TYPES DE MAGASINS

INSERT INTO `shopstype`(`idshopstype`, `shopstype`)

VALUES

(1, 'epicerie'),

(2, 'bijouterie'),

(3, 'boulangerie'),

(4, 'pharmacie'),

(5, 'chocolaterie'),

(6, 'grandes surfaces') ;

UNITES

INSERT INTO `units`(`idunits`, `unitsname`)

VALUES

(1, 'gramme'),

(2, 'milligramme'),

(3, 'kg'),

(4, 'litre'),

(5, 'piece');

SHOPS

INSERT INTO `shops` (`idshops`, `shopsName`, `shopsstreetnb`, `shopsstreetname`, `shopspostal`, `shopscountry`, `shopsimg`, `shopslogo`, `shopstype\_idshopstype`)

VALUES ('1', 'Vival', '33', 'Rue Portalis', '83330', 'Le Beausset', 'vival.jpg', 'vivalLogo.png', '1');

INSERT INTO `shops` (`idshops`, `shopsName`, `shopsstreetnb`, `shopsstreetname`, `shopspostal`, `shopscountry`, `shopsimg`, `shopslogo`, `shopstype\_idshopstype`)

VALUES ('2', 'Saoya', '42', 'Avenue des Fleurs', '83330', 'Le Beausset', 'saoya.jpg', 'saoyaLogo.png', '2');

INSERT INTO `shops` (`idshops`, `shopsName`, `shopsstreetnb`, `shopsstreetname`, `shopspostal`, `shopscountry`, `shopsimg`, `shopslogo`, `shopstype\_idshopstype`)

VALUES ('3', 'Pharmacie des Arcades', '10', 'Avenue Victor Hugo', '83330', 'Le Beausset', 'pharmacieDesArcades.jpg', 'pharmacieArcadesLogo.png', '4');

INSERT INTO `shops` (`idshops`, `shopsName`, `shopsstreetnb`, `shopsstreetname`, `shopspostal`, `shopscountry`, `shopsimg`, `shopslogo`, `shopstype\_idshopstype`)

VALUES ('4', 'Carrefour', '123', 'Avenue du Commerce', '83330', 'Le Beausset', 'carrefour.jpg', 'carrefourLogo.png', '6');

INSERT INTO `shops` (`idshops`, `shopsName`, `shopsstreetnb`, `shopsstreetname`, `shopspostal`, `shopscountry`, `shopsimg`, `shopslogo`, `shopstype\_idshopstype`)

VALUES ('5', 'Casino', '12', 'Avenue du Pre', '83330', 'Le Beausset', 'casino.jpg', 'casinoLogo.png', '6');

RAYONS DES MAGASINS

INSERT INTO `department`(`iddepartment`, `departmentname`) VALUES ('1','bracelet');

INSERT INTO `department`(`iddepartment`, `departmentname`) VALUES ('2','fruits\_et\_legumes');

INSERT INTO `department`(`iddepartment`, `departmentname`) VALUES ('3','epicerie\_salee');

PRODUCTS

INSERT INTO `products`(`idproducts`, `productsname`, `productsdescription`, `productsprice`, `productsimg`, `productsweight`, `units\_idunits`, `department\_iddepartment`)

VALUES (1, 'Bracelet en or', 'Bracelet en or 18 carats avec un design élégant.', '499.99', 'braceletOr.jpg', '0.02', '1','1');

INSERT INTO `products`(`idproducts`, `productsname`, `productsdescription`, `productsprice`, `productsimg`, `productsweight`, `units\_idunits`, `department\_iddepartment`)

VALUES (2, 'Bracelet en argent', 'Bracelet en argent blanc', '199.99', 'braceletArgent.jpg', '0.02', '1','1');

INSERT INTO `products`(`idproducts`, `productsname`, `productsdescription`, `productsprice`, `productsimg`, `productsweight`, `units\_idunits`, `department\_iddepartment`) VALUES ('3','Chips Lays','Délicieuses chips de pomme de terre Lays', '2.50', 'chipsLays.jpg', '150', '1','3');

INSERT INTO `products`(`idproducts`, `productsname`, `productsdescription`, `productsprice`, `productsimg`, `productsweight`, `units\_idunits`, `department\_iddepartment`) VALUES ('4', 'Poireau', 'Frais et croquant, idéal pour vos recettes de soupes et de plats', '1.20', 'poireau.jpg', '200', '1','2');

INSERT INTO `products` (`idproducts`, `productsname`, `productsdescription`, `productsprice`, `productsimg`, `productsweight`, `units\_idunits`, `department\_iddepartment`)

VALUES ('5', 'Salade', 'Salade fraîche et croquante, parfaite pour accompagner vos plats', '2.50', 'salade.jpg', '150', '1', '2');

SHOPS HAS ROLE

INSERT INTO `shops\_has\_department`(`shops\_idshops`, `department\_iddepartment`, `products\_idproducts`) VALUES ('2','1','1')

INSERT INTO `shops\_has\_department`(`shops\_idshops`, `department\_iddepartment`, `products\_idproducts`) VALUES ('2','1','2')

INSERT INTO `shops\_has\_department`(`shops\_idshops`, `department\_iddepartment`, `products\_idproducts`) VALUES ('1','2','4')

Selection des produits par rayons (peu importe le magasin)

SELECT \* FROM products

JOIN department ON products.department\_iddepartment = department.iddepartment

WHERE department.iddepartment = 1

SELECT productsname,productsprice FROM products

JOIN department ON products.department\_iddepartment = department.iddepartment

JOIN shops\_has\_department ON shops\_has\_department.department\_iddepartment = department.iddepartment

JOIN shops ON shops\_has\_department.shops\_idshops = shops.idshops

WHERE shops\_has\_department.department\_iddepartment = 1 AND shops\_has\_department.shops\_idshops = 2;

* **password\_hash()** prend deux paramètres : le mot de passe à hasher et l'algorithme de hachage à utiliser. **PASSWORD\_DEFAULT** utilise l'algorithme de hachage recommandé actuellement (actuellement Bcrypt), mais vous pouvez également spécifier un algorithme spécifique si nécessaire.
* Le mot de passe haché est ensuite inséré dans la base de données.

Lorsque vous vérifiez les mots de passe lors de la connexion (comme vous l'avez déjà fait avec **password\_verify()**), vous comparez le mot de passe soumis avec le mot de passe haché stocké dans la base de données. Voici un exemple :

phpCopy code

// Récupérer le mot de passe haché à partir de la base de données en utilisant l'email soumis $email = $\_POST['email']; $query = "SELECT password FROM user WHERE email = ?"; $stmt = $pdo->prepare($query); $stmt->execute([$email]); $hash = $stmt->fetchColumn(); // Vérifier si le mot de passe soumis correspond au mot de passe haché if (!$hash || !password\_verify($\_POST['password'], $hash)) { echo "Identifiants invalides."; }

// class Cart {

//   constructor() {

//     let cart = localStorage.getItem("cart);

//     if (cart === null) {

//       this.cart = [];

//     } else {

//       this.cart = JSON.parse(cart);

//     }

//   }

//   saveCart() {

//     localStorage.setItem("cart", JSON.stringify(this.cart));

//   }

//   addToCart(product) {

//     let foundProduct = this.cart.find((p) => p.id === product.id);

//     if (foundProduct !== undefined) {

//       foundProduct.quantity++;

//     } else {

//       product.quantity = 1;

//       this.cart.push(product);

//     }

//     this.saveCart();

//   }

//   removeFromCart(product) {

//     this.cart = this.cart.filter((p) => p.id !== product.id);

//     this.saveCart();

//   }

//   changeQuantity(product, quantity) {

//     let foundProduct = this.cart.find((p) => p.id === product.id);

//     if (foundProduct !== undefined) {

//       foundProduct.quantity += quantity;

//       if (foundProduct.quantity <= 0) {

//         this.removeFromCart(foundProduct);

//       }

//     }

//     this.saveCart();

//   }

//   getNumberProduct() {

//     let number = 0;

//     for (let product of this.cart) {

//       number += product.quantity;

//     }

//     return number;

//   }

//   getTotalPrice() {

//     let total = 0;

//     for (let product of this.cart) {

//       total += product.quantity \* product.price;

//     }

//     return total;

//   }

// }