Hiilikori

Jani Repo jani.repo@student.lut.fi, 0600539
Aino Solin aino.solin@student.lut.fi, 0587335
Vesa Ylämäki vesa.ylamaki@student.lut.fi, 0507933

Kuvaus ohjelmasta

Hiilikori ohjelman avulla käyttäjä voi suunnitella itselleen oman CO2 tavoitteiden mukaisen päivittäisen ateriakokonaisuuden ja tarkastella tekemiään valintoja Suomen keskiarvoon veratillen. Ohjelman ominaisuuksia:

- Käyttäjäprofiili kirjautumistietojen perusteella
- Ruoka-annoksien lisääminen päivittäiseen ruokalistaan
- Ruoka-annoksien sisältämien ainesosien tuottama CO2 ilmoittaminen
- Ostoslistan tekeminen suunniteltujen annoksien pohjalta
- Ruoasta aiheutuvan CO2 lukeman keskiarvon näyttäminen ohjelmassa, johon käyttäjä voi verrata omaa päivittäistä ruokalistaansa
- Tehtyjen ruokalistojen historiatietojen tarkastelu erilliseltä sivulta
- Omien tietojen päivitys

Tekijät

- Aino – UI suunnittelu ja toteutus, UI-backend integrointia video
- Jani – Backend ja UI-backend integrointia
- Vesa – Backend

Ohjelman toteutus

- Millaisella teknisellä alustalla ohjelma toimii?
    - Android API28 toteutettu, testattu foldable + Nexus 6p emulaattorilla
- Kirjastoriippuvuudet:

  androidx.appcompat:appcompat:1.1.0
  androidx.core:core:1.3.0
  com.google.android.material:material:1.1.0
  androidx.constraintlayout:constraintlayout:1.1.3
  androidx.navigation:navigation-fragment:2.2.2
  androidx.navigation:navigation-ui:2.2.2
  androidx.lifecycle:lifecycle-livedata-ktx:2.2.0
  androidx.lifecycle:lifecycle-viewmodel-ktx:2.2.0
  com.google.code.gson:gson:2.8.6
  com.github.PhilJay:MPAndroidChart:v3.1.0
  com.ramijemli.percentagechartview:percentagechartview:0.3.0
  com.android.volley:volley:1.2.0
  BCrypt module (Damien Miller)
- Mitä työkaluja on käytetty?
    - Ryhmätyökalut
        - Trello, git, Teams, Whatsapp
    - Ohjelmistokehitystyökalut
        - Android Studio, git, Figma
    - Dokumentoitityökalut
        - Word, Visio, draw.io

Luokkakaavio

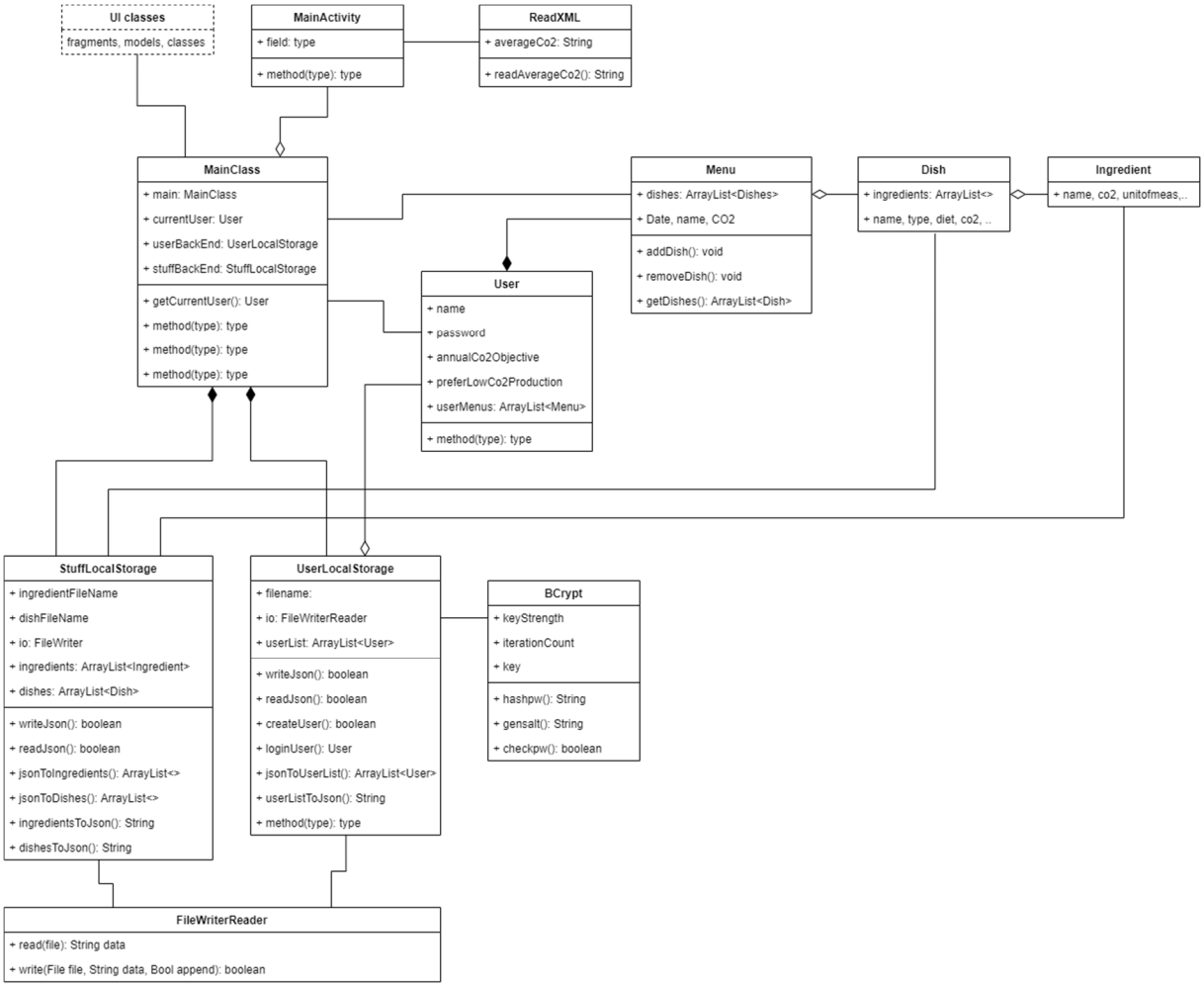

Figure 1 - Hiilikori luokkakaavio, oleellisimmat metodit ja parametrit

Toteutetut ominaisuudet

Ominaisuus	Perustelut	Pisteet
Olio-ohjelmoitu	Perustuu OO	-
Vähintään viisi erilaista luokkaa & oliota (käyttöliittymäluokkia eilasketa)	12 luokkaa, mutta toteutus poikkesi hieman suunnittelusta koska tiedosto accessit haluttiin pitää erossa UI luokista ja jälkikäteen ajatellen ei ehkä paras ratkaisu.	-

Vähintään yhden APIn käyttö, esim. Ilmastodieetti: https://ilmastodieetti.ymparisto.fi/ilmastodieetti/swagger/ui/index	CO2 referenssi syötettyjen tietojen pohjalta	-
Sovellus tallentaa käyttäjän toiminnan (käyttäjän syöttämät arvot / tulokset) logiin (JSON, XML jne.)	Käyttäjätiedot, raaka-aineet ja annokset tallennetaan: User.json Ingredients.json Dishes.json	-
Logia on mahdollista tarkastella (puhtaana tekstinä, graafisilla käppyröillä jne.), eli voidaan tutkia arvojen (esim. oma massa) kehitystä kirjausten edetessä.	Historiasivulla näytetään ruokalistojen toteutunut CO2	-
Ohjelma on rakennettu hyvin suunnitelluista UI-komponenteista	UI toimii lähes täydellisesti ja on johdonmukainen. Integroitu toimivasti backendiin. Paljon erilaisia elementtejä käytetty.	5
Kirjautuminen applikaatioon	Ohjelmaan pääsee kirjautumaan syöttämällä omat tiedot. Tiedot lisättävä ensin.	3
Sovelluksella voi olla useampi käyttäjä (ja niiden luominen), tietojen tallennus järkevästi jonnekin	Käyttäjiä voi luoda ”Rekisteröi käyttäjä” sivulla	3
Kirjautumisen salasana noudattaa hyvän salasanan sääntöjä (sisältää vähintään yhden numeron, erikoismerkin, ison ja pienen kirjaimen, on vähintään 12 merkkiä pitkä)	Toteutus regexpillä UserLocallStoragessa, tosin vain 6 merkkiä vaaditaan!	2
Salasanan tallennus käyttää jonkinlaista hash-menetelmää ja suolausta (esim SHA-512 + salt)	Toteutus tehty BCrypt moduulin avulla jonka avulla siis hash + salt salasana tallennetaan tekstitiedostoon laitteelle salattuna	2
Ohjelmaan on mahdollista syöttää perustiedot (esim. pituus, paino, ikä(/syntymävuosi), kuva, asuinkunta) käyttäjästä ja näitä arvoja käytetään jossakin	Käyttäjän suosima ruokavalio tallennetaan käyttäjän rekisteröinti-ikkunassa ja tätä käytetään ilmastodieettikyselyssä	2
Ohjelma kerää käyttäjän massan kehityksestä dataa ja näyttää muutokset graafisesti havainnollistaen ruudulla	UI:ssa graafille placeholder historia sivulla. Myös annoksien kertymä ja vertailu omaan päivän tavoitteeseen näytetään ruudulla.	3
Asynkronisten HTTP-kutsujen käyttö dataa haettaessa 2 pistettä	Kysely asetetaan Volleyn tekemään jonoon ja response odotetaan asynkronisesti?	2?
Fragmenttien hyödyntäminen aktiviteettien sijasta käyttöliittymiä rakennettaessa	Sivuvalikko toimii periaatteella Main Activity + fragmentit	2

Scoped storagen käyttäminen tiedon tallennuksessa (ei vaadi käyttäjän myöntämiä oikeuksia laitteen massamuistiin, vaan toimii omassa “hiekkalaatikossaan”)	Toimii applikaation omalla alueella	2
Responsiivinen käyttöliittymä	Testattu foldable, Pixel 3, Nexus 6 ja Nexus 5 laitteilla	2
Jokin oma hieno ominaisuus	Ruokapankki ja ostolista ominaisuudet. Joissakin moduuleissa pyritty hyvään virheenkäsittelyyn 😊	2
Summa		13 + 30 = 43

Työmäärät

Tekijä	Tehtävät	Tunnit
Aino	UI, backend-UI integrointi, testaus, video	~40h?
Jani	Backend, backend-ui integrointi, testaus, dokumentaatio	30-50h?
Vesa	Backend, alustava luokkakaaviosuunnitelma	15-25h?
Summa		85-105

Mitä opin harjoitustyöstä?

Jani: Kannattaa tehdä oikea tietokanta eikä käyttää json tiedostoja tai käyttää jotain kolmannen osapuolen palveluja esim. firebase tai aws. Valmiita kirjastoja löytyy androidille aika hyvin ja niillä voi toteuttaa kohtuullisella vaivalla esim. salasanojen kryptauksen. Salasanojen hashauksesta opin paljon. Liika olio-ohjelmointi ja rajojen vetäminen hidastaa välillä kehitystä, varsinkin jos tulee suunnanmuutoksia. Esim. UI luokat haluttiin keskustelevan MainClassin välityksellä muiden luokkien kanssa ja tämä oli aluksi hyvä ajatus ja tässä on hyviäkin puolia. Ominaisuuksien lisääminen vaatii kuitenkin päivitykset myös ns. rajapintaan. Lopputuloksen näkee nyt melko sekavana luokkakaaviona! Virheiden käsittelyn tekeminen ilman keskitettyjä toimintoja ja käytänteitä on aika hidasta ja tekee koodista vaikeasti luettavaa eli tämäkin kannattaisi suunnitella etukäteen.

Aino: Kurssin alussa ei olisi tullut mieleenkään, että loppujen lopuksi pystyisi koodaamaan Android sovelluksen. Minulla oli lähinnä vain muutamia funktionaalisen / proseduraalisen ohjelmoinnin kurssien perusteita takana (R ja Python) sekä vähän tietokantaosaamista, mutta ensikosketus olio-ohjelmointiin ja softakehitykseen tuli tällä kurssilla. Tämä on osasyy siihen, miksi tunteja kului melko

paljon, sillä Java ja OOP oli itselle todella uutta. Harjoitustyössä pääsin oppimaan hyvin laajasti erilaisia asioita, kuten UI/UX suunnittelua ja siihen liittyvien työkalujen käyttöä, miten gittiä käytetään ryhmätyössä, ja tietenkin olio-ohjelmointia, mutta opin myös ohjelmistokehityksen eri vaiheista sekä Android sovelluksen koodauksen perusteet.

Vesa: Java Android on hyvä käyttö- ja kehitysjärjestelmä. Android studio vie paljon RAM:ia. Backend suunnittelu on haastavaa ilman Java android kokemusta.

Palaute harjoitustyöstä (vapaaehtoinen)

- Mitkä ominaisuudet / toiminnot olivat helppoja / vaikeita toteuttaa?
- Oliko jokin asia aivan syvältä?
- Oliko jokin asia todella hyvää tässä työssä?
- Mitä toivoisit ensi vuoden harjoitustyöhön?

Aino:

- UI-komponenttien rakentaminen sekä suunnittelu oli itselleni mieluisaa, varsinkin kun siihen pääsi muutaman kokeilun jälkeen paremmin sisälle. Viikkotehtävät tein lähinnä käsin vetämällä komponentteja ruudulle, mutta harjoitustyössä oli pakko ymmärtää XML-koodia ja miten eri layoutit (Linear, Relative, Constraint) toimivat keskenään että komponentit saa nätisti alignattua toistensa kanssa. Vaikein osuus varmasti oli backend puolella saada tiedot järkevästi tallennettua ja että ne saa tallennettua käyttäjäkohtaisesti, tämän osan koodaukseen en tosin osallistunut, mutta se vaikutti vaikeimmalta.
- Todella hyvää oli nähdä se, miten palaset loksahtelivat yhteen ja lopuksi saimme koodattua jokseenkin järkevän sovelluksen, mistä saisi jatkojalostamalla varmasti hyödyllisenkin sovelluksen. Vaikeutta tuotti aiheen keksiminen harjoitustyölle ja löytää sitä tukevaa dataa, sillä varsinaista ”CO2 pankkia”-ruoka-aineksille ei löytynyt.

Jani:

- Mitkä ominaisuudet / toiminnot olivat helppoja / vaikeita toteuttaa?
    - Vaikeita:
        - UI oli itselleni vaikein osuus, kuinka tehdä hyvä ja käytettävä UI
        - Järkevä tapa toteuttaa sisäinen tiedon hallinta ilman tietokantaa. Nyt on toimiva, mutta ei todellakaan järkevä. Myös vaikeasti ylläpidettävä.
        - Etukäteen tehtävässä suunnitelmassa pysyminen
- Oliko jokin asia aivan syvältä?
    - Ei
- Oliko jokin asia todella hyvää tässä työssä?
    - Tietty vapaus aiheessa teki siitä mielenkiintoisen, vaikka raamit annettiin ulkopuolelta.
    - Aiheen ajankohtaisuus
- Mitä toivoisit ensi vuoden harjoitustyöhön?

- Vapautta valita jonkin aihepiirin sisältä jolloin siitä saa tehtyä itselleen sopivan eli sama kuin nyt.
- Kovan ja backendia vaativan työn sijaan myös mahdollisuus keskittyä esim. graafisten kirjastojen käyttämiseen esim. joku 3d grafiikka käyttävä AR juttu tms. Toki tämä on kandikurssi joka asettanee joitain rajoitteita.