Vi har ved fremstillingen af vores UML diagrammet benyttet PlantUML (<http://plantuml.com/>) og PlantText (<https://www.planttext.com/>). ER-diagrammerne er lavet ved hjælp af MySQL workbench og har derfor ikke kildekode.

## Kildetekst til figur 1:

@startuml

title Domain - Before

class User {

username

password

balance

}

class Order{

user

date

orderline List

}

class Orderline {

cupcake

quantity

price

}

class Cupcake{

bottom

topping

price

}

class Topping {

name

price

}

class Bottom {

name

price

}

User "!" -- "\*" Order

Order "1" -- "\*" Orderline

Orderline "1" -- "\*" Cupcake

Bottom "1" -- "\*" Cupcake

Topping "1" -- "\*" Cupcake

@enduml

## Kildetekst til figur 2:

@startuml

title Domain - After

class User {

name

password

balance

email

admin

}

class Order{

date : String

total\_price

}

class Orderline {

quantity

price

}

class ShoppingCart{

total\_price

}

class Topping {

name

price

}

class Bottom {

name

price

}

ShoppingCart "1" -- "\*" Orderline

User "1" -- "\*" Order

Order "1" -- "\*" Orderline

Bottom "1" -- "\*" Orderline

Topping "1" -- "\*" Orderline

@enduml

## Kildetekst til figur 4:

@startuml

Title Cupcake: Navigationsdiagram

State index.jsp

State topMenu.jsp

State shopCart.jsp : topmenu (user)

State adminPage.jsp : topmenu (admin)

State registration.jsp

State register\_completed.jsp

State showOrders.jsp : topmenu (user)

State showOrders.jsp : topmenu (admin)

State showOrderInfo.jsp : topmenu (user)

State showOrderInfo.jsp : topmenu (admin)

State orderFinished.jsp : topmenu (user)

[\*] -> index.jsp

index.jsp --> shopCart.jsp : Login\_registration.java

index.jsp --> adminPage.jsp : Login\_registration.java

index.jsp --> registration.jsp : Login\_registration.java

registration.jsp --> register\_completed.jsp

register\_completed.jsp --> index.jsp

shopCart.jsp --> showOrders.jsp :

adminPage.jsp --> showOrders.jsp

shopCart.jsp --> shopCart.jsp : GenerateOrderLine.java

showOrders.jsp --> showOrderInfo.jsp

shopCart.jsp --> orderFinished.jsp : GenerateOrder.java

@enduml

## Kildetekst til figur 5:

1. @startuml
2. scale 400 width
3. [\*] --> topMenu.jspNotLoggedIn
4. state topMenu.jspNotLoggedIn {
5. [\*] --> index.jsp
6. index.jsp --> registration.jsp
7. registration.jsp --> register\_completed.jsp
8. registration.jsp --> index.jsp
9. register\_completed.jsp --> index.jsp
10. index.jsp --> topMenu.jspAdmin
11. index.jsp --> topMenu.jspUser
12. }
13. state topMenu.jspAdmin {
14. [\*] --> adminPage.jsp
15. adminPage.jsp --> showOrders.jspAdmin
16. showOrders.jspAdmin --> showOrderInfo.jspAdmin
17. showOrderInfo.jspAdmin --> showOrders.jspAdmin
19. }
21. state topMenu.jspUser {
22. [\*] --> shopCart.jsp
23. shopCart.jsp --> orderFinished.jsp
24. orderFinished.jsp --> shopCart.jsp
25. orderFinished.jsp --> showOrders.jspUser
26. shopCart.jsp --> showOrders.jspUser
27. showOrders.jspUser --> shopCart.jsp
28. showOrders.jspUser --> showOrderInfo.jspUser
29. showOrderInfo.jspUser --> showOrders.jspUser
30. showOrderInfo.jspUser --> shopCart.jsp
31. }
33. topMenu.jspAdmin --> [\*]
34. topMenu.jspUser --> [\*]
36. }
37. @enduml

## Kildetekst til figur 6:

@startuml

title "Finish Order. - Sequence Diagram"

actor Kunde

boundary shopCart.jsp

boundary orderFinished.jsp

participant HttpSession

participant HttpRequest

control GenerateOrder

participant UserMapper

participant OrderMapper

participant CupcakeMapper

entity ShoppingCart

entity User

entity Order

entity Orderline

collections ResultSet

database cupcakeSQL

Kunde -> shopCart.jsp : "Finish Order"

shopCart.jsp -> GenerateOrder: http GET

GenerateOrder -> HttpSession: getAttribute()

HttpSession --> GenerateOrder: shopCart

GenerateOrder -> ShoppingCart: getTotal\_price()

ShoppingCart --> GenerateOrder: totalPrice

GenerateOrder -> HttpSession: getAttribute()

HttpSession --> GenerateOrder: user

GenerateOrder -> User: getBalance()

User --> GenerateOrder: balance

GenerateOrder -> Order: createOrder()

Order --> GenerateOrder: newOrder

GenerateOrder -> OrderMapper: putToOrderTable()

OrderMapper -> cupcakeSQL: executeUpdate()

OrderMapper --> GenerateOrder: orderNumber

GenerateOrder -> OrderMapper: getInitOrderById()

OrderMapper -> cupcakeSQL: executeQuery()

cupcakeSQL --> ResultSet: rs

OrderMapper -> UserMapper: createUserMapper()

UserMapper --> OrderMapper

OrderMapper -> ResultSet: getInt()

ResultSet --> OrderMapper: user\_id

OrderMapper -> ResultSet: getString()

ResultSet --> OrderMapper: date

OrderMapper -> UserMapper: getUserByID()

UserMapper --> OrderMapper: user

OrderMapper -> Order: createOrder()

Order --> OrderMapper: order

OrderMapper -> Order: setDate()

Order --> OrderMapper: order

OrderMapper -> Order: setOrder\_id()

Order --> OrderMapper: order

OrderMapper --> GenerateOrder: newOrder

GenerateOrder -> HttpRequest: setAttribute()

GenerateOrder -> ShoppingCart: getOrderlines()

ShoppingCart --> GenerateOrder: orderlines

loop Orderline ol : orderlines

GenerateOrder -> OrderMapper: putToOrderLineTable()

OrderMapper --> GenerateOrder: orderlineNumber

GenerateOrder -> Orderline: getQuantity()

Orderline --> GenerateOrder

GenerateOrder -> OrderMapper: putToOrderdetailsTable()

OrderMapper -> cupcakeSQL: executeUpdate()

end

GenerateOrder -> User: getBalance()

User --> GenerateOrder: balance

GenerateOrder -> GenerateOrder: newBalance = balance - totalPrice

GenerateOrder -> UserMapper: updateUserBalanceById()

GenerateOrder --> orderFinished.jsp: forward

orderFinished.jsp --> Kunde: "Order Finished.."

@enduml

## SQL script (create tables + insert data):

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='TRADITIONAL,ALLOW\_INVALID\_DATES';

-- -----------------------------------------------------

-- Schema cupcake

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `cupcake` DEFAULT CHARACTER SET utf8 ;

USE `cupcake` ;

-- -----------------------------------------------------

-- Table `cupcake`.`users`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `cupcake`.`users` (

`user\_id` INT NOT NULL AUTO\_INCREMENT,

`username` VARCHAR(45) NOT NULL,

`password` VARCHAR(45) NOT NULL,

`balance` DECIMAL(10,2) UNSIGNED NOT NULL,

`email` VARCHAR(45) NULL,

`administrator` TINYINT NOT NULL,

PRIMARY KEY (`user\_id`))

ENGINE = InnoDB;

INSERT INTO users (username,password,balance,email,administrator) VALUE

('admin','1234',5000.00,'admin@cupcake.dk',1),

('testuser','1234',10000.00,'testuser@cupcake.dk',0);

-- -----------------------------------------------------

-- Table `cupcake`.`orders`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `cupcake`.`orders` (

`order\_id` INT NOT NULL AUTO\_INCREMENT,

`date` DATETIME NOT NULL DEFAULT CURRENT\_TIMESTAMP,

`users\_user\_id` INT NOT NULL,

PRIMARY KEY (`order\_id`),

INDEX `fk\_orders\_users1\_idx` (`users\_user\_id` ASC),

CONSTRAINT `fk\_orders\_users1`

FOREIGN KEY (`users\_user\_id`)

REFERENCES `cupcake`.`users` (`user\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `cupcake`.`bottoms`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `cupcake`.`bottoms` (

`bottom\_id` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NOT NULL,

`price` DECIMAL(10,2) UNSIGNED NOT NULL,

PRIMARY KEY (`bottom\_id`))

ENGINE = InnoDB;

INSERT INTO bottoms (name,price) VALUE

('Chocolate',5.00),

('Vanilla',5.00),

('Nutmeg',6.00),

('Pistacio',6.00),

('Almond',7.00);

-- -----------------------------------------------------

-- Table `cupcake`.`toppings`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `cupcake`.`toppings` (

`topping\_id` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NOT NULL,

`price` DECIMAL(10,2) UNSIGNED NOT NULL,

PRIMARY KEY (`topping\_id`))

ENGINE = InnoDB;

INSERT INTO toppings (name,price) VALUE

('Chocolate',5.00),

('Blueberry',5.00),

('Rasberry',5.00),

('Crispy',6.00),

('Strawberry',6.00),

('Rum/Raisin',7.00),

('Orange',8.00),

('Lemon',8.00),

('Blue cheese',9.00);

-- -----------------------------------------------------

-- Table `cupcake`.`orderlines`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `cupcake`.`orderlines` (

`orderline\_id` INT NOT NULL AUTO\_INCREMENT,

`price` DECIMAL UNSIGNED NOT NULL,

`bottoms\_bottom\_id` INT NOT NULL,

`toppings\_topping\_id` INT NOT NULL,

PRIMARY KEY (`orderline\_id`),

INDEX `fk\_orderlines\_bottoms1\_idx` (`bottoms\_bottom\_id` ASC),

INDEX `fk\_orderlines\_toppings1\_idx` (`toppings\_topping\_id` ASC),

CONSTRAINT `fk\_orderlines\_bottoms1`

FOREIGN KEY (`bottoms\_bottom\_id`)

REFERENCES `cupcake`.`bottoms` (`bottom\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_orderlines\_toppings1`

FOREIGN KEY (`toppings\_topping\_id`)

REFERENCES `cupcake`.`toppings` (`topping\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `cupcake`.`orderdetails`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `cupcake`.`orderdetails` (

`orders\_order\_id` INT NOT NULL,

`orderlines\_orderline\_id` INT NOT NULL,

`quantity` INT NOT NULL,

PRIMARY KEY (`orders\_order\_id`, `orderlines\_orderline\_id`),

INDEX `fk\_orderdetails\_orders1\_idx` (`orders\_order\_id` ASC),

CONSTRAINT `fk\_orderdetails\_orderlines1`

FOREIGN KEY (`orderlines\_orderline\_id`)

REFERENCES `cupcake`.`orderlines` (`orderline\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_orderdetails\_orders1`

FOREIGN KEY (`orders\_order\_id`)

REFERENCES `cupcake`.`orders` (`order\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

CREATE VIEW cupcake.vieworderlinedetails AS

SELECT \* FROM cupcake.orderdetails JOIN cupcake.orderlines ON orderdetails.orderlines\_orderline\_id = orderlines.orderline\_id;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;