



Minna Karhu, Katja Peltonen, Wille Pulkkinen, Hanne Salmi

Budjettisovellus

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintäteknologian tutkinto-ohjelma

Toteutusdokumentti

27.4.2023

Sisällys

1 Johdanto	1
2 Tuotteen vaatimukset	1
3 Käyttäjäroolit ja käyttötapaukset	2
4 Ohjelmiston tietomalli	4
5 Ohjelmiston rakenne	6
6 Ohjelmiston toiminta	9
7 Kehitysprosessi ja kehitysvaiheen tekniikat	12
8 Tuotteen testaus	14
9 Jatkokehitysideat	14
10 Yhteenveto	15

1 Johdanto

Budjettisovellus on rahankäytöstä kiinnostuneille ihmisille suunniteltu sovellus. Sovellus on hyvin helppokäyttöinen ja sen avulla voidaan seurata rahankulutusta muun muassa eri profiileja käyttäen. Budjettisovellukseen on mahdollista luoda tarpeen mukaan useampia profiileja eri aihealueen rahankäyttöä varten tai useampaa käyttäjää varten, jos sovellusta halutaan käyttää samasta laitteesta.

Sovellukseen syötetään kuluja ja sovellus tallentaa nämä kulut tietokantaan. Tietokannasta sovellus hakee käyttäjälle listan kuluista sekä laskee haettujen tietojen perusteella, paljonko käyttäjällä on jäljellä budjettia kuluvalle kuukaudelle. Maksimibudjetti sovellukseen syötetään profiilia tehdessä.

Tämän dokumentin tavoite on selventää budjettisovelluksen toimintaa. Dokumentissa käydään läpi niin tuotteen vaatimukset kuin myös ohjelmiston rakenne ja toiminta. Lisäksi dokumentissa perehdytään tuotteen kehitysprosessiin.

2 Tuotteen vaatimukset

Sovellus on kehitetty omaa rahankäyttöään seuraaville ihmisille ja niille, jotka haluavat säästää rahaa tai pysyä tietyssä budjetissa. Budjettisovellus auttaa näkemään, mihin tietyn ajanjakson aikana on rahaa käyttänyt. Sovelluksesta löytyy myös ennuste, joka arvioi, kuinka paljon rahaa tulee kuukauden aikana kulumaan, jos rahankäyttö ei muutu.

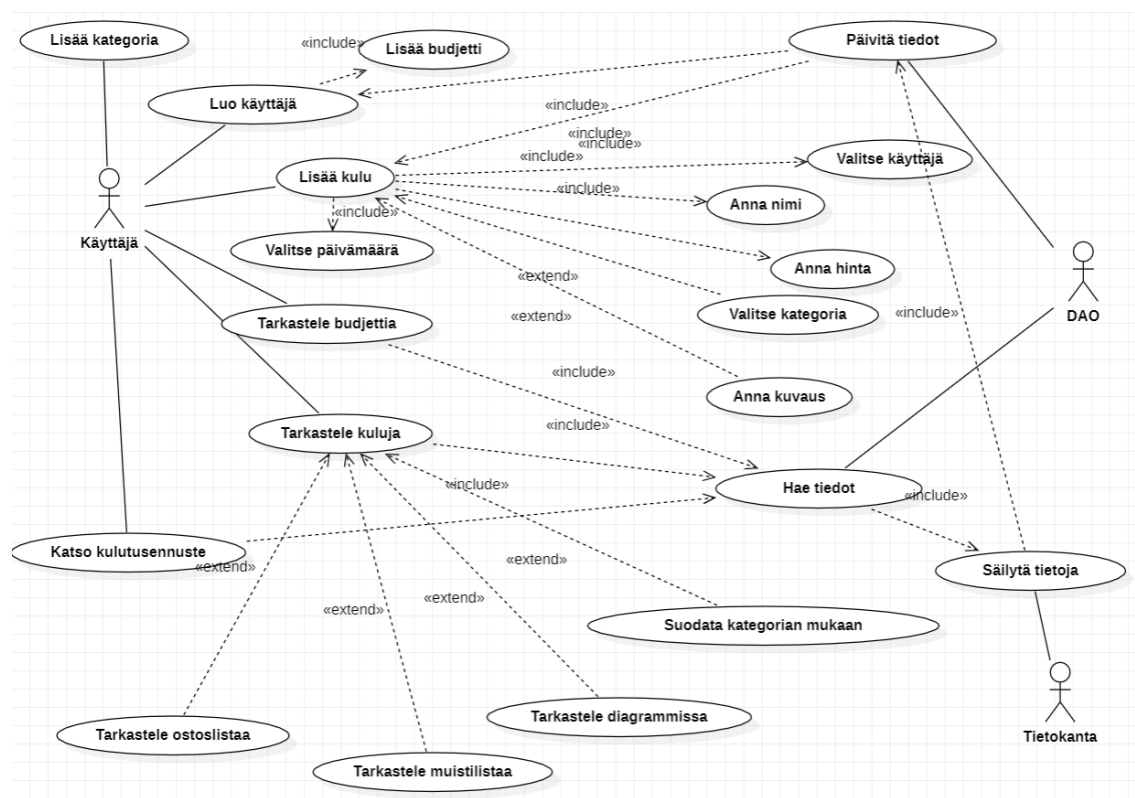
Sovellus pitää yllä käyttäjän budjettia laskien jäljellä olevan budjetin muutosta aina uuden kulun luonnin yhteydessä sekä kuluja muokatessa tai poistettaessa. Maksimibudjetti syötetään sovellukseen käyttäjää tehdessä ja sitä voidaan aina muokata käyttäjän asetuksista.

Sovelluksessa on myös mahdollista luoda uusia kategorioita ja lisätä kulut tiettyyn kategoriaan. Kategorioiden pohjalta sovellus luo käyttäjälle diagrammin, josta kulutustaan voi tarkastella kategorioittain. Lisäksi sovelluksesta löytyy tekoälyn tuottama ostoslista sekä muistilista. Ostoslistalla on käyttäjän yleisesti ostamia tuotteita. Muistilistalle taas on kasattu toistuvia kuluja, kuten vuokra ja kotitalouslaskut.

Budjettisovellus on suunniteltu kaikenikäisten ihmisten käyttöön. Tämän vuoksi vaatimuksena on, että sovelluksen käyttöliittymä on selkeä ja että sovellus on kokonaisuudessaan mahdollisimman helppokäyttöinen.

3 Käyttäjäroolit ja käyttötapaukset

Budjettisovelluksen käyttäjäroolit ja käyttötapaukset on kuvattu kaaviossa 1. Sovellukseen voi luoda useamman käyttäjän, mutta jokaiselle käyttäjälle on tarjolla samat toiminnot. Lisäksi sovelluksesta löytyy tietokanta, joka säilöo käyttäjän tiedot, sekä Data Access Objecteja, jotka hakevat ja päivittävät tietokannassa olevia tietoja.



Kaavio 1: Budjettisovelluksen käyttötapauskaavio.

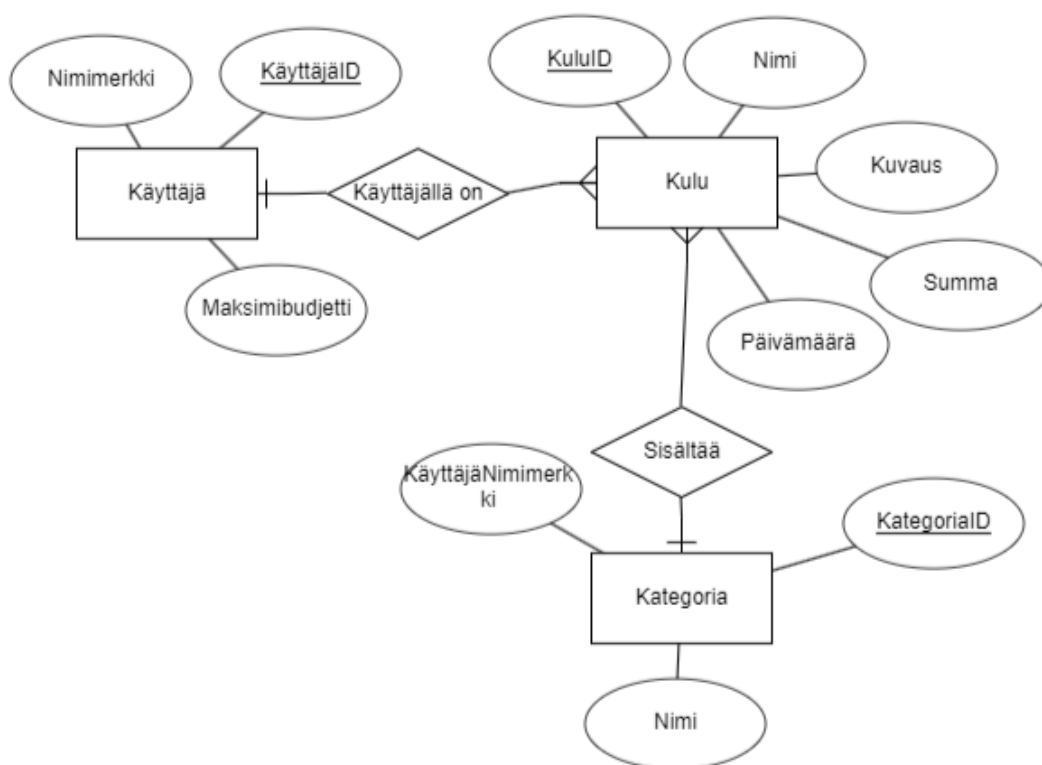
Ensin sovellukseen tulee luoda vähintään yksi käyttäjä. Käyttäjälle tulee antaa maksimibudjetti. Tämän jälkeen sovellukseen voi lisätä kuluja. Kuluille tulee valita käyttäjä sekä päivämäärä ja antaa nimi sekä hinta. Halutessaan kululle voi myös valita kategorian ja antaa kuvauksen. Jos sopivaa kategoriaa ei löydy, käyttäjä voi luoda sellaisen. Data Access Object päivittää käyttäjän lisäämät kulut, kategoriat ja käyttäjät tietokantaan.

Sovellukseen lisättyjä kuluja voi tarkastella joko listana tai diagrammin muodossa. Listalla olevat kulut saa halutessaan suodatettua kategorian mukaan. Käyttäjä voi myös tarkastella budjettiaan, eli katsoa, paljonko kuukauden budjetissa on vielä rahaa jäljellä. Halutessaan käyttäjä voi tarkastella myös kulutusennustetta, joka näyttää, paljonko rahaa tulee kuukauden aikana kulumaan, jos kulutus jatkuu samalla tasolla. Data Access Object hakee nämä kuluihin ja budjettiin liittyvät tiedot tietokannasta.

Lisäksi sovelluksesta löytyy ostoslista ja muistilista. Tekoäly generoi nämä listat käyttäjän sovellukseen lisäämien kulujen pohjalta. Ostoslistalta löytyy käyttäjän usein ostamia tuotteita. Muistilistalla sen sijaan on erilaisia kuukausittain toistuvia kuluja.

4 Ohjelmiston tietomalli

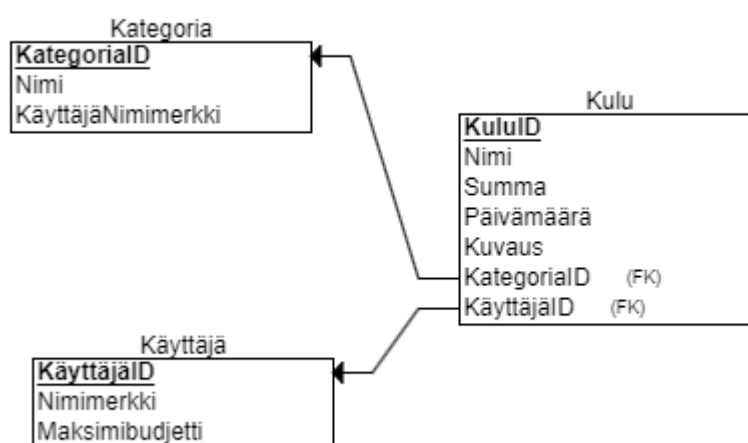
Kaaviossa 2 on kuvattu budjettisovelluksen entiteettien suhteet diagrammina. Kululla on omien data-attribuuttien lisäksi viittaukset Käyttäjä ja Kategoria -entiteetteihin monen suhteessa yhteen.



Kaavio 2: Entiteettisuhdediagrammi (eng. ER-diagram).

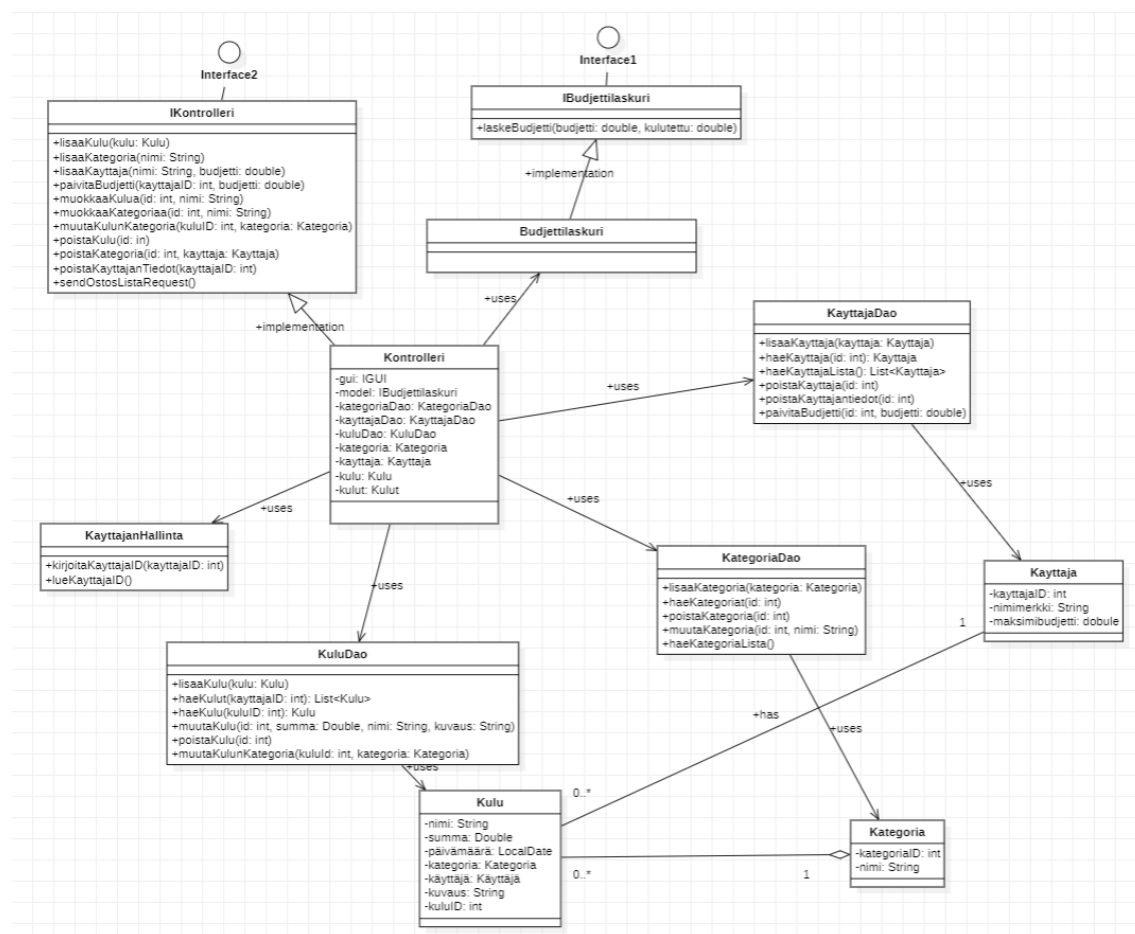
Kaaviossa 3. on kuvattu budjettisovelluksen relaatiotietokannan skeema diagrammina. Kululla on pääasiallinen uniikki avainattribuutti KululD (eng. primary key) ja omien data-attribuuttien nimen, kuvauksen, summan ja päivämäärän lisäksi viittaukset Käyttäjä ja Kategoria-entiteetteihin

vierasavaimien (eng. foreign key) “KäyttäjälD” ja “KategoriaID” perusteella .
 Kategoria-entiteetillä on pääasiallinen uniikki avain KategoriaID, Nimi ja
 KäyttäjänNimimerkki-attribuutit. Käyttäjällä on pääasiallinen uniikki avain
 KäyttäjälD ja Maksimibudjetti sekä KäyttäjäNimimerkki attribuutit.



Kaavio 3: Relaatiotietokannan skeema (eng. relational schema).

5 Ohjelmiston rakenne



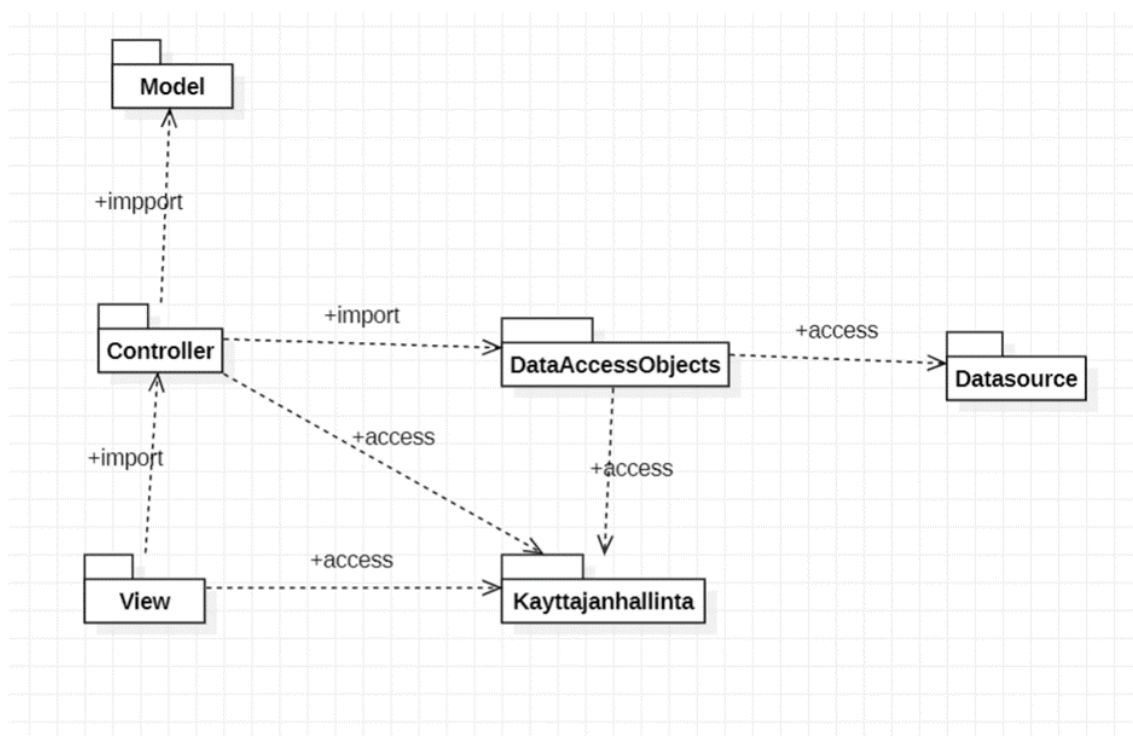
Kaavio 4: Luokkakaavio.

Kaaviossa 4 on kuvattu ohjelmiston rakenne luokkakaavion muodossa. Luokkakaavioon on otettu mukaan vain olennaisimmat Model, Controller, kayttajanHallinta ja DataAccessObjects-pakkauksiin kuuluvat luokat. Siihen ei ole sisällytetty View-pakkauksen luokkia. Kaavion selkeyttämiseksi myös getterit ja setterit on jätetty pois.

Kaaviosta 4 on nähtävissä, että käyttäjä voi luoda sovellukseen käyttäjän, joka on muotoa *Kayttaja*. Käyttäjälle voi lisätä kuluja, jotka ovat muotoa *Kulu*. Yhdellä käyttäjällä voi olla monta kuluja, mutta sama kulu ei voi liittyä useampaan eri käyttäjään. Kuluja ei voi lisätä sovellukseen luomatta käyttäjää, joten kulut liittyvät aina johonkin käyttäjään. Sovellukseen voi myös luoda

kategorian, joka on muotoa *Kategoria*. Kuluille voi lisäksi yhteydessä valita kategorian. Yhdessä kategoriassa voi olla monia kuluja, mutta sama kulu ei voi olla useammassa eri kategoriassa. Jos kuluille ei valita kategoriata, se menee yleiseen kategoriaan.

Käyttäjä lisää käyttäjät, kulut ja kategoriat sovellukseen graafisessa käyttöliittymässä. Graafisesta käyttöliittymästä tiedot kulkeutuvat kontrollerin kautta *KayttajaDao*, *KuluDao* ja *KategoriaDao*-objekteille. Nämä Data Access Objectit huolehtivat tiedon tallennuksesta tietokantaan, jossa tiedot säilyvät myös sovelluksen sulkemisen jälkeen. Sovelluksen käynnistyessä tietokantaan tallennetut tiedot haetaan graafiseen käyttöliittymään samalla tavalla. Kontrolleri siis hakee käyttäjät, kulut ja kategoriat tietokannasta Data Access Objectien (*KayttajaDao*, *KuluDao* ja *KategoriaDao*) avulla.



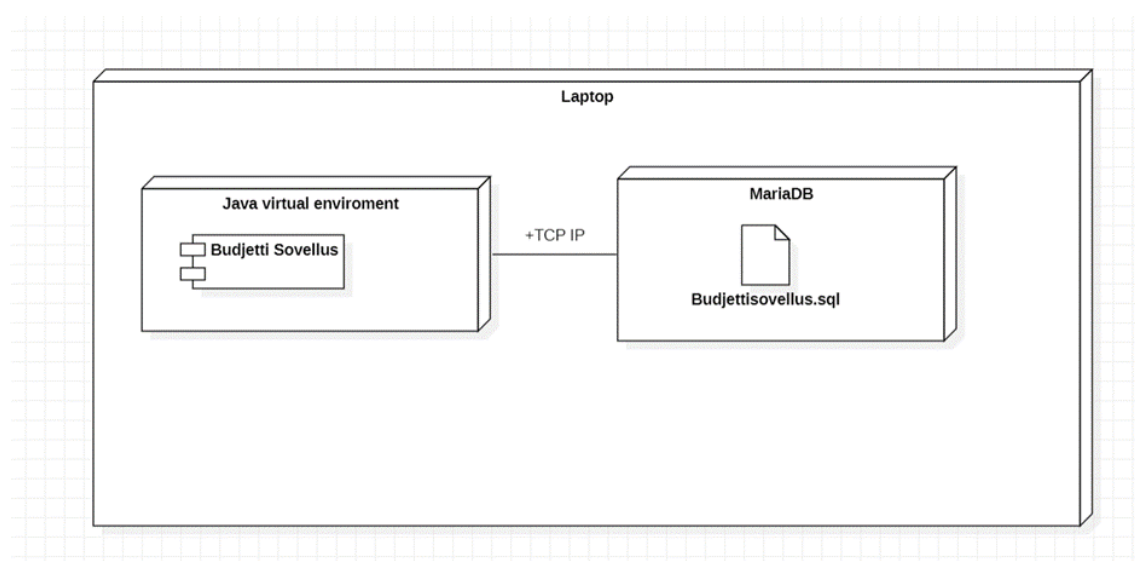
Kaavio 5: Pakkauskaavio.

Kaaviosta 5 näkee, että ohjelmisto on jaoteltu kuuteen eri pakkaukseen.

View-pakkaus pitää sisällään käyttöliittymän joka importtaa

Controller-pakkauksen, joka toimii kommunikoinnin solmukohtana molempiin

suuntiin Model ja DataAccessObjects-pakkauksiin importtaamalla nämä pakkaukset. View, Controller ja DataAccessObjects pääsevät käsiksi Kayttajanhallinta-pakkaukseen käyttäjätietojen käsittelyä varten. DataAccessObjects on ainoa, joka pääsee käsiksi Datasource-pakkaukseen joka on yhteydessä tietokantaan.

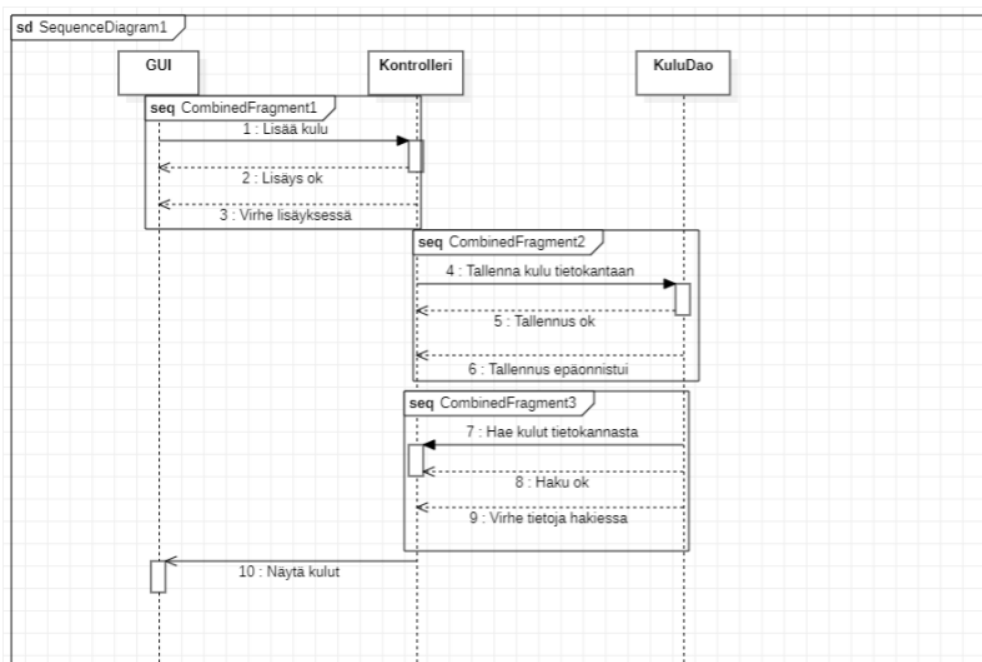


Kaavio 6: Sijoittelukaavio (eng. Deployment diagram).

Kaaviossa 6 on kuvattu sijoittelukaavio budjettisovelluksesta tuotantokäytössä. Kannettavalla on virtuaalinen javan suoritussympäristö, joka suorittaa budjettisovellusta. Budjettisovellus initialisoi ja kommunikoi MariaDB-suoritussympäristön kanssa TCP/IP protokollan välityksellä, joka suorittaa budjettisovelluksesta muodostettua tietokantamallia.

6 Ohjelmiston toiminta

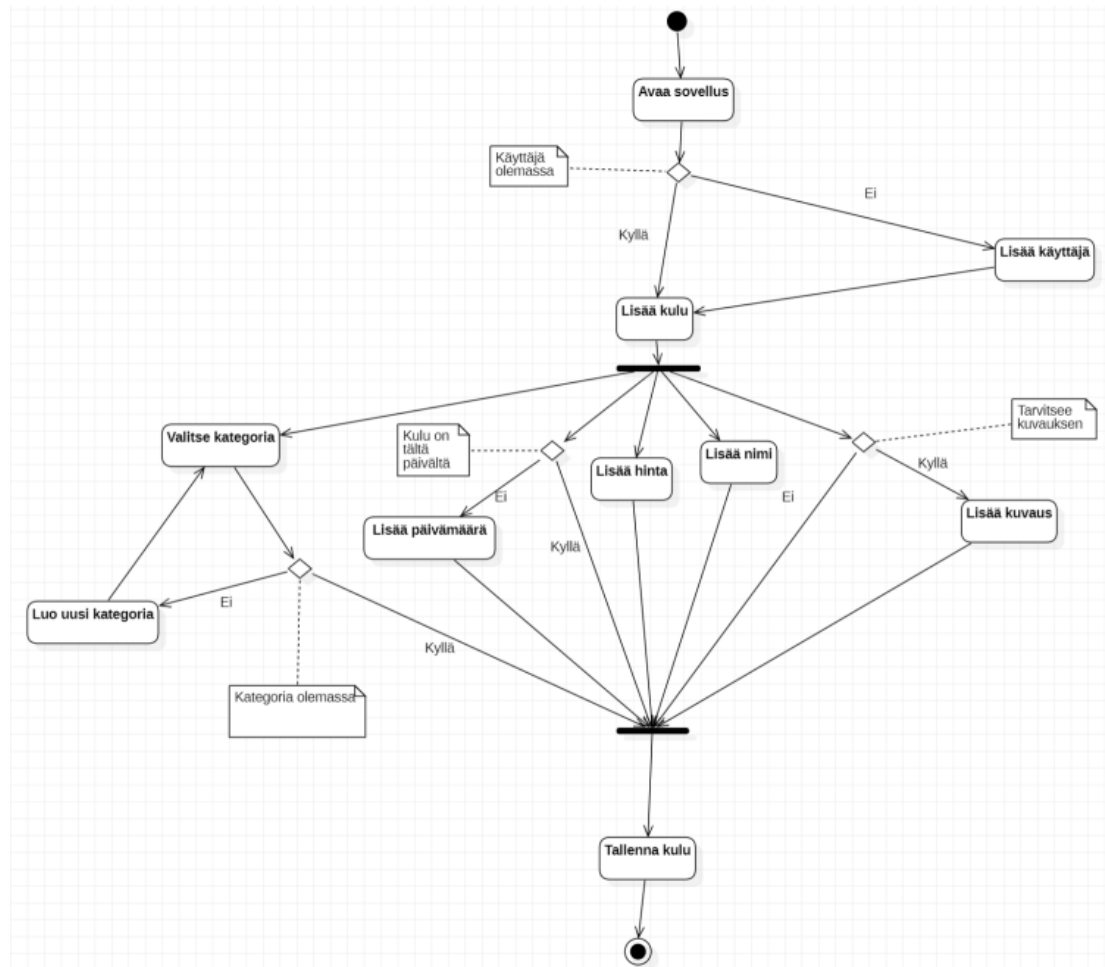
Yksi sovelluksen keskeisin käyttötapaus on kulun lisääminen. Tämä käyttötapaus on kuvattu sekvenssikaaviona kaaviossa 7 ja aktiviteettikaaviona kaaviossa 8.



Kaavio 7: Sekvenssikaavio kulun lisäämisestä.

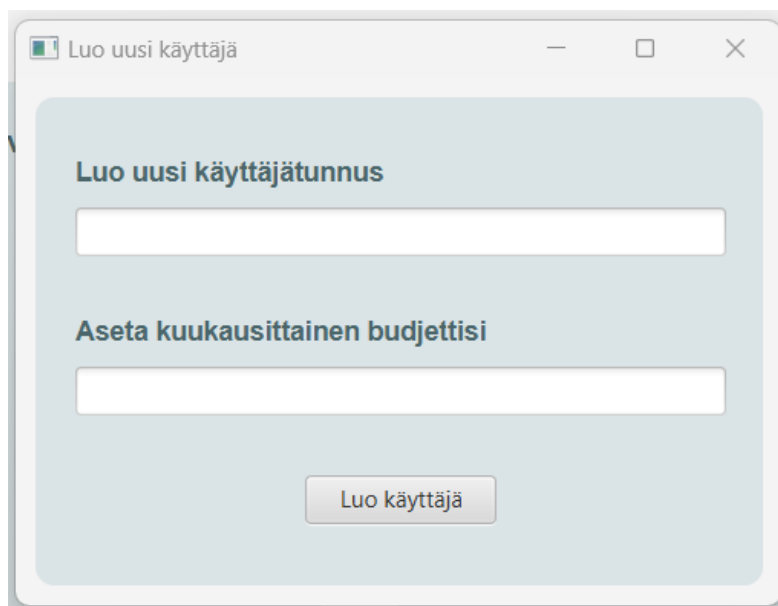
Kuten kaaviosta 7 nähdään, käyttäjä lisää ensin sovellukseen kulun graafisen käyttöliittymän kautta. Graafisesta käyttöliittymästä kulu siirtyy kontrollerille. Jos tässä vaiheessa ei kohdata ongelmia, kontrolleri vie kulun edelleen KuluDAO:lle, joka tallentaa kulun tietokantaan.

Jos kulun tietokantaan tallennus onnistuu, kontrolleri hakee KuluDAO:n avulla tietokannasta päivitetyn listan kuluista. Mikäli mitään virhettä ei tapahdu, lista kuluista siirtyy graafiselle käyttöliittymälle, joka näyttää päivitetyn kululistan käyttäjälle.



Kaavio 8: Aktiviteettikaavio kulun lisäämisestä.

Kaaviosta 8 nähdään, kuinka kulun lisäys tarkalleen tapahtuu käyttäjän näkökulmasta katsottuna. Aivan ensimmäiseksi käyttäjän tulee avata sovellus ja luoda käyttäjä kuvassa 1 näkyvän ikkunan kautta, jos käyttäjää ei ole jo luotu aikaisemmin. Käyttäjälle tulee antaa käyttäjätunnus sekä kuukausittainen budjetti.



Kuva 1: Käyttäjän luonti-ikkuna.

Kun käyttäjä on luotu, käyttäjä pääsee lisäämään kulua. Kulun lisäys tapahtuu Kulut-välilehdellä kuvassa 2 näkyvässä kohdassa. Kululle tulee lisätä ainakin nimi, päivämäärä ja hinta. Halutessaan kululle voi myös valita kategorian. Jos haluttua kategoriaa ei ole olemassa, käyttäjä voi luoda uuden kategorian. Jos kategoriaa taas ei valita ollenkaan, menee kulu yleiseen kategoriaan, joka löytyy sovelluksesta oletuksena. Lisäksi kululle voi vielä lisätä kuvauksen, mutta tämäkin on vapaaehtoinen.



Kuva 2: Kulun lisääminen sovellukseen.

Kun kaikki vaaditut tiedot on täytetty, käyttäjä voi lisätä kulun sovellukseen klikkaamalla "Lisää ostos"-painiketta. Tällöin tieto lisätystä kulusta kulkeutuu kontrollerille ja edelleen KuluDAO:lle, joka tallentaa kulun tietokantaan. Tämän

Toisessa ja kolmannessa sprintissä ohjelmistoon lisättiin ominaisuuksia, jotka mahdollistavan sovelluksen sujuvan käytön ja havainnollistavat käyttäjälle hänen rahan kulutustaan. Käyttäjäprofiilit, diagrammi kulutuksen tarkasteluun kategorioittain ja graafi kulutetun rahan määrään seurantaan ja ennustamiseen lisättiin. Sen lisäksi käyttäjälle luotiin mahdollisuus muokata jo aiemmin sovellukseen syötettyjä käyttäjä- ja kulutietoja. Neljännessä sprintissä sovellukseen lisättiin vielä viimeiset toiminnalliset ominaisuudet, kuten virheilmoitukset ja kategorioiden muokkaus, ja sovelluksen ulkoasu viimeisteltiin.

Viidennessä sprintissä aloitettiin sovelluksen lokalisointi, graafisen käyttöliittymän hiominen, ostos- ja muistilistatoimintojen lisääminen sekä sovelluksen testikattavuuden parantaminen. Valtaosaa näistä tehtävistä jatkettiin viimeiseen eli seitsemänteen sprinttiin saakka. Viimeisessä sprintissä päivitettiin myös sovelluksen dokumentaatiota, kommentoitiin koodia ja tuotettiin Javadoc.

Pääasiassa sprinttiin kuuluvat asiat saatiin valmiiksi sprintin aikana. Poikkeuksen muodostivat laajat kokonaisuudet, jotka viimeisteltiin seuraavien sprintien aikana ja mahdolliset vasta seuraavaan sprinttiin suunnitellut lisätehtävät, jotka aloitettiin jo edellisen sprintin aikana. Saimme jokaisen suunnitellun käyttötapauksen toteutettua projektille varatussa ajassa, joten projektia voi pitää onnistuneena.

Ryhmä kokoontui säännöllisesti sekä Discordissa että koulussa palavereihin, joissa käytiin läpi tehdyt asiat ja suunnitelmat tulevista töistä. Tapaamisten ulkopuolella jokainen ryhmän jäsen ilmoitti Discordiin, kun sai valmiiksi kehittämänsä ominaisuuden, joten sovelluksen edistymisen seuranta oli helppoa sekä mobiilissa että pöytäkoneella. Seuranta tehtiin myös Nektionissa, jossa pidettiin kirjaa työtunneista ja kaikista projektiin sisältyvistä tehtävistä.

8 Tuotteen testaus

Sovelluksessa on JUnit 5 testejä, joilla testataan DAO-, kontrolleri- ja view-pakkauksiin kuuluvien toimintaa. DAO-testeissä testataan sovelluksen vuorovaikutusta tietokannan kanssa ja katsotaan, että tiedon haku, poistaminen, muokkaaminen ja lisääminen onnistuvat. Kontrolleri-testeissä taas katsotaan, että kontrollerin metodit kutsuvat oikeita metodeja DAO-luokista oikeilla parametreilla. View-testeissä taas testataan, että view-luokkien metodit kutsuvat kontrollerista oikeita metodeja oikeilla parametreilla.

Model, Controller ja DAO-pakkauksiin kuuluvat luokkiin saimme toteutettua melko kattavat testit. View-pakkauksen luokkiin taas ei ehditty toteuttaa kovinkaan kattavia testejä. Tämä johtuu kahdesta asiasta: sovelluksen käyttöliittymä tuli valmiiksi vasta projektin loppuvaiheessa ja testien toteuttaminen veikin oletettua enemmän aikaa.

Testaus on automatisoitu Jenkinsin avulla. Sovellus hyödyntää myös JaCoCo-kirjastoa, jonka avulla saadaan tuotettua selkeä raportti testikattavuudesta. Tarkastelimme tätä raporttia säännöllisin väliajoin ja mietimme, millaisia testejä sovellukseen tulisi vielä toteuttaa.

9 Jatkokehitysideat

Sovellukseen saatiin toteutettua kaikki alun perin suunnitellut ominaisuudet ja siitä löytyvätkin tällä hetkellä kaikki käyttäjän kannalta olennaisimmat toiminnot. Halutessa sovellusta voitaisiin kuitenkin vielä kehittää pidemmälle ja lisätä siihen uusia ominaisuuksia.

Sovellukseen voitaisi esimerkiksi lisätä toiminto, joka tarjoaa käyttäjälle tekoälyn generoimia säästövinkkejä. Näissä säästövinkeissä voitaisi ottaa huomioon käyttäjän sovellukseen lisäämät kulut. Tekoäly voisi esimerkiksi huomauttaa, mihin käyttäjällä on kulunut erityisen paljon rahaa ja miten käyttäjä voisi säästää tällä saralla. Jos tätä ominaisuutta haluaisi kehittää vielä entistäkin pidemmälle,

voisi tekoäly jopa ehdottaa, mistä kaupasta tietyn tuotteen saisi ostettua halvemmalla.

Lisäksi sovellukseen voitaisi vielä lisätä enemmän kieliä, jolloin se voisi saavuttaa entistä suuremman yleisön. Tällä hetkellä sovellus on saatavilla vain suomeksi ja englanniksi. Toki myös sovelluksen ulkoasua voisi aina kehittää vielä entistäkin tyylikkäämmäksi ja sovellukseen voitaisi vaikkapa lisätä erilaisia teemoja, joista käyttäjä voi valita itselleen mieluisimman.

10 Yhteenveto

Budjettisovelluksen tarkoituksena on auttaa henkilöä seuraamaan kuukausittaista rahankäyttöään. Tässä dokumentissa on kuvattu sovelluksen rakennetta, toimintaa ja kehitysprosessia sekä tekstimuodossa että kaavioiden ja kuvien avulla. Dokumentista saa käsityksen sovelluksen luokista ja pakkauksista ja niiden kommunikoinnista toistensa kanssa. Myös tietokannan rakenne käydään läpi.

Saavutimme projektille asetetut tavoitteet hyvin. Saimme projektin valmiiksi ajoissa ja kaikki suunnitellut käyttötapaukset toteutettua. Kaikista olennaisimpien ominaisuuksien lisäksi saimme myös toteutettua niin sanotusti ylimääräisiä ominaisuuksia, kuten ostoslistan ja muistilistan. Sovelluksen ulkoasusta saatiin myös tehtyä selkeä ja moderni. Sovellusta voisi halutessa vielä jatkokehittää ja lisätä siihen vielä entistäkin edistyneempiä ominaisuuksia. Tämänhetkisestä sovelluksesta löytyvät kuitenkin kaikki käyttäjän kannalta olennaisimmat ominaisuudet.