Talousdatan visualisointiohjelma

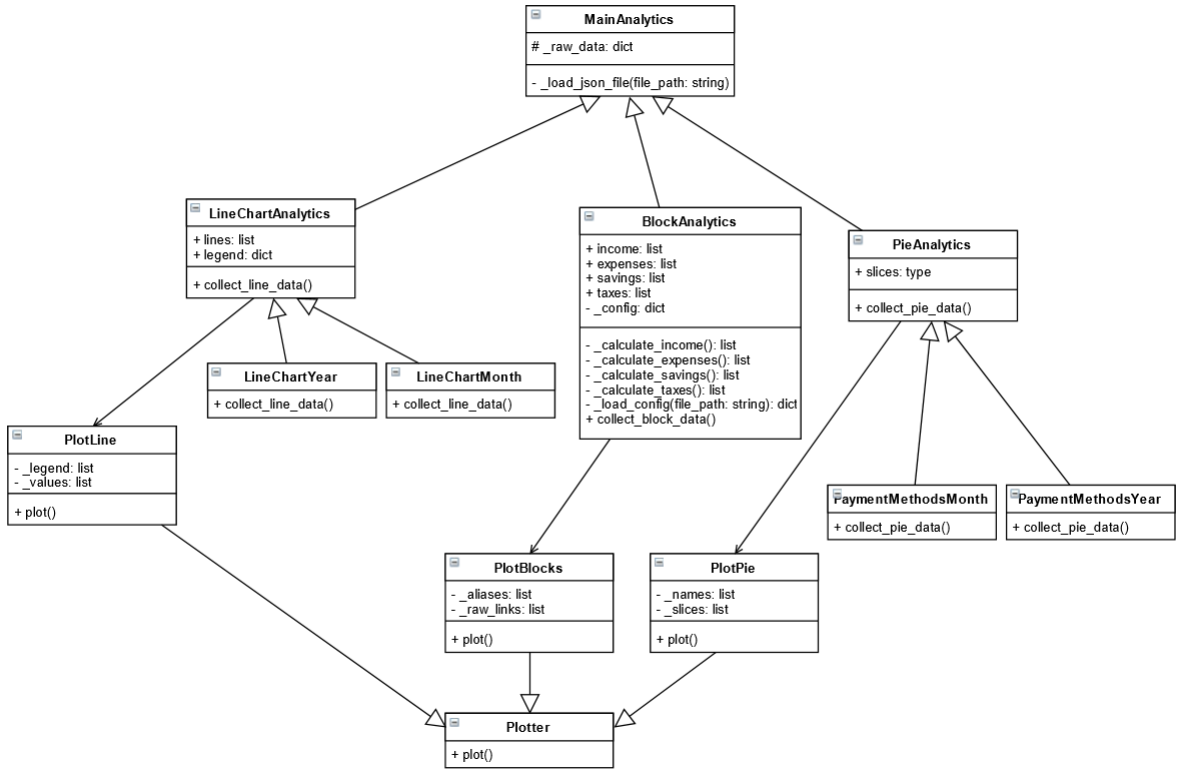
Ohjelmoinnin peruskurssi Y2, oma projekti: tekninen suunnitelma

Markus Säynevirta, 594684, ELEC EST, 3. vsk

6.3.2020

Ohjelman rakennesuunnitelma

Ohjelma koostuu PyQt-pohjaisista käyttöliittymäluokista ja niiden toimintaa tukevista apuluokista, joiden metodit käsittelevät päätietokantaa ja keräävät sen datan haluttuja osia graafisen liittymän toteutusten vaatimaan muotoon.

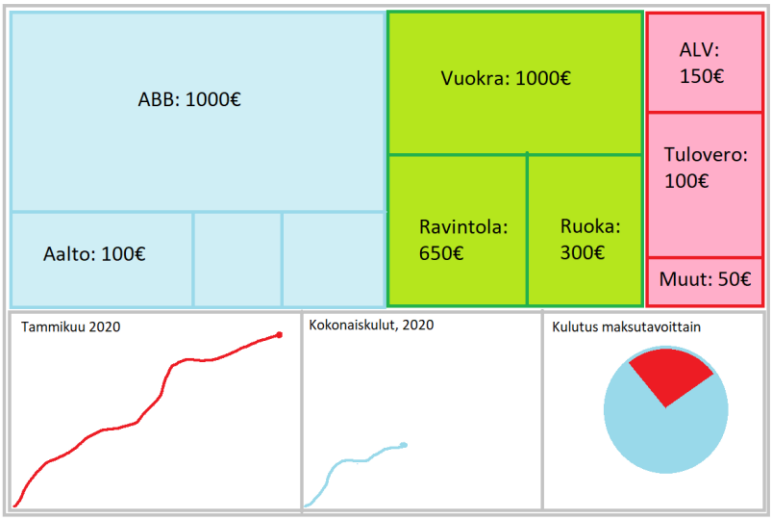


Kuva 1: Luokkarakenteen alustava UML-kaavio.

Apuluokkiin kuuluvat analytiikkaluokat MainAnalytics ja sen perivät LineChartAnalytics, BlockAnalytics ja PieAnalytics. Näistä ensimmäinen tuottaa viivakuvaajien, toinen blokkidiagrammien ja kolmas piirakkakuvaajien vaatimaa dataa. Kuvaajakohtaisia luokkia perivät alemmat PaymentMethods- ja LineChart-luokat, jotka piirtävät kyseisiä kuvaajia tietyn aikavälin datasta. BlockAnalytics.

Plotter ja sen alaluokat PlotLine, PlotBlocks ja PlotPie käsittelevät PyQt-kirjastoa ja toteuttavat kuvaajien graafiseen piirtämiseen vaaditut metodit.

Käyttötapauskuvaus



Kuva 2: Yleissuunnitelman visio ohjelman käynnistyksen yhteydessä aukeavasta dashboard-näkymästä.

Ohjelma käynnistetään työpöytäikonin kautta, minkä jälkeen käyttäjälle aukeaa dashboard-näkymä. Näkymän muodostaminen tapahtuu ohjelman koko luokkarakennetta käyttäen, sillä käynnistyksen yhteydessä kuvaajat luodaan tietokannan datan pohjalta uudelleen. Dashboard-näkymässä on neljä erilaista kuvaajaa: Ikkunan pääosan käyttää tutkibudjettia.fi blokkidiagrammi, joka kuvaa koko vuoden tulojen, menojen ja verojen kategorisesta jakaumasta. Blokkidiagrammin data haetaan BlockAnalytics-metodilla ja piirretään PlotBlocks-luokan plot()-metodilla.

Ikkunan alaosaan tulee kolme osaa, joista kaksi ensimmäistä ovat kuluvan kuukauden ja vuoden kumulatiiviset menot ja tulot viivagraafina. Viivagraafien datan kokoaminen tehdään LineChartMonth ja LineChartYear-luokkien avulla ja kuvaajat piirretään PlotLine-luokan metodilla plot(). Kolmantena kuvaajana on menneen vuoden kulutuksen piirakkakuvaaja maksutapojen mukaan, jonka data kootaan PaymentMethodsYear-metodilla ja piirretään PlotPie-luokan plot()-metodilla. Piirakkakuvaajan aikaskaalaa voidaan vaihtaa kuukauteen kuvaajan alapuolella sijaitsevalla painikkeella, jonka painamisen jälkeen PaymentMethodsMonth-luokka kerää halutun kuukauden tiedot, jotka piirretään uudelleen PlotPie-luokan plot()-metodilla.

Dashboard-näkymän yläosan blokkidiagrammi on interaktiivinen, ja sen kenttien painaminen avaa kyseisen luokan tarkemman osittelun. Osittelun hakeminen tapahtuu PlotBlocks-luokan metodeilla BlockAnalytics-luokan keräämän datan pohjalta.

Ohjelma voidaan sulkea suorituksen lopussa graafisen ikkunan kulmassa olevasta painikkeesta.

Tietorakenteet ja algoritmit

Ohjelman toteutuksen pohjana toimii vuosien 2019 ja 2020 kulutus- ja tulotiedoista json-muotoon koostetut tilastotiedot, joiden aineisto on kerätty pankkien tiliotteiden ja talteen otettujen ostokuittien pohjalta. Ohjelma ei tee muutoksia käyttämäänsä json-tietokantaan, vaan ainoastaan lukee sitä.

Pidemmällä aikavälillä päätietokanta voisi olla järkevää muuttaa esimerkiksi relaatiotietokannaksi, kasvavan datamäärän ja tehokkaan haettavuuden seurauksena. Tämän toteuttaminen ei ole kevään projektin aikataulurajoitteissa todennäköisesti mahdollista.

Kuvaajien konfiguraation määrittämiseen käytetään json-tiedostoja, joissa määritellään esimerkiksi, mihin datan tietyt kategoriat sisällytetään kuvaajassa (esim. lounaat, illalliset ja pikaruoka yhden ravintola-pääkategorian alle).

Tietokannan ja setup-tiedoston esimerkkitiedostojan on repositorion doc-kansion example_dataset-alakansiossa.

Projektin laskentapuoli on pääasiassa yksinkertaisia kategoriakohtaisia yhteenlaskuja, joilla kerätään eri transaktioiden yhteissummia. Blokkidiagrammin laatikoiden järjestely vaatii yksinkertaisen lajittelualgoritmin, ja sen toteutukseen käytetään projektin puitteissa Pythonin standardikirjaston sort()-metodia.

Verotietojen hakeminen transaktiosanakirjoista voidaan tehdä toistorakenteella ja summauksella.

Kirjastot

Ohjelman graafisen käyttöliittymän toteutuksessa käytetään PyQt-kirjastoa, ja sen PyQtChart-osaa. Json-tiedostojen käsittelyssä käytetään Pythonin standardikirjaston json-modulia (json enkooderi ja dekooderi). Kurssi asettaa rajoitteita kirjastojen käytölle, ja projekti siirtyy sen päättymisen jälkeen todennäköisesti johonkin monipuolisempaan ja modernimpaan kuvaajakirjastoon (esim. Plotly).

Projektin tavoitteena ei ole oman kuvaajakirjaston tai tiedostojen en/dekooderin toteuttaminen, jolloin kirjastojen käytöllä päästään keskittymään projektitavoitteeseen, eli taloustietokannan sisällön kuvaamiseen mielenkiintoisilla tavoilla.

Aikataulu

Viikko	Tavoitteet	Työtunnit
10	Yleissuunnitelma ja tekninen suunnitelma	4
11	**Projektitapaaminen 11.3.** Käyttöliittymän runko, blokkidiagrammin aloitusnäkymä ja viivadiagrammi, analytiikka viivadiagrammille ja blokkidiagrammille	**9**
12	Käyttöliittymän runko, blokkidiagrammi aloitusnäkymä ja viivadiagrammi, analytiikka viivadiagrammille ja blokkidiagrammille	9
13	**Checkpoint?** Blokkidiagrammin aloitusnäkymä ja piirakkadiagrammi, analytiikka piirakkadiagrammille ja blokkidiagrammille	**9**
14	Blokkidiagrammin interaktiivisuus ja piirakkadiagrammi, analytiikka piirakkadiagrammille ja blokkidiagrammille	9
15	Blokkidiagrammin ja piirakkadiagrammin interaktiivisuus	9
16	Blokkidiagrammin ja piirakkadiagrammin interaktiivisuus	9
17	Blokkidiagrammin interaktiivisuus	9
18	(Vappuviikko)	4
19	**Projektin määräaika + loppudemo**	**9**

Yksikkötestaussuunnitelma

Projektin toteuttamiseen käytettävissä oleva aika on varsin rajallinen ja yksilöprojektina testaus syö kehitysaikaa toiminnallisen ohjelmakoodin kehityksestä. Nämä rajoittavat tekijät huomioiden testauksen painopiste tulee olemaan ohjelman sisältämien metodien yksikkötestauksessa, missä tavoitteena on täysi kattavuus. Ohjelmaa voidaan tietyiltä osin myös integraatioitestata. Ohjelman kehityksen lähtökohtana on testipohjainen ohjelmistokehitys, millä voidaan välttyä merkittäviltä testiaukoilta.

Yksikkötestauksessa on tarkoitus käyttää pääasiassa unittest-kirjastoa, ja testejä on pääasiassa tarkoitus tuottaa dataa käsittelevillä apumetodeille, sillä graafisen käyttöliittymän testaaminen automaattisesti tai manuaalitestein olisi todennäköisesti liian aikaintensiivistä.

Yksikkötestauksen metodeilla varmistetaan ohjelmakoodin metodikohtainen toiminta syöttämällä koodille erilaisia variaatioita syötedatasta. Talousdatalaskennan kohdalla metodien syötedatana ovat päätietokannan variaatiot ja erilaiset konfiguraatiotiedostot, joiden yhdistelmä tuottaa erilasia ulostulevia datasettejä. Testimetodit ovat toteutukseltaan hyvin saman tyyppisiä eri visualisointikomponenttien välillä, joskin niille annettava syötedata vaihtelee ainakin konfiguraatiotiedostojen osalta.

Yksikkötestejä ajamista automaattisesti repositorioon tehdyille commiteille, mikäli Aallon tarjoama kehitysympäristö tukee ominaisuutta.

Viitteet

- Vastaavat sovellukset
  - Nordea Wallet
  - OP, Talouden tasapaino
  - S-Kanava, omat ostot
  - tutkibudjettia.fi
- Kirjastot
  - PyQt5
  - PyQtChart
  - json-parseri