Project documentation

The project contains a total of 16 files, which will be detailed below:

1. chage\_state.php

In this file we make the inclusion of the file SolicitudSaldoController.php, to access its static method chageState, which we use to perform the action of approval or rejection of the balance request. After performing this action we perform the redirection to the page listado.php.

2. Connection.php

In this file we create the class Connection which handles everything related to the connection to the database, within this we declare the static attributes $servername, $database, $username and $password, which will serve to make the connection to the database, the static attribute $con will store the connection to the database.

In the static function start() using the function mysqli\_connect(), passing the parameters $servername, $database, $username and $password, we connect to the database and store the connection in the variable $con, then we verify that a connection has been established if not we use the function die() that will show us an error message, if everything went correctly we return the value of the variable $con.

In the static function getConexion() we return the value of the variable $con.

In the static function isConnected() we verify if we have established a connection with the database and depending on this we return true or false.

In the static function close() using the mysqli\_close() function we close the connection to the database.

3. footer.php

In this file we close the tags that are open from the header.php file and add the script tag with the javascript cdn of the bootstrap framework.

4. form.php

In this file we create the form for the balance request, in principle we include the files header.php (This file gives us the structure of the main links menu that is handled in the project), SolicitudSaldoController.php and Session.php (These files give us the functionality they contain).

After including the necessary files, we verify by means of the static function hasSession of the class Session if the user has already entered to the system by means of its user data, and if it has not entered to the system it redirects it to the login file. php, if the user has already logged into the system with your username and password is checked if you have clicked on the button btnSolictudSaldo, if the button has been pressed by the static function create of the class SolicitudSaldoController perform the action of creating this request by passing the requested value as a parameter to the function, if the request was successful redirected to the file listado.php but we show an error message.

Finally the following is html code in which we create a form with the quantity input and the submit button, this form is sent to the same file using the post method and the verification process of pressing the btnSolicitudSaldo button is performed.

5. header.php

In this file we create the main links menu that we will use in the application, inside its head tag we add the link tag with the css cdn of the bootstrap framework, in the first line we include the Session.php file to use its functions, inside the links menu using the static function hasSession of the Session class we show or not some links depending on whether the user is logged into the system or not.

6. home.php

In this file we include the files header.php (This file gives us the structure of the main links menu that is handled in the project), Session.php and UserController.php (These files give us the functionality they contain).

After including the necessary files, we verify by means of the static function hasSession of the class Session if the user has already entered to the system by means of its user data, and if it has not entered to the system it redirects it to the login.php file.

Then, if we do not redirect to the login, using the static function getUser() of the UserController class we obtain the current user data and store it in the variable $user.

Finally we show the user's data using the variable $user, depending on the value of the role stored in the global variable $\_SESSION we show or not the user's balance.

In the last line we include the footer.php file to close the tags that are open from the header.php file and include the cdn that we already mentioned.

7. index.php

In this file we make the inclusion of the header.php files (This file gives us the structure of the main links menu that is handled in the project). Then we add an image using the img tag.

In the last line we include the footer.php file to close the tags that are open from the header.php file and include the cdn that we already mentioned.

8. list.php

In this file we visualize the list of balance requests, in principle we include the files header.php (This file gives us the structure of the main links menu that is handled in the project), SolicitudSaldoController.php and Session.php (These files give us the functionality they contain).

After including the necessary files, we verify by means of the static function hasSession of the class Session if the user has already entered to the system by means of its user data, and if it has not entered to the system it redirects it to the login.php file.

Then, if we do not redirect to the login, using the static function getAll() of the class SolicitudSaldoController we obtain the balance requests depending on the current user, if it is of role type ADMINISTRATOR we obtain all the requests, if it is of role type USER we obtain all the request that the current user has made, and we store the requests in the variable $res.

Then depending on whether the role type of the current user stored in the global variable $\_SESSION is equal to USER we show the Create Balance Request button.

Then using HTML code we display the data of the requests stored in the variable $res, also depending on whether the role type of the current user stored in the global variable $\_SESSION is equal to ADMINISTRATOR and the status of the request is equal to pending we show the action column in which we find two buttons that perform the actions of approving and rejecting.

In these buttons which are actually a tags, we create the links that lead to the change\_state.php file with parameters depending on the button that is clicked.

In the last line we include the footer.php file to close the tags that are open from the header.php file and include the cdn we already mentioned.

9. login.php

In this file we create the login form, in principle we include the files header.php (This file gives us the structure of the main links menu that is handled in the project), LoginController.php and Session.php (These files give us the functionality they contain).

Initially we verify if the btnLogin button has been clicked, if the button has been clicked using the static function login of the LoginController class we perform the action of obtaining the user data by passing the user and password as parameters to the function, if the request was successful (the user and password are correct) we redirect to the home.php file, otherwise we show an error message.

The following is html code in which we create a form with the user input, the password input and the enter button, this form is sent to the same file using the post method and the verification process of pressing the btnLogin button is performed.

In the last line we include the footer.php file to close the tags that are open from the header.php file and include the cdn that we already mentioned.

10. LoginController.php

In this file we include the file Connection.php to use its functions.

In the static function login that receives two parameters $usuario and $clave, by means of the function to initiate of the class Connection we store in the variable $con a connection with the database, and using this variable $con we make the consultation to the database with the variables of $usuario and $clave, we store the result of this consultation in the variable $result and we return it.

11. logout.php

In this file we make the inclusion of the Session.php file, to be able to access its static function close, which we use to delete the data and close the current session. After performing this action we perform the redirection to the base path coolac/.

12. registration.php

In this file we create the form for user registration, in principle we include the files header.php (This file gives us the structure of the main links menu that is handled in the project) and RegistroController.php (This file gives us the functionality it contains).

Initially we verify if the btnRegister button has been clicked, if the button has been clicked using the static function register of the RegistroController class we perform the action of obtaining the user's data passing all the inputs requested in the form as parameters to the function, if the request was successful we redirect to the login.php file, otherwise we show some error messages.

The following is html code in which we create the form, the inputs we need and the register button, this form is sent to the same file using the post method and the verification process of pressing the btnRegister button is performed.

In the last line we include the footer.php file to close the tags that are open from the header.php file and include the cdn that we already mentioned.

13. RegistrationController.php

In this file we include the file Connection.php to use its functions.

In the static function register that receives eight parameters $document\_type, $document\_num, $first\_name, $last\_name, $email, $phone, $user and $key, using the function start of the Connection class we store in the variable $with a connection to the database, and using this variable $con we make the insertion to the database with the variables of $document\_type, $document\_num, $first\_name, $last\_name, $email, $phone, $user and $key, we store the result of this insertion in the variable $result and we return it.

14. Session.php

In this file we create the Session class which handles everything related to the user session.

In the static function start() we receive three parameters $id, $user and $role, we verify through the function session\_id() if a session has already started if not we start it, and we create the variables $\_SESSION["id"], $\_SESSION["user"] and $\_SESSION["role"] assigning them their corresponding with the parameters received.

In the static function hasSession() we verify by means of the function session\_id() if a session has already been started, if not we start it and depending on whether the variable $\_SESSION["user"] has been started we return true or false.

In the static function close() we verify by means of the session\_id() function if a session has already been started if we did not start it and we close the user session by means of the session\_unset() and session\_destroy() functions.

15. SolicitudSaldoController.php

In this file we include the files Connection.php and Session.php to use their functions.

In the static function getAll() using the function start of the Connection class we store in the variable $con a connection to the database, and using this variable $con we make the query to the database depending on the role of the current user, which we get from the variable $\_SESSION["role"], and according to this we make a query that will obtain all the balance requests if it is ADMINISTRATOR role or only the balance requests of the current user if it is USER role, sending in the query the id that we obtain from the variable $\_SESSION["id"] in the second case, we store the result of the query that has been executed in the variable $result and we return it.

In the static function create() that receives the parameter $quantity, by means of the function initiate of the class Connection we store in the variable $con a connection with the database, and using this variable $con we make the insertion to the database with the variables of $quantity and $id( that we obtain from the variable $\_SESSION["id"] of the session of the current user), we store the result of this insertion in the variable $result and we return it.

In the static function changeState() that receives two parameters $id and $state, using the function initiate of the Connection class we store in the variable $con a connection with the database, and using this variable $con we perform the update to the database with the variables of $state and $id, with which we update the state of the balance request and if the variable $state is equal to 6 (state = approve) we perform the query to the database to obtain the balance request that is being processed, the result of this query we store it in the variable $result and from this variable we obtain the user\_id and the amount that we store in the variables $user\_id and $amount respectively, later with these two new variables created we make another update in the database in which according to the user\_id we increase the balance in the value that we have in the variable $amount, finally we store the result of the last query to the database that we have made, whatever it was, in the variable $result and we return it.

16. UserController.php

In this file we include the files Connection.php and Session.php to use its functions.

In the static function getUsuario() by means of the function initiate of the class Connection we store in the variable $con a connection with the database, and using this variable $con we make the consultation to the database to obtain the current user, sending in the consultation the id that we obtain of the variable $\_SESSION["id"] and that we store in the variable $id, we store the result of the consultation in the variable $result and we return it.