**Sprintin tehtävät**

**Gitin käyttöönotto:**

Loimme ryhmällemme GithHub-tilan. Yhdistimme GithHubiin myös ryhmämme Slack-tiliin, jotta ryhmäläiset olisivat mahdollisimman ajan tasalla mahdollisista muutoksista.

**Tietokantasuunnittelu:**

Suunnittelimme tietokannan relaatiokaavion.

Toteutimme Pizza-luokan, Pizzan täytteet-luokan ja Täyte-luokan MariaDB-tietokantaan.



**Täyte-luokka:**

Loimme tietokantaan Täyte-luokan.

CREATE TABLE Tayte

(

tayteid INT UNSIGNED NOT NULL AUTO\_INCREMENT,

taytenimi VARCHAR(25) NOT NULL,

saatavilla BOOLEAN NOT NULL,

PRIMARY KEY(tayteid)

);

Lisäsimme Tayte-luokkaan arvoja.

INSERT INTO Tayte (taytenimi, saatavilla)

VALUES ('Kinkku', 1);

​

INSERT INTO Tayte (taytenimi, saatavilla)

VALUES ('Juusto', 1);

​

INSERT INTO Tayte (taytenimi, saatavilla)

VALUES ('Jauheliha', 1);

​

INSERT INTO Tayte (taytenimi, saatavilla)

VALUES ('Ananas', 1);

​

INSERT INTO Tayte (taytenimi, saatavilla)

VALUES ('Tomaatti', 1);

​

INSERT INTO Tayte (taytenimi, saatavilla)

VALUES ('Kebab', 1);

​

INSERT INTO Tayte (taytenimi, saatavilla)

VALUES ('Salami', 1);

​

INSERT INTO Tayte (taytenimi, saatavilla)

VALUES ('Kana', 1);

**Pizza-luokka:**

Loimme tietokantaan Pizza-luokan.

CREATE TABLE Pizza

(

pizzaid INT UNSIGNED NOT NULL AUTO\_INCREMENT,

pizzanimi VARCHAR(25) NOT NULL,

hinta DECIMAL(6,2) NOT NULL,

PRIMARY KEY(pizzaid)

);

Lisäsimme Pizza-luokkaan arvoja.

INSERT INTO Pizza (pizzanimi,hinta)

VALUES ('Margareta', 10.00);

​

INSERT INTO Pizza (pizzanimi,hinta)

VALUES ('Tropicana', 14.99);

​

INSERT INTO Pizza (pizzanimi,hinta)

VALUES ('Porakana', 18.00);

​

INSERT INTO Pizza (pizzanimi,hinta)

VALUES ('Alkopala', 19.50);

​

**Pizzantaytteet-luokka:**

Loimme tietokantaan Pizzantaytteet-luokan.

CREATE TABLE Pizzantaytteet

(

pizzaid INT UNSIGNED NOT NULL,

tayteid INT UNSIGNED NOT NULL,

PRIMARY KEY(pizzaid, tayteid),

FOREIGN KEY(pizzaid) REFERENCES Pizza(pizzaid),

FOREIGN KEY(tayteid) REFERENCES Tayte(tayteid)

);

Pizzantaytteet-luokassa loimme suhteet Pizza-luokan ja Tayte-luokan välille. Näin tietty pizza saa omat täytteensä automaattisesti.

insert into Pizzantaytteet (pizzaid, tayteid) values (?, ?);

**Tietokannan tulostus käyttäjälle:**

Loimme Eclipse-ohjelman avulla Kontrolleri.java-servletin.

Do-Get-metodi:

pDAO.avaaYhteys();

List<Pizza> lista;

List<Tayte> lista2;

try {

lista = pDAO.haePizzat();

request.setAttribute("pizzalista", lista);

lista2 = pDAO.haeTaytteet();

request.setAttribute("taytelista", lista2);

} catch (NumberFormatException e) {

e.printStackTrace();

} catch (SQLException e) {

e.printStackTrace();

} finally {

pDAO.suljeYhteys();

}

// Pistetään tieto eteenpäin list.jsp:lle

RequestDispatcher disp = request.getRequestDispatcher("list.jsp");

disp.forward(request, response);

DoPost-metodi:

//luodaan pizzalista

List<Pizza> lista = new ArrayList<Pizza>();

//haetaan parametrit käyttäjältä(submitista) ja lisätään nelopuksi pizza-olioon

String nimi = request.getParameter("nimi");

double hinta = Double.parseDouble(request.getParameter("hinta"));

int id = lista.size() +1;

String taytteet = "hahaa";

Pizza a = new Pizza(id, nimi, hinta, taytteet);

//lisätään pizza-olio tietokantaan PizzaDAO-java luokan avulla.

PizzaDAO pDao = new PizzaDAO();

pDao.avaaYhteys();

pDao.lisaaPizza(a);

pDao.suljeYhteys();

response.setContentType("text/html");

//java.io.PrintWriter wout = response.getWriter();

System.out.println("<b>Nimi:</b> " + a.getPizzanimi());

System.out.println("<br>");

System.out.println("<b>Hinta:</b> " + f.format(a.getHinta()));

//ohjataan takaisin alkuun

response.sendRedirect("Kontrolleri?added=true");

Loimme Eclipse-ohjelman avullaPizzaDAO.java(Data Access Object)-luokan.

PizzaDAO-luokan alku, jossa luodaan listat:

public PizzaDAO() {

try {

Class.forName("org.mariadb.jdbc.Driver");

} catch (java.lang.ClassNotFoundException ex) {

System.out.print("ClassNotFoundException: ");

System.out.println(ex.getMessage());

}

}

public ArrayList<Pizza> pizzalista = new ArrayList<Pizza>();

public ArrayList<Tayte> taytelista = new ArrayList<Tayte>();

Connection yhteys = null;

AvaaYhteys-metodi:

public void avaaYhteys() {

// Käyttäjätiedot ja DB:n osoite - lisää puuttuvat tiedot!

String username = "a1500925";

String password = "nySUxd46r";

String url = "jdbc:mariadb://localhost/a1500925";

try {

// Yhdistetään tietokantaan

yhteys = DriverManager.getConnection(url, username, password);

} catch (SQLException ex) {

System.out.println("Virhe yhteyden avaamisessa");

}

}

SuljeYhteys-metodi:

public void suljeYhteys() {

try {

if (yhteys != null && !yhteys.isClosed())

yhteys.close();

} catch (Exception e) {

System.out.println("Tietokantayhteys ei jostain syystÃ¤ suostu menemÃ¤Ã¤n kiinni.");

e.printStackTrace();

}

}

haePizzat-metodi:

public List<Pizza> haePizzat() throws NumberFormatException, SQLException {

String sql = "SELECT p.pizzaid, p.pizzanimi, p.hinta, GROUP\_CONCAT(t.tayteid) as tayteidt, GROUP\_CONCAT(t.taytenimi) as taytenimet FROM Pizza p LEFT JOIN Pizzantaytteet pt ON p.pizzaid = pt.pizzaid LEFT JOIN Tayte t ON t.tayteid = pt.tayteid GROUP BY p.pizzaid order by pizzaid, null";

Statement haku = yhteys.createStatement();

ResultSet tulokset = haku.executeQuery(sql);

while (tulokset.next()) {

int id = tulokset.getInt("pizzaid");

String nimi = tulokset.getString("pizzanimi");

double hinta = tulokset.getDouble("hinta");

String taytteet = tulokset.getString("taytenimet");

// lisätään pizza listaan

Pizza pizza = new Pizza(id, nimi, hinta, taytteet);

pizzalista.add(pizza);

}

return pizzalista;

}

haeTaytteet-metodi:

public List<Tayte> haeTaytteet() throws NumberFormatException, SQLException {

String sql = "SELECT \* FROM Tayte";

Statement haku = yhteys.createStatement();

ResultSet tulokset = haku.executeQuery(sql);

while (tulokset.next()) {

int tayteid = tulokset.getInt("tayteid");

String taytenimi = tulokset.getString("taytenimi");

boolean saatavilla = tulokset.getBoolean("saatavilla");

// lisätään pizza listaan

Tayte tayte = new Tayte(tayteid, taytenimi, saatavilla);

taytelista.add(tayte);

}

return taytelista;

}

lisaaPizza-metodi:

public void lisaaPizza(Pizza p) {

try {

// alustetaan sql-lause

String sql = "INSERT INTO Pizza(pizzanimi, hinta) VALUES(?,?)";

PreparedStatement lause = yhteys.prepareStatement(sql);

// täydennetään puuttuvat tiedot (eli pizzan nimi ja hinta)

lause.setString(1, p.getPizzanimi());

lause.setDouble(2, p.getHinta());

// suoritetaan lause

lause.executeUpdate();

System.out.println("Pizza" + p + "lisätty tietokantaan");

} catch (Exception e) {

// Tapahtui jokin virhe

System.out.println("Pizzan lisäämisyritys aiheutti virheen!");

}

}

Loimme Eclipse-ohjelman avulla Pizza-luokan, Tayte-luokan ja Pizzantaytteet-luokan.

public class Pizza {

private int pizzaid;

private String pizzanimi;

private double hinta;

private String taytteet;

...konstruktorit, setterit, getterit ja tostringit...

}

public class Tayte {

private int tayteid;

private String taytenimi;

private boolean saatavilla;

...konstruktorit, setterit, getterit ja toStringit...

}

Loimme Eclipse-ohjelman avulla list.jsp dynamic web project-tiedoston.

list.jsp:ssä oleva JSTL-koodi, joka tulostaa pizzan täytteineen:

<c:forEach items="${pizzalista}" var="pizza">

<tr><td><c:out value="${pizza.pizzaid}"/> <td><c:out value="${pizza.pizzanimi}"/><br><div class="taytenimi"><c:out value="${pizza.taytteet}"/></div></td><td><c:out value="${pizza.hinta}"/></td></tr>

</c:forEach>

list.jsp:ssä oleva JSTL-koodi, joka tulostaa täytteet:

<c:forEach items="${taytelista}" var="tayte">

<tr><td><c:out value="${tayte.tayteid}"/> <td><div class="taytenimi"><c:out value="${tayte.taytenimi}"/></div></td><td><c:if test="${tayte.saatavilla}"><c:out value="kyllä"/></c:if><c:if test="${!tayte.saatavilla}"><c:out value="ei"/></c:if></td></tr>

</c:forEach>

Tulostimme tietokannan serverille. Serverille tuli näkymään pizzan id, nimi, hinta, täytteet ja täytteiden saatavuus:

SELECT p.pizzaid, p.pizzanimi, p.hinta, GROUP\_CONCAT(t.tayteid) as tayteidt,

GROUP\_CONCAT(t.taytenimi) as taytenimet

FROM Pizza p

LEFT JOIN Pizzantaytteet pt ON p.pizzaid = pt.pizzaid

LEFT JOIN Tayte t ON t.tayteid = pt.tayteid

GROUP BY p.pizzaid

order by pizzaid, null;

**Dokumentointi:**

Dokumentoimme sprintti 1. vaiheet word-tiedostoon.