



Jami Hämäläinen, Iina Laamo, Ville Yli-Kivistö

HouseholdHero-sovellus

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikan tutkinto-ohjelma

Tekninen dokumentaatio

15.12.2022

Sisällys

1	Johdanto	1
2	Tuotteen vaatimukset	1
2.1	Tarve ja toimintaidea	1
2.2	Vaatimukset	2
2.3	Käsitteet ja määritelmät	4
3	Käyttäjäroolit ja käyttötapaukset	6
4	Ohjelmiston tietomalli	7
5	Ohjelmiston rakenne	8
6	Ohjelmiston toiminta	9
7	Kehitysprosessi ja kehitysvaiheen tekniikat	11
8	Lokalisointi	12
9	Sovelluksen testaaminen	12
10	Yhteenveto	13

1 Johdanto

Tässä dokumentissa kerromme tarkemmin sovelluksestamme, sen kehityksestä ja esittelemme sen rakennetta. Dokumentti on suunnattu sovelluksemme kehitysprosessista kiinnostuneille.

HouseholdHero on sovelluksemme, jolla haluamme helpottaa ihmisten taloudenhallintaa. Kehitimme sovelluksen OTP1-kurssilla ja jatkoimme sovelluksen parissa OTP2-kurssilla toteuttaen lisäominaisuuksia, lokalisoinnin sekä testausta.









HouseholdHero-sovellus mahdollistaa kotitalouden ruokaostosten budjetoinnin ja tuotteiden käytön seuraamisen. Sovelluksella käyttäjä voi luoda budjetin haluamalleen summalle ja haluamalleen aikavälille. Käyttäjä voi lisätä ostamansa elintarvikkeet sovelluksen jääkaappinäkymään, minkä jälkeen sovelluksesta voi kätevästi katsoa mitkä tuotteet ovat käytettävissä olevia ja tuoreita, pian vanhentuvia tai jo vanhentuneita.

2 Tuotteen vaatimukset

2.1 Tarve ja toimintaidea

HouseholdHero-sovellus tekee oman jääkaapin sisällön ja ruokaan käytettävän rahamäärän suunnittelusta ja seuraamisesta helpompaa. Kun asiakas näkee omassa jääkaapissaan olevat tuotteet kätevästi yhdestä näkymästä, hän voi priorisoida pian vanhaksi menevien tuotteiden käyttöä ja näin ehkäistä ruokahävikin syntymistä. Ruokahävikin välttämisessä yhdistyvät ekologisuus ja säästäminen, kun rahaa ei ole turhaan käytetty pois heitettäviin tuotteisiin. Kun asiakas näkee helposti omassa jääkaapissaan jo olevat tuotteet, on vaivattomampaa ostaa kaupasta vain tuotteita, joille on tarve. Lisäksi ruokaan käytettävän budjetin asettaminen ja seuraaminen auttavat säästämään ruokakustannuksissa.

Ruoan kuluttamiseen liittyvä ekologisuus ja harkittu rahankäyttö ovat erityisen tärkeitä nykytilanteessa ruoan hinnan noustua. HouseholdHero-sovelluksesta hyötyvät erityisesti ihmiset, jotka haluavat tehostaa omaa kotitalouden hallintaansa sekä säästää luonnonvaroja ja rahaa. Sovelluksen kohderyhmä on 20-40-vuotiaat ihmiset, joille teknologian käyttäminen on luontaista.

HouseholdHero	Products In Fridge					Categories ▼
	Name	Category		Price	Best Before	Edit
	Apple	Fruits and Berries		0.30	2022-10-24	
	Broccoli	Vegetables		2.57	2022-10-13	
Fridge	Feta Cheese	Dairy		2.99	2022-09-24	
	Budget And Waste					
Add Product	Expired Products In Fridge					Categories ▼
	Name	Category		Price	Best Before	Put To Waste
	Tofu	Plant-based Protein		3.29	2022-09-24	

Kuvio 1. HouseholdHeron visiokuva

HouseholdHero-sovelluksella asiakas voi asettaa ruokaostoksilleen budjetin haluamalleen aikavälille, sekä seurata ja muokata budjettia. Sovelluksen keskeinen osa on jääkaappi, jonka sisällön asiakas näkee, ja jonne asiakas voi lisätä tuotteita. Asiakas voi muokata ja poistaa tuotteita sekä siirtää tuotteita käytetyksi ja hävikkiin.

2.2 Vaatimukset

Vaatimuksiksi asetettiin, että toteutettava sovellus on työpöytäsovellus, joka on tehty Javalla, JavaFX:llä, SceneBuilder -työkalulla ja CSS:llä. Sovellus käyttää MariaDB-tietokantaa.

OTP2-kurssin aikana toteutetun sovelluksen toiminnallisia vaatimuksia ovat:

- **Budjetin asettaminen, muokkaaminen ja poistaminen:**

- Käyttäjä voi asettaa ruokaostoksilleen budjetin haluamalleen aikavälille ja haluamalleen summalle
- Käyttäjä voi muokata budjetin päivämääriä ja suunniteltua summaa
- Käyttäjä voi poistaa budjetin

- **Jääkaappiin liittyvät toiminnot:**

- Käyttäjä voi lisätä tuotteen jääkaappiin
- Käyttäjä voi nähdä mitä tuotteita jääkaapissa on
- Käyttäjä voi järjestää jääkaapin sisältöä eri ominaisuuksien perusteella
- Käyttäjä voi muokata jääkaappiin lisättyä tuotetta
- Käyttäjä voi poistaa tuotteen kokonaan
- Käyttäjä voi siirtää tuotteen käytetyksi
- Käyttäjä voi siirtää tuotteen hävikkiin

- **Budjetin ja hävikin seuraaminen:**

- Käyttäjä voi tarkastella nykyistä tai mennyttä budjettia
- Käyttäjä voi seurata suunniteltua, käytettyä ja jäljellä olevaa budjettia

- Käyttäjä voi seurata budjetin jakautumista eri kategorioihin visuaalisesta piirakkadiagrammista
- Käyttäjä voi seurata budjetin aikana ostettujen tuotteiden tietoja

OTP2-kurssin aikana toteutetun sovelluksen laadullisia vaatimuksia ovat:

- Sovellus on helppokäyttöinen sovelluksen kohderyhmälle
- Sovelluksen tietoturva on otettu huomioon siten, että tietokannan käsittelyssä on käytetty parametrisoituja kyselyjä, joten SQL-injektioita ei pysty tekemään
- Suorituskyvyltään sovellus reagoi nopeasti erilaisiin käyttäjän syötteisiin
- Sovellus on rakennettu huomioimaan erilaiset poikkeukset sekä varoittamaan käyttäjää mahdollisista virheellisistä syötteistä

2.3 Käsitteet ja määritelmät

Sovelluksemme on toteutettu englanniksi.

Fridge on jääkaappi, eli sovelluksen päänäköymä, jossa on Products In Fridge -taulukko ja Add Product –painike.

Products In Fridge on jääkaapin osio, jossa näkyvät taulukkomuodossa jokaisen jääkaappiin lisätyn tuotteen nimi (Name), kategoria (Category), kategorian ikoni, hinta (Price (€)), parasta ennen -päivämäärä (Best Before), statusikoni, tuotteen muokkauspainike (Edit), Put To Used –painike ja Put To Waste -painike.

Add Product -painikkeella käyttäjä avaa näkymän, jossa hän voi syöttää lisättävän tuotteen tiedot.

Kategorian ikoni on määritelty kullekin kategorialle valmiiksi.

Name on tuotteen nimi, jonka käyttäjä voi tuotteelle itse kirjoittaa.

Category on tuotteen kategoria. Tuotekategorioita on 23, ja ne on määritelty sovellukseen valmiiksi. Käyttäjä voi valita kategorian, johon tuote kuuluu.

Price (€) on tuotteen hinta euroina, jonka käyttäjä voi tuotteelle itse kirjoittaa.

Best Before on parasta ennen -päivämäärä, jonka käyttäjä voi asettaa tuotteelle.

Statusikoni näyttää käyttäjälle, miten lähellä vanhenemista tuote on.

- Punainen ympyrä tarkoittaa, että parasta ennen päivä on jo mennyt.
- Keltainen kolmio tarkoittaa, että tuote vanhenee kahden päivän sisällä.
- Vihreä neliö tarkoittaa, että tuote vanhenee kolmen tai useamman päivän päästä.

Edit-sarake Products In Fridge -osiossa sisältää kullekin tuotteelle painikkeen, jolla tuotetta pääsee muokkaamaan.

Put To Used -sarake Products In Fridge -osiossa sisältää kullekin tuotteelle painikkeen, jolla tuotteen pääsee siirtämään käytetyksi.

Put To Waste -sarake Products In Fridge -osiossa sisältää kullekin tuotteelle painikkeen, jolla tuotteen pääsee siirtämään hävikkiin.

Budget on sovelluksen budjettinäköymä, jossa käyttäjä voi seurata kaikkia luomiaan budjetteja. Näkymässä on valikko budjetin valitsemelle, valitun budjetin tiedot (Budget Information), taulukko budjetin aikana ostetuista tuotteista (Products Purchased During Budget), piirakkakaavio budjetin jakautumisesta (Budget Distribution) ja painike budjetin muokkaukselle (Edit Budget).

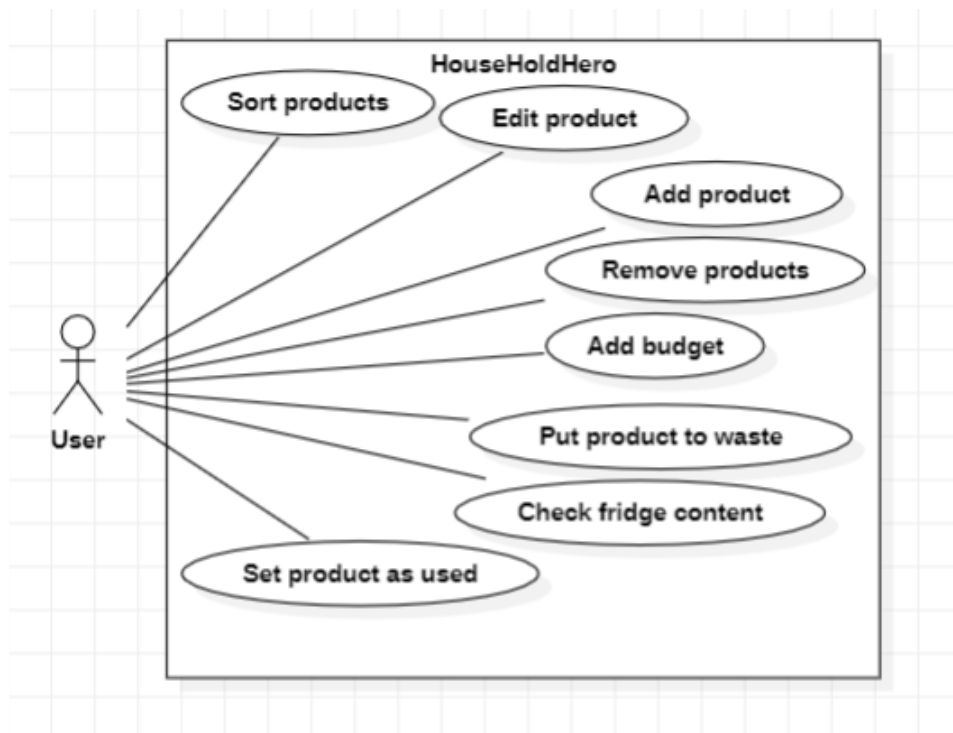
Budget Information osiossa valitusta budjetista näkyy aloituspäivämäärä (Start Date), päättymispäivämäärä (End Date), suunniteltu budjetti (Planned Budget), käytetty budjetti (Spent Budget) ja jäljellä oleva budjetti (Remaining Budget).

Budget Distribution on piirakkakaavio, jossa näkyy, miten budjetin rahamäärä on jakautunut.

- Vihreä osio edustaa jääkaapissa oleviin tuotteisiin kulunutta euromäärää (Fridge).
- Keltainen osio edustaa käytettyihin tuotteisiin kulunutta euromäärää (Used).
- Punainen osio edustaa hävikkiin kulunutta euromäärää (Waste).
- Sininen osio edustaa jäljellä olevaa budjettia (Remaining Budget).

Products Purchased During Budget on taulukko, jossa näkyvät kaikki valitun budjetin aikana ostetut tuotteet ja niiden nimi (Name), kategoria (Category), hinta (Price), parasta ennen -päiväys (Best Before) ja tuotteen tila (Status).

3 Käyttäjäroolit ja käyttötapaukset



Kuvio 2. HouseholdHeron käyttötapauskaavio

Sovelluksellamme **käyttäjä** (User) on ainoa käyttäjärooli.

Sovelluksemme käyttötapauksia ovat:

Tuotteiden järjestäminen (Sort products): Käyttäjä voi järjestää jääkaapin sisältöä kuvaavan listan eri ominaisuuksien perusteella.

Tuotteen muokkaaminen (Edit product): Käyttäjä voi muokata jääkaapissa olevaa tuotetta.

Tuotteen lisäys (Add product): Käyttäjä voi lisätä jääkaappiin tuotteita.

Tuotteen poistaminen (Remove products): Käyttäjä voi poistaa tuotteen jääkaapista, jolloin se menee hävikkiin.

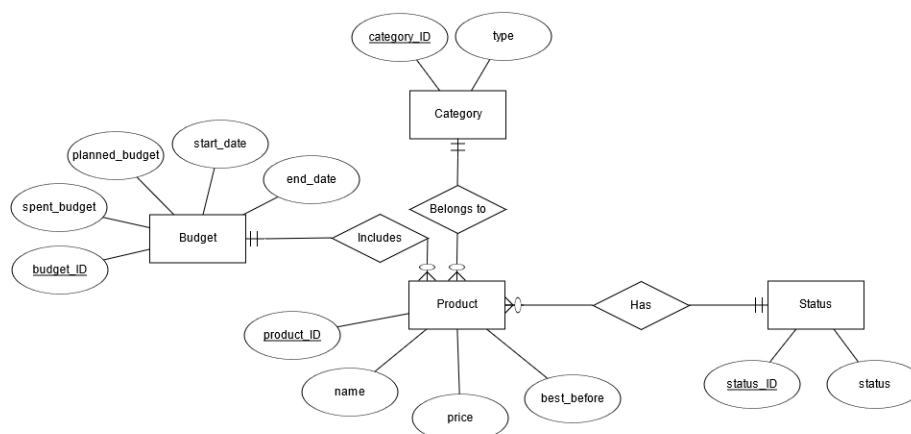
Budjetin luominen (Add budget): Käyttäjä voi sovelluksessa luoda budjetin.

Laita tuote hävikkiin (Put product to waste): Käyttäjä voi laittaa tuotteen hävikkiin.

Jääkaapin sisällön tarkistus (Check fridge content): Käyttäjä voi seurata jääkaapin sisältöä listalta.

Merkitse tuote käytetyksi (Set product as used): Käytettyään tuotteen, käyttäjä voi merkitä sen käytetyksi.

4 Ohjelmiston tietomalli



Kuvio 3. HouseholdHeron ER-kaavio

Sovelluksessa käytettävä tietokanta koostuu neljästä eri taulusta.

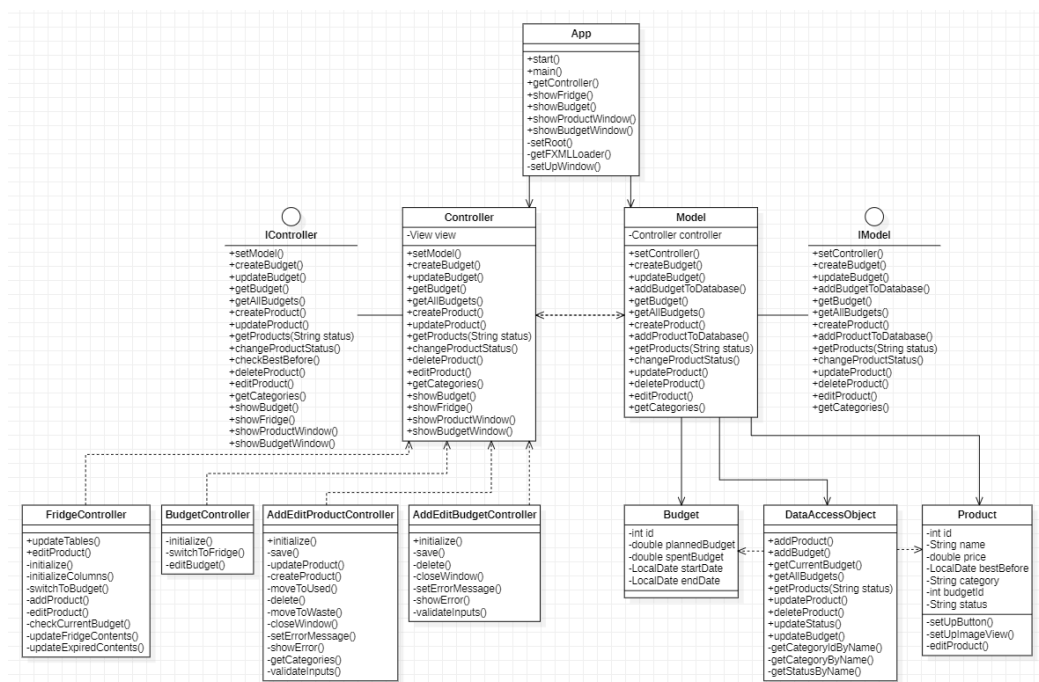
Budget-taulussa on kaikki käyttäjän lisäämät budjetit. Budjetti voi sisältää useita tuotteita.

Category-taulussa on 23 ennalta määritettyä kategoriää.

Status-taulussa on kolme ennalta määritettyä tilaa (Fridge, Used ja Waste).

Product-taulussa on kaikki käyttäjän lisäämät tuotteet. Jokainen tuote on osa yhtä budjettia. Tuotteilla on myös kategoria (Category) ja tila (Status).

5 Ohjelmiston rakenne



Kuvio 4. HouseholdHeron luokkakaavio

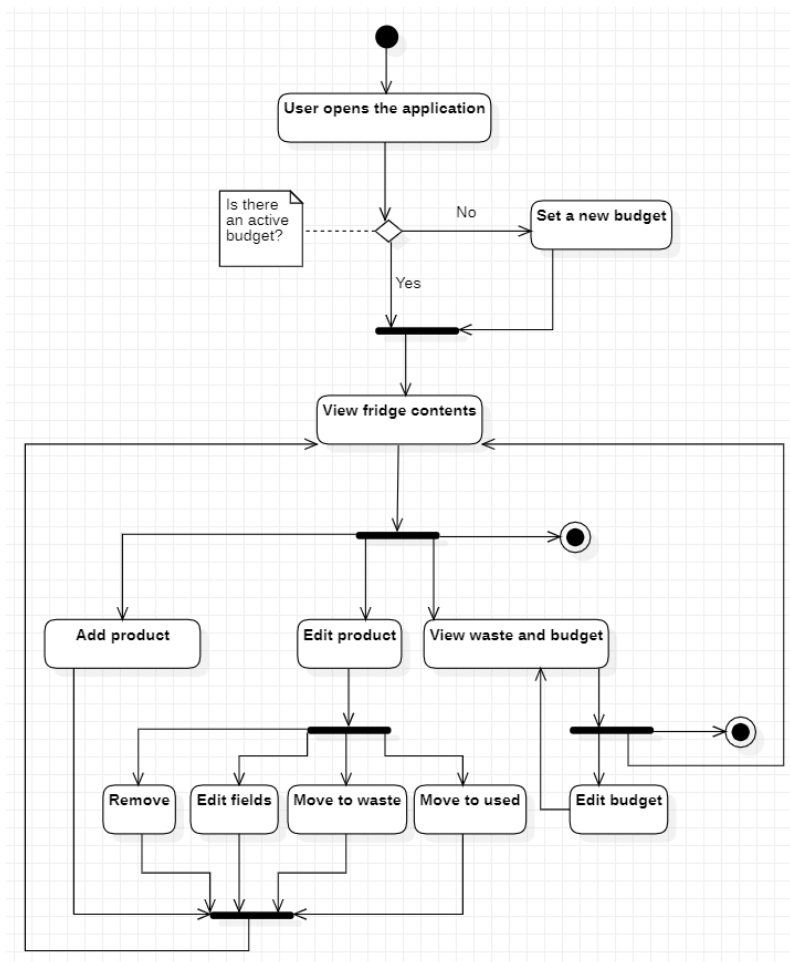
Sovellus käyttää MVC-mallia, eli sen logiikka on jaettu kolmeen eri osaan (Model, View ja Controller). Sovellus käynnistyy App-luokasta, joka myös luo

Controller- ja Model-luokat. FridgeController, BudgetController, AddEdit-ProductController ja AddEditBudgetController ovat FXML controllereita. Model-luokassa luodaan Budget- ja Product-olioita. DataAccessObject käsittelee tietokantaa.

6 Ohjelmiston toiminta

Havainnollistamme sovelluksen toimintaa aktiviteetti- ja sekvenssikaaviolla.

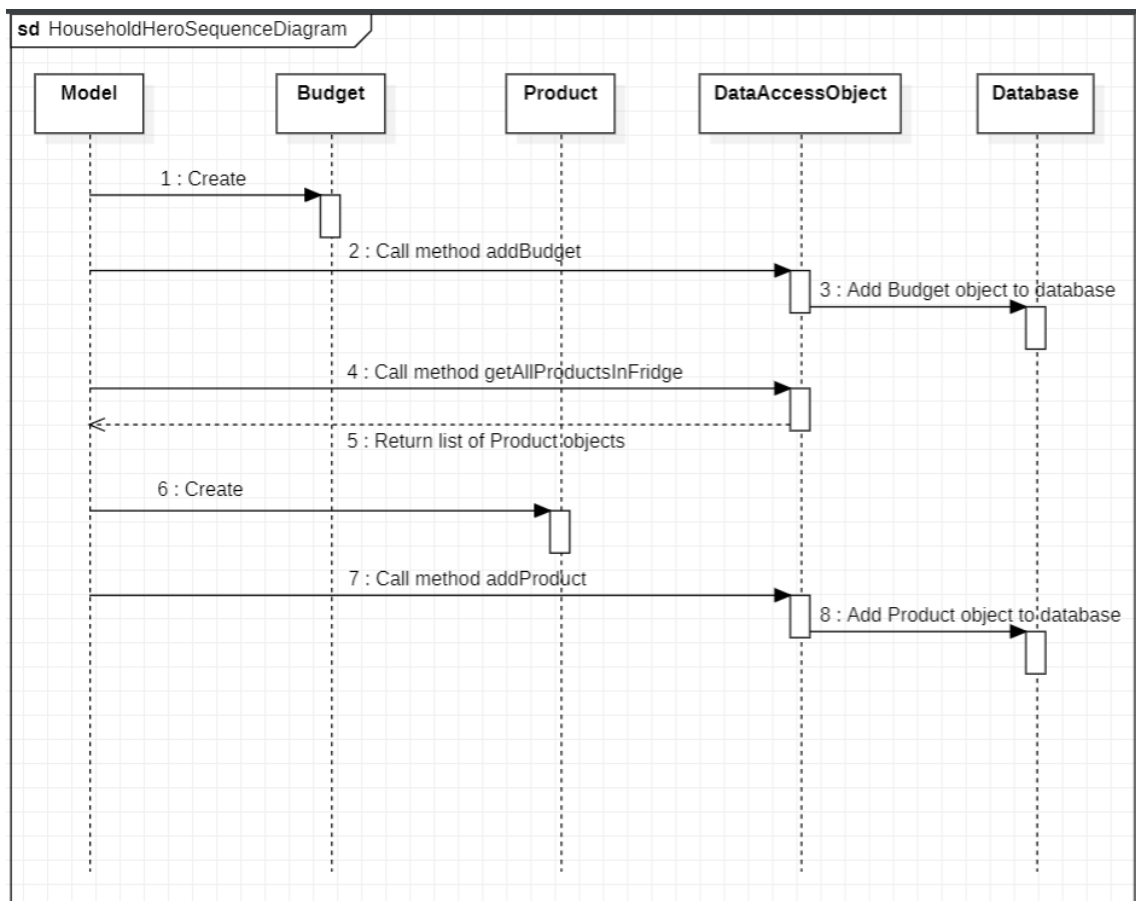
Aktiviteettikaaviossa kuvataan sovelluksen toiminnallisuutta käyttäjän näkökulmasta.



Kuvio 5. HouseholdHeron aktiviteettikaavio

Kun sovellus avataan, se tarkistaa onko tietokannassa aktiivista budjettia. Jos sellaista ei ole, uuden budjetin luontinäköymä aukeaa. Kun budjetti on asetettu, aukeaa jääkaappinäköymä. Tästä näköymästä käyttäjä voi lisätä uuden tuotteen, muokata olemassa olevia tuotteita tai siirtyä budjettinäköymään. Tuotteen muokausnäköymässä käyttäjä voi poistaa tuotteen tietokannasta pysyvästi, muokata sen kenttiä tai siirtää sen joko hävikkiin tai käytetyksi. Budjettinäköymässä voi tarkastella ja muokata kaikkien budjettien kenttiä, sekä nähdä listan niiden aikana ostetuista tuotteista.

Sekvenssikaaviossa esitetään käyttötapaukset "Add budget", "Check fridge content" ja "Add product".



Kuvio 6. HouseholdHeron sekvenssikaavio

Käyttötapauksessa "Add budget" Model luo ensin Budget-olion. Model kutsuu DataAccessObject:in addBudget()-metodia, joka lisää Budget-olion tietokantaan.

Käyttötapauksessa "Check fridge content" Model kutsuu DataAccessObject:in getAllProductsInFridge()-metodia. DataAccessObject palauttaa listan Product-olioista.

Käyttötapauksessa "Add product" Model kutsuu DataAccessObject:in addProduct()-metodia, joka lisää Product-olion tietokantaan.

7 Kehitysprosessi ja kehitysvaiheen tekniikat

Sovelluksemme suunnittelu alkoi ideoinnista, jolla pyrimme keksimään hyödyllisen ja laajennettavissa olevan sovelluksen. Ideasta yhteisymmärrykseen päädyttyämme, aloimme kaavioiden avulla luoda raameja sovelluksellemme.

Alussa kehitimme myös käyttäjätarinoita, päätimme ohjelmointikielen ja kehitimme tarkempaa suunnitelmaa sovelluksen ominaisuuksista. Sovelluksen tyylyttelystä vastuussa oleva tiimin jäsen loi Figmalla wireframen sovelluksemme näkymistä sekä kehitti sovelluksellemme logon.

Hyvän suunnittelun jälkeen alkoi sovelluksen kehittäminen. Loimme Maven-projektin ja projektille Git-repositorion. GitHubissa loimme jokaiselle sovelluskehittäjälle oman branchin, jotta kaikki voisivat kehittää sovellusta omaan tahtiin. Sovelluksen kehittäminen oli karkeasti jaettu tyylyttelyä, logiikkaa ja tietokantaa painottaviin osiin.

Sovelluksen kehitystyössä hyödynsimme Nektionia, jossa pidimme kirjaa tehdyistä työtunneista ja sovelluksen kehitysvaiheista. Ohjelmointikieleksi valitsimme Javan. Sovelluskehitysympäristönä toimi Eclipse, jossa loimme Maven-projektin. Näiden lisäksi hyödynsimme Scene Builderia tyylyttelyssä. Toteutimme ohjelmistolle MariaDB-tietokannan, jota käsitelimme SQL-kielellä. Jotta koodimme pysyi ajan tasalla, käytimme Gitiä ja GitHubia versionhallintaan.

Projektikokonaisuus yhdistettiin Discordilla, jota käytimme ryhmän jäsenien väliseen kommunikoimiseen.

8 Lokalisointi

Lokalisoimme sovelluksemme niin, että sitä on mahdollista käyttää kahdella kielellä: englanniksi ja iiriksi. Olimme toteuttaneet sovelluksen englanniksi ja iiri valikoitui sovelluksen toiseksi kieleksi, koska saimme OTP2-kurssin aikana kuvitteelliseksi asiakkaaksemme irlantilaisen tukkuliikkeen. Toteutimme lokalisoinnin niin, että kieliä on helppo tarvittaessa lisätä.

Lokalisointityö alkoi lokalisointi-Excelin tuottamisella. Kokosimme Exceliin kaikki sovelluksemme tekstit ja teimme niille iirinkieliset käännökset.

Excelin pohjalta tuotimme TextProperties-tiedostot, joihin asetimme tekstien avain-arvo-parit. Avaimet ovat kummassakin TextProperties-tiedostossa samat, mutta englanninkielisessä tiedostossa arvot ovat englanniksi ja iirinkielisessä tiedostossa iiriksi.

Tämän jälkeen asetimme avaimet kaikkien tekstien tilalle sovelluksen näkymissä joko Scene Builderin avulla tai suoraan FXML-tiedostoihin. Myös kontroloreissa oli erilaisia virheviestitekstejä, jotka täytyi korvata avaimilla.

Lokalisoinnin viimeisenä vaiheena teimme sovelluksen näkymiin kielivalikot (Language Choice Boxes) sekä toiminnallisuuden kielen vaihtamiselle englannin ja iirin välillä.

9 Sovelluksen testaaminen

Sovellustamme olemme testanneet JUnit-testeillä, sekä testitapauksilla. JUnit-testejä on tietokannan ja sovelluksen välisellä kommunikointiluokalla eli DataAccessObject-luokalla, sekä FridgeController-, BudgetController-, Localiser- ja CategoryLocaliser-luokilla. Testitapauksia on tehty sovelluksen eri

toiminnallisuuksille. Jenkins ajaa JUnit-testejämme aina, kun GitHub-repositoriomme päivittyy. Jostain syystä emme saaneet testejä, jotka käyttävät JavaFX-komponentteja, toimimaan Jenkinsissä. Nämä testit kuitenkin toimivat, kun niitä ajetaan manuaalisesti Eclipsessä.

Testitapaukset ovat käyttäjälle luotuja tapauksia, joissa on yksinkertainen tehtävä. Käyttäjän tulee toteuttaa tehtävä ja tehtävän suorittamisesta kirjataan tapahtumat mahdollisimman tarkasti. Testitapauksia on yhteensä seitsemän, joissa testattiin budjetin lisäys ja muuttaminen, tuotteen lisäys, muuttaminen ja "status"-kentän kuvakkeet. Viimeinen testitapaus testasi kielen vaihtamista. Kaikki testitapaukset onnistuivat ongelmitta.

Test Case ID	Bud- get_01	Test Case Description	Test creating new budget		
Tester's Name	Ville	Date Tested	December 5, 2022	Test (Pass/Fail/Not executed)	Case Exe- Pass
S #	Prerequisites:		S #	Test Data	
1	The user has <u>no</u> active budget		1		
2			2		
3			3		
4			4		
<u>Test Scenario</u>	Create budget				
Step #	Step Details	Expected Results	Actual Results		Pass / Fail / Not executed / Sus- pended
1	Open software	Software should open with add new budget view	As Expected		Pass
2	Select starting date, ending date and fill in Planned Budget	Planned Budget, starting and ending day should be visible	As Expected		Pass
3	Click Save	Budget is active	As Expected		Pass

Kuvio 7. Testitapaus uuden budjetin luomisesta

10 Yhteenveto

Tässä dokumentissa olemme kattavasti esitelleet sovelluksemme vaatimukset ja kehitysprosessin, sekä käyttäytymiskaavioiden ja rakennekaavioiden avulla

sovelluksemme käyttäjäroolit ja käyttötapaukset, tietomallin, rakenteen sekä toiminnan.

Koko projektin aloitusvaiheessa ideoimme laajasti millaisia toimintoja HouseholdHero-sovellukseen voisi sisällyttää, ja tiedostimme, että näistä ideoista riittäisi sisältöä sekä OTP1- että OTP2-kurssille. HouseholdHero-sovelluksen keskeinen tavoite on tehdä oman jääkaapin sisällön ja ruokaan käytettävän rahamäärän suunnittelusta ja seuraamisesta helpompaa. OTP1-kurssin aikana toteutimme jääkaappiin liittyvät toiminnot ja budjettiin liittyvät toiminnot.

Jääkaappiin liittyviksi toiminnoiksi valitsimme jääkaapin sisällön näyttämisen, jääkaapin sisällön järjestämisen eri ominaisuuksien perusteella sekä tuotteen lisäämisen, muokkaamisen, poistamisen sekä käytetyksi ja hävikkiin siirtämisen. Budjettiin liittyviksi toiminnoiksi valitsimme budjetin asettamisen, muokkaamisen ja seuraamisen. Saavutimme tavoitteemme toteuttaa nämä toiminnallisuudet ja näyttää ne käyttöliittymässä.

OTP2-kurssin aikana toteutimme sovellukseen OTP1-kurssilta jääneet jatkokehitysideat. Muokkasimme jääkaappinäkymää niin, että kaikki tuotteet ovat yhdessä taulukossa, eivätkä enää erikseen tuoreiden ja vanhentuneiden tuotteiden taulukoissa. Lisäsimme jääkaappinäkymään tuotteille statukset, joista näkee, onko tuote vielä tuore, vanhenemassa alle kahden päivän päästä vai jo vanhentunut.

Kehitimme uuden budjetinäkymän, jossa voi tarkastella nykyistä tai jotakin menneistä budjeteista. Budjetinäkymässä voi tarkastella budjetin tietoja, mihin kategorioihin budjetti jakautuu sekä mitä tuotteita budjetin aikana on ostettu. Budjetin tietoina näkyvät budjetin alku- ja päättymispäivämäärä (Start Date, End Date) sekä suunniteltu budjetti (Planned Budget), käytetty budjetti (Spent Budget) ja jäljellä oleva budjetti (Remaining Budget).

Budjetin jakautuminen eri kategorioihin näytetään visuaalisesti piirakkadiagrammin avulla. Piirakkadiagrammissa on budjetin jakautumiselle kategoriat Jääkaapissa (Fridge), Käytetty (Used), Hävikki (Waste) ja Jäljellä oleva budjetti

(Remaining Budget). Budjetin aikana ostetut tuotteet (Products Purchased During Budget) näytetään taulukossa, jossa on tuotteen kategorian kuva, tuotteen nimi (Name), tuotteen kategoria (Category), hinta euroina (Price (€)), parasta ennen -päivämäärä (Best Before) ja tuotteen tila (Status). Budjettinäköymän taulukossa tuotteiden tilat ovat Jääkaapissa (Fridge), Käytetty (Used) ja Hävikki (Waste). Budjettinäköymän Edit Budget -painikkeesta voi siirtyä muokkaamaan budjettia.

Myös tavoitteet tietokannan suhteen täyttyivät, eli tietokantayhteys toimii, sekä tietoja haetaan, lisätään, päivitetään ja poistetaan tietokannasta. Kaikki tiimin jäsenet osallistuivat aktiivisesti suunnitteluun ja kehitystyöhön projektin aikana, eli saavutimme tavoitteemme toimivasta tiimityöstä.

Jatkokehitysideoina meillä on kauppalistan lisääminen sovellukseen, kuittien tai tuotteiden skannaus ja kotitalouden muiden kulujen lisäys budjetointiin. Kauppalista toimisi sovelluksen sisäisenä listana, johon olisi helppo lisätä tuotteita. Kuittien tai tuotteiden skannaus lisäisi tuotteen suoraan sovellukseen, joka nopeuttaisi sovelluksen käyttöä huomattavasti. Budjetin sisältöä voisi laajentaa niin, että sinne voisi lisätä ruoan lisäksi muita kotitalouden kuluja, kuten sähkö- ja vesilaskut.