VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Jari Matti Hakala

HENKILÖKOHTAINEN KUNTO-OHJAAJA -SOVELLUS

Tekniikka ja liikenne
2009

VAASAN AMMATTIKORKEAKOULU

Tietotekniikan koulutusohjelma

TIIVISTELMÄ

Tekijä	Jari Hakala
Opinnäytetyön nimi	Henkilökohtainen kunto-ohjaaja -sovellus
Vuosi	2009
Kieli	suomi
Sivumäärä	35
Ohjaaja	Pirjo Prosi

Tämän työn tarkoituksena oli toteuttaa web-pohjainen sovellus joka toimii henkilökohtaisena kunto-ohjaajana. Sovellus tarjoaa käyttäjilleen työkaluja, joilla voidaan seurata omia ruokailu- ja liikuntatottumuksiaan sekä ohjeistaa käyttäjää saavuttamaan tavoitteitaan.

Sovellus toteutettiin JavaServer Faces -tekniikkaa käyttäen ja tietojen varastointiin käytetään MySQL tietokantahallintajärjestelmää. Aineistona on käytetty alan kirjallisuutta sekä käytettyjen tekniikoiden ja järjestelmien spesifikaatioita.

Valmis sovellus sisältää lähes kaikki vaatimusmäärittelyssä esitetyt toiminnot. Sovelluksen käyttöliittymä ei vielä tätä raporttia kirjoitettaessa ole täysin valmis ja vaatii jatkokehitystä.

Asiasanat ohjelmointi, JavaServer Faces, tietokanta, MySQL

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Tietotekniikan koulutusohjelma

ABSTRACT

Author	Jari Hakala
Title	Personal Trainer Application
Year	2009
Language	Finnish
Pages	35
Name of Supervisor	Pirjo Prosi

The purpose of this thesis was to produce a web-application that functions as a personal trainer. The application offers the user a variety of tools that helps the user to keep track of his eating and exercise habits. The application also coaches the user towards proper nutrition and effective exercise regimes.

The application was programmed using JavaServer Faces -technology and MySQL database system for data storage. For study material I have used literature about the relevant subject and specifications of the techniques used.

Finished application contains almost every feature from the requirement analysis. The user interface is still incomplete and it requires much more development.

Keywords Programming, JavaServer Faces, Database, MySQL

KÄYTETYT TERMIT JA LYHENTEET

JSF, eli JavaServer Faces on Sunin kehittämä sovelluskehys Java -pohjaisten web-sovelluksien toteutukseen. /5/

SQL, eli Structured Query Language on standardoitu relaatiotietokannan kyselykieli, jonka avulla annetaan käskyjä tietokannalle. /4/

JDBC, eli Java Database Connectivity on ohjelmointirajapinta, jonka avulla voidaan luoda yhteys tietokannan ja Java -sovelluksen välillä sekä suorittaa SQL-kyselyitä Java-luokkien sisällä. /3/

MVC, eli Model-View-Controller on arkkitehtuuri, jonka mukaan sovellus on jaettu kolmeen osaan; malliin (model), näkymään (view) ja ohjaimeen (controller). /2/

SISÄLLYS

1 JOHDANTO

Opinnäytetyön tarkoituksena on suunnitella ja toteuttaa Internet-pohjainen sovellus, joka toimii henkilökohtaisena kunto-ohjaajana. Sovellus on tarkoitettu varsinkin aloitteleville kuntoilijoille, mutta sopii myös edistyneemmillekin. Sovellus pyrkii ohjaamaan kuntoilijaa ennen kaikkea terveellisiin elämäntapoihin ja opettamaan kuntoilijaa seuraamaan ruokailu- ja liikuntatottumuksiaan, jolloin mahdollisia epäkohtia on helpompi korjata. Kuntoilijaa ohjataan tavoitteeseensa antamalla ohjeita esimerkiksi ravinnon ja harjoittelun suhteen. Sovellus ei suosi mitään kuureja tai muitakaan nopeita ratkaisuja, vaan sovelluksen perimmäisenä ajatuksena on määrätietoinen harjoittelu ja laadukas ravinnon saanti, jotka pyritään saamaan osaksi jokapäiväistä elämää, jolloin saavutetut tulokset ovat pysyviä.

Opinnäytetyönä toteutettava sovellus toimii Internet-selaimella ja rekisteröitymisen jälkeen käyttäjä voi kirjautua sisälle palveluun ja aloittaa sovelluksen käytön. Käyttäjän tunnistus suoritetaan käyttäjätunnuksella ja salasanalla, jotka ovat varastoituina tietokantaan. Ohjelmistoprojekti toteutetaan käyttämällä JavaServer Faces -tekniikkaa, joka on standardoitu sovelluskehys Java-pohjaisten web-sovellusten luomiseksi. Sovellus toimii Apache Tomcat 6.0 web-palvelimella ja käyttää tietojen varastointiin MySQL -tietokantahallintajärjestelmää.

2 KÄYTETYT OHJELMISTOT JA TEKNIIKAT

2.1 Ohjelmointiympäristö

Ohjelmointiympäristönä toimii NetBeans IDE 6.5, joka on avoimen lähdekoodin kehitysympäristö ja se tukee monia eri ohjelmointikieliä ja sovelluskehyksiä, mutta on alun perin kehitetty toimimaan Java-ohjelmointikielen kehitysympäristönä. Kehitysympäristö tarjoaa laajan työkaluvalikoiman ohjelmakoodin kirjoittamiseen ja hallintaan.

NetBeans IDE valittiin työssä käytettäväksi kehitysympäristöksi, koska se tukee työssä käytettävää JavaServer Faces -tekniikkaa ja se on hyvin yhteensopiva Apache Tomcat 6.0 sekä MySQL Server 5.0 -palvelinten kanssa, jolloin ylimääräisiltä lisäosien asennusprojekteilta vältytään.

2.2 Ohjelmointitekniikka

Opinnäytetyö toteutettiin JavaServer Faces -tekniikalla, jäljempänä JSF. JSF on sovelluskehys, joka on suunniteltu Java-pohjaisten palvelinpuolen sovellusten ripeään käyttöliittymäkehitykseen. JSF kuuluu Java Enterprise Edition -standardiin ja se sisältää kolme pääosaa; kokoelman valmiita käyttöliittymäkomponentteja, tapahtumapohjaisen ohjelmointimallin sekä komponenttimallin, joka mahdollistaa kolmansien osapuolten valmistamien komponenttien tuonnin. Lyhyesti kerrottuna JSF on web-sovelluksiin tarkoitettu Swing -käyttöliittymäkomponenttikirjasto. /2/

Käytännössä JSF jakaa sovelluksen kahteen eri osaan, jotka ovat esitysosa sekä toiminnallisuusosa. Esitysosa Internet-pohjaisessa sovelluksessa on yleensä web-sivu, joka sisältää JSF tagikirjaston komponentteja, jotka kutsuvat sovelluksen toiminnallista osaa, joka on normaalisti Java-papu (Java Bean). Java-papu on Java-luokka, joka tarjoaa sovelluksen ominaisuudet ja tapahtumankäsittelijät, se on myös vastuussa kommunikoinnista sovelluksen varsinaisten liiketoimintaluokkien kanssa. /2/

Kirjoitettaessa JSF -sovellusta tarvitaan kaksi tagikirjastoa, jotta JSF komponentit saadaan käyttöön. Nämä kaksi kirjastoa ovat core -kirjasto sekä html -kirjasto ja ne lisätään jsp -sivun alkuun kirjoittamalla niille deklaraatiot:

```
<%@ taglib prefix="f" uri="http://java.sun.com/jsf/core" %>
<%@ taglib prefix="h" uri="http://java.sun.com/jsf/html" %>
```

Kun nämä kaksi tagikirjastoa on otettu mukaan, voidaan niiden sisältämiä komponentteja käyttää sovelluksessa. Tagikirjastot täytyy sisältää jokaiseen sovelluksen sivuun, joissa käytetään JSF -komponentteja. Kaikki JSF -komponentit on sijoitettava *<f:view>* tagien sisään, jotta komponentit toimisivat halutulla tavalla. *<f:view>* tagien sisään voidaan lisätä *<h:form>* tagi, joka muodostaa lomakkeen, johon voidaan lisätä itse komponentteja, kuten tekstikenttiä ja nappeja. /5/

Toimiakseen halutulla tavalla JSF -sovellus tarvitsee faces-config.xml -tiedoston, johon on määritelty sovelluksen kaikki navigaatiosäännöt. Navigaatiosäännöt määrittelevät mille sovelluksen sivulle käyttäjä ohjataan tietyn tapahtuman sattuessa. Navigaatiosäännöt toimivat merkkijonojen avulla, joita saadaan esimerkiksi jonkin metodin paluuarvona (dynaaminen navigointi) tai JSF -komponenttien action -ominaisuudesta (staattinen navigointi). Alla on esimerkki JSF -sovelluksen navigaatiosäännöstä. /2/

```
<navigation-rule>
<from-view-id>/hei.jsp</from-view-id>
<navigation-case>
<from-outcome>ok</from-outcome>
<to-view-id>/tervetuloa.jsp</to-view-id>
</navigation-case>
</navigation-rule>
```

Yllä olevassa esimerkissä on määritelty yksi navigaatiosääntö (*<navigation-rule>*), joka sisältää yhden navigaatiotapauksen (*<navigation-case>*). Sääntö käsittelee mihin käyttäjä ohjataan kun sattuu tapahtuma hei.jsp -sivulla. Hei.jsp -sivulla voi olla esimerkiksi *commandLink* -komponentti, jonka action ominaisuuden arvona on ”ok”. Kun komponenttia painetaan saa sovellus

merkkijonon ”ok” ja sen jälkeen sovellus tarkastaa faces-confing.xml -tiedostosta mitä täytyy tehdä. Yllä olevan navigaatiosäännön mukaan sovellus ohjaa käyttäjän tervetuloa.jsp -sivulle.

JSF -sovellus toteuttaa MVC -arkkitehtuuria, eli sovelluksen arkkitehtuuri on jaettu kolmeen osaan; malliin (model), näkymään (view) ja ohjaimeen (controller). Mallilla tarkoitetaan sovelluskohtaista dataa jota käyttäjä voi manipuloida, esimerkiksi päivämääriä tai merkkijonoja. Näkymällä määritellään kuinka data esitetään käyttöliittymässä. Ohjain määrittää datan prosessoinnin ja käyttöliittymän päivityksen. JSF -sovelluskehys yhdistää mallin ja näkymän käyttämällä komponentteja, joilla pystytään välittämään tietoa lomakkeen ja sovelluksen logiikkaosan välillä. Esimerkiksi voidaan käyttää *inputText* -komponenttia, joka on sidottu Java -pavun muuttujaan. JSF -sovelluskehys toimii myös ohjaimena, joka ohjaa lomakkeelta annetun tiedon sovelluksen logiikkaosan asianmukaiselle metodille ja päivittää käyttöliittymän tapahtumien mukaan. Esimerkiksi voidaan käyttää *commandButton* -komponenttia, joka on sidottu Java -pavun metodiin, joka tallentaa tai muokkaa dataa. /2/

JSF -sovelluskehys tarjoaa mahdollisuuden käyttää datakonverttereita ja validaattoreita. Normaalissa web -sovelluksessa käyttäjä syöttää sovelluksen tiedot *String* -tyyppisenä, jotka täytyy syötön jälkeen konvertoida käsin sovelluksen logiikkaosassa. JSF -sovelluksessa voidaan käyttää datakonversiota suoraan lomakkeessa, jolloin sovellus konvertoi *String* -tyyppisen tiedon automaattisesti haluttuun muotoon, kuten päivämääräksi tai numeroksi jolloin se voidaan ottaa vastaan sovelluksen logiikkaosassa suoraan oikeanmuotoiseen muuttujaan. Validaattoreita voidaan käyttää tarkistamaan onko syötetty tieto oikeassa muodossa. Esimerkiksi voidaan tarkastaa onko päivämäärä syötetty oikein JSF:n sisäänrakennetulla *convertDateTime* validaattorilla. Seuraavassa on esimerkki päivämäärän konversiosta ja validoinnista. /5/

```
PVM (vvvv-kk-pp): <h:inputText id="date" value="#{JavaBean.date}" >
                    <f:convertDateTime pattern="yyyy-MM-dd" />
                  </h:inputText>
                  <h:message for="date" style="color:red" />
```

Edellä olevassa esimerkissä päivämäärä täytyy syöttää muodossa vvvv-kk-pp, esimerkiksi 2009-09-29. Mikäli päivämäärä on väärässä muodossa, antaa sovellus käyttäjälle virheilmoituksen, mutta jos päivämäärä on oikeassa muodossa, lähetetään se *JavaBean* -luokan *date* -muuttujaan joka on muotoa *java.util.Date*. Vaikka JSF tarjoaa suuren valikoiman valmiita konverttereita ja validaattoreita voidaan niitä myös määritellä helposti itse. /5/

2.3 Tietokannat

Internet-pohjaisen tietokantasovelluksen toteuttamiseen tarvitaan nopea, toimiva ja siirrettävä tietokantojen hallintajärjestelmä. Tämän projektin tarpeisiin sopii MySQL relaatiotietokantojen hallintajärjestelmä, joka on avoimen lähdekoodin tuote ja ilman kaupallista tuotetukea se on ilmainen käyttää. /4/

MySQL on hyvin yhteensopiva Java-ohjelmointikielen kanssa ja kommunikointi näiden kahden välillä toimii JDBC-ajureita käyttäen. JDBC on ohjelmointirajapinta, jonka avulla voidaan luoda yhteys tietokantaan sekä suorittaa SQL-kyselyitä Java-luokkien sisällä. /3/

2.4 Palvelinympäristö

Internet-pohjainen sovellus vaatii toimiakseen palvelimen, jonka päällä ohjelmakoodia suoritetaan. Tässä opinnäytetyössä käytetään Apache Tomcat 6.0 -palvelinta, joka on avoimen lähdekoodin JavaServer Pages ja Servlet -säiliö. Apache Tomcat 6.0 -versio tukee myös työssä käytettävää JavaServer Faces -tekniikkaa.

3 VAATIMUSMÄÄRITTELY

3.1 Yleiskuvaus

Opinnäytetyön lopputuotteena syntyvä sovellus toimii web-palvelimella, johon otetaan yhteys Internet-selaimella ja se käyttää tietojen varastointiin tietokantoja, jotka sijaitsevat tietokantapalvelimella. Sovellus toimii henkilökohtaisena kunto-ohjaajana, joka opastaa käyttäjää terveellisiin elämäntapoihin. Sovellus tarjoaa käyttäjälleen erilaisia työkaluja omien ruokailu- ja liikuntatottumusten seuraamiseen sekä tilastoi kehitystä ja tarjoaa ohjeistusta ravinnon ja liikunnan suhteen.

3.2 Toiminnallisuuteen kohdistuvat vaatimukset

3.2.1 Toimintovaatimukset

Sovelluksen tulee toimia käyttäjän henkilökohtaisena kunto-ohjaajana, joka avustaa ja ohjaa käyttäjää saavuttamaan haluamansa tavoitteet. Käyttäjän tulee voida pitää kirjaa kehityksestään, ruokailuistaan sekä liikkumisestaan. Syötettyjen tietojen perusteella ohjelmisto laskee päivässä saadun energian, kulutetun energian sekä kokonaisenergian. Sovellus laskee myös erilaisia kehityksen indikaattoreita, muodostaa tilastoiduista tiedoista graafisia esityksiä, antaa ohjeita ruokavalion toteuttamiseen ja harjoitusten suorittamiseen sekä antaa varoituksia tai neuvoo ottamaan yhteyttä lääkäriin jos kehityksessä havaitaan huolestuttavia piirteitä. Taulukossa 1 on lueteltuna sovelluksen ominaisuudet ja niiden prioriteetti.

Taulukko 1: Toimintovaatimukset

Toiminto	**Kuvaus**	**Prioriteetti**
T1	Uuden käyttäjän rekisteröinti	1
T2	Käyttäjän tunnistus	1
T3	Ruokapäiväkirja	1
T4	Harjoituspäiväkirja	1
T5	Mittauspäiväkirja	1
T6	Tilastojen tarkkailu halutulla aikavälillä	1
T7	Tavoitteen valinta	1
T8	Ravintoainetietokanta	1
T9	Harjoitustietokanta	1
T10	Harjoitusohjeiden antaminen	2
T11	Ruokailuohjeiden antaminen	2
T12	Ohjaaminen tavoitteen mukaan	2
T13	Saavutusten seuranta	2
T14	Varoituksien antaminen	2
T15	Graafisten esitysten luonti	3

3.2.1 Yhteensopivuus ja toimintaympäristö

Sovelluksen tulee olla yhteensopiva yleisimpien nykyaikaisten Internet-selaimien, kuten Microsoft Internet Explorerin, Operan ja Mozilla Firefoxin kanssa. Käyttöliittymän tulee olla skaalautuva asiakaskoneen näytön resoluution mukaan.

Toimintaympäristönä tulee olemaan Apache Tomcat 6.0 web-palvelin sekä MySQL Server 5.0 -tietokantapalvelin. Sovellus asennetaan web-palvelimelle ja siihen otetaan yhteys Internet-selaimella HTTP-yhteyden yli. Web-palvelin kommunikoi tietokantapalvelimen kanssa JDBC-ohjelmointirajapinnan kautta.

3.3 Luotettavuuteen kohdistuvat vaatimukset

Sovelluksen täytyy olla vikasietoisuudeltaan hyvä ja virhetilanteen tapahtuessa sen tulee ilmoittaa käyttäjälle virheestä ja toivuttava nopeasti virhettä

edeltäneeseen tilaan, aiheuttamatta tiedon katoamista. Virheilmoitus kertoo virheen syyn ja kehottaa käyttäjää ottamaan yhteyttä sovellusta hallinnoivaan tahoon. Sovelluksen tulee ilmoittaa käyttäjälle myös onnistuneesta tiedon tallennuksesta. Sovellus ei saa aiheuttaa asiakaskoneelle aiheettomia tietoturvariskejä.

3.4 Käytettävyyteen kohdistuvat vaatimukset

Sovelluksen käyttöliittymän tulee olla selkeä, helppokäyttöinen, helposti ymmärrettävä ja opittava. Sovelluksen toimintojen on toimittava odotetulla tavalla ja sovelluksen käyttö tulee olla miellyttävää.

Komponenttien ja linkkien tulee olla hyvin järjesteltyjä ja selkeitä sekä sovelluksen käyttöön täytyy olla saatavilla selkeät ja huolitellut ohjeet. Navigointi sovelluksen sisällä tulee toteuttaa siten, että jokaisesta näkymästä pääsee perusnäkymään ja samankaltaisiin näkymiin.

Sovelluksen käyttökieli on suomi. Kielen tulee olla helppolukuista ja yksiselitteistä siten, että sovelluksen käyttö on helppoa ja ohjeet ymmärrettäviä jokaiselle käyttäjälle.

3.5 Ylläpidettävyyteen kohdistuvat vaatimukset

Sovellus tulee olla helposti päivitettävissä web-palvelimella ilman, että syntyy pitkiä käyttökatkoksia. Sovellus täytyy toteuttaa modulaarisesti siten, että uusia ominaisuuksia voidaan lisätä ilman koko ohjelmakoodin muuttamista.

Sovellus muodostaa tietokantayhteyden ainoastaan silloin, kun sitä tarvitaan ja tarvittaessa se voidaan liittää myös muihin tietokantoihin kuin MySQL tietokantaan mahdollisimman vähillä ohjelmakoodin muutoksilla.

Sovelluksen tulee toimia luotettavasti ja vakaasti erilaisissa käyttötilanteissa, eikä se saa kuormittaa palvelimia turhaan.

Kehitysvaiheessa sovellusta täytyy voida testata suoraan kehitysympäristössä, ilman sen siirtoa varsinaiselle sovelluspalvelimelle. Valmiin sovelluksen suorituskykyä ja toimivuutta on voitava testata.

Valmiin sovelluksen siirtäminen web-palvelimelta toiselle tulee olla suoraviivaista ja ohjelmakoodin muuttaminen vähäistä.

4 TOIMINNALLINEN MÄÄRITTELY

4.1 Johdanto

Opinnäytetyönä suoritettavan ohjelmistoprojektin tavoitteena on toteuttaa web-pohjainen sovellus, joka toimii henkilökohtaisena kunto-ohjaajana. Sovellus on tarkoitettu yleiseen käyttöön ja kaikille halutaan tarjota mahdollisuus kehittää omia fyysisiä ominaisuuksiaan lähtökohdista riippumatta.

4.1.1 Tarkoitus

Sovelluksen tarkoituksena on toimia henkilökohtaisena kunto-ohjaajana, joka auttaa käyttäjiä seuraamaan omia ruokailu- ja liikuntatottumuksiaan sekä ohjata muuttamaan elämäntapoja terveellisemmiksi.

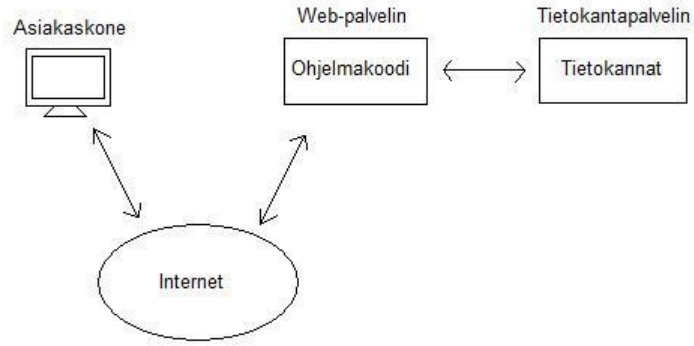
4.1.2 Tuote ja ympäristö

Vuorovaikutus käyttäjän kanssa tapahtuu Internet-selaimella, joka esittää sovelluksen käyttöliittymän. Varsinainen sovellus suoritetaan web-palvelimella ja varastoidut tiedot sijaitsevat tietokantapalvelimella. Sovelluksen käyttö ei vaadi mitään erityisiä lisäohjelmia käyttäjän koneella.

4.2 Yleiskuvaus

4.2.1 Ympäristö ja toiminta

Sovellus sijaitsee web-palvelimella, johon otetaan yhteys Internet-selaimella joka esittää myös sovelluksen käyttöliittymän. Tietojen varastointiin käytetään tietokantaa, joka sijaitsee tietokantapalvelimella ja tietojen esitykseen käytetään sovelluksen käyttöliittymää. Web-palvelimella oleva toiminnallinen osuus hakee tiedot tietokannasta ja toimittaa ne edelleen esitettäviksi sovelluksen käyttöliittymälle. Kuvassa 1 on periaatteellinen esitys sovelluksen toimintaympäristöstä.



Kuva 1: Käyttöympäristö

4.2.2 Käyttäjät

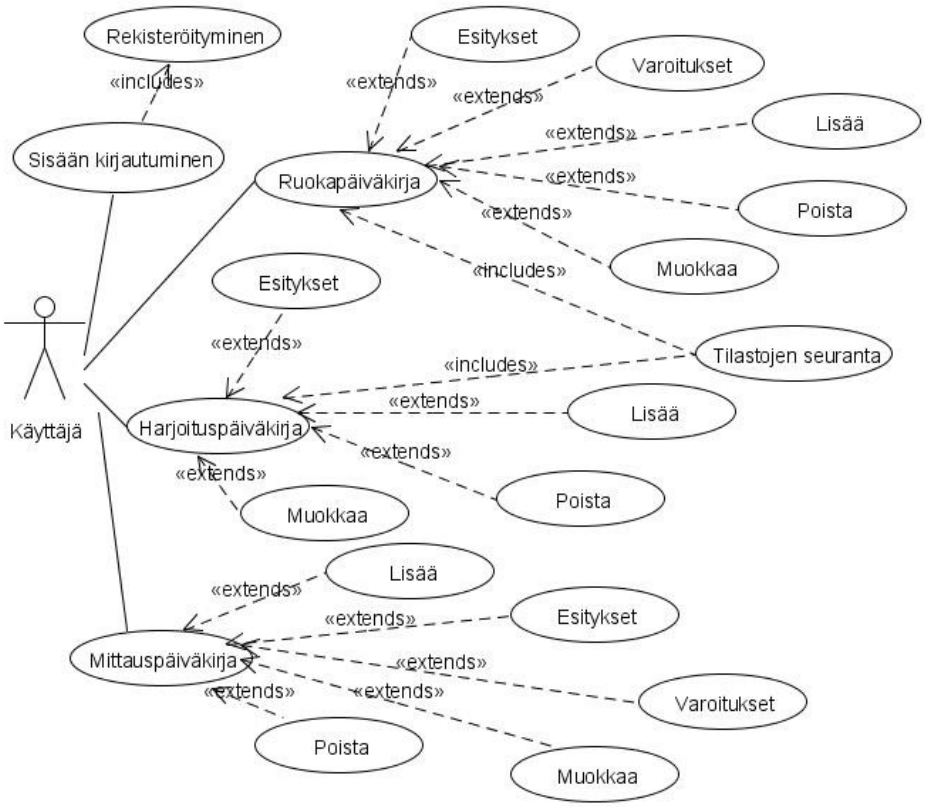
Sovellus on suunniteltu sellaisille käyttäjille, jotka haluavat muuttaa elämäntapojaan terveellisemmiksi ja parantaa omia fyysisiä ominaisuuksiaan. Suurin hyöty sovelluksesta on aloitteleville kuntoilijoille, mutta suunnittelussa on otettu huomioon myös edistyneemmät kuntoilijat jotka haluavat lisätukea harrastukselleen.

4.2.3 Oletukset ja riippuvuudet

Ohjelman toiminnan kannalta käyttäjällä oletetaan olevan käytössään toimiva ja avoinna oleva Internet-yhteys, koska sovellus toimii web-palvelimella eikä mitään ohjelmiston osia ladata käyttäjän koneelle. Sovelluksen käynnistys vaatii yhteyden luomisen käyttäjän koneen ja web-palvelimen välille.

4.3 Toiminnot

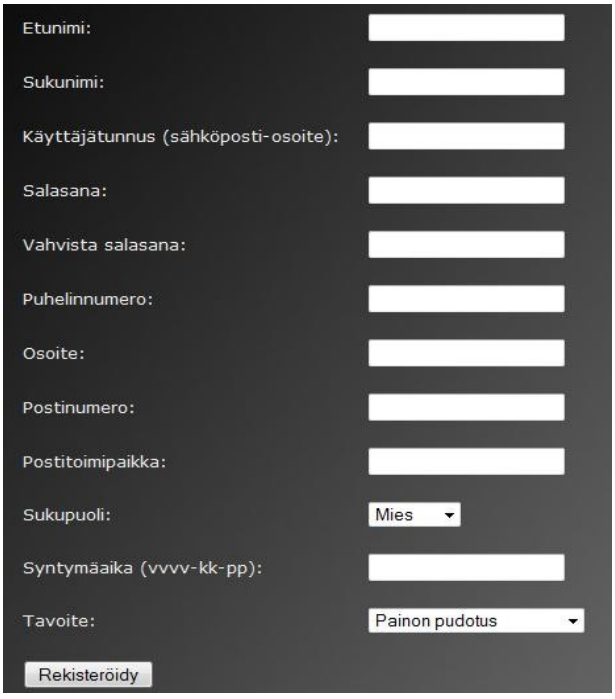
Sovelluksen tärkeimmät toiminnot on esitettynä kuvassa 2 olevassa käyttötapakaaviossa.



Kuva 2: Käyttötapakaavio

4.3.1 Rekisteröityminen

Sovelluksen käyttäjäksi voidaan rekisteröityä täyttämällä rekisteröintilomake, johon uusi käyttäjä voi syöttää sovelluksen tarvitsemat henkilötiedot ja valita haluamansa tavoitteen. Rekisteröintilomakkeeseen käyttäjän tulee syöttää sähköpostitunnus, salasana, salasanan vahvistus, etunimi, sukunimi, puhelinnumero, katuosoite, postitoiminumero, postitoimipaikka, sukupuoli, syntymäaika ja tavoite. Sähköpostitunnus toimii sovelluksessa myös käyttäjätunnuksena. Tietojen syöttämisen jälkeen lomakkeen tiedot lähetetään sovellukselle valitsemalla Rekisteröidy -painike, jolloin tiedot tallennetaan tietokantaan, käyttäjä ohjataan sovelluksen aloitusnäkymään ja hänelle avataan uusi istunto. Kuvassa 3 on käyttöliittymäesimerkki rekisteröitymisestä.



Kuva 3: Rekisteröityminen

Rekisteröitymisen jälkeen käyttäjä voi tarkastella ja muokata omia henkilötietojaan sekä valita uuden tavoitteen.

4.3.2 Sisäänkirjautuminen

Sisäänkirjautumiseen käytetään käyttäjätunnusta ja salasanaa, jotka ovat tallennettuina tietokantaan rekisteröitymisen yhteydessä. Käyttäjätunnus ja salasana syötetään sisäänkirjautumissivun lomaketta käyttäen. Tietojen syötön jälkeen sovellus noutaa tietokannasta asianmukaisen salasanan ja vertaa sitä käyttäjän toimittamaan salasanaan. Mikäli salasanat täsmäävät, käyttäjä ohjataan sovelluksen aloitusnäkymään ja hänelle avataan uusi istunto. Sisäänkirjautumisen epäonnistuessa näytetään virhesivu, jossa käyttäjälle ilmoitetaan virheestä. Kuvassa 4 on käyttöliittymäesimerkki sisäänkirjautumisesta.

Kuva 4: Sisäänkirjautuminen

4.3.4 Ruokapäiväkirja

Ruokapäiväkirja koostuu kahdesta tietokannan taulusta, joita ovat Ruoka -taulu ja Ruokapäiväkirja -taulu. Ruoka -taulussa on syötettynä erilaisia elintarvikkeita ravintosisältöineen, joita voidaan lisätä itse Ruokapäiväkirja -tauluun.

Ruokapäiväkirjan rivejä voidaan seurata näkymästä, johon on listattu käyttäjän aiemmin lisäämät rivit, joista käy ilmi päivämäärä, elintarvikkeen nimi, kokonaisenergia, proteiinit, hiilihydraatit ja rasvat. Tästä näkymästä käyttäjä voi myös lisätä, poistaa tai muokata rivejä. Kuvassa 5 on käyttöliittymäesimerkki ruokapäiväkirjan rivien seurannasta.

Kuva 5: Ruokapäiväkirjan rivit

Ruokapäiväkirja -tauluun voidaan lisätä uusi rivi täyttämällä lomake, johon voidaan syöttää päivämäärä, elintarvikkeen nimi sekä sen määrä. Päivämäärälle ja ruoan määrälle on varattuna tekstikentät joihin tiedot voidaan syöttää. Elintarvikkeen nimi voidaan valita alasvetovalikosta, johon sovellus hakee kaikkien Ruoka -taulussa olevien elintarvikkeiden nimet. Tietojen syötön jälkeen käyttäjä valitsee joko Peruuta tai Tallenna -painikkeen. Peruuta -painike ei tallenna tietoja tietokantaan ja ohjaa käyttäjän takaisin ruokapäiväkirjan näkymään, jossa on esillä ruokapäiväkirjan kaikki rivit. Tallenna -painike lähettää syötetyt tiedot sovelluksen logiikkaosalle, joka hakee Ruoka -taulusta ruoan ravintosisällön sekä laskee Ruokapäiväkirja -taulun riville tarvittavat määrät perustuen ruoan ravintosisältöön sekä nautittuun määrään. Laskutoimitusten jälkeen sovellus tallentaa tiedot tietokantaan ja ohjaa käyttäjän ruokapäiväkirjan näkymään, jossa on esitettynä ruokapäiväkirjan kaikki rivit. Kuvassa 6 on käyttöliittymäesimerkki rivin lisäyksestä.

Kuva 6: Ruokapäiväkirjan rivin lisäys

Rivin poisto tapahtuu valitsemalla kuvan 3 näkymästä Poista -painike rivin lopusta, jolloin sovellus poistaa kyseisen rivin tietokannasta ja päivittää avoinna

olevan näkymän. Rivin muokkaus tapahtuu valitsemalla kuvan 3 näkymästä Muokkaa -painike, jolloin käyttäjä ohjataan rivin muokkausnäkymään, jossa riville voidaan vaihtaa ruoan nimi ja määrä. Rivin muokkauksen jälkeen valitaan Tallenna -painike, jolloin sovellus tallentaa muutokset tietokantaan. Kuvassa 7 on käyttöliittymäesimerkki rivin muokkauksesta.

Kuva 7: Ruokapäiväkirjan rivin muokkaus

Saatujen ravintoaineiden kokonaismääriä ja suhteita voidaan seurata taulukosta tai graafisesta esityksestä yhden tai useamman päivän ajalta. Sovellus hakee tietokannasta ruokapäiväkirjan rivit halutun ajanjakson ajalta ja suorittaa niille laskutoimituksia, joista käy ilmi ravintoaineiden määrät ja suhteet.

Sovellus antaa käyttäjälle varoituksen mikäli joitain ravintoaineita ei saada tarpeeksi tai huomautuksen mikäli joitain ravintoaineita saadaan liikaa.

Ruoka -taulun ravintoainemäärien lähteenä on käytetty erilaisten elintarvikkeiden tuoteselostetta.

4.3.4 Harjoituspäiväkirja

Harjoituspäiväkirjatoiminto on samankaltainen kuin ruokapäiväkirjatoiminto. Harjoituspäiväkirja koostuu Harjoite -taulusta ja Harjoituspäiväkirja -taulusta. Harjoite -taulussa on syötettynä erilaisia harjoitusmuotoja ja niiden energiankulutus minuuttia kohden. Energiankulutus riippuu käyttäjän painosta,

joten jokaiselle harjoitusmuodolle on syötetty energiankulutus usealle eri painoluokalle. Harjoituspäiväkirjan riveiltä käy ilmi harjoitteen suorituspäivämäärä, harjoitteen muoto, kesto ja kokonaiskulutus.

Harjoituspäiväkirjaan voidaan lisätä rivejä samankaltaisella lomakkeella kuin ruokapäiväkirjassa. Lisäyslomakkeessa on paikat harjoitemuodolle ja harjoitteen kestolle. Lisäyksen jälkeen sovellus laskee kulutetun energian ja tallentaa rivin tietokantaan. Rivien muokkaus ja poisto toimivat samalla tavalla kuin ruokapäiväkirjassa.

Päivän kulutettu kokonaisenergia saadaan laskemalla yhden päivän kaikkien rivien kulutus yhteen sekä lisäämällä siihen henkilön aineenvaihdunnasta aiheutuva kulutus. Harjoituspäiväkirjan tietoja voidaan seurata taulukkomuodossa.

Harjoitusten energiankulutuksen lähteenä on käytetty Kuopion yliopiston energiankulutuslaskuria. /1/

4.3.5 Mittauspäiväkirja

Mittauspäiväkirjaan käyttäjä voi lisätä omia fyysisiä tietojaan, joita ovat pituus, paino, eri kehonosien ympärysmitat, leposyke ja maksimisyke. Sovellus laskee automaattisesti käyttäjän painoindeksin, joka lisätään myös mittauspäiväkirjaan.

Rivien lisäys, muokkaus ja poisto toimii samoin kuin ruokapäiväkirjassa ja harjoituspäiväkirjassa. Mittauspäiväkirjan tietoja voidaan seurata taulukkomuodossa sekä graafisena esityksenä.

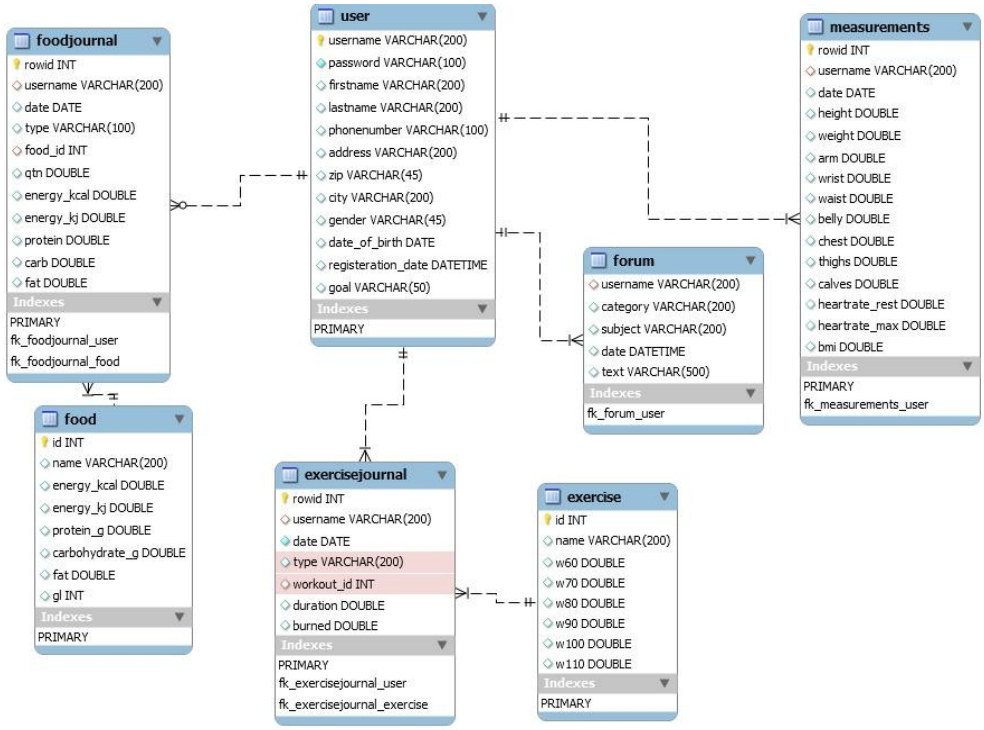
4.3.6 Tilastojen seuranta

Sovelluksen yksi tärkeimmistä ominaisuuksista on tilastojen seuranta, josta saadaan yhteenveto päivittäisestä energiansaannista ja kulutuksesta. Tämän toiminnon avulla pystytään ohjaamaan käyttäjää haluamansa tavoitteen mukaan.

Toiminto hakee ruokapäiväkirjasta ja harjoituspäiväkirjasta yhden päivän kaikki tiedot ja laskee niistä päivän energiatasapainon. Tietoja voidaan seurata taulukkomuodossa sekä graafisena esityksenä.

4.4 Tiedot ja tietokanta

Sovelluksen tarvitsemat tiedot tallennetaan tietokantaan, joka on erillisellä palvelimella. Tietokannassa ovat kuvassa 8 olevan kaaviokuvan mukaiset taulut.



Kuva 8: Tietokanta

4.5 Ulkoiset liittymät

Sovellus toimii web-palvelimella, joka tarjoaa sovellukselle valmiin toimintaympäristön. Sovelluksen käyttö tapahtuu Internet-selaimella, joten käyttäjän ei tarvitse asentaa mitään lisäohjelmia käyttääkseen sovellusta. Sovelluksen tietoliikenne kulkee HTTP-yhteyden yli web-palvelimen ja käyttäjän päätelaitteen välillä.

5 TEKNINEN MÄÄRITTELY

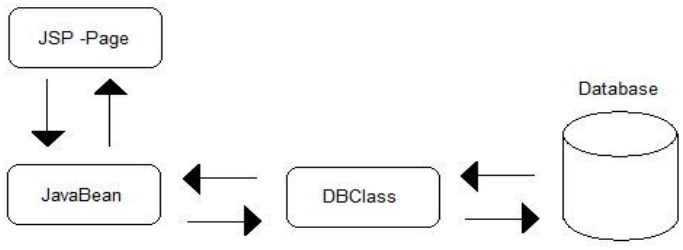
5.1 Johdanto

Henkilökohtainen kunto-ohjaaja -sovellus on työkalu, jonka tarkoituksena on auttaa eritasoisia kuntoilijoita saavuttamaan tavoitteensa ja säilyttämään saavutetut tulokset. Sovellus on tarkoitettu yleiseen käyttöön kaikkien halukkaiden saataville.

Sovellus toteutetaan JavaServer Faces -tekniikkaa käyttäen ja se toimii Apache Tomcat palvelimella, johon käyttäjä ottaa HTTP-yhteyden Internet-selainta käyttäen. Sovellus ei vaadi toimiakseen mitään lisäohjelmia, vaan käyttämiseen riittää nykyaikainen Internet-selain ja avoinna oleva verkkoyhteys.

5.2 Arkkitehtuurin kuvaus

Sovelluksen käyttöliittymän esitys toteutetaan .jsp päätteisillä tiedostoilla, jotka sisältävät käyttöliittymäkomponentit ja funktiokutsut. Kutsuttavat funktiot sijaitsevat Java-luokissa, jotka suorittavat ohjelmakoodin tai kutsuvat tarvittaessa tietokantaluokkien funktioita. Tietokantaluokat hoitavat kommunikoinnin tietokantojen kanssa ja välittävät tiedot edelleen sovelluksen luokille, jotka käsittelevät tiedot ja lähettävät ne sovelluksen käyttöliittymälle esitettäviksi. Kuvassa 9 on esitetty sovelluksen arkkitehtuuri.



Kuva 9: Sovelluksen arkkitehtuuri

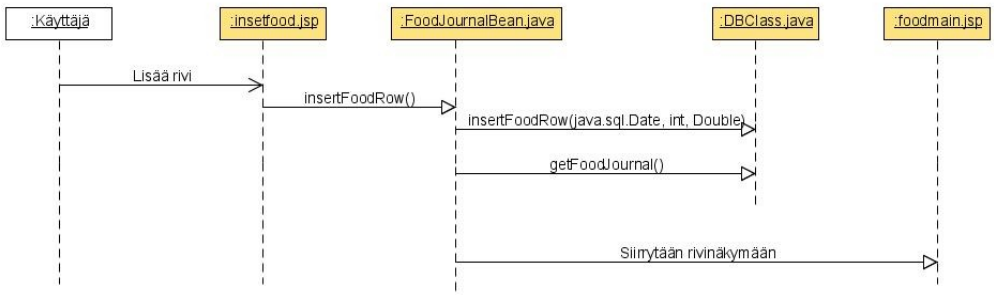
5.3 Moduulisuunnittelu

Sovellus on suunniteltu modulaariseksi siten, että uusien ominaisuuksien lisäys tai vanhojen poisto tapahtuu mahdollisimman vähällä vaivalla. Sovelluksen jokainen ominaisuus koostaa yhden moduulin, johon kuuluu käyttöliittymän esitys sekä Java-papuja, jotka käsittelevät käyttöliittymältä saadut tiedot. Jokaisen moduulin tiedostot jakautuvat kahteen kansioon, joista toinen pitää sisällään .jsp päätteiset tiedostot, jotka esittävät sovelluksen käyttöliittymän ja toinen .java päätteiset Java-pavut, jotka hoitavat tiedon käsittelyn. Oman moduulinsa muodostavat myös tietokantaluokat, jotka kommunikoivat tietokannan kanssa. Sovelluksen keskeisimmät moduulit ovat käyttäjähallinta, ruokapäiväkirja, harjoituspäiväkirja, mittauspäiväkirja, tilastojen seuranta sekä tietokantarajapinta. Seuraavassa käydään läpi selventävänä esimerkkinä käyttäjähallinnan moduuli.

Kansiossa *profile* on käyttäjähallinnan kannalta tarpeelliset .jsp päätteiset sivut, jotka esittävät moduulin käyttöliittymän. Käyttäjähallinnan käyttöliittymän muodostaa kolme eri käyttöliittymänäkymää; modifyUser.jsp, jonka avulla voidaan muokata käyttäjän tietoja, modifyFail.jsp, joka esitetään jos jokin menee vikaan käyttäjätietojen muokkauksessa sekä viewUser.jsp, joka esittää käyttäjän tiedot. Kansiossa *ptrainer.user* sijaitsee moduulin tarvitsemat Java-pavut, jotka suorittavat tietojen muokkauksen sekä kutsuvat tarvittaessa tietokantaluokkien funktioita. Moduuli vaatii toimiakseen kaksi eri Java-papua; *UserBean.java*, joka hoitaa uuden käyttäjän luomisen sekä tietojen toimittamisen käyttöliittymälle ja *ModifyUserBean.java*, joka hoitaa käyttäjätietojen muokkauksen.

6 TOTEUTUS

Tässä luvussa esitellään sovelluksen toteutusta käymällä läpi ruokapäiväkirjarivin lisäys -toiminto. Aluksi toimintoa voidaan tarkastella kuvan 10 sekvenssikaaviosta.



Kuva 10: Ruokapäiväkirjarivin lisäystoiminnon sekvenssikaavio

Käyttäjän saapuessa ruokapäiväkirjan rivin lisäyssivulle näytetään lomake, johon käyttäjä voi lisätä rivin tiedot. Kun käyttäjä on lisännyt tarvittavat tiedot, tallennetaan ne *FoodJournalBean* -luokan muuttujiin, joita voidaan sen jälkeen käyttää kutsuttaessa muita metodeja. Lomakkeen tiedot sidotaan Java -pavun muuttujiin käyttämällä JSF -komponentteja. Seuraavassa on esimerkki päivämäärän välityksestä Java -pavulle.

Päivämäärä sijoitetaan lomakkeessa olevaan tekstikenttään, jonka koodi on esitetty alla.

```
PVM(vvvv-kk-pp): <h:inputText id="insdate" value="#{FoodJournalBean.d1}"  required="true">
                        <f:convertDateTime pattern="yyyy-MM-dd" />
                 </h:inputText>
                 <h:message for="insdate" style="color:red" />
```

Päivämäärä on pakollinen tieto, jonka muodon oikeellisuus tarkistetaan konvertterilla, joka antaa käyttäjälle virheilmoituksen mikäli päivämäärä on väärässä muodossa. Lomakkeen tekstikenttä on sidottu *FoodJournalBean* -luokan muuttujaan *d1* antamalla ominaisuuden *value* arvoksi *#{FoodJournalBean.d1}*.

Kun käyttäjä on lisännyt tarvittavat tiedot lomakkeeseen, painetaan lomakkeessa olevaa Lisää -painiketta, joka kutsuu *FoodJournalBean* -luokan metodia *insertFoodRow*. Painikkeen koodi on esitetty alla.

```
<h:commandButton id="submit" type="submit" value="Lisää"
        action="#{FoodJournalBean.insertFoodRow}"/>
```

Painikkeen ominaisuus action määrittelee mitä tapahtuu kun painiketta painetaan. Tässä tapauksessa painiketta painettaessa kutsutaan *FoodJournalBean* -luokan *insertFoodRow* -metodia.

FoodJournalBean -luokan *insertFoodRow* -metodi muuttaa lomakkeesta saadun päivämäärän sql -muotoiseksi päivämääräksi, kutsuu tietokantaluokan *insertFoodRow* -metodia, sekä palauttaa merkkijonon jonka perusteella sovellus ohjaa käyttäjän asianmukaiselle sivulle.

Metodin palauttama merkkijono on määritelty sovelluksen faces-config.xml -tiedostossa, joka sisältää kaikki sovelluksen tarvitsemat navigaatiosäännöt. Seuraavassa on esitelty sovelluksen tämän osan tarvitsemat navigaatiosäännöt.

```
<navigation-rule>
    <from-view-id>/main/food/insertfood.jsp</from-view-id>
    <navigation-case>
      <from-outcome>insertok</from-outcome>
      <to-view-id>/main/food/foodmain.jsp</to-view-id>
    </navigation-case>
    <navigation-case>
      <from-outcome>insertfail</from-outcome>
      <to-view-id>/main/food/fooderror.jsp</to-view-id>
    </navigation-case>
 </navigation-rule>
```

Mikäli *insertFoodRow* metodi palauttaa merkkijonon ”insertok”, ohjataan käyttäjä ruokapäiväkirjan rivinäkymään. Jos metodi palauttaa merkkijonon ”insertfail”, ohjataan käyttäjä virhesivulle.

Tietokantaluokan metodi *insertFoodRow* -metodille annetaan argumentteina päivämäärä, ruoan tunniste sekä ruoan määrä. Ensin metodi hakee käyttäjänimen, joka on tallennettu istunnon tietoihin sisäänkirjautumisen yhteydessä.

```
FacesContext context = FacesContext.getCurrentInstance();
HttpSession session = (HttpSession) context.getExternalContext().getSession(false);
uname = session.getAttribute("username").toString();
```

Seuraavaksi haetaan ruoan tiedot tietokannasta luomalla asianmukainen kysely, jonka jälkeen tiedot tallennetaan luokan muuttujiin.

```
connectDatabase();
        st = con.createStatement();
        String query = "SELECT * FROM  food where id='" + food_id + "'";
        res = st.executeQuery(query);
        if (res.next()) {
            foodname = res.getString("name");
            energy_kcal = res.getDouble("energy_kcal");
            energy_kj = res.getDouble("energy_kj");
            protein = res.getDouble("protein_g");
            carb = res.getDouble("carbohydrate_g");
            fat = res.getDouble("fat"); }
```

Ruoan tietojen haun jälkeen sovellus laskee ruoan ravintosisällön määrän perusteella ja seuraavaksi lisätään saadut tiedot tietokannan Ruokapäiväkirja -tauluun luomalla asianmukainen sql -lause.

```
ps = con.prepareStatement("insert into foodjournal (username, date, type, food_id, qtn,
energy_kcal, energy_kj, protein, carb, fat ) values (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)");
        ps.setString(1, uname);
        ps.setDate(2, date);
        ps.setString(3, foodname);
        ps.setInt(4, food_id);
        ps.setDouble(5, qtn);
        ps.setDouble(6, energy_kcal);
        ps.setDouble(7, energy_kj);
        ps.setDouble(8, protein);
        ps.setDouble(9, carb);
        ps.setDouble(10, fat);
```

```
int updateResult = ps.executeUpdate();
```

Tietojen lisäyksen jälkeen suljetaan tietokantayhteydet ja palautetaan merkkijono *FoodJournalBean* -luokan *insertFoodRow* -metodille merkiksi tietojen lisäyksen onnistumisesta.

Kun tietokantaluokan *insertFoodRow* -metodi on suoritettu haetaan ruokapäiväkirjan rivit uudelleen kutsumalla *FoodJournalBean* -luokan *getFoodJournal* -metodia, joka kutsuu tietokantaluokan *getFoodJournal*- ja *getFoodNames* -metodeja, jolloin pystytään päivittämään uusimmat tiedot rivinäkymään. *getFoodJournal*- ja *getFoodNames* -metodit palauttavat tiedot *ArrayList* -muodossa *FoodJournalBean* -luokalle.

Kun kaikki metodit on suoritettu onnistuneesti, ohjataan käyttäjä ruokapäiväkirjan rivinäkymään, jossa on esillä kaikki käyttäjän lisäämät rivit. Rivinäkymä on toteutettu JSF:n *dataTable* -komponentilla, jossa voidaan helposti esittää kokoelmamuuttujassa olevia tietoja. *dataTable* -komponentti alustetaan seuraavanlaisesti.

```
<h:dataTable var="item" value="#{FoodJournalBean.items}"
binding="#{FoodJournalBean.dataTable}" border="1" >
```

Yllä olevassa koodissa *dataTable* -komponentti on sidottu *FoodJournalBean* -luokan *HtmlDataTable* -muuttujaan *dataTable*, jonka avulla saadaan muodostettua taulukko *items* nimisestä *ArrayList* -kokoelmasta. Tämän alustuksen jälkeen voidaan *items* -kokoelman tietoja esittää sovelluksen sivulla. Alla olevassa koodissa on esimerkki päivämäärän esityksestä.

```
<h:column>
        <f:facet name="header">
                <h:outputText value="PVM"/>
        </f:facet>
        <h:outputText value="#{item.date}"/>
</h:column>
```

Taulukon sarakkeelle voidaan määritellä otsikko luomalla uusi *facet* -objekti, jonka nimeksi annetaan header ja arvoksi haluttu otsikko. Itse tieto esitetään

outputText -komponentilla asettamalla arvo-ominaisuus vastaamaan oikeaa tietoa kokoelmassa, tässä tapauksessa arvoksi tulee *item.date*.

7 TESTAUS

7.1 Johdanto

Sovelluksen testaus on suoritettu toteutuksen aikaisena testauksena sekä järjestelmätestauksena. Toteutuksen aikaista testausta on suoritettu jokaisen toiminnon ja moduulin valmistumisen jälkeen. Järjestelmätestaus on suoritettu valmiin sovelluksen toiminnan testaukseen.

Toteutuksen aikaisessa testauksessa jokainen toiminto ja moduuli on testattu poikkeustilanteiden varalta syöttämällä sovellukselle hylättäviä sekä hyväksyttyjä syötteitä. Näitä olivat vääränmuotoiset ja oikeanmuotoiset päivämäärät ja sähköpostitunnukset, merkkijonot numeerisiin kenttiin, tyhjät syötteet sekä hyväksyttävät syötteet.

Järjestelmätestauksessa testattiin kaikkien toimintojen ja ominaisuuksien toimintaa integroituna kokonaisuutena testiraportin avulla, missä käydään läpi mahdollisia poikkeustilanteita.

7.2 Testausympäristö

Käyttöliittymän testaus on suoritettu Microsoft Internet Explorer ja Mozilla Firefox -selaimilla. Koodin kääntämistä ja tietokannan toimintaa on testattu NetBeans IDE -kehitystyökalulla.

7.3 Testaustulokset

Testauksen tuloksia voidaan tarkastella taulukosta 2.

Taulukko 2: Testiraportti

Toiminto	**Poikkeustilanne**	**Tulos**
Sisäänkirjautuminen	Käyttäjätunnus tai salasana väärin	ok
Sisäänkirjautuminen	Käyttäjä jo sisäänkirjautuneena	ok
Rekisteröityminen	Puutteelliset tiedot	ok
Rekisteröityminen	Syntymäaika väärässä muodossa	ok
Rekisteröityminen	Käyttäjätunnus on jo rekisteröity	ok
Salasanan vaihto	Salasana väärin	ok
Salasanan vaihto	Uusi salasana ja salasanan vahvistus ei täsmää	ok
Profiilin muokkaus	Syntymäaika väärässä muodossa	ok
Profiilin muokkaus	Puutteelliset tiedot	ok
Ruokapäiväkirja	Rivin lisäyksessä puutteelliset tiedot	ok
Ruokapäiväkirja	Rivin muokkauksessa puutteelliset tiedot	ok
Ruokapäiväkirja	Aikavälin päivämäärät väärässä muodossa	ok
Harjoituspäiväkirja	Rivin lisäyksessä puutteelliset tiedot	ok
Harjoituspäiväkirja	Rivin muokkauksessa puutteelliset tiedot	ok
Harjoituspäiväkirja	Aikavälin päivämäärät väärässä muodossa	ok
Mittauspäiväkirja	Rivin lisäyksessä puutteelliset tiedot	ok
Mittauspäiväkirja	Rivin muokkauksessa puutteelliset tiedot	ok
Mittauspäiväkirja	Aikavälin päivämäärät väärässä muodossa	ok
Tilastojen seuranta	Ei rivejä ruokapäiväkirjassa tai harjoituspäiväkirjassa	ok

8 YHTEENVETO

Opinnäytetyönä toteutettu sovellus vastasi hyvin asetettuja tavoitteita ja lähes kaikki vaatimusmäärittelyssä määritellyt toiminnallisuudet toteutuivat lukuun ottamatta graafisia esityksiä. Tätä raporttia kirjoitettaessa sovelluksen ulkoasu on vielä puutteellinen, mikä jättää sovellukselle runsaasti jatkokehitysmahdollisuuksia.

Työ oli haastava ja mielenkiintoinen, koska sovellus toteutettiin minulle ennestään tuntemattomalla tekniikalla, mikä tarjosi hyvän mahdollisuuden kehittää sekä syventää omaa osaamistani.

KIRJALLISUUTTA

/1/ EnergyNet - Energiankulutuslaskuri eri liikuntamuodoissa [online]. Kuopion yliopisto. Päivitetty 2.1.2009 [viitattu 29.9.2009]. Saatavilla www-muodossa: <http://ffp.uku.fi/cgi-bin/energynet03/energycosts.pl>

/2/ Geary, David – Horstmann, Cay 2007. 2 p. Core JavaServer Faces. Prentice Hall.

/3/ Hamilton, Graham – Cattell, Rick – Fisher, Maydene 1998. 3 p. JDBC Databace Access with Java: A Tutorial and Annotated Reference. Addison-Wesley.

/4/ Meloni, Julie 2003. MySQL Trainer Kit. Edita.

/5/ Schutta, Nathaniel – Asleson, Ryan 2006. Pro Ajax and Java. Apress.