CUPCAKE WEB-SHOP

Technical overview

Project developed from 26-02-2018 to 18-03-2018
Report finalized 18-03-2018

Thomas Rosenkrans Vestergaard
cph-tv55@cphbusiness.dk
Github: Thomas-Rosenkrans-Vestergaard

Mikkel Koefod Lindtner
ml-509@cphbusiness.dk
Github:mklindtner

Contents

Introduction

Our goal with this project was to implement a web-shop for a bakery that sells cupcakes. Users of the web-shop can either create their own or buy one of the predefined cupcakes. The purchasable cupcakes consist of a bottom and topping.

Users of the web-shop can create and maintain a profile using a login system. Users log in using a username and password. For some functions of the web-shop, a profile is required. These functions include:

- Adding a product to the users shopping cart.
- Placing orders.
- Viewing user details.
- Viewing funds available to purchase products with.

Once a user has created their cupcake, this product can then be added to their personal shopping cart along with the quantity (the number of that type of cupcake to order). Once the user has added the products they want to their shopping cart, the user can then choose to order these products by placing an order.

Funds are uses to place orders for products. More funds can be added to the wallet using the `/funds` page.

Technologies

The following technologies were used in the making of the web-shop.

Java 8

Java 8 is the general programming language used to program the web-shop.

Gradle (v3.0)

The project uses Gradle 3.0 to manage dependencies the project requires. Gradle is currently only used to manage dependencies and does not use any custom tasks needed to execute the project.

MySQL server (v5.7)

To store persistent data, the web-shop uses the MySQL database system. The database contains information about:

- The topping from which custom cupcakes can be created by users.
- The bottoms from which custom cupcakes can be created by users.
- The preset cupcakes that can be bought by users.
- The user profile details of users who have created a profile.
- The orders placed by authorized users.

Ubuntu 17.4 and Tomcat 8

The server hosting the project runs Ubuntu 17.4 and serves HTTP requests using the Tomcat 8 webserver.

JDBC

The web-shop uses the JDBC API to query the MySQL database. The web-shop furthermore uses prepared statements for all querying of the MySQL database.

MySQL Connector (v8.0.8)

The MySQL Connector is used as the JDBC driver allowing JDBC to connect to the MySQL database system described above. This dependency is included and managed using Gradle.

JavaEE Servlets 3.0 and JSP

The pages of the web-shop are served by JavaEE Servlets and rendered using Java Server Pages. The Java libraries needed to use these technologies are included and managed using Gradle.

Frontend

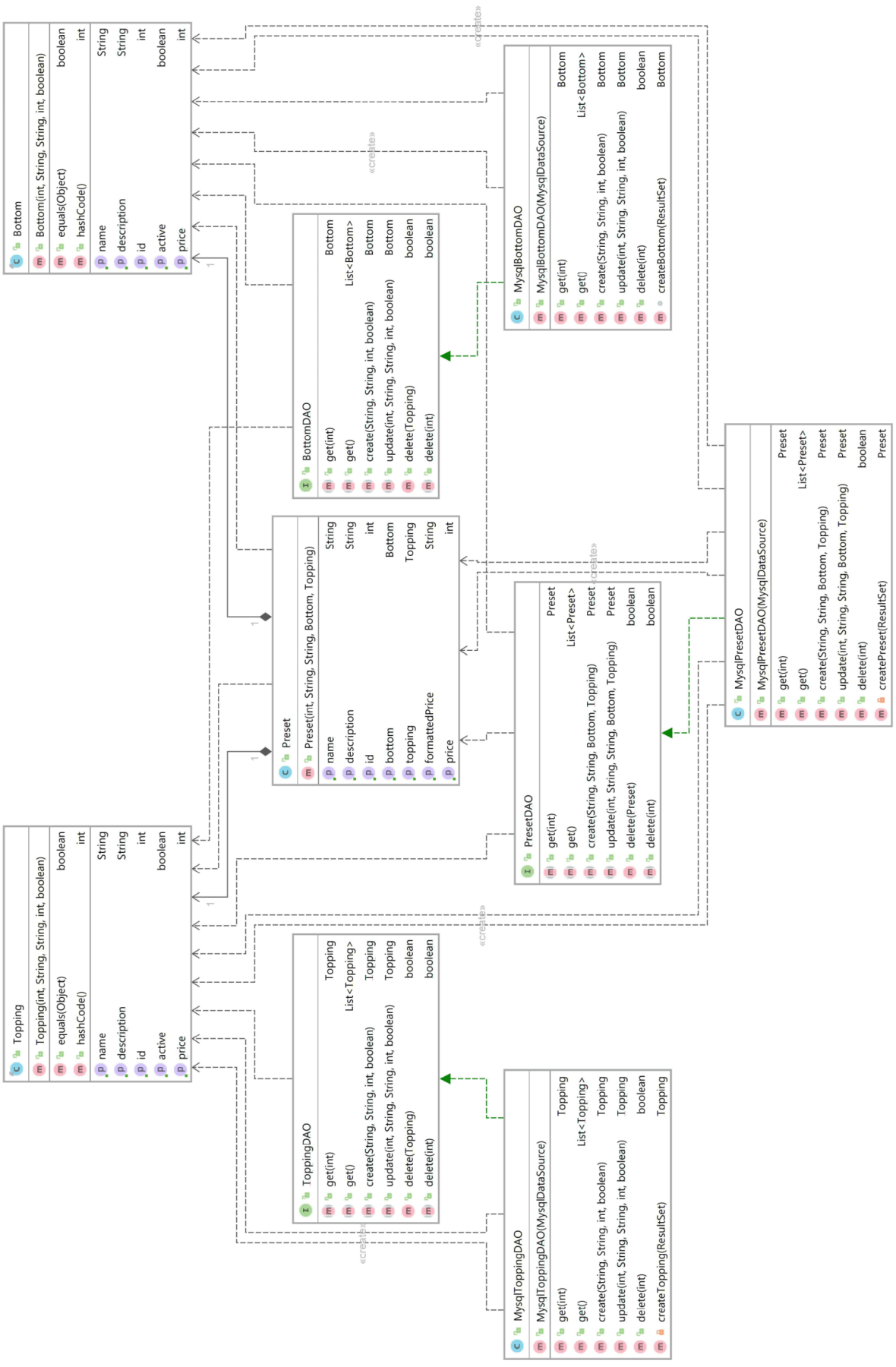
The frontend of the web-shop is served using.

- HTML
- CSS
- JavaScript

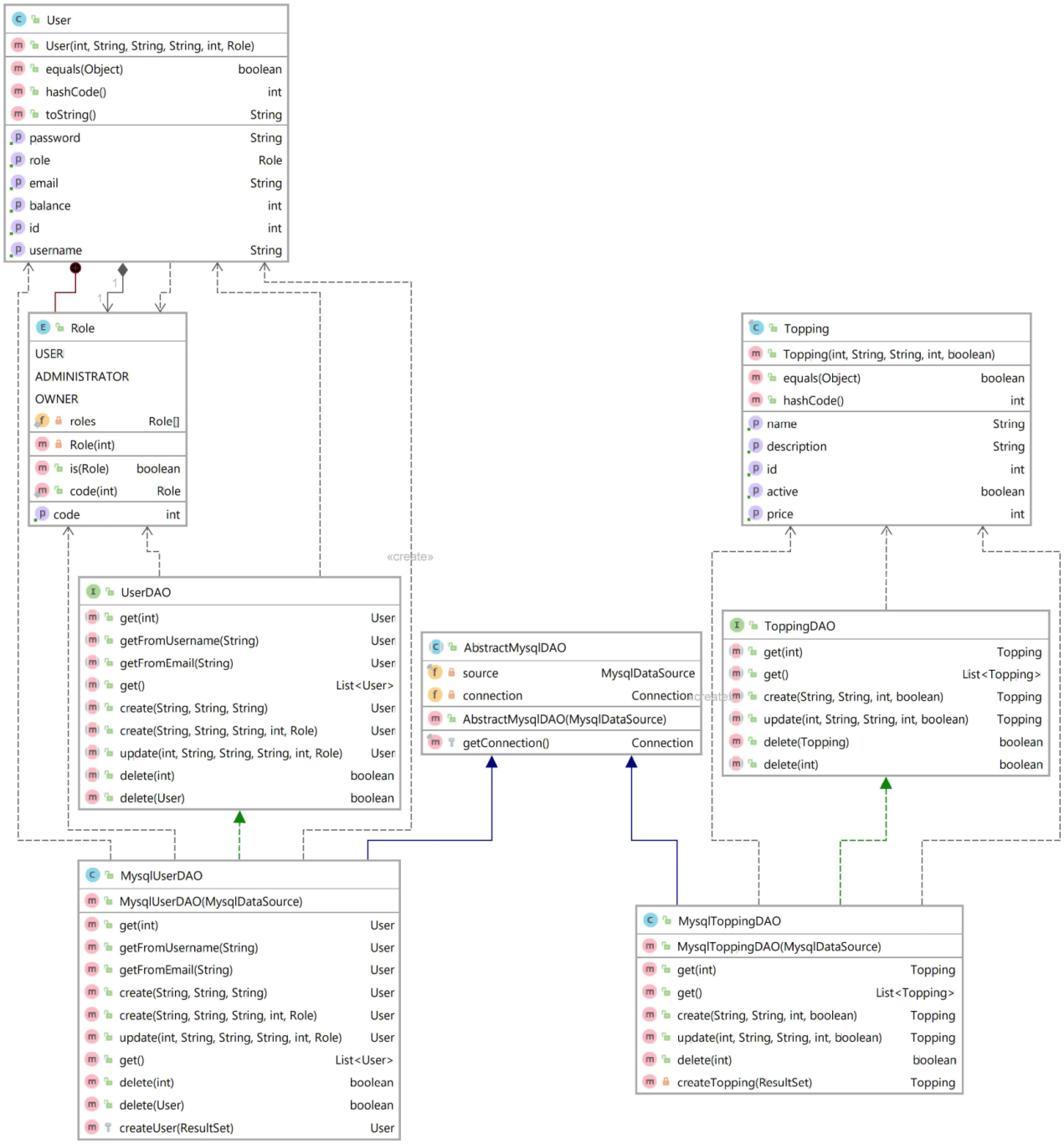
The project also utilizes the the following libraries to speed up and ease development of the frontend web-shop:

- MaterializeCSS (v1).
- jQuery (v3.2.1).

DAO class diagram

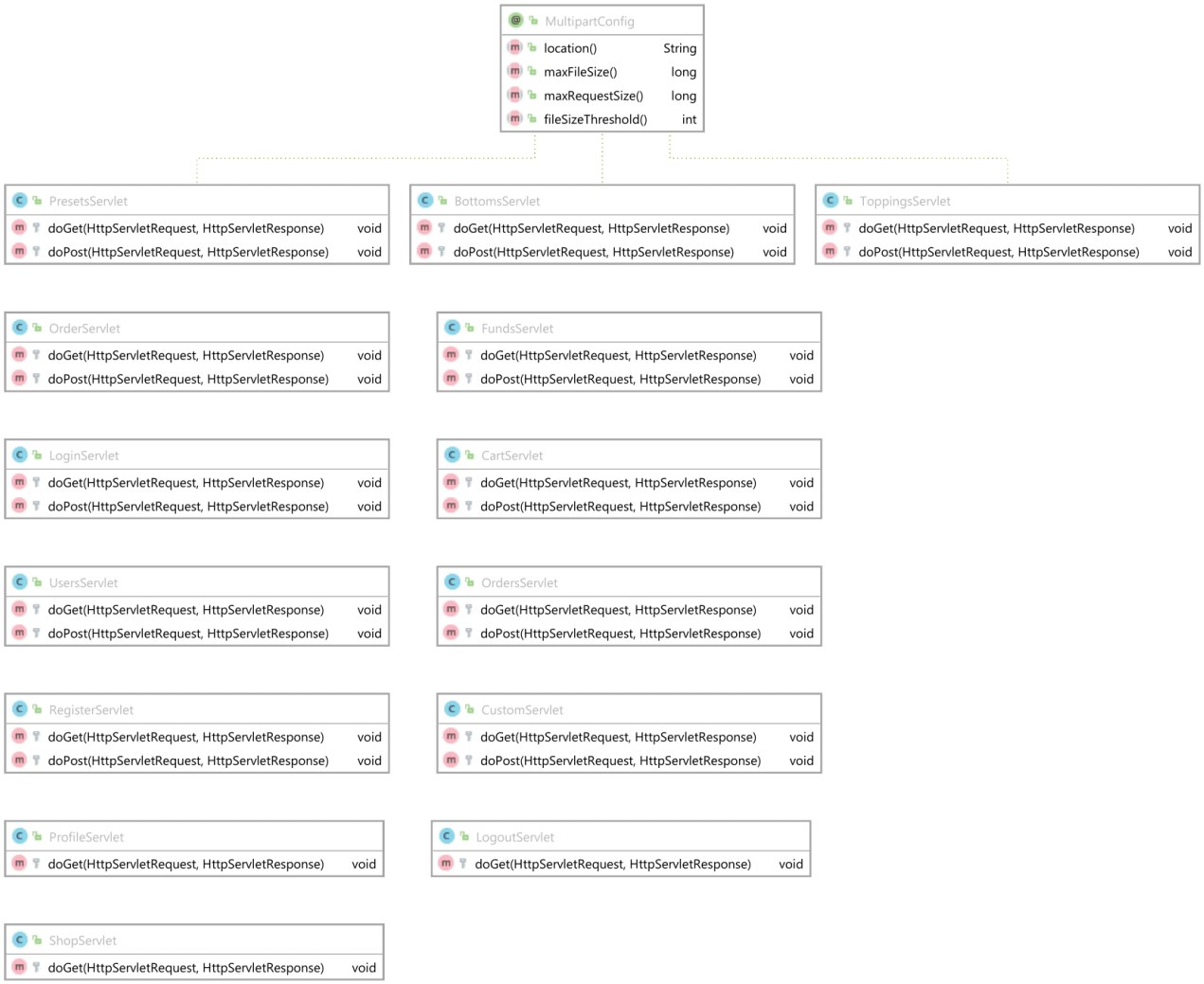


The above class diagram describes the structure of the DAO layer concerned with bottoms, toppings and presets. The DAO classes concerned with users and orders are described here.



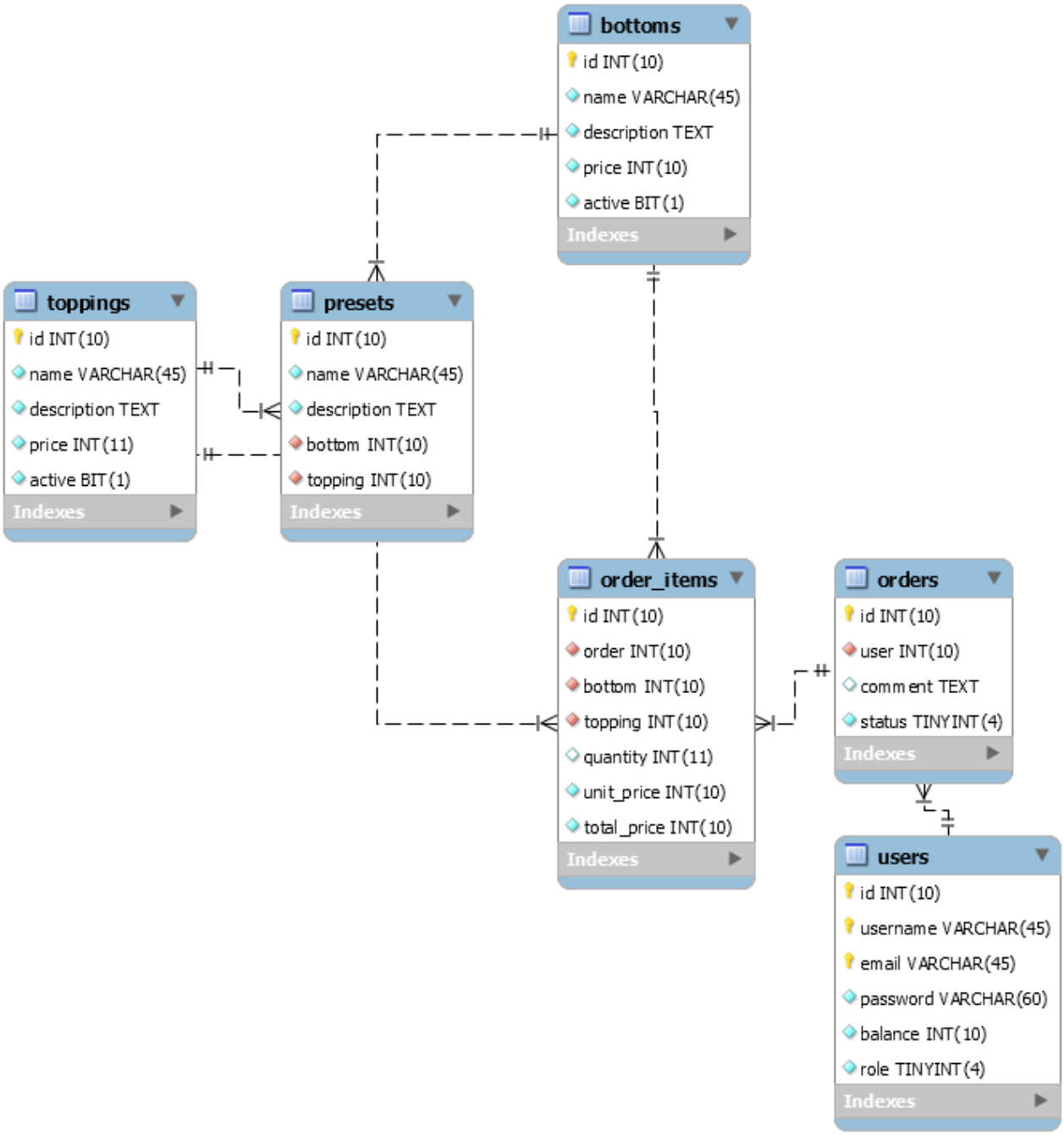
Servlets class diagram

The following diagram provides an overview of the servlets in the view.



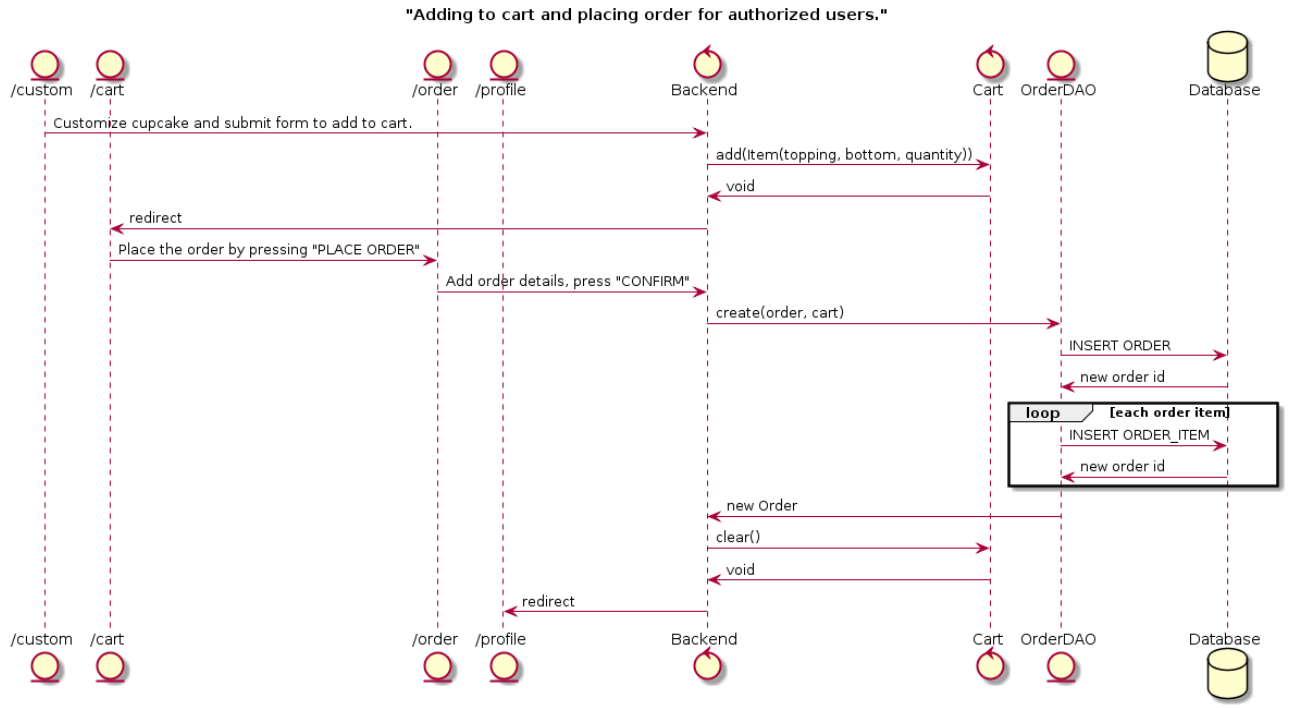
ER diagram

Below you can find a diagram showing the structure of the database used by the web-shop. All currency related attributes are in cents. The active attribute on bottoms and toppings dictate whether the bottom or topping can currently be purchased. The bottom and topping are considered discontinued when `active` is `0`.



Sequence diagram

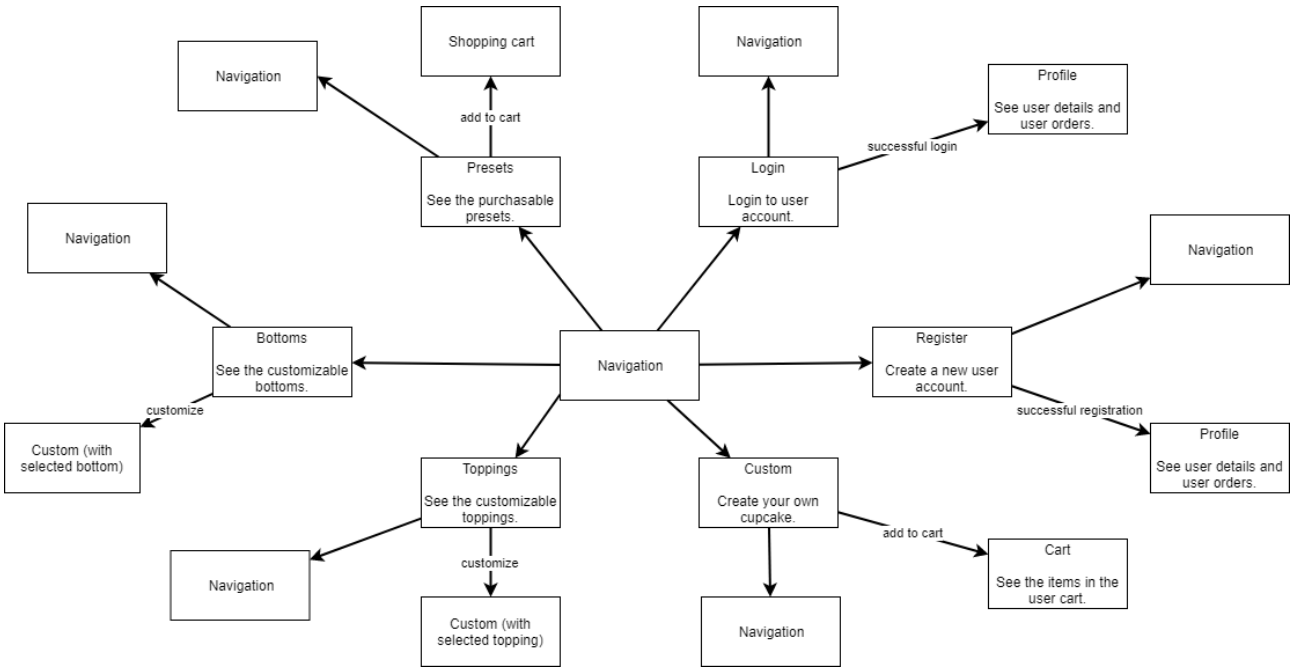
Following is a Sequence Diagram describing the steps taken for an authorized user to place an item in their shopping cart and then ordering the products in the cart.



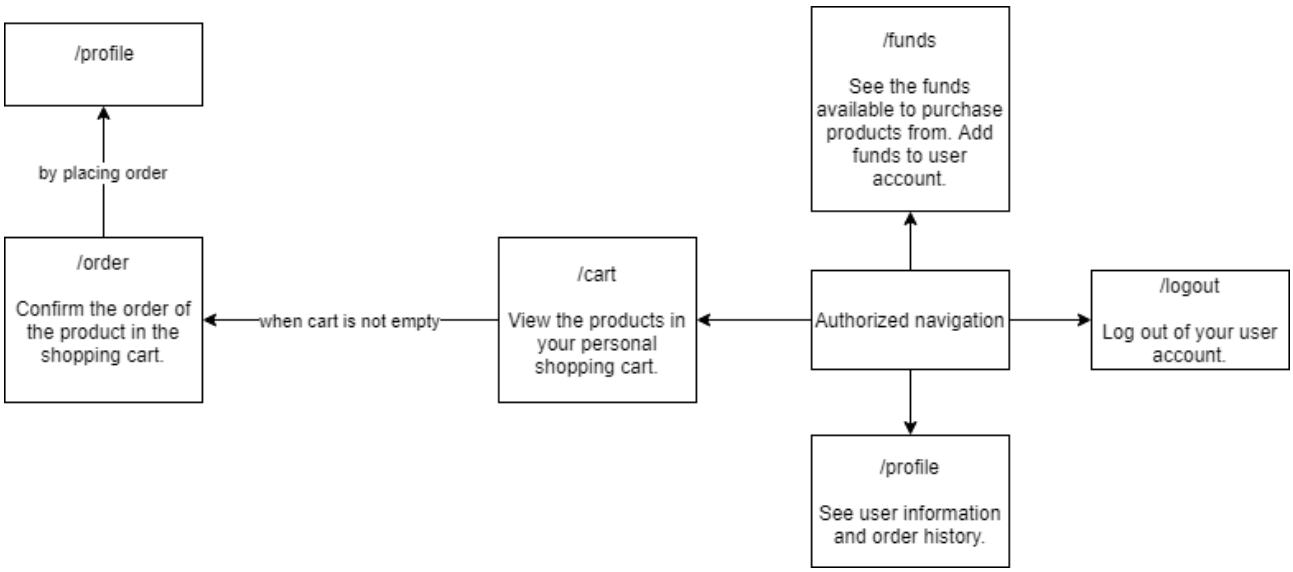
Notice that the above diagram only describes the use case for an authorize user with enough funds. The `Backend` boundary consists of many `Servlets`.

Navigation diagram

First the navigation diagram showing the pages that are accessible by both authorized and unauthorized users.



Authorized users gain access to the following pages too:



Additional information

Sessions

The web-shop maintains a session for all users of the web-shop.

- Users that are logged in have a shopping cart object in session that contains the items the users have put in their cart. The object still exists in session even when the user has added no items to their cart.
- Users that are logged in have a `User` object in session, that contains the information about that user. This session attribute is mostly used to check if a user is authorized.
- The session of both authorized and unauthorized users is used to store notifications for that user.

Exceptions

Exceptions are handled gracefully on most pages in the web-shop. When an exception occurs, the user is redirected to the /shop page and notified that an error occurred. If the `/shop` page also throws an exception, the browser will most likely not show the page because of too many redirects. A dedicated JSP error page should be used in case the `/shop` page also throws an exception.

User-input

Most user-input is validation on all pages. Instead of crashing the user is presented with friendly messages describing the cause of the error that occurred.

Validation of user input is mostly done using the `Parameters` helper class. This class could be improved and should be tested in the future since lots of servlets depends on it.

One place where validation of user input has not been implemented, is the page where users can add funds to their profile. No validation or checks are performed on the credit card provided to this form.

A system for transferring money from the provided credit card, has also not yet been implemented.

Security

The login system cannot currently be considered safe, since the web-shop does not currently utilize HTTPS, meaning that the user details sent from the login and registration pages are sent as plaintext

Currently user passwords are hashed using the B-crypt one-way cryptographic function to protect user passwords in case the database is compromised.

Although the website is protected against XSS attacks, the website is still vulnerable to Cross Site Request Forgery attacks.

User roles

Every user in the database has a user role. A user role defines the actions that can be taken by the user. The user type is stored using the `TINYINT` datatype. Currently three user types exist.

Identifier	Name	Actions
0	`USER`	• Use shopping cart • Place orders • View profile • Add funds
1	`ADMINISTRATOR`	• Can access the administration panel

		• Can make changes to presets, bottoms, toppings and orders.
2	`OWNER`	• Can make changes to `ADMINISTRATOR` user accounts.

The only thing stored in the database is the Identifier of the user type. The names of the user types are mapped using the java enumeration `User.Role`.

Notifications

When a user needs to be notified of some situation, like an error or success message, a notification is sent using the Notifications class. This class adds a new Notification to a queue in the session of the user to receive the notification. The notification queue is then accessed from `JSP` to display them to users.

Requirements

Most of the requirements have been fulfilled, however some were overlooked.

- The method for adding funds to a user profile is flawed. No validation of the provided credit card information is validated. This means that any users that add an arbitrary amount of funds to their user profile without providing a credit card.
- Administrators can only perform update actions on the orders table.

Improvements

Several improvements can be made to the usability and security of the web-shop.

- Allow unauthorized users to add products to their shopping cart.
- Improve the login system, so users prompted to login or create a profile don't lose progress on their current task. For example: When an unauthorized user attempts to place a cupcake into their cart, they lose the progress they made on customizing their cupcake.

- Fix the `/logout` page, allowing users to log out of their user account.
- Crop images uploaded through the administration panel. Currently no images are being cropped, meaning that the images on the site could have varying dimensions.
- Add constraint violation error messages to the administration panel.
- Add better error messages when values provided through the administration panel are out of bounds, too long or too short.

Security

- Validate the ownership of the email provided during registration. Sending a conformation email to new users with a one-time conformation link should fix this problem.
- Once allow `OWNER` users to edit and create new users with `ADMINISTRATOR` role.
- Check that the topping and bottom being ordered are currently purchasable. Currently it's possible to order bottoms and toppings that are inactive. Inactive bottoms and toppings should also not be shown on the `/shop` page.
- Validate credit card information provided through the `/funds` page.

Maintenance

- Beginning to log the `exceptions` that are raised during the execution of the program would help the developer find errors quicker than currently possible.
- Delete images left over after a bottom, topping or preset is deleted.

Testing

The only classes currently being tested are the are the `ShoppingCart` and `Notifications` class.