Group 10

Members:
Hampus Lilja, *881025-4972, hampusn@student.chalmers.se ,* HampusLilja
Tomas Arnesson, *860606-2951, tomasar@student.chalmers.se ,* therealtomas
Mattias Herzog, *900201-0073, herzog@student.chalmers.se ,* h3rzog, (*076-25 71 911)*
Mathias Dosé, *901107-1090, dose@student.chalmers.se ,* mathiasdose

What is this? In which area is the system supposed to be used. What is it supposed to do?
An "online" liquor shop.

Possible users/roles (admin, others,...) and permissions.
There are three different "roles" you can have on the website: admin, user and "not logged in".

1. **"Not logged in"** : Can see the homepage, , shop,contact page and the signup/login page. If you try to enter another page where you are not allowed you will find a page with no content.
2. **User:** Can do everything you can do as "Not logged in" with the addition of being able to create a wish list.
3. **Admin:** Can do everything the "user" can but also add,edit and delete products. Can edit and remove customers.

There's a boolean on each page's bean that gets altered depending on which "role" you have on the page.

A list of fully functional use cases (short description, one sentence).

1. **Welcome-cookie**: The first time you go to the index page you will see a popup dialog box and be asked to enter a name, when you return to index.xhtml you will find a dialog box that welcomes you back.
2. **Login**: It's possible to login on the site. If you have an user or admin account a bean will give you the permission to join the pages you are supposed to have access to.
3. **MD5**: When you sign up or enter your password when you login the password gets hashed with salt before it's added in the database or checked against the database.
4. **Password check 1**: There is a javascript that makes sure that the user that wants to sign up has to re-enter the password to make sure it's the intended password.
5. **Password check 2:** Forces the user to pick a password that contains at least six characters and the password must consist of upper and lower case letters and at least one number.
6. **Admin products**: If you are logged in as an admin you can add/edit/delete products
7. **Admin customers:** If you are logged in as an admin you can edit and delete customers.
8. **iFrame**: On the contact page you can find an iFrame where a google map is displayed and gives the user the possibility to find out where we are located and how to find us.

9. **Contact form**: If the form is filled in correctly and the submit button is pressed a mail client will appear as a popup where the user only has to press send to give us the contact information.
10. **Header image**: If you press the header image, no matter where you are located on the site, you will be redirected to the index page.
11. **Upload image**: When you have added a product's specs you will be redirected to a page where you can upload an image of the product.
12. **Wish list**: Gives the user the possibility to create a list of products they want to be able to buy in the shop in the future.
13. **Shop**: The user can drag products to a shopping cart and also remove items.
14. **Display product image:** If no picture is uploaded to a product, a default picture is shown.

Known issues:

1. All the products are displayed at shop.xhtml, with a large database this page can be very large, no limit on size or length.
2. Shop does not reset total price and total products when products are removed from the shop.
3. Wish list is supposed to "help" the user to send a mail with the list to the owners, it doesn't.
4. We don't have a way for the customer to pay for the products from the cart, so for now when the user press the buy button you will simply get redirected to the index page.
5. We use an absolute url to decide where the uploaded picture should be stored. This need to be changed to a relative url to work in a different environment.

Tests:

We got a few tests, all located in the JavaApp_Project in the edu.chl.group10.javaapp_project.core of "Test Packages". The test we have created are quite self explanatory.

Physical setup (tiers)

Java Derby database(name: group10db) is needed to store customers and products.
Glassfish 3.1.2-b14. We had to change the glassfish version to be able to get our image upload to work. It was very time consuming to figure out that we needed to to this.

Participating software components (applications, middleware, libs, ...) distributed over the tiers. Responsibility for each component. Communication between the components.

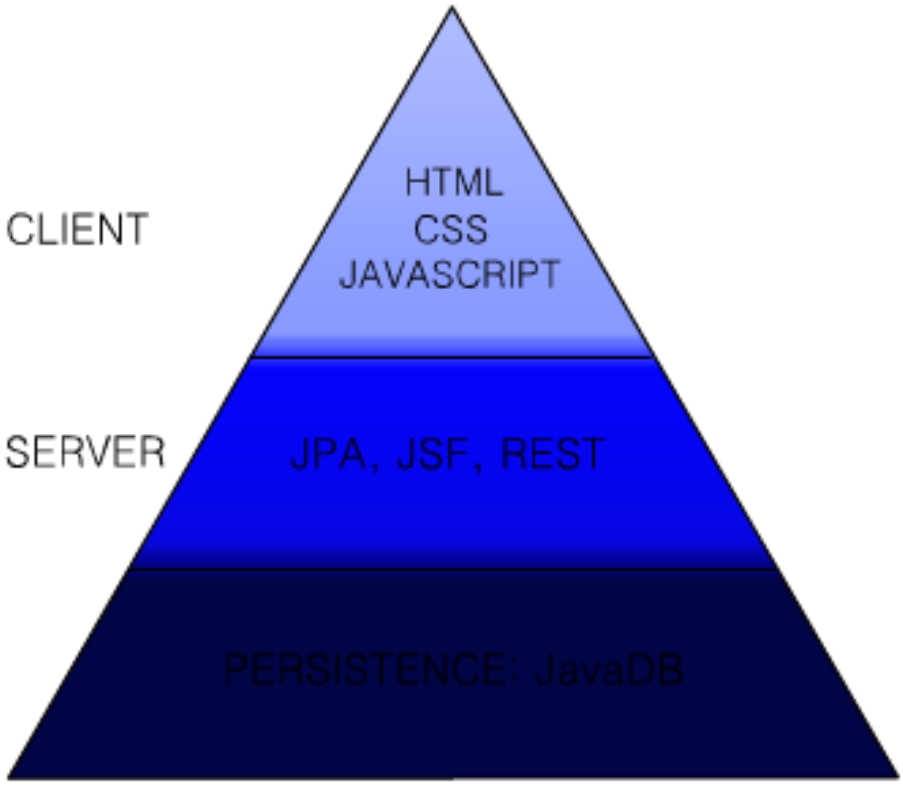
Netbeans, Github, Git Shell.

The modules (packages) of each component and the responsibility for each module.

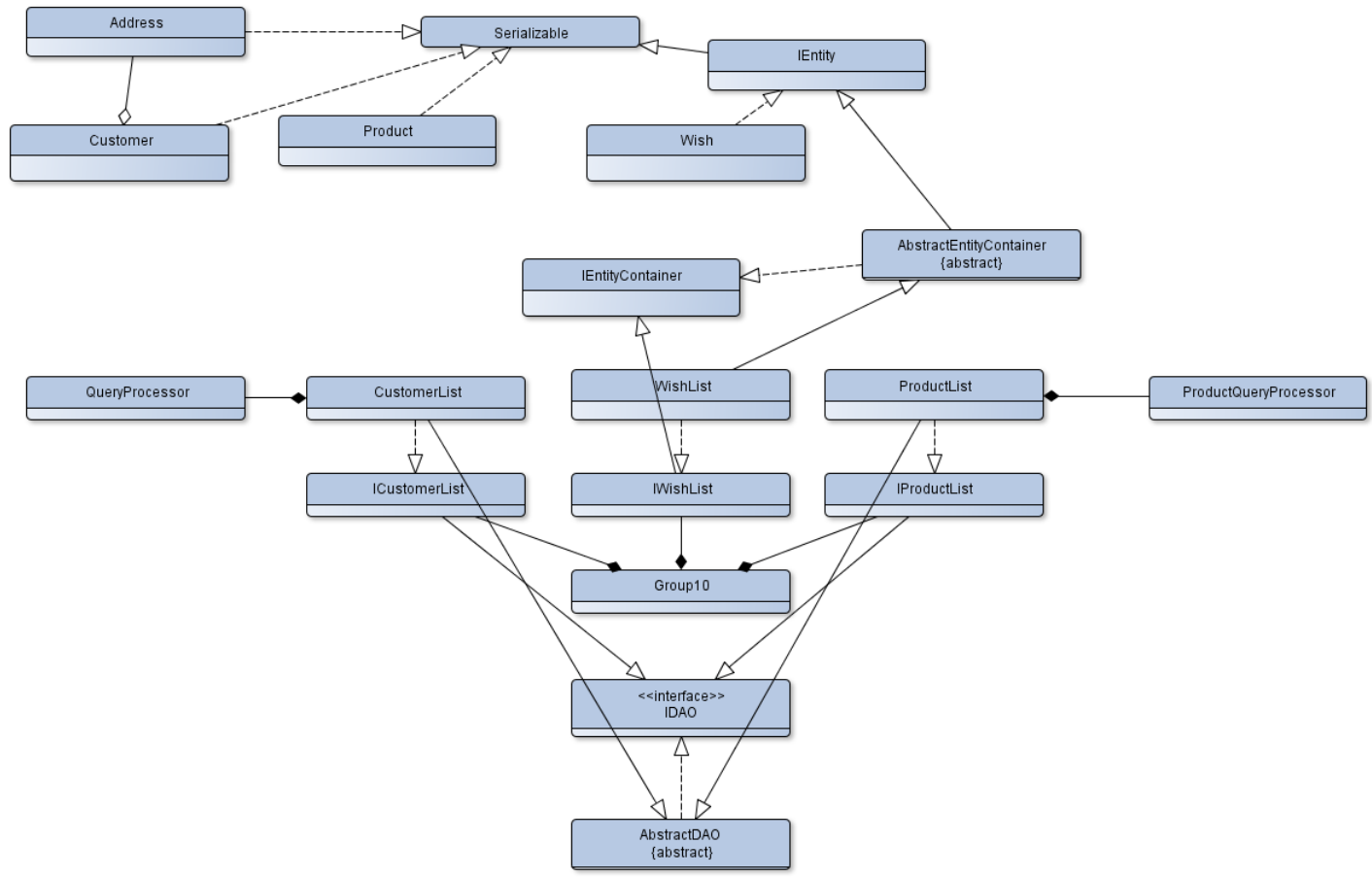
- **edu.chl.group10.core:** Handling various entitymanagers and entitys for connections with the database. With persistence handling and JPQL processing.
- **edu.chl.group10.webapp_project**: Contain the files needed to use RESTful services on the project. Also contains a bean that's used during the picture upload.
- **edu.chl.group10.webapp_project.CustomerCRUDBB:** Contain the beans necessary for Customer to get full CRUD. "customerConversation" is used to communicate between the beans.

- **edu.chl.group10.webapp_project.Login:** Contain the beans for the login page.
- **edu.chl.group10.webapp_project.ProductCRUDBB:** Contain the beans necessary for Product to get full CRUD. ”productConversation” is used to communicate between the beans.
- **Web layer javascripts:**
  welcomeCookie.js keeps track of visiting user.
  form-validation.js validates all inputforms from user input, before registering any data.
  checkPassword.js makes sure the username and passwords entered follow enforces safety standards.
  cart/script.js handles the shopping cart and product jquerys.
  Jquery.jfollow.js makes sure the shopping cart always is visible when scrolling the page.
  PreviewImageJavascript.js enables a preview of the uploaded image.

A layered view of the application.



The object oriented model as a UML class diagram.



Which technologies are used, where? A list.

- **HTML5:** Featured in all shown webcontent.
- **CSS:** Featured in all shown webcontent.
- **JPA/ORM:** Persistence JPA is used on Customer.java and Product.java. The persistence file is located at JavaApp_Project /Other Sources/"src/main/resources"/META-INF/persistence.xml
- **JSP:** Featured in all pages that has relation to the Derby database.
- **JQuery:** Featured in shop.xhtml and addpicturetoproduct.xhtml
- **JSF:** Templates, Facelets: All pages inherits design från JSF Faces Template. Also almost all pages use JSF html.
- **Javascript:** Featured in RESTful service, welcome-cookie, shop and imageupload.
- **Java:** JavaApp_Project consists only of Java-files in .core package. WebApp_Project uses Java language in packages .*, .CustomerCRUDBB, .Login, .ProductCRUDBB. For handling servlets and beans.
- **JavaDB Derbys**: Information about products,customers stored in group10db, jdbc:derby.
- **BackingBeans:** Featured in each page that shares information, via a ConversationBean.
- **RESTful services:** Used in wish list
- **JUnit:** Java testing classes in JavaApp_Project. Testing entitymanagers, navigation, adding customer, JPQL and mapping.
- **JPQL:** Java Persistence Query language used then communicating with database.