OLSKER CUPCAKE

Presented by
Esben Moldt
Styrbjørn Gullacksen
Gruppe 14

Kundens behov

Olsker Cupcakes har mange bestillinger og vil gøre det nemt for kunderne.

Kunderne ønsker at designe deres cupcakes online.

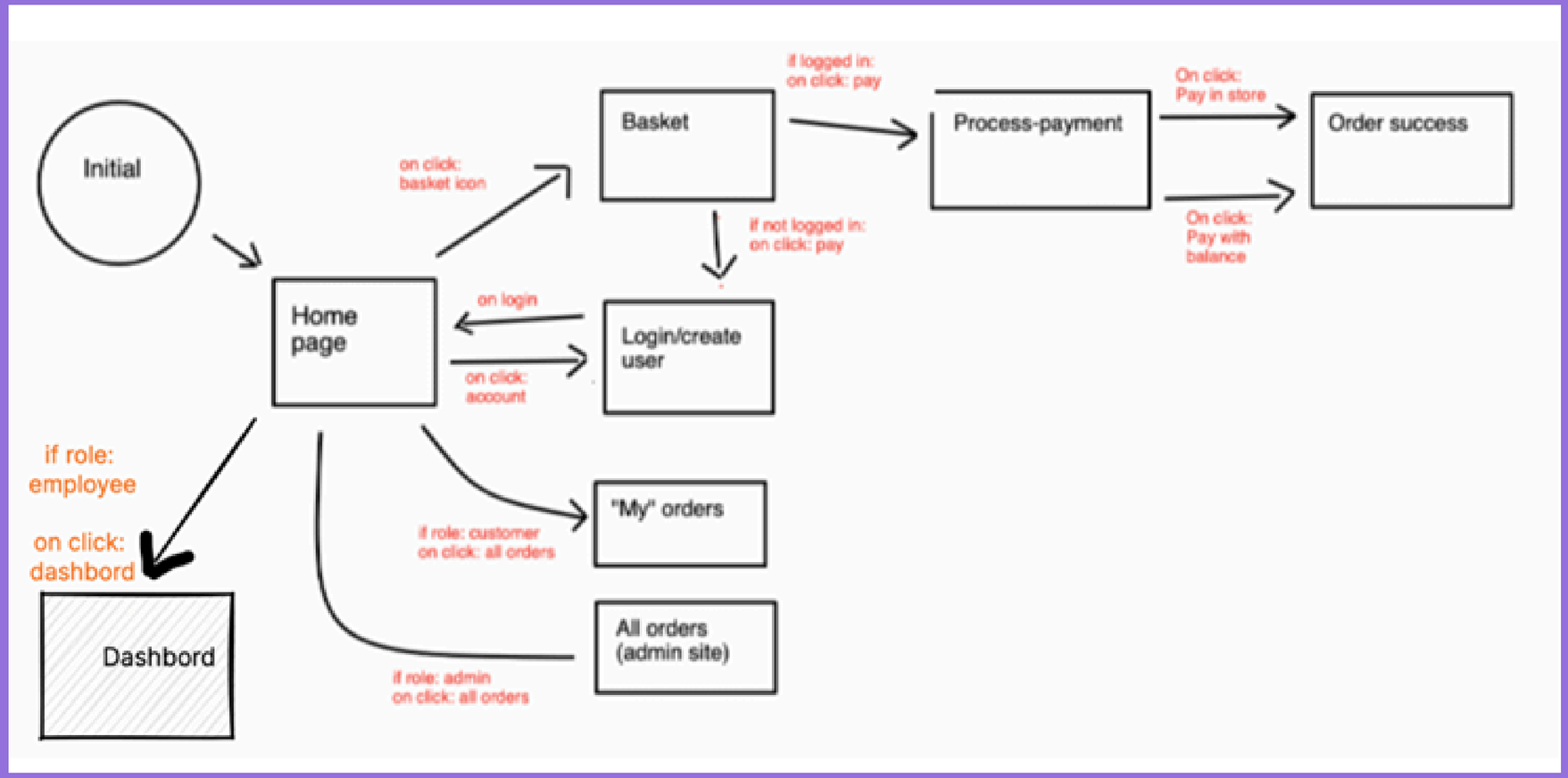
De vil hurtigt kunne hente dem i butikken uden ventetid.

Vores mål

Bygge en nem og brugervenlig webshop.

Mulighed for at bestille og betale online.

Fokus på en simpel første version (MVP) – så vi kan teste og videreudvikle.


Initial
Home page
Basket
on click: basket icon
if logged in: on click: pay
Process-payment
On click: Pay in store
Order success
On click: Pay with balance
if not logged in: on click: pay
on login
on click: account
Login/create user
if role: employee
on click: dashbord
Dashbord
"My" orders
if role: customer on click: all orders
All orders (admin site)
if role: admin on click: all orders

Opfyldte krav

User case 1-6

User case 7-9

Vigtigste ekstra features

Ekstra betaling muligheder

Visualisering af cupcakes

Brugerprofil opdatering

VIDEREUDVIKLING

UX

Kvittering pr mail

Udvidet produktinformation

Levering

Økonomi

Nye varegrupper

Levering

POS

USER STORY 1


Cupcake
Pick cupcake bottom
Chocolate
Price: 5.0
Vanilla
Price: 5.0
Nutmeg
Price: 5.0
Pistachio
Price: 6.0
Almond
Price: 7.0
Pick cupcake top
Chocolate
Price: 5.0
Blueberry
Price: 5.0
Raspberry
Price: 5.0
Crispy
Price: 6.0
Strawberry
Price: 6.0
Rum/Raisin
Price: 7.0
Orange
Price: 8.0
Lemon
Price: 8.0
Blue cheese
Price: 9.0
1
Total price: 0.00 DKK
Add to basket

Pick cupcake bottom

Chocolate
Price: 5.0

Vanilla
Price: 5.0

Nutmeg
Price: 5.0

Pistachio
Price: 6.0

Pick cupcake top

Chocolate
Price: 5.0

Blueberry
Price: 5.0

Raspberry
Price: 5.0

Crispy
Price: 6.0

Strawberry
Price: 6.0

Rum/Raisin
Price: 7.0

Orange
Price: 8.0

Lemon
Price: 8.0

Blue cheese
Price: 9.0

1

Total price: 10.00 DKK

Add to basket

Pick cupcake bottom

Chocolate
Price: 5.0

Vanilla
Price: 5.0

Nutmeg
Price: 5.0

Pistachio
Price: 6.0

Almond
Price: 7.0

Pick cupcake top

Chocolate
Price: 5.0

Blueberry
Price: 5.0

Raspberry
Price: 5.0

Crispy
Price: 6.0

Strawberry
Price: 6.0

Rum/Raisin
Price: 7.0

Orange
Price: 8.0

Lemon
Price: 8.0

Blue cheese
Price: 9.0

1

Total price: 11.00 DKK

Add to basket

```java
private static void addToBasket(Javalin app) {  1 usage  Styrse
    app.post( path: "/", ctx -> {
        //USERS VALG AF BOTTOM OG TOP
        int inputBottomId = Integer.parseInt(ctx.formParam( key: "cupcakeBottom"));
        int inputTopId = Integer.parseInt(ctx.formParam( key: "cupcakeTop"));
        int quantity = Integer.parseInt(ctx.formParam( key: "cupcakeQuantity"));

        try {
            // ALLE BOTTOMS OG TOPS FRA DB
            CupcakeBottom cupcakeBottom = CupcakeMapper.getCupcakeBottomById(inputBottomId);
            CupcakeTop cupcakeTop = CupcakeMapper.getCupcakeTopById(inputTopId);

            BasketItem basketItem = new BasketItem(quantity, new Cupcake(cupcakeBottom, cupcakeTop));

            if (basket.isEmpty()) {
                basket.add(basketItem);
            } else {
                for (BasketItem item : basket) {
                    if (item.getItem().equals(basketItem.getItem())) {
                        item.addToBasket(quantity);
                        ctx.redirect( location: "/");
                        return;
                    }
                }
                basket.add(basketItem);
            }

            ctx.redirect( location: "/");
        } catch (DatabaseException e) {
            throw new RuntimeException(e);
        }
    });
}
```

Java

Bruger input

Cupcake bund og top

Cupcake objekt og kurv

“Dobbelt” tjekker

```java
public static CupcakeBottom getCupcakeBottomById(int bottomId) throws DatabaseException {  2 usages  Styrse
    List<CupcakeBottom> cupcakeBottoms = CupcakeMapper.getCupcakeBottoms();

    CupcakeBottom cupcakeBottom = null;

    for (CupcakeBottom bottom : cupcakeBottoms) {
        if (bottom.getId() == bottomId) {
            cupcakeBottom = bottom;
            break;
        }
    }
    return cupcakeBottom;
}
```

Java/Mapper

Bruger input

Finder bund

```java
public static List<CupcakeBottom> getCupcakeBottoms() throws DatabaseException {  4 usages  Styrse
    List<CupcakeBottom> bottoms = new ArrayList<>();

    String sqlBottom = "SELECT * FROM \"Cupcake_bottom\"";

    try (Connection connection = connectionPool.getConnection()) {
        try (PreparedStatement ps = connection.prepareStatement(sqlBottom)) {

            ResultSet rs = ps.executeQuery();

            while (rs.next()) {
                int bottom_id = rs.getInt( columnLabel: "cupcake_bottom_id");
                String bottom_flavor = rs.getString( columnLabel: "cupcake_bottom_flavor");
                float bottom_cost_price = rs.getFloat( columnLabel: "cupcake_bottom_cost_price");
                float bottom_sales_price = rs.getFloat( columnLabel: "cupcake_bottom_sales_price");
                boolean bottom_gluten_free = rs.getBoolean( columnLabel: "cupcake_bottom_gluten_free");
                int bottom_calories = rs.getInt( columnLabel: "cupcake_bottom_calories");
                String bottom_description = rs.getString( columnLabel: "cupcake_bottom_description");
                String bottom_path = rs.getString( columnLabel: "cupcake_bottom_path");

                CupcakeBottom bottom = new CupcakeBottom(
                        bottom_cost_price,
                        bottom_sales_price,
                        bottom_calories, bottom_description,
                        bottom_gluten_free,
                        bottom_id,
                        bottom_flavor,
                        bottom_path
                );
                bottoms.add(bottom);
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
        throw new DatabaseException("Error executing query");
    }
    return bottoms;
}
```

```java
public static List<BasketItem> basket = new ArrayList<>();  15 usages
```

```java
private static void showBasket(Javalin app) {  1 usage  Styrse *
    app.get( path: "/basket", ctx -> {
        Map<String, Object> model = new HashMap<>();
        model.put("basket", basket);

        ctx.sessionAttribute("basket", basket);
        ctx.attribute("totalPrice", BasketUtils.getTotalPrice(basket));

        ctx.render( filePath: "basket.html", model);
    });
}
```

```html
<table>
    <thead>
    <tr>
        <th>Item</th>
        <th>Quantity</th>
        <th>Item price</th>
        <th>Total price</th>
        <th>Remove line</th>
    </tr>
    </thead>
    <tbody>
    <tr th:each="item, iterStat : ${basket}">
        <td th:text="${item.item}"></td>
        <td th:text="${item.quantity}"></td>
        <td th:text="${item.itemPrice}"></td>
        <td th:text="${item.totalPrice}"></td>
        <td>
            <form action="/basket/remove" method="post">
                <input type="hidden" name="index" th:value="${iterStat.index}">
                <button type="submit" class="remove-line-button">Remove</button>
            </form>
        </td>
    </tr>
    </tbody>
</table>
```

Statisk kurv

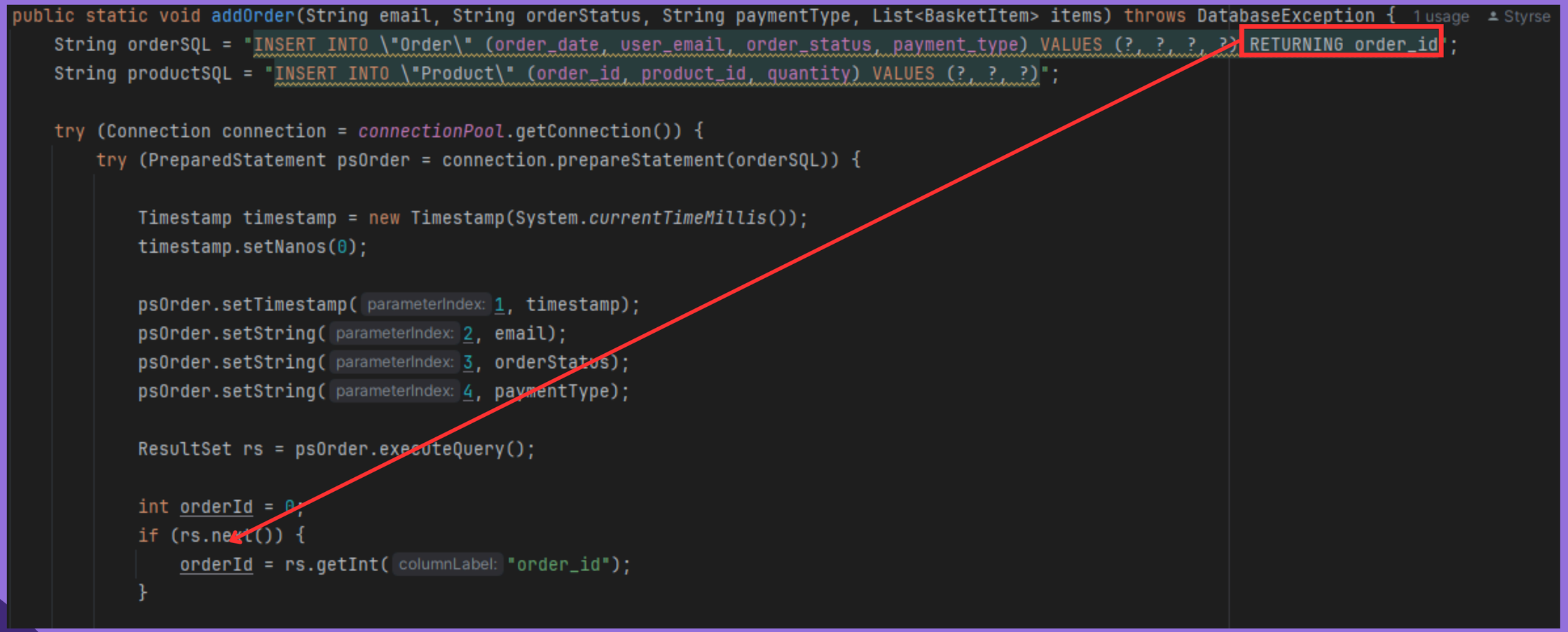
Kurven gemt i session

Kurv vist i html

```java
private static void processPayment(Javalin app) {  1 usage  Styrse
    app.get( path: "/process-payment", ctx -> {
        User user = ctx.sessionAttribute( key: "user");

        if (user == null) {
            ctx.redirect( location: "/login");
            return;
        }

        if (basket.isEmpty()) {
            ctx.redirect( location: "/");
            return;
        }

        double totalPrice = BasketUtils.getTotalPrice(basket);

        ctx.attribute("basketTotal", totalPrice);
        ctx.attribute("user", user);
        ctx.render( filePath: "process-payment.html");
    });
}
```

```java
private static void handlePayment(Javalin app) {  1 usage  Styrse +1
    app.post( path: "/process-payment", ctx -> {
        User user = ctx.sessionAttribute( key: "user");
        assert user != null;
        String userEmail = user.getEmail();
        float basketTotal = (float) BasketUtils.getTotalPrice(basket);
        float currentBalance = user.getBalance();

        String paymentMethod = ctx.formParam( key: "paymentMethod");

        String orderStatus = "";
        String paymentType = "";

        if (paymentMethod.equals("balance") || paymentMethod.equals("mobilepay")) {
            orderStatus = "Confirmed";

            if (paymentMethod.equals("balance")) {
                if (currentBalance >= basketTotal) {
                    paymentType = "Balance";

                    double newBalance = currentBalance - basketTotal;
                    updateUserBalance(userEmail, newBalance);

                    user.setBalance(currentBalance - basketTotal);
                }
            } else if (paymentMethod.equals("mobilepay")) {
                paymentType = "MobilePay";
            }
        } else if (paymentMethod.equals("cash")) {
            orderStatus = "Pending";
            paymentType = "Cash";
        }

        OrderMapper.addOrder(userEmail, orderStatus, paymentType, basket);
        basket.clear();

        ctx.sessionAttribute("user", user);
        ctx.redirect( location: "/");
    });
}
```

```java
public static void addOrder(String email, String orderStatus, String paymentType, List<BasketItem> items) throws DatabaseException {  1 usage  Styrse
    String orderSQL = "INSERT INTO \"Order\" (order_date, user_email, order_status, payment_type) VALUES (?, ?, ?, ?) RETURNING order_id";
    String productSQL = "INSERT INTO \"Product\" (order_id, product_id, quantity) VALUES (?, ?, ?)";

    try (Connection connection = connectionPool.getConnection()) {
        try (PreparedStatement psOrder = connection.prepareStatement(orderSQL)) {

            Timestamp timestamp = new Timestamp(System.currentTimeMillis());
            timestamp.setNanos(0);

            psOrder.setTimestamp( parameterIndex: 1, timestamp);
            psOrder.setString( parameterIndex: 2, email);
            psOrder.setString( parameterIndex: 3, orderStatus);
            psOrder.setString( parameterIndex: 4, paymentType);

            ResultSet rs = psOrder.executeQuery();

            int orderId = 0;
            if (rs.next()) {
                orderId = rs.getInt( columnLabel: "order_id");
            }

            for (BasketItem item : items) {
                try (PreparedStatement psItem = connection.prepareStatement(productSQL)) {

                    int productId = -1;
                    int quantity = item.getQuantity();

                    if (item.getItem() instanceof Cupcake) {
                        productId = CupcakeMapper.getCupcakeId(((Cupcake) item.getItem()).getCupcakeBottom().getId(), ((Cupcake) item.getItem()).getCupcakeTop().getId());
                    }

                    psItem.setInt( parameterIndex: 1, orderId);
                    psItem.setInt( parameterIndex: 2, productId);
                    psItem.setInt( parameterIndex: 3, quantity);
                    psItem.executeUpdate();
                }
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
        throw new DatabaseException("Error executing query");
    }
}
```


public static void addOrder(String email, String orderStatus, String paymentType, List<BasketItem> items) throws DatabaseException { 1 usage Styrse
String orderSQL = "INSERT INTO \"Order\" (order_date, user_email, order_status, payment_type) VALUES (?, ?, ?, ?) RETURNING order_id";
String productSQL = "INSERT INTO \"Product\" (order_id, product_id, quantity) VALUES (?, ?, ?)";
try (Connection connection = connectionPool.getConnection()) {
try (PreparedStatement psOrder = connection.prepareStatement(orderSQL)) {
Timestamp timestamp = new Timestamp(System.currentTimeMillis());
timestamp.setNanos(0);
psOrder.setTimestamp(parameterIndex: 1, timestamp);
psOrder.setString(parameterIndex: 2, email);
psOrder.setString(parameterIndex: 3, orderStatus);
psOrder.setString(parameterIndex: 4, paymentType);
ResultSet rs = psOrder.executeQuery();
int orderId = 0;
if (rs.next()) {
orderId = rs.getInt(columnLabel: "order_id");
}

```java
for (BasketItem item : items) {
    try (PreparedStatement psItem = connection.prepareStatement(productSQL)) {

        int productId = -1;
        int quantity = item.getQuantity();

        if (item.getItem() instanceof Cupcake) {
            productId = CupcakeMapper.getCupcakeId(((Cupcake) item.getItem()).getCupcakeBo
        }

        psItem.setInt( parameterIndex: 1, orderId);
        psItem.setInt( parameterIndex: 2, productId);
        psItem.setInt( parameterIndex: 3, quantity);
        psItem.executeUpdate();
    }
}
```

product_id [PK] integer	bottom_id integer	top_id integer
1	1	1
2	1	2
3	1	3
4	1	4
5	1	5
6	1	6
7	1	7
8	1	8
9	1	9
10	2	1
11	2	2
12	2	3
13	2	4
14	2	5
15	2	6
16	2	7
17	2	8
18	2	9
19	3	1
20	3	2
21	3	3
22	3	4
23	3	5
24	3	6
25	3	7
26	3	8

TAK