Documentación del proyecto

El proyecto contiene en total 16 archivos, que se detallarán a continuación:

1. chage\_state.php

En este archivo realizamos la inclusión del archivo SolicitudSaldoController.php, para poder acceder a su método estático chageState, el cual lo utilizamos para realizar la acción de aprobación o rechazo de la solicitud de saldo. Luego de realizar esta acción realizamos el redireccionamiento a la página listado.php.

1. Conexión.php

En este archivo creamos la clase Conexión la cual maneja todo lo relacionado con la conexión a la base de datos, dentro de esta declaramos los atributos estáticos $servername, $database, $username y $password, que nos servirán para realizar la conexión con la base de datos, el atributo estatico $con almacenará la conexión con la base de datos.

En la función estática iniciar() mediante la función mysqli\_connect(), pasando los parámetros $servername, $database, $username y $password, nos conectamos a la base de datos y almacenamos la conexión en la variable $con, luego verificamos que se haya establecido una conexión de no ser asi empleoamos la función die() que nós mostrará un mensaje de error, si todo ha salido correctamente retornamos el valor de la variable $con.

En la función estática getConexion() retornamos el valor de la variable $con.

En la función estática isConectado() verificamos si tenemos establecida una conexión con la base de datos y dependiendo de esto retornamos true o false

En la función estática cerrar() mediante la función mysqli\_close() cerramos la conexión con la base de datos.

1. footer.php

En este archivo cerramos las etiqueta que están abiertas desde el archivo header.php y agregamos la etiqueta script con el cdn de javascript del framework bootstrap.

1. form.php

En este archivo creamos el formulario para la solicitud de saldo, en principio incluimos los archivos header.php (Este archivo nos brinda la estructura del menú de links principal que se maneja en el proyecto), SolicitudSaldoController.php y Session.php (Estos archivos nos brindan las funcionalidades que contienen).

Luego de incluir los archivos necesarios, verificamos mediante la función estática hasSession de la clase Sessión si el usuario ya ha ingresado al sistema mediante sus datos de usuario, y si no ha ingresado al sistema lo redirecciona al archivo login.php, si el usuario ya ha ingresado al sistema con su usuario y contraseña se verifica si se ha pulsado en el botón btnSolictudSaldo, de haber sido pulsado el botón mediante la función estática create de la clase SolicitudSaldoController realizamos la acción de crear esta solicitud pasándole el valor solicitado como parámetro a la función, si la solicitud se realizó correctamente redireccionamos al archivo listado.php sino mostramos un mensaje de error.

Finalmente lo siguiente es código html en el cual creamos un formulario con el input de cantidad y el botón de enviar, este formulario se envía al mismo archivo mediante el método post y se realiza el proceso de verificación de pulsado del botón btnSolicitudSaldo.

1. header.php

En este archivo creamos el menú de links principal que emplearemos en la aplicación, dentro de su etiqueta head agregamos la etiqueta link con el cdn de css del framework de bootstrap, en la primera línea incluimos el archivo Session.php para utilizar sus funciones, dentro de menú de links empleando la función estática hasSession de la clase Session mostramos o no algunos enlace dependiendo de si el usuario ha o no iniciado sesión en el sistema.

1. home.php

En este archivo realizamos la inclusión de los archivos header.php (Este archivo nos brinda la estructura del menú de links principal que se maneja en el proyecto), Session.php y UsuarioController.php (Estos archivos nos brindan las funcionalidades que contienen).

Luego de incluir los archivos necesarios, verificamos mediante la función estática hasSession de la clase Sessión si el usuario ya ha ingresado al sistema mediante sus datos de usuario, y si no ha ingresado al sistema lo redirecciona al archivo login.php.

Después, si no redireccionamos al login, mediante la función estática getUsuario() de la clase UsuarioController obtenemos los datos del usuario actual y los almacenamos en la variable $usuario.

Finalmente mostramos los datos del usuario empleando la variable $usuario, dependiendo del valor del rol almacenado en la variable global $\_SESSION mostramos o no el saldo del usuario.

En la última línea incluimos el archivo footer.php para cerrar las etiquetas que vienen abiertas del archivo header.php e incluir el cdn que ya mencionamos.

1. index.php

En este archivo realizamos la inclusión de los archivos header.php (Este archivo nos brinda la estructura del menú de links principal que se maneja en el proyecto). Despues agregamos una imagen mediante la etiqueta img.

En la última línea incluimos el archivo footer.php para cerrar las etiquetas que vienen abiertas del archivo header.php e incluir el cdn que ya mencionamos.

1. listado.php

En este archivo visualizamos el listado de solicitudes de saldo, en principio incluimos los archivos header.php (Este archivo nos brinda la estructura del menú de links principal que se maneja en el proyecto), SolicitudSaldoController.php y Session.php (Estos archivos nos brindan las funcionalidades que contienen).

Luego de incluir los archivos necesarios, verificamos mediante la función estática hasSession de la clase Sessión si el usuario ya ha ingresado al sistema mediante sus datos de usuario, y si no ha ingresado al sistema lo redirecciona al archivo login.php.

Después, si no redireccionamos al login, mediante la función estática getAll() de la clase SolicitudSaldoController obtenemos las solicitudes de saldo dependiendo del usuario actual, si es de rol tipo ADMINISTRADOR obtenemos todas las solicitudes, si es de rol tipo USUARIO obtenemos todas la solicitud que ha realizado el usuario actual, y almacenamos las solicitudes en la variable $res.

Luego dependiendo de si el tipo de rol del usuario actual almacenado en la variable global $\_SESSION es igual a USUARIO mostramos el botón Crear Solicitud de Saldo.

Posteriormente mediante código HTML realizamos la visualización de los datos de las solicitudes almacenadas en la variable $res, también dependiendo de si el tipo de rol del usuario actual almacenado en la variable global $\_SESSION es igual a ADMINISTRADOR y el estado de la solicitud es igual a pendiente mostramos la columna acción en la cual encontramos dos botones que realizan las acciones de aprobar y rechazar.

En estos botones que en realidad son etiquetas a, creamos los enlaces que llevan al archivo change\_state.php con parámetros dependiendo del botón que es pulsado.

En la última línea incluimos el archivo footer.php para cerrar las etiquetas que vienen abiertas del archivo header.php e incluir el cdn que ya mencionamos.

1. login.php

En este archivo creamos el formulario de login, en principio incluimos los archivos header.php (Este archivo nos brinda la estructura del menú de links principal que se maneja en el proyecto), LoginController.php y Session.php (Estos archivos nos brindan las funcionalidades que contienen).

Inicialmente se verifica si se ha pulsado en el botón btnLogin, de haber sido pulsado el botón mediante la función estática login de la clase LoginController realizamos la acción de obtener los datos del usuario pasándole el usuario y la clave como parámetros a la función, si la petición se realizó correctamente (el usuario y la clave son correctas) redireccionamos al archivo home.php sino mostramos un mensaje de error.

Lo siguiente es código html en el cual creamos un formulario con el input de usuario, el input de clave y el botón de entrar, este formulario se envía al mismo archivo mediante el método post y se realiza el proceso de verificación de pulsado del botón btnLogin.

En la última línea incluimos el archivo footer.php para cerrar las etiquetas que vienen abiertas del archivo header.php e incluir el cdn que ya mencionamos.

1. LoginController.php

En este archivo incluimos el archivo Conexión.php para usar sus funciones.

En la función estática login que recibe dos parámetros $usuario y $clave, mediante la función iniciar de la clase Conexión almacenamos en la variable $con una conexión con la base de datos, y utilizando esta variable $con realizamos la consulta a la base de datos con las variables de $usuario y $clave, almacenamos el resultado de esta consulta en la variable $result y la retornamos.

1. logout.php

En este archivo realizamos la inclusión del archivo Session.php, para poder acceder a su función estático cerrar, el cual lo utilizamos para eliminar los datos y cerrar la sesión actual. Luego de realizar esta acción realizamos el redireccionamiento a la ruta base coolac/.

1. registro.php

En este archivo creamos el formulario para el registro de usuarios, en principio incluimos los archivos header.php (Este archivo nos brinda la estructura del menú de links principal que se maneja en el proyecto) y RegistroController.php (Este archivo nos brinda las funcionalidades que contiene).

Inicialmente se verifica si se ha pulsado en el botón btnRegistrar, de haber sido pulsado el botón mediante la función estática registrar de la clase RegistroController realizamos la acción de obtener los datos del usuario pasándole todos los inputs que solicitamos en el formulario como parámetros a la función, si la petición se realizó correctamente redireccionamos al archivo login.php sino mostramos unos mensajes de error.

Lo siguiente es código html en el cual creamos el formulario, los inputs que necesitamos y el botón de registrarse, este formulario se envía al mismo archivo mediante el método post y se realiza el proceso de verificación de pulsado del botón btnRegistrar.

En la última línea incluimos el archivo footer.php para cerrar las etiquetas que vienen abiertas del archivo header.php e incluir el cdn que ya mencionamos.

1. RegistroController.php

En este archivo incluimos el archivo Conexión.php para usar sus funciones.

En la función estática registrar que recibe ocho parámetros $tipo\_documento, $num\_documento, $nombre, $apellido, $email, $telefono, $usuario y $clave, mediante la función iniciar de la clase Conexión almacenamos en la variable $con una conexión con la base de datos, y utilizando esta variable $con realizamos la inserción a la base de datos con las variables de $tipo\_documento, $num\_documento, $nombre, $apellido, $email, $telefono, $usuario y $clave, almacenamos el resultado de esta insercion en la variable $result y la retornamos.

1. Session.php

En este archivo creamos la clase Session la cual maneja todo lo relacionado con la sesión de usuarios.

En la función estática iniciar() recibimos tres parámetros $id, $usuario y $rol, verificamos mediante la función session\_id() si ya se ha iniciado una sesión si no la iniciamos, y creamos las variables $\_SESSION["id"], $\_SESSION["usuario"] y $\_SESSION["rol"] asignándoles sus correspondientes con los parámetros recibidos.

En la función estática hasSession() verificamos mediante la función session\_id() si ya se ha iniciado una sesión si no la iniciamos y según si se tiene iniciada la variable $\_SESSION["usuario"] retornamos true o false.

En la función estática cerrar() verificamos mediante la función session\_id() si ya se ha iniciado una sesión si no la iniciamos y cerramos la sesión de usuario mediante las funciones session\_unset() y session\_destroy().

1. SolicitudSaldoController.php

En este archivo incluimos los archivos Conexión.php y Session.php para usar sus funciones.

En la función estática getAll() mediante la función iniciar de la clase Conexión almacenamos en la variable $con una conexión con la base de datos, y utilizando esta variable $con realizamos la consulta a la base de datos dependiendo de rol del usuario actual, que obtenemos de la variable $\_SESSION["rol"], y según esto realizamos una consulta que obtendrá todos las solicitudes de saldo si es rol ADMINISTRADOR o solo las solicitudes de saldo del usuario actual si es rol USUARIO, enviando en la consulta el id que obtenemos de la variable $\_SESSION["id"] en el segundo caso, almacenamos el resultado de la consulta que se haya ejecutado en la variable $result y la retornamos.

En la función estática create() que recibe el parámetro $cantidad, mediante la función iniciar de la clase Conexión almacenamos en la variable $con una conexión con la base de datos, y utilizando esta variable $con realizamos la inserción a la base de datos con las variables de $cantidad y $id( que obtenemos de la variable $\_SESSION["id"] de la sesión del usuario actual), almacenamos el resultado de esta inserción en la variable $result y la retornamos.

En la función estática changeState() que recibe dos parámetros $id y $state, mediante la función iniciar de la clase Conexión almacenamos en la variable $con una conexión con la base de datos, y utilizando esta variable $con realizamos la actualización a la base de datos con las variables de $state y $id, con lo cual actualizamos el estado de la solicitud de saldo y si la variable $state es igual a 6 (estado = aprobar) realizamos la consulta a la base de datos para obtener la solicitud de saldo que se está tratando, el resultado de esta consulta lo almacenos en la variable $result y de esta variable obtenemos el usuario\_id y la cantidad que almacenamos en las variables $usuario\_id y $cantidad respectivamente, posteriormente con estas dos nuevas variables creadas realizamos otra actualización en la base de datos en la cual según el usuario\_id incrementamos el saldo en el valor que tenemos en la variable $cantidad, finalmente almacenamos el resultado del ultimo query a la base de datos que hayamos realizado, cualquiera que fuera, en la variable $result y la retornamos.

1. UsuarioController.php

En este archivo incluimos los archivos Conexión.php y Session.php para usar sus funciones.

En la función estática getUsuario() mediante la función iniciar de la clase Conexión almacenamos en la variable $con una conexión con la base de datos, y utilizando esta variable $con realizamos la consulta a la base de datos para obtener el usuario actual, enviando en la consulta el id que obtenemos de la variable $\_SESSION["id"] y que almacenamos en la variable $id, almacenamos el resultado de la consulta en la variable $result y la retornamos.