```php
<?php
    require "./lib/constantes.inc.php";
    /**
     * Function which return a pdo object
     * @return PDO
     */
    function dbConnect()
    {
        static $dbc = null;
        if ($dbc == null) {
            try {
                $dbc = new PDO('mysql:host=' . HOST . ';dbname=' . DBNAME, DBUSER, DBPWD, array(
                    PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8",
                    PDO::ATTR_PERSISTENT => true
                ));
            }
            catch (PDOException $e) {
                echo 'Erreur : ' . $e->getMessage() . '<br />';
                echo 'N° : ' . $e->getCode();
                die('Could not connect to MySQL');
            }
        }
        return $dbc;
    }

    /**
     * Function which create a user in the database
     * @param $email
     * @param $passwordHash
     * @param $streetName
     * @param $streetNumber
     * @param $postalCode
     * @param $city
     * @return bool
     */
    function createUser($email, $passwordHash, $streetName, $streetNumber, $postalCode, $city)
    {
        $answer = false;
        if (!checkUserExists($email))
        {
            static $ps = null;
            $sql = "INSERT INTO USER (`email`, `passwordHash`, `isAdmin`, `streetName`, `streetNumber`, `city`, `postalCode`) VALUES (:EMAIL, :PASSWORD_HASH, 0, :STREET_NAME, :STREET_NUMBER, :CITY, :POSTAL_CODE);";
            if ($ps == null)
            {
                $ps = dbConnect()->prepare($sql);
            }
            try {
                $ps->bindParam(':EMAIL', $email, PDO::PARAM_STR);
                $ps->bindParam(':PASSWORD_HASH', $passwordHash, PDO::PARAM_STR);
                $ps->bindParam(':STREET_NAME', $streetName, PDO::PARAM_STR);
                $ps->bindParam(':STREET_NUMBER', $streetNumber, PDO::PARAM_STR);
                $ps->bindParam(':POSTAL_CODE', $postalCode, PDO::PARAM_STR);
                $ps->bindParam(':CITY', $city, PDO::PARAM_STR);

                $answer = $ps->execute();
            }
            catch (PDOException $e) {
                echo $e->getMessage();
            }
        }
        return $answer;
    }

    /**
     * Function which check if a user exists in the database
     * @param $email
     * @return bool
     */
    function checkUserExists($email)
    {
        static $ps = null;
        $sql = "SELECT * FROM USER WHERE email = :EMAIL;";
        if ($ps == null)
        {
            $ps = dbConnect()->prepare($sql);
        }
        $answer = false;
        try {
            $ps->bindParam(':EMAIL', $email, PDO::PARAM_STR);
            if ($ps->execute())
            {
                $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
            }
        }
        catch (PDOException $e) {
            echo $e->getMessage();
        }
        return $answer;
    }

    /**
     * Function which if a user
     * @param $email
     * @return bool|array
     */
    function getUserInfo($email)
    {
        static $ps = null;
        $sql = 'SELECT * FROM USER WHERE email = :EMAIL;';

        if ($ps == null)
        {
            $ps = dbConnect()->prepare($sql);
        }
        $answer = false;
        try {
            $ps->bindParam(':EMAIL', $email, PDO::PARAM_STR);
            if ($ps->execute())
            {
                $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
            }
        }
        catch (PDOException $e) {
            echo $e->getMessage();
        }
        return $answer[0];
    }

    /**
     * Function which return the products in the database
     * @return bool|array
     */
    function getAllProducts()
    {
        static $ps = null;
        $sql = 'SELECT * FROM PRODUCT JOIN PICTURE_PRODUCT ON PRODUCT.idProduct = PICTURE_PRODUCT.idProduct JOIN PICTURE ON PICTURE.idPicture = PICTURE_PRODUCT.idPicture WHERE isDefaultPicture = 1 ORDER BY productName ASC;';

        if ($ps == null)
        {
            $ps = dbConnect()->prepare($sql);
        }
        $answer = false;
        try {
            if ($ps->execute())
            {
                $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
            }
        }
        catch (PDOException $e) {
            echo $e->getMessage();
        }
        return $answer;
    }

    /**
     * Function which return the product whit the id given in parameter
     * @param $idProduct
     * @return bool|array
     */
    function getProduct($idProduct)
    {
        static $ps = null;
        $sql = 'SELECT * FROM PRODUCT WHERE idProduct = :ID_PRODUCT;';

        if ($ps == null)
        {
            $ps = dbConnect()->prepare($sql);
        }
        $answer = false;
        try {
            $ps->bindParam(':ID_PRODUCT', $idProduct, PDO::PARAM_INT);
            if ($ps->execute())
            {
                $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
            }
        }
        catch (PDOException $e) {
            echo $e->getMessage();
        }
        return $answer[0];
    }

    /**
```

```php
     * Function which return the pictures of a product
     * @param $idProduct
     * @return bool|array
     * @throws PDOException
     *
     */
    function getPictures($idProduct)
    {
        static $ps = null;
        $sql = 'SELECT * FROM PICTURE JOIN PICTURE_PRODUCT ON PICTURE_PRODUCT.idPicture = PICTURE.idPicture WHERE idProduct = :ID_PRODUCT ORDER BY isDefaultPicture DESC;';

        if ($ps == null)
        {
            $ps = dbConnect()->prepare($sql);
        }
        $answer = false;
        try {
            $ps->bindParam(':ID_PRODUCT', $idProduct, PDO::PARAM_INT);
            if ($ps->execute())
            {
                $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
            }
        }
        catch (PDOException $e) {
            echo $e->getMessage();
        }
        return $answer;

    }

    /**
     * Function which return the products which have a name or a description which is like the term given in parameter
     * @param $termToSearch
     * @return bool|array
     */
    function searchProducts($termToSearch)
    {
        static $ps = null;
        $sql = 'SELECT * FROM PRODUCT JOIN PICTURE_PRODUCT ON PRODUCT.idProduct = PICTURE_PRODUCT.idProduct JOIN PICTURE ON PICTURE.idPicture = PICTURE_PRODUCT.idPicture WHERE isDefaultPicture = 1 AND (productName LIKE :TERM_TO_SEARCH OR description LIKE :TERM_TO_SEARCH) ORDER BY PRODUCT.productName ASC;';
        $termToSearch = "%" . $termToSearch . "%";
        if ($ps == null)
        {
            $ps = dbConnect()->prepare($sql);
        }
        $answer = false;
        try {
            $ps->bindParam(':TERM_TO_SEARCH', $termToSearch, PDO::PARAM_STR);
            if ($ps->execute())
            {
                $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
            }
        }
        catch (PDOException $e) {
            echo $e->getMessage();
        }
        return $answer;
    }

    /**
     * Function which add a product in the $_SESSION['shoppingBasket']
     * @param $idProduct
     * @param $quantity
     * @return bool
     *
     */
    function addProductToShoppingBasketSession($idProduct, $quantity)
    {
        session_start();
        if (isset($_SESSION['shoppingBasket'][$idProduct]))
        {
            $_SESSION['shoppingBasket'][$idProduct] += $quantity;
        }
        else {
            $_SESSION['shoppingBasket'][$idProduct] = $quantity;
        }
    }

    /**
     * Function which add a product to the shopping basket
     * @param $email
     * @param $idProduct
     * @param $quantity
     * @return bool
     *
     */
    function addProductToShoppingBasket($email, $idProduct, $quantityToChange)
    {
        static $ps = null;
        $answer = false;
        if (orderExist($email))
        {
            updateRemainingNumber($quantityToChange, $idProduct, true);
            $totalPrice = getTotalPrice(getIdOrder($email));
            $totalPrice += $quantityToChange * getProductPrice($idProduct);
            updateTotalPrice($totalPrice, $_SESSION["email"]);
            if (productOrderedExists($email, $idProduct))
            {
                $sql = "UPDATE PRODUCT_ORDERED SET quantity = (SELECT quantity FROM PRODUCT_ORDERED WHERE idProduct = :ID_PRODUCT AND idOrder = :ID_ORDER) + :QUANTITY WHERE idProduct = :ID_PRODUCT AND idOrder = :ID_ORDER;";
                if ($ps == null)
                {
                    $ps = dbConnect()->prepare($sql);
                }
                try {
                    $idOrder = getIdOrder($email);
                    $ps->bindParam(':ID_PRODUCT', $idProduct, PDO::PARAM_INT);
                    $ps->bindParam(':ID_ORDER', $idOrder, PDO::PARAM_INT);
                    $ps->bindParam(':QUANTITY', $quantityToChange, PDO::PARAM_INT);
                    $ps->bindParam(':EMAIL', $email, PDO::PARAM_STR);
                    $answer = $ps->execute();
                }
                catch (PDOException $e) {
                    echo $e->getMessage();
                }
            }
            else {
                $sql = "INSERT INTO PRODUCT_ORDERED (`idProduct`, `idOrder`, `quantity`) VALUES (:ID_PRODUCT, (SELECT idOrder FROM ORDERED WHERE email = :EMAIL AND isPaid = 0), :QUANTITY);";
                if ($ps == null)
                {
                    $ps = dbConnect()->prepare($sql);
                }
                try {
                    $ps->bindParam(':ID_PRODUCT', $idProduct, PDO::PARAM_INT);
                    $ps->bindParam(':EMAIL', $email, PDO::PARAM_STR);
                    $ps->bindParam(':QUANTITY', $quantityToChange, PDO::PARAM_INT);
                    $answer = $ps->execute();
                }
                catch (PDOException $e) {
                    echo $e->getMessage();
                }
            }
        }
        else {
            newOrder($email, 0);
            $answer = addProductToShoppingBasket($email, $idProduct, $quantityToChange);
        }
        return $answer;
    }

    /**
     * Function which return the price in CHF of a product
     * @param $idProduct
     * @return bool|array
     *
     */
    function getProductPrice($idProduct)
    {
        static $ps = null;
        $sql = 'SELECT priceInCHF FROM PRODUCT WHERE idProduct = :ID_PRODUCT;';
        if ($ps == null)
        {
            $ps = dbConnect()->prepare($sql);
        }
        $answer = false;
        try {
            $ps->bindParam(':ID_PRODUCT', $idProduct, PDO::PARAM_INT);
            if ($ps->execute())
            {
                $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
            }
        }
        catch (PDOException $e) {
            echo $e->getMessage();
        }
        return $answer[0]["priceInCHF"];
    }

    /**
     * Function which return the total price of the order
     * @param $idOrder
     * @return bool|array
     *
     */
    function getTotalPrice($idOrder)
    {
        static $ps = null;
        $sql = 'SELECT totalPrice FROM ORDERED WHERE idOrder = :ID_ORDER AND isPaid
= 0;';
        if ($ps == null)
        {
            $ps = dbConnect()->prepare($sql);
        }
        $answer = false;
        try {
            $ps->bindParam(':ID_ORDER', $idOrder, PDO::PARAM_INT);
            if ($ps->execute())
            {
                $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
            }
        }
        catch (PDOException $e) {
            echo $e->getMessage();
        }
        return $answer[0]["totalPrice"];
    }

    /**
     * Function which add a product for a user which is not connected
     * @param $productId
     * @param $quantityToAdd
     *
     */
    function addProductToShoppingBasketInSession($idProduct, $quantityToAdd)
    {
        session_start();
        updateRemainingNumber($quantityToAdd, $idProduct, true);
        $_SESSION['shoppingBasket'][$idProduct] += $quantityToAdd;
    }

    /**
     * Function which return the id of the "active" order of a user
     * @param $email
     * @return bool|int
     *
     */
    function getIdOrder($email)
    {
        static $ps = null;
        $sql = 'SELECT idOrder FROM ORDERED WHERE email = :EMAIL AND isPaid = 0;';
        if ($ps == null)
        {
            $ps = dbConnect()->prepare($sql);
        }
        $answer = false;
        try {
            $ps->bindParam(':EMAIL', $email, PDO::PARAM_STR);
            if ($ps->execute())
            {
                $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
            }
        }
        catch (PDOException $e) {
            echo $e->getMessage();
        }
        return $answer[0]["idOrder"];
    }

    /**
     * Function which return the products which are in the shopping basket
     * @param $idProduct
     * @param $idOrder
     * @return bool|array
     *
     */
    function getUserQuantityForProduct($idProduct, $idOrder)
    {
        static $ps = null;
        $sql = 'SELECT quantity FROM PRODUCT_ORDERED WHERE idProduct = :ID_PRODUCT 
AND idOrder = :ID_ORDER;';
        if ($ps == null)
        {
            $ps = dbConnect()->prepare($sql);
        }
        $answer = false;
        try {
            $ps->bindParam(':ID_PRODUCT', $idProduct, PDO::PARAM_INT);
            $ps->bindParam(':ID_ORDER', $idOrder, PDO::PARAM_INT);
            if ($ps->execute())
            {
                $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
            }
        }
        catch (PDOException $e) {
            echo $e->getMessage();
        }
        return $answer[0]['remainingNumber'];
    }

    /**
     * Function which calculate the remaining number of a product and change it
     * @param $quantity
     * @param $idProduct
     * @param $hasReduce
     *
     */
    function updateRemainingNumber($quantity, $idProduct, $hasReduce)
    {
        $remainingNumber = getQuantity($idProduct);
        if ($hasReduce)
        {
            $remainingNumber -= $quantity;
        }
        else {
            $remainingNumber += $quantity;
        }
        updateQuantity($idProduct, $remainingNumber);
    }

    /**
```

```php
     * Function which update the quantity of a product
     * @param $idProduct
     * @param $quantity
     * @return bool
     *
     */
    function updateQuantity($idProduct, $quantity)
    {
        static $ps = null;
        $sql = "UPDATE PRODUCT SET remainingNumber = :QUANTITY WHERE idProduct = 
:ID_PRODUCT;";
        if ($ps == null)
        {
            $ps = dbConnect()->prepare($sql);
        }
        $answer = false;
        try {
            $ps->bindParam(':QUANTITY', $quantity, PDO::PARAM_INT);
            $ps->bindParam(':ID_PRODUCT', $idProduct, PDO::PARAM_INT);
            $answer = $ps->execute();
        }
        catch (PDOException $e) {
            echo $e->getMessage();
        }
        return $answer;
    }

    /**
     * Function which return the quantity of a product
     * @param $idProduct
     * @return int
     */
    function getQuantity($idProduct)
    {
        static $ps = null;
        $sql = 'SELECT remainingNumber FROM PRODUCT WHERE idProduct = 
:ID_PRODUCT;';
        if ($ps == null)
        {
            $ps = dbConnect()->prepare($sql);
        }
        $answer = false;
        try {
            $ps->bindParam(':ID_PRODUCT', $idProduct, PDO::PARAM_INT);
            if ($ps->execute())
            {
                $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
            }
        }
        catch (PDOException $e) {
            echo $e->getMessage();
        }
        return $answer[0]['remainingNumber'];
    }

    /**
     * Function which check if the user has already ordered a product
     * @param string $email
     * @param int $idProduct
     * @return bool|array
     *
     */
    function productOrderedExists($email, $idProduct)
    {
        static $ps = null;
        $sql = 'SELECT * FROM PRODUCT_ORDERED JOIN ORDERED ON PRODUCT_ORDERED.idOrder = ORDERED.idOrder WHERE isPaid = 0 AND EMAIL = :EMAIL AND idProduct = :ID_PRODUCT;';
        if ($ps == null)
        {
            $ps = dbConnect()->prepare($sql);
        }
        $answer = false;
        try {
            $ps->bindParam(':EMAIL', $email, PDO::PARAM_STR);
            $ps->bindParam(':ID_PRODUCT', $idProduct, PDO::PARAM_STR);
            if ($ps->execute())
            {
                $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
            }
        }
        catch (PDOException $e) {
            echo $e->getMessage();
        }
        return $answer;
    }

    /**
     * Function which check if an order is already "active" for a user
     * @param string $email
     * @return bool|array
     *
     */
    function orderExist($email)
    {
        static $ps = null;
        $sql = 'SELECT * FROM ORDERED WHERE isPaid = 0 AND email = :EMAIL;';
        if ($ps == null)
        {
            $ps = dbConnect()->prepare($sql);
        }
        $answer = false;
        try {
            $ps->bindParam(':EMAIL', $email, PDO::PARAM_STR);
            if ($ps->execute())
            {
                $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
            }
        }
        catch (PDOException $e) {
            echo $e->getMessage();
        }
        return $answer;
    }

    /**
     * Function which create an "active" order for the user in the database
     * @param $email
     * @param $totalPrice
     * @return bool
     *
     */
    function newOrder($email, $totalPrice)
    {
        static $ps = null;
        $sql = "INSERT INTO ORDERED (`isPaid`, `isSent`, `totalPrice`, `email`)
VALUES (0, 0, :TOTAL_PRICE, :EMAIL);";
        if ($ps == null)
        {
            $ps = dbConnect()->prepare($sql);
        }
        $answer = false;
        try {
            $ps->bindParam(':EMAIL', $email, PDO::PARAM_STR);
            $ps->bindParam(':TOTAL_PRICE', $totalPrice, PDO::PARAM_STR);
            $answer = $ps->execute();
        }
        catch (PDOException $e) {
            echo $e->getMessage();
        }
        return $answer;
    }

    /**
     * Function which return the detailed description of an order
     * @param $email
     * @return bool|array
     *
     */
    function getShoppingBasket($email)
    {
        static $ps = null;
        $sql = 'SELECT PRODUCT.idProduct, productName, description, priceInCHF,
fileName, quantity FROM PRODUCT_ORDERED JOIN ORDERED ON PRODUCT_ORDERED.idOrder =
ORDERED.idOrder JOIN PRODUCT ON PRODUCT.idProduct = PRODUCT_ORDERED.idProduct JOIN
PICTURE_PRODUCT ON PICTURE_PRODUCT.idProduct = PRODUCT.idProduct JOIN PICTURE ON
PICTURE_PRODUCT.idPicture = PICTURE.idPicture WHERE isPaid = 0 AND email = :EMAIL
AND isDefaultPicture = 1;';
        if ($ps == null)
        {
            $ps = dbConnect()->prepare($sql);
        }
        $answer = false;
        try {
            $ps->bindParam(':EMAIL', $email, PDO::PARAM_STR);
            if ($ps->execute())
            {
                $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
            }
        }
        catch (PDOException $e) {
            echo $e->getMessage();
        }
        return $answer;
    }

    /**
     * Function which update the total price of an order
     * @param $newPrice
     * @param $email
     * @return bool|array
     *
     */
    function updateTotalPrice($newPrice, $email)
    {
        $idOrder = orderExist($email)[0]['idOrder'];
        static $ps = null;
        $sql = "UPDATE ORDERED SET totalPrice = :NEW_PRICE WHERE idOrder = :ID_ORDER;";
        if ($ps == null)
        {
            $ps = dbConnect()->prepare($sql);
        }
        $answer = false;
        try {
            $ps->bindParam(':NEW_PRICE', $newPrice, PDO::PARAM_INT);
            $ps->bindParam(':ID_ORDER', $idOrder, PDO::PARAM_INT);
            $answer = $ps->execute();
        }
        catch (PDOException $e) {
            echo $e->getMessage();
        }
        return $answer;
    }

    /**
     * Function which update the profil of an user
     * @param $oldEmail
     * @param $newEmail
     * @param $passwordHash
     * @param $streetName
     * @param $streetNumber
     * @param $postalCode
     * @param $city
     *
     */
    function updateProfil($oldEmail, $newEmail, $passwordHash, $streetName, $streetNumber, $postalCode, $city)
    {
        $answer = false;
        if (!checkUserExists($newEmail) || $oldEmail == $newEmail)
        {
            static $ps = null;
            $sql = "UPDATE USER SET email = :NEW_EMAIL, passwordHash = :PASSWORD_HASH, streetName = :STREET_NAME, streetNumber = :STREET_NUMBER, city = :CITY, postalCode = :POSTAL_CODE WHERE email = :OLD_EMAIL;";
            if ($ps == null)
            {
                $ps = dbConnect()->prepare($sql);
            }
            try {
                $ps->bindParam(':NEW_EMAIL', $newEmail, PDO::PARAM_STR);
                $ps->bindParam(':PASSWORD_HASH', $passwordHash, PDO::PARAM_STR);
                $ps->bindParam(':STREET_NAME', $streetName, PDO::PARAM_STR);
                $ps->bindParam(':STREET_NUMBER', $streetNumber, PDO::PARAM_STR);
                $ps->bindParam(':POSTAL_CODE', $postalCode, PDO::PARAM_STR);
                $ps->bindParam(':CITY', $city, PDO::PARAM_STR);
                $ps->bindParam(':OLD_EMAIL', $oldEmail, PDO::PARAM_STR);

                $answer = $ps->execute();
            }
            catch (PDOException $e) {
                echo $e->getMessage();
            }
        }
        return $answer;
    }

    /**
     * Function which return all the orders of a user
     * @param $email
     * @return bool|array
     *
     */
    function getOrders($email)
    {
        static $ps = null;
        $sql = 'SELECT * FROM ORDERED WHERE email = :EMAIL;';
        if ($ps == null)
        {
            $ps = dbConnect()->prepare($sql);
        }
        $answer = false;
        try {
            $ps->bindParam(':EMAIL', $email, PDO::PARAM_STR);
            if ($ps->execute())
            {
                $orders = $ps->fetchAll(PDO::FETCH_ASSOC);
            }
        }
        catch (PDOException $e) {
            echo $e->getMessage();
        }
        foreach ($orders as $order)
        {
            $answer[$order['idOrder']]["orderInfo"] = getOrder($order['idOrder'])
[0];
            $answer[$order['idOrder']]["detailOrder"] =
getProductsOfAnOrder($order['idOrder']);
        }
        return $answer;
    }

    /**
     * Function which return all the users in the database
     * @return bool|array
     *
     */
    function getClients()
    {
        static $ps = null;
        $sql = 'SELECT * FROM USER;';
        if ($ps == null)
        {
            $ps = dbConnect()->prepare($sql);
        }
        $answer = false;
        try {
            $ps->bindParam(':EMAIL', $email, PDO::PARAM_STR);
            if ($ps->execute())
            {
                $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
            }
        }
        catch (PDOException $e) {
            echo $e->getMessage();
        }
        return $answer;
    }

    /**
     * Function which return the order of a user
     * @param $idOrder
     * @return bool|array
     *
     */
    function getOrder($idOrder)
    {
        static $ps = null;
        $sql = 'SELECT * FROM ORDERED WHERE idOrder = :ID_ORDER;';
        if ($ps == null)
        {
            $ps = dbConnect()->prepare($sql);
        }
        $answer = false;
        try {
            $ps->bindParam(':ID_ORDER', $idOrder, PDO::PARAM_INT);
            if ($ps->execute())
            {
                $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
            }
        }
        catch (PDOException $e) {
            echo $e->getMessage();
        }
        return $answer;
    }

    /**
     * Function which return the products of an order
     * @param $idOrder
     * @return bool|array
     *
     */
    function getProductsOfAnOrder($idOrder)
    {
        $answer = false;
        static $ps = null;
        $sql = 'SELECT PRODUCT.idProduct, productName, description, priceInCHF,
remainingNumber, fileName, quantity FROM PRODUCT JOIN PICTURE_PRODUCT ON
PRODUCT.idProduct = PICTURE_PRODUCT.idProduct JOIN PICTURE ON PICTURE.idPicture =
PICTURE_PRODUCT.idPicture JOIN PRODUCT_ORDERED ON PRODUCT_ORDERED.idProduct =
PRODUCT.idProduct WHERE isDefaultPicture = 1 AND idOrder = :ID_ORDER;';

        if ($ps == null)
        {
            $ps = dbConnect()->prepare($sql);
        }

        try {
            $ps->bindParam(':ID_ORDER', $idOrder, PDO::PARAM_INT);
            if ($ps->execute())
            {
                $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
            }
        }
        catch (PDOException $e) {
            echo $e->getMessage();
        }
        return $answer;
    }

    /**
     * Function which rename the pictures and give them a unique name
     * @param $pictures
     * @return array
     *
     */
    function renamePictures($pictures)
    {
        for ($i=0; $i < count($pictures["name"]); $i++)
        {
            $pictures["name"][$i] = "picture_" . date('Y-m-d-H-i-s') . "_" .
uniqid() . "." . substr($pictures["type"][$i], 6);
        }
        return $pictures;
    }

    /**
     * Function which add a picture to the database
     * @param $fileName
     * @return bool
     *
     */
    function addPicture($fileName)
    {
        static $ps = null;
        $sql = "INSERT INTO `PICTURE` (`fileName`) VALUES (:FILENAME);";

        if ($ps == null)
        {
            $ps = dbConnect()->prepare($sql);
        }
        $answer = false;
        try {
            $ps->bindParam(':FILENAME', $fileName, PDO::PARAM_STR);
            $answer = $ps->execute();
        } catch (PDOException $e) {
            echo $e->getMessage();
        }
        return $answer;
    }

    /**
     * Function which link a picture to a product
     * @param $idProduct
     * @param $idPicture
     * @param $isDefaultPicture
     * @return bool
     *
     */
    function linkPictureToProduct($idProduct, $idPicture, $isDefaultPicture)
    {
        static $ps = null;
        $sql = "INSERT INTO `PICTURE_PRODUCT` (`idProduct`, `idPicture`, `isDefaultPicture`) VALUES (:ID_PRODUCT, :ID_PICTURE, :IS_DEFAULT_PICTURE);";

        if ($ps == null)
        {
            $ps = dbConnect()->prepare($sql);
        }
        $answer = false;
        try {
            $ps->bindParam(':ID_PRODUCT', $idProduct, PDO::PARAM_INT);
            $ps->bindParam(':ID_PICTURE', $idPicture, PDO::PARAM_INT);
            $ps->bindParam(':IS_DEFAULT_PICTURE', $isDefaultPicture, PDO::PARAM_BOOL);

            $answer = $ps->execute();
        } catch (PDOException $e) {
            echo $e->getMessage();
        }
        return $answer;
    }

    /**
     * Function which add a product to the database
     * @param $productName
     * @param $description
     * @param $priceInCHF
     * @param $remainingNumber
     * @return bool
     *
     */
    function addProduct($productName, $description, $priceInCHF, $remainingNumber)
    {
        static $ps = null;
        $sql = "INSERT INTO `PRODUCT` (`productName`, `description`, `priceInCHF`, `remainingNumber`) VALUES (:PRODUCT_NAME, :DESCRIPTION, :PRICE_IN_CHF, :REMAINING_NUMBER);";
        /*if ($ps == null)
        {
            $pdo = dbConnect();
            $pdo->beginTransaction();
            $ps = $pdo->prepare($sql);
        }*/
        if ($ps == null)
        {
            $ps = dbConnect()->prepare($sql);
        }
        $answer = false;
        try {
            $ps->bindParam(':PRODUCT_NAME', $productName, PDO::PARAM_STR);
            $ps->bindParam(':DESCRIPTION', $description, PDO::PARAM_STR);
            $ps->bindParam(':PRICE_IN_CHF', $priceInCHF, PDO::PARAM_STR);
            $ps->bindParam(':REMAINING_NUMBER', $remainingNumber, PDO::PARAM_INT);

            $answer = $ps->execute();

        } catch (PDOException $e) {
            //$pdo->rollBack();
            echo $e->getMessage();
        }
        return $answer;
    }

    /**
     * Function which add a product and its pictures to the database and in the
server
     * @param $productName
     * @param $description
     * @param $priceInCHF
     * @param $remainingNumber
     * @param $hasOthersPictures
     * @param $defaultPicture
     * @param $othersPictures
     *
     */
    function addProductWithPictures($productName, $description, $priceInCHF,
$remainingNumber, $hasOthersPictures, $defaultPicture, $othersPictures,
$addProduct, $idProduct = 0)
    {
        try {
            if ($addProduct)
            {
                addProduct($productName, $description, $priceInCHF,
$remainingNumber);
                $idProduct = getIdProduct($productName, $description, $priceInCHF,
$remainingNumber);
            }
            if (isset($defaultPicture["name"][0]))
            {
                addPicture($defaultPicture["name"][0]);
                $idPicture = getIdPicture($defaultPicture["name"][0]);
                linkPictureToProduct($idProduct, $idPicture, true);
            }
            if ($hasOthersPictures)
            {
                for ($i=0; $i < count($othersPictures["name"]); $i++)
                {
                    addPicture($othersPictures["name"][$i]);
                    $idPicture = getIdPicture($othersPictures["name"][$i]);
                    linkPictureToProduct($idProduct, $idPicture, false);
                }
            }
            savePictures($defaultPicture);
            savePictures($othersPictures);
        }
        catch (Exception $e) {
            return false;
        }
        return true;
    }

    /**
     * Function which return the id of a picture
     * @param $fileName
     * @return int
     *
     */
    function getIdPicture($fileName)
    {
        static $ps = null;
        $sql = 'SELECT idPicture FROM PICTURE WHERE fileName = :FILENAME;';

        if ($ps == null)
        {
            $ps = dbConnect()->prepare($sql);
        }
        $answer = false;
        try {
            $ps->bindParam(':FILENAME', $fileName, PDO::PARAM_STR);
            if ($ps->execute())
            {
                $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
            }
        }
        catch (PDOException $e) {
            echo $e->getMessage();
        }
        return $answer[0]["idPicture"];
    }

    /**
     * Function which return the id of a product
     * @param $productName
     * @param $description
     * @param $priceInCHF
     * @param $remainingNumber
     * @return int
     *
     */
    function getIdProduct($productName, $description, $priceInCHF,
$remainingNumber)
    {
        static $ps = null;
        $sql = 'SELECT idProduct FROM PRODUCT WHERE productName = :PRODUCT_NAME AND
description = :DESCRIPTION AND priceInCHF = :PRICE_IN_CHF AND remainingNumber =
:REMAINING_NUMBER;';
        if ($ps == null)
        {
            $ps = dbConnect()->prepare($sql);
        }
        $answer = false;
        try {
            $ps->bindParam(':PRODUCT_NAME', $productName, PDO::PARAM_STR);
            $ps->bindParam(':DESCRIPTION', $description, PDO::PARAM_STR);
            $ps->bindParam(':PRICE_IN_CHF', $priceInCHF, PDO::PARAM_STR);
            $ps->bindParam(':REMAINING_NUMBER', $remainingNumber, PDO::PARAM_INT);
            if ($ps->execute())
            {
                $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
            }
        }
        catch (PDOException $e) {
            echo $e->getMessage();
        }
        return $answer[0]["idProduct"];
    }

    /**
     * Function which save the pictures on the server
     * @param $pictures
     * @return bool|array
     *
     */
    function savePictures($pictures)
    {
        $picturesNb = count($pictures["name"]);
        $picturesMoved = 0;
        for ($i=0; $i < count($pictures["name"]); $i++)
        {
            $pictureMoved = false;
            $pictureMoved = move_uploaded_file($pictures['tmp_name'][$i],
PICTURES_FOLDER . $pictures["name"][$i]);
            if ($pictureMoved)
            {
                $picturesMoved++;
            }
        }
        if ($picturesMoved != $picturesNb)
        {
            return false;
        }
        else {
            return $pictures;
        }
    }

    /**
     * Function which delete a product and its associated pictures from the
database and the server
     * @param $idProduct
     * @param $deleteProduct
     *
     */
    function deleteProductAndPictures($idProduct, $deleteProduct)
    {
        $pictures = getPictures($idProduct);
        for ($i = 0; $i < count($pictures); $i++)
        {
            deleteLinkPictureToProduct($pictures[$i]["idPicture"]);
            unlink(PICTURES_FOLDER . $pictures[$i]["fileName"]);
            deletePicture($pictures[$i]["idPicture"]);
        }
        if ($deleteProduct)
        {
            deleteProduct($idProduct);
        }
    }

    /**
     * Function which delete a product from the database
     * @param $idProduct
     * @return bool|array
     *
     */
    function deleteProduct($idProduct)
    {
        static $ps = null;
        $sql = "DELETE FROM PRODUCT WHERE idProduct = :ID_PRODUCT;";
        if ($ps == null)
        {
            $ps = dbConnect()->prepare($sql);
        }
        $answer = false;
        try {
            $ps->bindParam(':ID_PRODUCT', $idProduct, PDO::PARAM_INT);
            $answer = $ps->execute();
        }
        catch (PDOException $e) {
            echo $e->getMessage();
        }
        return $answer;
    }

    /**
     * Function which delete the link in the database between the product and the picture
     * @param $idPicture
     * @return bool|array
     */
    function deleteLinkPictureToProduct($idPicture)
    {
        static $ps = null;
        $sql = "DELETE FROM PICTURE_PRODUCT WHERE idPicture = :ID_PICTURE;";
        if ($ps == null)
        {
            $ps = dbConnect()->prepare($sql);
        }
        $answer = false;
        try {
            $ps->bindParam(':ID_PICTURE', $idPicture, PDO::PARAM_INT);
            $answer = $ps->execute();
        }
        catch (PDOException $e) {
            echo $e->getMessage();
        }
        return $answer;
    }

    /**
     * Function which delete a picture from the database
     * @param $idPicture
     * @return bool|array
     *
     */
    function deletePicture($idPicture)
    {
        static $ps = null;
        $sql = "DELETE FROM PICTURE WHERE idPicture = :ID_PICTURE;";
        if ($ps == null)
        {
            $ps = dbConnect()->prepare($sql);
        }
        $answer = false;
        try {
            $ps->bindParam(':ID_PICTURE', $idPicture, PDO::PARAM_INT);
            $answer = $ps->execute();
        }
        catch (PDOException $e) {
            echo $e->getMessage();
        }
        return $answer;
    }

    /**
     * Function which update the product and its pictures using the fonction to add a product
     * @param $idProduct
     * @param $productName
     * @param $description
     * @param $priceInCHF
     * @param $remainingNumber
     * @param $hasOthersPictures
     * @param $defaultPicture
     * @param $othersPictures
     *
     */
    function updateProductPictures($idProduct, $productName, $description, $priceInCHF, $remainingNumber, $hasOthersPictures, $defaultPicture, $othersPictures)
    {
        try {
            updateProductInfos($idProduct, $productName, $description, $priceInCHF, $remainingNumber);
            addProductWithPictures($productName, $description, $priceInCHF, $remainingNumber, $hasOthersPictures, $defaultPicture, $othersPictures, false, $idProduct);
        }
        catch (Exception $e) {
            return false;
        }
        return true;
    }

    /**
     * Function which update the informations of a product
     * @param $idProduct
     * @param $productName
     * @param $description
     * @param $priceInCHF
     * @param $remainingNumber
     * @return bool|array
     *
     */
    function updateProductInfos($idProduct, $productName, $description, $priceInCHF, $remainingNumber)
    {
        $answer = false;
        static $ps = null;
        $sql = 'UPDATE PRODUCT SET productName = :PRODUCT_NAME, description = :DESCRIPTION, priceInCHF = :PRICE_IN_CHF, remainingNumber = :REMAINING_NUMBER WHERE idProduct = :ID_PRODUCT;';
        if ($ps == null)
        {
            $ps = dbConnect()->prepare($sql);
        }
        try {
            $ps->bindParam(':PRODUCT_NAME', $productName, PDO::PARAM_STR);
```

```php
            $ps->bindParam(':DESCRIPTION', $description, PDO::PARAM_STR);
            $ps->bindParam(':PRICE_IN_CHF', $priceInCHF, PDO::PARAM_STR);
            $ps->bindParam(':REMAINING_NUMBER', $remainingNumber, PDO::PARAM_INT);
            $ps->bindParam(':ID_PRODUCT', $idProduct, PDO::PARAM_INT);
            $answer = $ps->execute();
        }
        catch (PDOException $e) {
            echo $e->getMessage();
        }
        return $answer;
    }

    /**
     * Function which delete a picture
     * @param $idPicture
     *
     */
    function deleteSelectedPicture($idPicture)
    {
        deleteLinkPictureToProduct($idPicture);
        deletePicture($idPicture);
    }


    /**
     * Function which return product's information
     * @param $idProduct
     * @return bool|array
     *
     */
    function getProductDetails($idProduct)
    {
        $answer = false;
        static $ps = null;
        $sql = 'SELECT PRODUCT.idProduct, productName, description, priceInCHF,
remainingNumber, fileName FROM PRODUCT JOIN PICTURE_PRODUCT ON PRODUCT.idProduct =
PICTURE_PRODUCT.idProduct JOIN PICTURE ON PICTURE.idPicture =
PICTURE_PRODUCT.idPicture WHERE isDefaultPicture = 1 AND PRODUCT.idProduct =
:ID_PRODUCT;';

        if ($ps == null)
        {
            $ps = dbConnect()->prepare($sql);
        }

        try {
            $ps->bindParam(':ID_PRODUCT', $idProduct, PDO::PARAM_INT);
            if ($ps->execute())
            {
                $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
            }
        }
        catch (PDOException $e) {
            echo $e->getMessage();
        }
        return $answer[0];
    }

    /**
     * Function which change the state of the payement
     * @param $idOrder
     * @param $isPaid
     * @return bool
     *
     */
    function switchPaid($idOrder, $isPaid)
    {
        $answer = false;
        static $ps = null;
        $sql = 'UPDATE ORDERED SET isPaid = :IS_PAID WHERE idOrder = :ID_ORDER;';
        if ($ps == null)
        {
            $ps = dbConnect()->prepare($sql);
        }
        try {
            $ps->bindParam(':ID_ORDER', $idOrder, PDO::PARAM_INT);
            $ps->bindParam(':IS_PAID', $isPaid, PDO::PARAM_BOOL);
            $answer = $ps->execute();
        }
        catch (PDOException $e) {
            echo $e->getMessage();
        }
        return $answer;
    }

    /**
     * Function which change the state of the sending
     * @param $idOrder
     * @param $isPaid
     * @return bool
     *
     */
    function switchSent($idOrder, $isSent)
    {
        $answer = false;
        static $ps = null;
        $sql = 'UPDATE ORDERED SET isSent = :IS_SENT WHERE idOrder = :ID_ORDER;';
        if ($ps == null)
        {
            $ps = dbConnect()->prepare($sql);
        }
        try {
            $ps->bindParam(':ID_ORDER', $idOrder, PDO::PARAM_INT);
            $ps->bindParam(':IS_SENT', $isSent, PDO::PARAM_BOOL);
            $answer = $ps->execute();
        }
        catch (PDOException $e) {
            echo $e->getMessage();
        }
        return $answer;
    }

    /**
     * Function which delete a product from the shoppingBasket
     * @param $idProduct
     * @param $quantity
     * @param $idOrder
     * @return bool
     *
     */
```

```php
    function deleteProductFromShoppingBasket($idProduct, $quantity, $idOrder)
    {
        static $ps = null;
        $sql = "DELETE FROM PRODUCT_ORDERED WHERE idProduct = :ID_PRODUCT AND idOrder = :ID_ORDER;";
        if ($ps == null)
        {
            $ps = dbConnect()->prepare($sql);
        }
        $answer = false;
        try {
            $ps->bindParam(':ID_PRODUCT', $idProduct, PDO::PARAM_INT);
            $ps->bindParam(':ID_ORDER', $idOrder, PDO::PARAM_INT);
            $answer = $ps->execute();
        }
        catch (PDOException $e) {
            echo $e->getMessage();
        }
        return $answer;
    }


    /**
     * Function which update a product from the shoppingBasket
     * @param $idProduct
     * @param $quantity
     * @param $idOrder
     * @return bool
     *
     */
    function updateProductInShoppingBasket($idProduct, $quantity, $idOrder)
    {
        static $ps = null;
        $sql = "UPDATE PRODUCT_ORDERED SET quantity = :QUANTITY WHERE idProduct = :ID_PRODUCT AND idOrder = :ID_ORDER;";
        if ($ps == null)
        {
            $ps = dbConnect()->prepare($sql);
        }
        $answer = false;
        try {
            $ps->bindParam(':ID_PRODUCT', $idProduct, PDO::PARAM_INT);
            $ps->bindParam(':QUANTITY', $quantity, PDO::PARAM_INT);
            $ps->bindParam(':ID_ORDER', $idOrder, PDO::PARAM_INT);
            $answer = $ps->execute();
        }
        catch (PDOException $e) {
            echo $e->getMessage();
        }
        return $answer;
    }
?>
```