työnjako
3. Ohjelmassa käytetyt tiedostot ja tietovarastot
4. Ohjelmassa olevat toiminnot
5. Ohjelman ulkoiset liittymät ja riippuvuudet
6. Ohjelman testaus

Projektin päämääränä on tuottaa määrittelyn mukainen prototyyppi tulevaa tuotantoversioa varten, jossa alustavat, sekä tarvittaessa lisätyt toiminnot.

2. Työnjako

Suunnittelu & dokumentaatiot

Suunnitellessamme projektia Lassi teki mallisuunnitelmat käyttöliittymästä ja PDF-tiedostosta, jossa määriteltiin miltä asiat voisivat näyttää ja missä niiden tulisi olla. Suunnittelimme tapoja miten toteutamme sovelluksen.

Laadimme kolme eri dokumenttia projektista. Suunnittelu-, toteutus- ja ohjedokumentin. Kaikki osallistuivat dokumentoimiseen yhtälailla.

Toiminnallisuus

Responsiivisuus

Kasper teki PDF-tiedoston responsiiviseksi lisäämällä toiminnon, joka muuttaa sivun sisällön kokoa sivun koon mukaan.

Säädata

Jami alkoi tutkimaan API-avaimen toimintaa ja sai tehtyä “prototyypin” säädatan hausta. Myöhemmin kuitenkin Kasper teki käyttämämme API-kutsu toiminnon.

Lassi piirsi Adobe Illustratorilla kuvat säätaulukkoon ja Kasper teki toiminnon, joka lisää oikean kuvan sään mukaan.

Kaaviot

Tomi kirjoitti koodin, joka hakee viimeisimmän Excel-tiedoston automaattisesti käyttäjään Downloads-kansiosta.

Tomi löysi tavan muuttaa Excel-dataa luettavampaan muotoon pandas-kirjaston avulla. Kasper muutti koodin myöhemmin kompaktimmaksi.

Aluksi Lassi sijoitti kaaviot manuaalisesti koordinaatteja käyttäen niin, että kaaviot olisi keskellä.

Myöhemmin Kasper teki kaavioiden sijoittamisesta ohjelmallisen.

Käyttöliittymä

Jami aloitti käyttöliittymän toteutuksen TkInter-kirjaston avulla. Tomi auttoi ulkonäön suunnittelussa.

Kasper siisti myöhemmin ulkoasua.

Lisäasetukset

Kasperin idea oli lisätä lisäasetukset, jotka Jami lisäsi käyttöliittymään.

Kasper viimeisteli ja paransi osaa asetuksista.

Sivukoko

Jami teki funktion, joka muuttaa tulostettavan PDF-tiedoston kokoa, jonka jälkeen Lassi siirsi kaaviot manuaalisesti oikeisiin paikkoihin ja oikean kokoisiksi. Kasper teki kaavioista ja säätaulukosta responsiiviset

Kasper lisäsi toimintoon tavan, jolla käyttäjä voi asettaa sivulle oman koon, jos niin tahtoo.

Säätaulun tulostus

Jami teki funktion joka joko lisää tai poistaa säätaulukon, Kasper viimeisteli funktion toiminnallisuuden.

Config-tiedosto

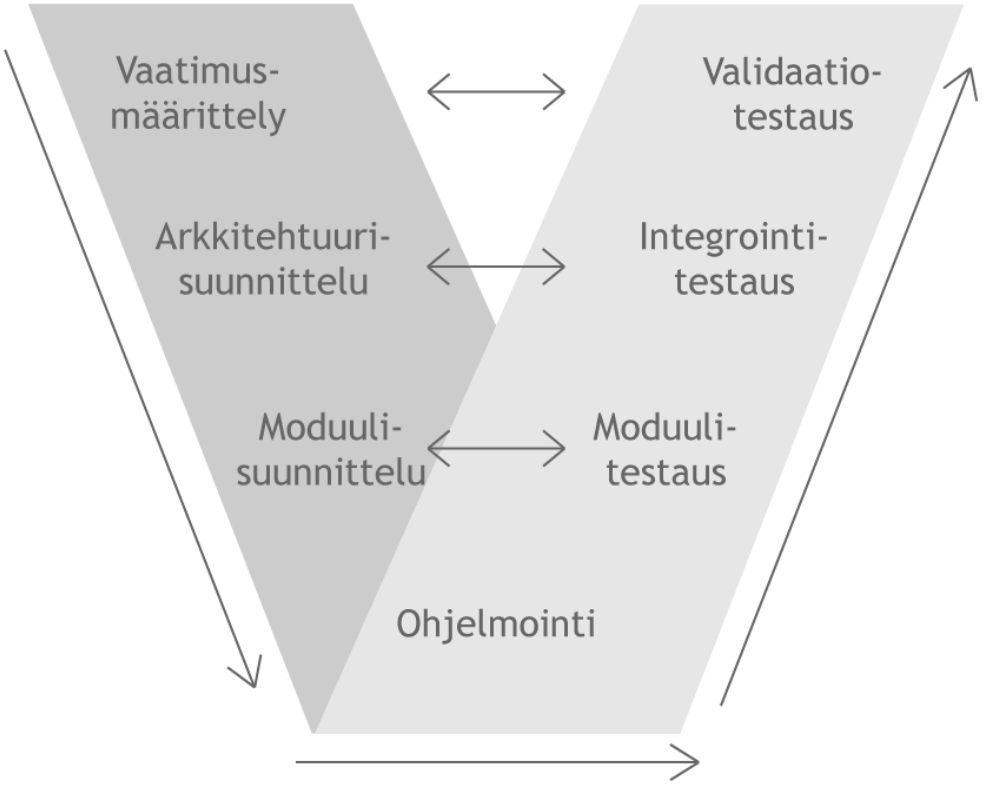
Jami rakensi tavan jolla API-avain, valittu koko, sekä tulostetaanko kaavio tallentuu tekstitiedostoon. Ohjelma lukee tästä tiedostosta nämä tiedot ja käyttää niitä, näin tallentaen annetut käskyt/muuttujat tulevaa käyttöä varten.

Kasper muunsi tätä toimintoa muuttaen tiedostotyypin JSON-kieleen ja muunsi rakennetta.

API-avaimen vaihto / tallennus

Jami teki toiminnon, jolla API-avain voidaan vaihtaa käyttöliittymän kautta. Tämä toiminto myös sitoutui tallennus-toiminnon, tallentaen API-avaimen config-tiedostoon.

3. Testaus



Ohjelmointi

Suunnittelimme, mitä ohjelmointikieltä ja tekstieditoria käytämme sekä mitä lisäosia tarvitsemme projektin toteuttamiseen. Päädyimme Python ohjelmointikieleen. Valitsimme Visual Studio Code tekstieditorin. Asensimme ohjelmaan tarvittavat lisäosat ja kirjastot.

Kirjoittaessa ohjelman koodia on huomioitu koodin helppolukuisuutta kommentoimalla osioiden ylle tietoa, mitä osio sisältää.

Vaatimusmäärittely & Validaatiotestaus

Vaatimusmäärittelyssä on mietitty kaikki vaatimukset, mitkä valmiin ohjelman tulee täyttää, ilman että on vielä mietitty miten nämä tullaan käytännössä toteuttamaan.

Validaatiotestauksella on varmistettu, että ohjelma täyttää kaikki määritellyt vaatimukset, kun kaikki ohjelman toiminnot on toteutettu, ja yhdistetty samaan tiedostoon.

Arkkitehtuurisuunnittelu & Integrointitestaus

Arkkitehtuurisuunnittelussa on suunniteltu, kuinka saisimme ohjelman komponentit toimimaan keskenään. Samalla etsimme ratkaisuja suunnitelman toiminnallisuuden saavuttamiseen.

Integrointitestauksella on testattu, että ohjelman yksittäiset komponentit toimivat toisten komponenttien kanssa yhteistyössä. Esimerkkinä testasimme, että haettua excel-dataa voidaan käyttää kaavioiden piirtämiseen.

Moduulisuunnittelu & Moduulitestaus

Moduulisuunnittelussa on mietitty mitä kaikkien ohjelman osien tulisi tehdä ja kuinka saamme ne toimimaan.

Moduulitestauksella on testattu, että ohjelman kaikki osat toimivat tarkoituksenmukaisesti ennen niiden yhdistämistä. Esimerkkinä varmistimme, että kaavioiden piirtäminen toimii staattisilla arvoilla ilman, että piirtämiseen käytettiin vielä excel-dataa.

4. Käytetyt kirjastot & riippuvuudet

Matplotlib

Matplotlib:in avulla saimme muodostettua kaavioita excel-datan perusteella.

svglib

Svglib muuttaa kaaviot SVG-muotoon, ja asettaa ne pdf-tiedostoon.

reportlab

Reportlab:in avulla saimme ohjelman luomaan PDF-tiedoston, ja lisäämään sinne sisältöä.

requests

Requests-kirjastolla saimme haettua säädatan verkosta HTTP-kutsulla.

Tkinter

Teimme Tkinter-kirjastolla visuaalisen käyttöliittymän.

xlrd

Käytimme xlrd:tä excel:in lukemiseen.

Pandas

Pandas teki excel-tiedoston muuttamisen ohjelmallisesti luettavampaan muotoon mahdolliseksi.

PyInstaller

PyInstaller:ia käyttämällä pakkasimme ohjelman .exe-muotoon. Käytännössä tämä tarkoittaa, että ohjelman suoritus tapahtuu tuplaklikkaamalla.

ClimaCell API

Käytimme ClimaCell:in API-palvelua säädatan noutamiseen