**Ruokapäiväkirja**

IIO13100 JavaEE-ohjelmointi

Olli Nissinen H8593

Miika Avela H4211

Dokumentaatio

4/2017

Ohjelmistotekniikan koulutusohjelma

Tekniikan ja liikenteen ala

Sisällysluettelo

[1 Yleiskuvaus 2](#_Toc480902671)

[1.1 Projektin kuvaus ja toimeksianto 2](#_Toc480902672)

[1.2 Tekijät 2](#_Toc480902673)

[2 Työnjako ja vastuualueet 3](#_Toc480902674)

[3 Käytetyt teknologiat 3](#_Toc480902675)

[3.1.1 Projektin suunnittelu 4](#_Toc480902676)

[3.1.2 Tietokantayhteys ja malli 5](#_Toc480902677)

[3.1.3 JAX-RS 5](#_Toc480902678)

[3.1.4 Spring Framework 6](#_Toc480902679)

[4 Aikataulu 8](#_Toc480902680)

[4.1 Suunniteltu ajankäyttö 8](#_Toc480902681)

[4.2 Toteutunut ajankäyttö 8](#_Toc480902682)

[5 Testaus, ongelmakohdat ja niiden ratkaisut 9](#_Toc480902683)

[6 Itsearviointi projektista 10](#_Toc480902684)

# Yleiskuvaus

## Projektin kuvaus ja toimeksianto

Projektiin toimeksiannon saimme JavaEE-ohjelmoinnin opettajaltamme, Ismo Harjunmaalta. Työn tavoitteena oli perehtyä JavaEE-ohjelmoinnissa käytettyihin yleisimpiin teknologioihin ja oppia näistä käytännössä mahdollisimman paljon. Aiheeksemme valikoitui ruokapäiväkirja, mikä on tehty käyttämällä JDBC/DAO-tietokantayhteyttä MySQL:aan, Spring servleteilla sekä RESTilla. Ruokapäiväkirja on suunnattu syömistottumuksiaan seuraaville henkilöille.

## Tekijät

Projektin tekijöinä toimivat kaksi Jyväskylän ammattikorkeakoulun, ohjelmistotekniikan linjan, kolmannen vuoden opiskelijaa Miika Avela (H4211) ja Olli Nissinen (H8593).

JavaEE-ohjelmointikurssille tultaessa kummankin Javan lähtötaso perustui kahteen aiemmin käytyyn olio-ohjelmointikurssiin, sekä Miikalla Android-kehityksen aikana käytyihin tekniikoihin. Varsinaista JavaEE-kokemusta kummaltakaan ei löytynyt.

# Työnjako ja vastuualueet

Vastuualueet jaoimme alustavasti suunnitelman jälkeen siten, että molemmille tulisi suunnilleen yhtä paljon tehtävää. Ongelman ilmestyessä miettisimme ratkaisua kuitenkin yhdessä. Alueet jakautuivat seuraavasti:

**Olli:**

DAO, DTO, JBDC, View, Google analytics

**Miika:**

Spring, Controllerit, REST, Model, POM määrittely, Interface DAO:n ja Springin välillä

**Yhteistä:**

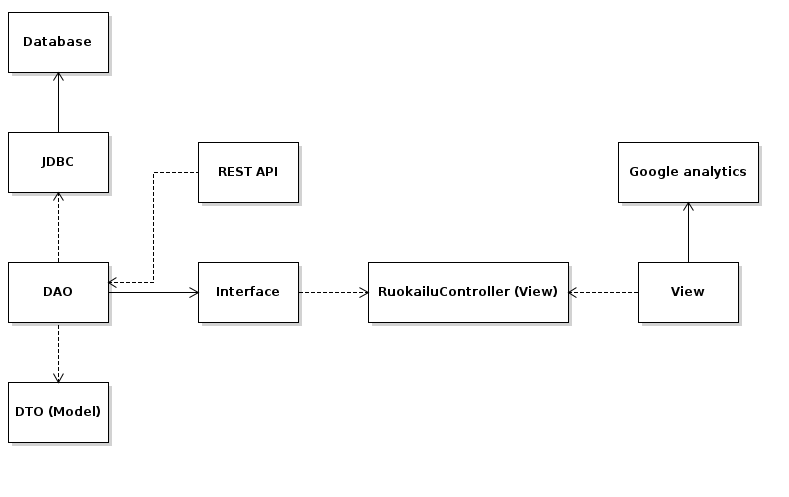
Suunnittelu, Kaaviot, Dokumentointi, Tietokanta

# Käytetyt teknologiat

Kaikki projektin ohjelmointi ja testaus on suoritettu Eclipsen JEE Neon -versiolla. Projektinhallinta ja -seuranta toteutettiin Github-repositoriolla, mitä päivitettiin jokaisten muutosten jälkeen, sekä yhteydenpito samanaikaisesti hoidettiin pääosin Mumble VoiP-ohjelman välityksellä. Myös projektipalavereja pidettiin kasvokkain muutaman kerran viikossa.

### Projektin suunnittelu

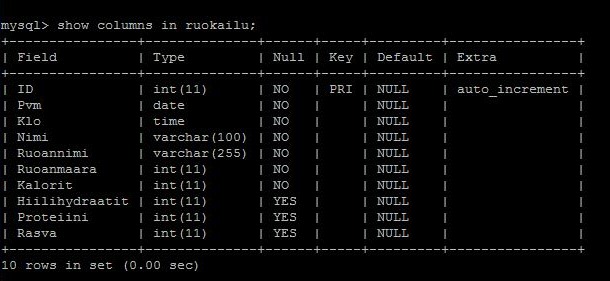
Projektin suunnitteluvaiheessa kirjasimme käsin ylös Ruokapäiväkirjan tärkeimmät ominaisuudet ja vaatimukset, sekä teimme karkean suunnitelman ohjelmiston rakenteesta ja toimivuudesta UML Violetilla. Tässä vaiheessa projekti näytti seuraavalta.



Kuva 1 UML-kaavio ohjelmiston toiminnasta

### Tietokantayhteys ja malli

Ruokapäiväkirjan tietokantana toimi MySQL, mihin loimme kuvan mukaisen taulun



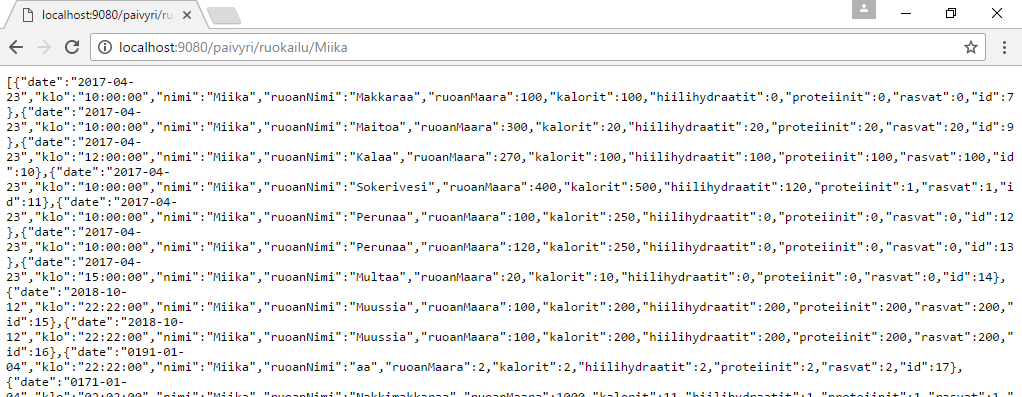
Kuva 2 MySQL taulu

Taulun perusteella luotiin Data Transfer Object (DTO), mikä vastasi attribuuteiltaan taulua. Seuraavaksi luotiin DbManager-luokka, mikä hyödyntää JDBC:ta varsinaiseen kommunikointiin kannan kanssa.

DTO sekä DbManager yhdistettiin Data Access Objectiin (DAO), minkä kautta kaikki kommunikointi kantaan tehdään, sekä mikä vastaa taulun kääntämisestä objekteiksi.

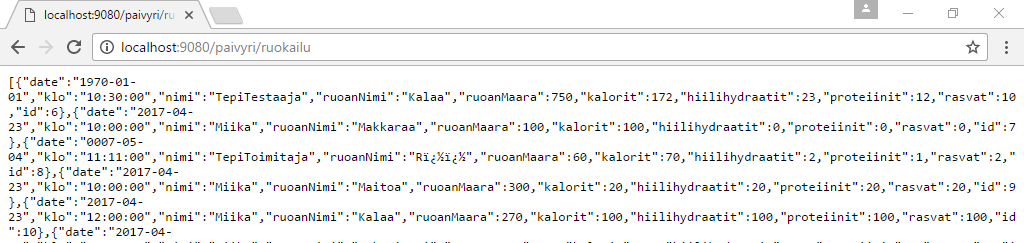
### JAX-RS

Tallennamme tietokantaan enemmän dataa kuin mitä näytämme palvelussa, joten tahdoimme antaa käyttäjille mahdollisuuden hakea ruokailutietonsa käyttöliittymämme ohi. Tähän tarpeeseen rakensimme JAX-RS:n, Java API for RESTful Services, mikä hakee HTTP-osoitteeseen kirjoitettujen parametrien avulla käyttäjäkohtaiset tiedot. Jos esimerkiksi Miika haluaa hakea kaikki ruokailunsa, tulee osoiteriville syöttää *localhost:9080/paivyri/ruokailu/Miika* ja vastaus palaa takaisin JSON-muotoisena seuraavan kuvan mukaisesti.



Kuva 3 JAX-RS Parametrisoidun haun vastaus

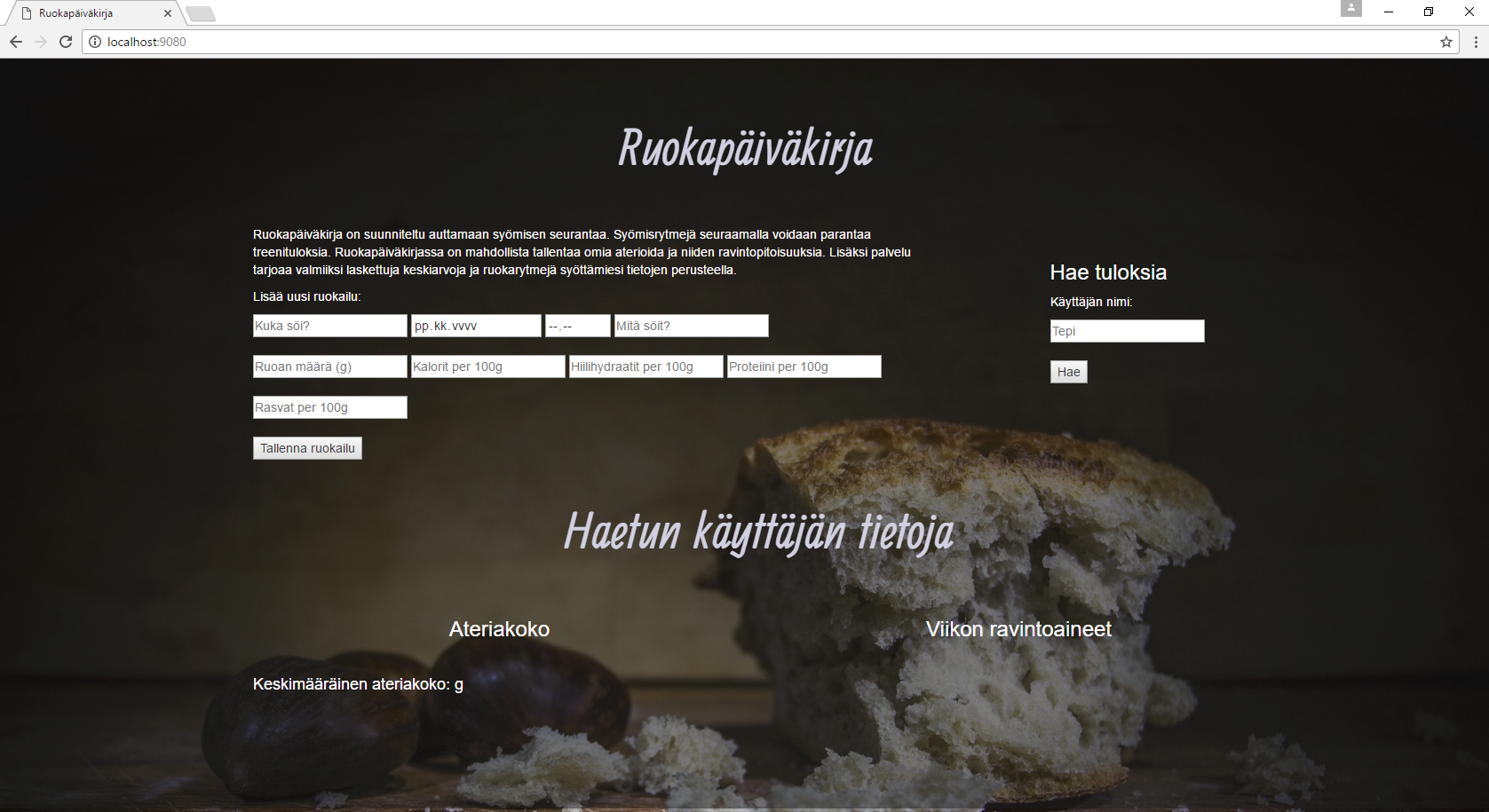
Palvelulta voidaan myös pyytää kaikki, mitä tietokantaan on komennettu jättämällä käyttäjäparametri pois osoiteriviltä: *localhost:9080/paivyri/ruokailu* milloin vastaus on seuraavan kuvan mukainen.



Kuva 4 JAX-RS Täyden haun vastaus

### Spring Framework

Ohjelmiston käyttöliittymä tehtiin selainpohjaiseksi ja tämä toteutettiin käyttämällä Spring Frameworkia. Syöttämällä osoitteen *localhost:9080* selaimelle avautuu palvelun pääsivu, mikä tarjoaa mahdollisuuden syöttää uusia ruokailujaan tietokantaan sekä hakea käyttäjäkohtaisia ruokailutottumuksia tulkittavaksi tietokannasta. Oletuksena pääsivulla ei ole mitään datasidonnaista.



Kuva 5 Etusivu

Kun palvelusta hakee käyttäjäkohtaisia tietoja, Controller saa HTTP GET -pyynnön, mikä ottaa pyynnön URL-parametreista käyttäjän nimen ja kutsuu interfacea (RuokailuService.class), mikä kysyy DAO:lta kyseisen käyttäjän ruokailut. Ruokailut palaavat listana DTO-objekteja, mistä lasketaan vaadittavat arvot Googlen Analytics:lle, mikä visualisoi datan näkymään, sekä muut palautettavat arvot.



Kuva 6 Käyttäjäkohtaisen haun visualisointi

Kun arvot lähetetään näkymälle, siellä olevaa JavaScript-koodia kutsutaan, mikä ottaa ne vastaan ja pyytää Google Analyticsia visualisoimaan niitä, sekä tämän jälkeen esittää ne käyttäjälle.

# Aikataulu

## Suunniteltu ajankäyttö

Lähtökohtaisena ajankäytön suunnitelmana oli käyttää tunteja seuraavasti:

View 5h

Google analytics 5h

Tietokannan pystytys 5h

Tietokanta yhteys + JDBC + DAO + DTO tai JPA 10h

POM määrittely 5h

REST 10h

Spring 15h

Suunnittely 5h

Dokumentointi 5h

Kaaviot 2h

Interface ( dao - spring ) 8h

Controllerit 10h

Model 5h

**Yhteensä 90h**

## Toteutunut ajankäyttö

Todellisuudessa työtunteja kertyi seuraavasti:

Projektin suunnittelu 6h

View 4h

JPA 11h -> vaihdettiin JDBC + DAO + DTO 10h

Tietokannan pystytys ja konfigurointi 4h

POM määrittely 7h

Google analytics 3h

REST 11h

Spring 16h

Kaaviot 3h

Interface 2h

Controllerit 8h

Model 4h

Dokumentointi 6h

**Yhteensä: 95h**

# Testaus, ongelmakohdat ja niiden ratkaisut

Sivuston tietokantaan syöttäminen on toteutettu Prepared Statementeilla joten perinteisiltä injektointihyökkäyksiltä on suojauduttu. Palvelussa on myös testattu, ettei kantaan pysty syöttämään vääränmallista dataa. Tämä on varmistettu, syöttökentillä, jotka pakottavat käyttäjän antamaan halutun mallista dataa. Myös RESTful API on testattu ja API:lta voi mm. pyytää omia tietoja ja hyödyntää niitä jollain muulla sivustolla.

Ongelmakohtia olivat JPA ja Eclipsen konfiguraatio. JPA:n kanssa ongelmaksi muodostui MySQL-ajuri. Ajuri ei saanut millään yhteyttä kantaan ja ongelmaan kului sen verran aikaa, että päätimme toteuttaa kantayhteyden vaihtoehtoisesti JDBC & DAO:n avulla. Vaihtoehtoinen yhteysmuoto onnistui hyvin.

Toisena ongelmana oli Eclipsen konfiguraatio. Oli pitkään epäselvää, miksi Jetty-serveriä käynnistäessä palvelun compilointi epäonnistui satunnaisesti. Useammalla yrityksellä tästä päästiin ohi ja ongelma sivuutettiin pidemmäksi aikaa, kunnes compilointi ei enää onnistunut ollenkaan, eikä projektia saanut enää ajettua. Tähän löytyi lopulta ratkaisu, kun Eclipse käytti oletuscompilerina JRE:ta ja projekti vaati sen olevan JDK.

# Itsearviointi projektista

Mielestämme projekti on ollut hyvin opettavainen ja mielenkiintoinen kokonaisuus. Tunnilla tehdyt asiat ovat olleet oikein hyviä ja hyödyllisiä esimerkkejä. Silti suuremman kokonaisuuden hallitseminen on haastavaa ilman aukotonta suunnitelmaa kun tietotaitoa tekniikoista ei vielä ole kertynyt tarpeeksi. Projektin edetessä olemme huomanneet UML-kaavion tärkeyden ja kuinka se helpottaa koodin kirjoittamista, sekä projektin toiminnallisuuden ymmärtämistä.

Lopputulokseen olemme tyytyväisiä, sillä aikaa projektin tekemiseen oli hyvin rajallisesti rankan kevään takia, silti täyttäen kaikki projektille osoitetut vaatimukset. Sovellus sisältää seuraavat osa-alueet: Tietokantayhteystaso (JDBC/DAO), Sovelluslogiikka ja vahvasti datasidonnainen sekä MVC-arkkitehtuurin mukainen käyttöliittymä. Tulevaisuudessa suunnittelimme tekevämme tallennetusta datasta monipuolisempia visualisointeja ja laskelmointeja Google Analyticsin avulla.