Krusty Kookies Sweden AB: Programmerings Projekt in Databasteknik (EDA216)

EDA, Institutionen för Datavetenskap, Lunds Tekniska Högskola

Per Holm

Adam Hansson Lyrén, C11 (dic11aha@student.lu.se)
Mergim Rama, C11 (dic11mra@student.lu.se)

4 april 2013

LUNDS UNIVERSITET

Lunds Tekniska Högskola

Innehåll

1 Introduktion

Projektets uppgift var att konstruera ett program som tar hand om alla delar som påverkas av en kakproduktion på företaget, allt från att tillverka en pall kakor, till att minska mängden råvaror på lagret. Vi ska även implementera funktioner för att blockera och söka på specifika pallar, vilka tillsammans presenteras i ett program som simulerar pallproduktionen. Allt detta kräver att man designat och implementerat en databas samt ett GUI för programmet.

2 System

Skriva om hur vi kopplar databasen till klasserna och vilka klasser det är som tar hand om kopplingen, vilka versioner av PHP MyQSL mm vi använder. Hur vi delat upp GUI klasserna.

3 Implementation

Projektet består i grunden av två huvudsakliga implementationer, vilka är databasimplementationen samt GUI implementationen. Dessa implementationer har olika uppgifter i projektet. Databasen kommer stå för sparandet av viktig information samt att kunna få ut det man vi ha ut ur informationen. GUIt ska göra programmet användarvänligt så att även den minst tekniske ska kunna använda det utan att ha någon förkunskap om funktionaliteten.

3.1 Databasimplementation

För att kunna implementera en databas

Databasen vi implementerade är av typen MySQL och går att hitta på institutionens egna server som de tillhandager EDA216 studenter. Serverns adress är puccini.cs.lth.se. För att göra våra querys säkra mot attacker använder vi oss av preparedStatements, vilka vi skriver i php.

Eftersom att flera ska vara inne samtidigt och utföra handlingar i databasen inte var ett krav så beslöt vi oss för att inte implementera transactions. All SQL kod vi använt för att skapa databasen går att hitta under *6. Databasdump.*

3.2 GUI Implementation

När vi började med vårt GUI till programmet ville vi göra det så enkelt som möjligt för användaren att interagera med tjänsten. Vi började först att skapa en egen design vilket tog sin tid och visade sig vara svårare än vi trodde. Då övergick vi till att använda Twitters egna designpaket, Bootstrap som är opensource där vem som helst får använda sig av den i sina egna sidor. När vi gick över till Bootstrap blev allt mycket lättare och vi kunde få till en väldigt snygg och användarvänlig design på programmet.

Vi började med att skapa en tydlig förstasida där användaren får välja vilken tjänst han vill använda. Det han har att välja emellan är:

- Rawmaterial & Recepies
- Production

- Orders & Deliveries

Eftersom vi endast skulle koncentrera oss på produktionen så fungerar inte de andra knapparna.

När man kommer till andra sidan, det vill säga efter att ha klickat sig in på produktion finns en vy med alla pallar som skapats av olika företag med olika ordrar. Det är väldigt snyggt upplagt vilket för det enkelt för användaren att navigera bland pallarna. Användaren kan också direkt från listan med pallar redigera en pall och få den ej godkänd samt ta bort den.

På övre delen av sidan finns det knappar som leder dig till just denna sidan, sidan för att skapa en ny pall.

Utöver detta finns det ett sökfält där användaren grafiskt kan bestämma intervallet på det datum han vill markera samt vad han vill söka på. Det smart med sökfältet är att användaren kan söka på vad som helst, förutsatt att det finns i tabellen. Det gör att användaren aldrig behöver följa någon hjälptext för att veta vad sökfältet söker efter.

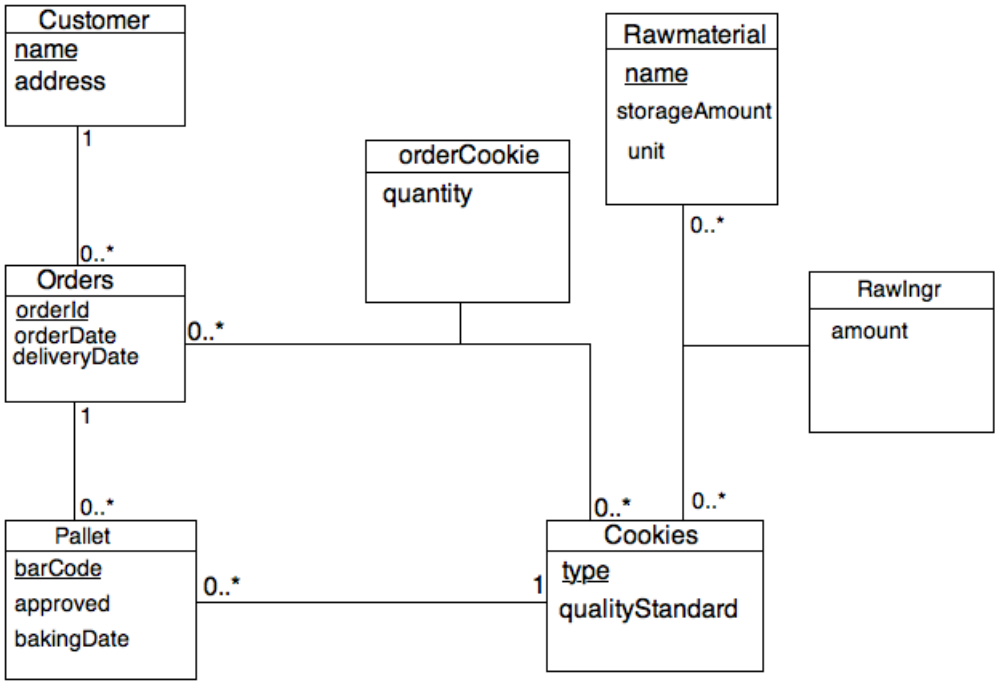
Alla knappar som finns på sidorna är väldigt stora, tydliga och färgglada. På så sätt har vi sett över behöven för i princip alla olika användare. På alla submenyer till produktionen finns även en inramning runt det som skall göras. Detta påvisar och underlättar för användaren att veta vad han skall göra samt att allt får en sorterad struktur som ger sidan en proffsigare utformning.

4 Modellering

Att få fram ett bra E/R diagram till programmet är inte lätt. Man måste läsa beskrivningen av problemet som introducerats och analysera det in i minsta detalj för att sedan kunna skapa ett skapligt diagram. Vi satt och och läste texten några gånger och fick slutligen fram en någorlunda design. Efter att ha filat och gjort om skickade vi in den till projektansvarige för att få bekräftelse på om diagrammet ser bra ut eller om det behöver någon ändring.

När vi fick tillbaka feedbacken var det enstaka punkter som korrigerades. Nu kunde vi övergå till relationsöversättning för att sedan kunna implementera databasen. E/R diagrammet syns på bilden nedan.

4.1 E/R Diagram



5 Relationer

Databasen vi har skapat består av 7 relationer varav två av relationerna är referenstabeller för många-till-många relationer. För att få fram en så korrekt databas som möjligt utgår man från det E/R diagram man modellerat och skapar relationerna utifrån den.

5.1 Relationerna

- Customer(name, address)
- Order(orderId, *customerName*, orderDate, deliveryDate)
- Cookies(type, qStandard)
- OrderCookie(*orderId*, *cookieType*, quantity)
- Pallet(barCode, *cookieType*, *orderId*, approved, bakingDate)
- Rawmaterial(name, storageAmount, unit)
- Rawingr(*cookieType*, *rawMaterialName*, amount)

Eftersom våra relationer saknar funktionella beroenden mellan attributen och att de endast har nyckelberoenden så är våra relationer i BCNF.

6 Databasdump

För att visa hur vi skapat databasen och för att andra ska kunna återskapa den har vi skapat en dump av alla de kommando vi använt. Koden som användes vid utförandet av databasen är bifogad precis under.

6.1 SQL Kod

```
CREATE TABLE `cookies` (

  `type` varchar(40) NOT NULL DEFAULT '',
  `qualityStandard` int(3) NOT NULL DEFAULT '90',
  PRIMARY KEY (`type`)
);

CREATE TABLE `rawingr` (
  `cookieType` varchar(40) NOT NULL DEFAULT '',
  `rawMaterialName` varchar(40) NOT NULL DEFAULT '',
  `amount` double(10,0) NOT NULL,
  PRIMARY KEY (`cookieType`,`rawMaterialName`),
   FOREIGN KEY (`cookieType`) REFERENCES `cookies` (`type`),
  FOREIGN KEY (`rawMaterialName`) REFERENCES `rawmaterial` (`name`)
);

CREATE TABLE `orders` (
  `orderId` int(10) NOT NULL AUTO_INCREMENT,
  `orderDate` date NOT NULL,
  `deliveryDate` date DEFAULT NULL,
  `customerName` varchar(40) NOT NULL DEFAULT '',
  PRIMARY KEY (`orderId`),
  FOREIGN KEY (`customerName`) REFERENCES `customer` (`name`)
);

CREATE TABLE `customer` (
  `name` varchar(40) NOT NULL DEFAULT '',
  `address` varchar(40) NOT NULL DEFAULT '',
  PRIMARY KEY (`name`)
);

CREATE TABLE `ordercookie` (
  `quantity` int(10) NOT NULL,
  `orderId` int(10) NOT NULL,
  `cookieType` varchar(40) NOT NULL,
  PRIMARY KEY (`orderId`,`cookieType`),
  FOREIGN KEY (`orderId`) REFERENCES `orders` (`orderId`),
  FOREIGN KEY (`cookieType`) REFERENCES `cookies` (`type`)
);

CREATE TABLE `rawmaterial` (
  `name` varchar(40) NOT NULL DEFAULT '',
  `storageAmount` double(10,0) NOT NULL,
  `unit` varchar(45) DEFAULT 'g',
  PRIMARY KEY (`name`)
);

CREATE TABLE `pallet` (
  `barCode` int(10) NOT NULL AUTO_INCREMENT,
  `cookieType` varchar(40) NOT NULL,
  `location` varchar(45) NOT NULL DEFAULT 'Freezing Storage',
  `orderId` int(10) NOT NULL,
```

```
  `approved` tinyint(4) NOT NULL DEFAULT '1',
  `bakingDate` date NOT NULL,
  PRIMARY KEY (`barCode`),
  FOREIGN KEY (`cookieType`) REFERENCES `cookies` (`type`),
  FOREIGN KEY (`orderId`) REFERENCES `orders` (`orderId`)
);
```