MANAGEMENT OF A BEACH STRUCTURE

INDIVIDUAL PROJECT

GENERAL DESCRIPTION

The system will show a main screen where you can register or login. Users will be able to register while employees will have to be added manually by the Admin. After authentication with appropriate checks and queries to the Database, the logged in user will be redirected to his personal page based on the "Type" that was assigned to him during registration / insertion by the administrator. The user will be able to perform actions based on their privilege. 4 types of users are identified: client, cashier, lifeguard and admin. The client user can book a "Seat", free it, order something to eat, keep track of the status of his orders and update his login credentials. The "Lifeguard" will have an interactive screen to monitor the status of the Seats and view their information. The “Cashier” will be able to view all the purchases that users make and view the total cash flow. The "Cook" through an interactive screen will display uncompleted orders and can change the status of the order to "completed" using the button. The Admin will be able to add new employees, delete users and view all registered users

SYSTEM ARCHITECTURE

Used technologies:

1. Java servlet classes, for accepting http requests, for the processing of data entered by the client and for forwarding to JSP pages;

2. JSP for the representation of the results processed by the servlets;

3. Java Beans for the creation, storage of Users, Deckchairs and Orders and for sharing the latter between the various classes of the web application;

4. HTML, CSS, JavaScript and jQuery for user interface management;

5. AJAX for the creation of interactive screens;

6. Apache Tomcat, as middleware for the deployment of the web application.

7. MySQL as DBMS, InnoDB for data storage.

FUNCTIONAL REQUIREMENTS

General:

1. All access cases must be checked;

2. Possible errors must be checked and managed;

3. The data entered by the user must be verified;

4. The interactive screens will be reloaded periodically and asynchronously;

5. Technical errors due to the database query will cancel the actions and return an error screen;

"Client" user:

1. It will be possible to register using the form;

2. You will be able to login;

3. Your information will be visible;

4. The access password can be updated;

5. It will be possible to book a "seat";

6. It will be possible to free a seat and exit;

7. You can order food / drink;

8. You will be able to view your purchases and their status.

"Lifeguard" user:

1. You will be able to login;

2. You can interactively view the status of the "seat" of the system;

3. You will be able to view information relating to a “seat”.

"Cashier" user:

1. You will be able to login;

2. You will be able to view the status of the "orders" and the total cash flow.

"Cook" user:

1. You will be able to login;

2. You can interactively view the status of the "orders";

3. You can change the status of your orders.

"Admin" user:

1. You will be able to login;

2. Will be able to add new employees;

3. Will be able to delete users from the system;

4. You will be able to view all registered users.

Server side:

1. Subdivision into packages: Servlets, Model, Database;

2. Processing of data provided by the client through the use of java servlet classes;

3. Handling of errors and exceptions;

4. Utility methods to query the DB;

5. Configuring a Data Source to manage connections;

6. Use of the PreparedStatement for security;

7. Authentication via Login;

8. Java Beans for creating and sharing objects between classes;

9. JSP for viewing the processed data;

10. Implementation of the MVC pattern;

DATA MODEL AND OTHER SPECIFICATIONS

Xampp (Apache, MySQL, Apache Tomcat v9.0);

The system has been made as simple as possible to speed up implementation:

user (ID, username, password, type);

orders (ID, idclient, description, quantity, price, status);

seat (ID, idclient, posX, posY, available);

orders: Status represents the status of the order ("Incomplete", "Complete"), Cook will update the status of an order;

user: each User will have a Type ("client" in the case of a simple customer, "admin", "cook", "lifeguard", "cashier" in the case of employees with specific duties);

seat: posX, posY represent the “coordinates” of the specific deckchair; Available (if the deckchair is free then in the DB the value will be equal to 1, on the contrary it will be equal to 0.