

# Selected files

## 60 printable files

```
src/assets/js/validateAddArticle.js
src/assets/js/validateAddCategory.js
src/assets/js/validateEmail.js
src/assets/js/validateLogin.js
src/assets/js/validateRegister.js
src/assets/js/validateUpdateArticle.js
src/assets/js/validateUpdateCategory.js
src/containers/Article.php
src/containers/CartItem.php
src/containers/Category.php
src/containers/City.php
src/containers/Image.php
src/containers/User.php
src/includes/checkAll.php
src/includes/checkSessionHeader.php
src/includes/mailer.php
src/includes/nav.php
src/includes/web.inc.all.php
src/model/article.php
src/model/cartItem.php
src/model/category.php
src/model/city.php
src/model/image.php
src/model/user.php
src/tests/testArticle.php
src/tests/testCartItem.php
src/tests/testCategory.php
src/tests/testCity.php
src/tests/testImage.php
src/tests/testsessions.php
src/tests/testUser.php
src/tools/addArticleTools.php
src/tools/articleDetailsTools.php
src/tools/indexTools.php
src/tools/registerTools.php
src/tools/saveCart.php
src/addArticle.php
src/validateOrder.php
src/updateCategory.php
src/updateArticle.php
src/search.php
src/register.php
src/logout.php
src/login.php
src/index.php
src/cart.php
src/addCategory.php
src/articleDetails.php
src/addArticle.php
src/session/SessionManager.php
src/session/SecureSession.php
src/model/user.php
src/includes/web.inc.all.php
src/db/database.php
src/containers/User.php
src/config/sessionparam.php
src/config/phpmailerparam.php
src/config/conparam.php
src/assets/js/validateUpdateCategory.js
src/assets/css/style.css
```

## src/assets/js/validateAddArticle.js

```
let featured = document.getElementById("featured");

// Permet de valider les entrer dans le formulaire
function validateForm()
{
    // Saisie de l'utilisateur
    let name = document.getElementById("name");                                // Récupère le nom
    let description = tinymce.get("description").getContent();                  // Récupère la description
    let price = document.getElementById("price");                                // Récupère le prix
    let image = document.getElementById("image");                                // Récupère les images
    let category = document.getElementById("categories");                        // Récupère la categorie
    let stock = document.getElementById("stock");                                // Récupère la quantité en
stock

    // Message d'erreur afficher dans une div
    let nameErrorMsg = document.getElementById("nameHelp");
    let descriptionErrorMsg = document.getElementById("descriptionHelp");
    let priceErrorMsg = document.getElementById("priceHelp");
    let imageErrorMsg = document.getElementById("imageHelp");
    let categoryErrorMsg = document.getElementById("categoryHelp");
    let stockErrorMsg = document.getElementById("stockHelp");
    let errorMsg = document.getElementById("errorMsg");

    toggleCheckboxValue(featured);

    // Vérification de la quantité en stock
    if (stock.value.length <= 0)
    {
        // Affiche un message d'erreur
        stockErrorMsg.innerText = "Veuillez saisir une quantité.";
        stock.focus();
    }
    else
    {
        // Efface le message d'erreur
        stockErrorMsg.innerText = "";
    }

    // Vérification de la categorie
    if (category.value === "0")
    {
        // Affiche un message d'erreur
        categoryErrorMsg.innerText = "Veuillez choisir une catégorie.";
        category.focus();
    }
    else
    {
        // Efface le message d'erreur
        categoryErrorMsg.innerText = "";
    }

    // Vérification des images
    if (image.files.length == 0)
    {
        // Affiche un message d'erreur
        imageErrorMsg.innerText = "Veuillez saisir une image.";
        image.focus();
    }
}
```

```
}

else
{
    // Efface le message d'erreur
    imageErrorMsg.innerText = "";
}

// Vérification du prix
if (price.value.length <= 0)
{
    // Affiche un message d'erreur
    priceErrorMsg.innerText = "Le prix doit être plus grand que 0.";
    price.focus();
}
else
{
    // Efface le message d'erreur
    priceErrorMsg.innerText = "";
}

// Vérification de la description
if (description.length < 10)
{
    // Affiche un message d'erreur
    descriptionErrorMsg.innerText = "La description doit contenir au moins 10
caractères.";
    tinymce.get("description").focus(); // Définit le focus sur le champ de texte
}
else
{
    // Efface le message d'erreur
    descriptionErrorMsg.innerText = "";
}

// Vérification du nom
if (name.value.length < 10 || name.value.length > 200)
{
    // Affiche un message d'erreur
    nameErrorMsg.innerText = "Le nom de l'article doit contenir entre 10 et 200
caractères.";
    name.focus();
}
else
{
    // Efface le message d'erreur
    nameErrorMsg.innerText = "";
}

// Si les messages d'erreur sont vides on retourne true sinon false
if (nameErrorMsg.innerText === "" && descriptionErrorMsg.innerText === "" &&
    priceErrorMsg.innerHTML === "" && imageErrorMsg.innerText === "" && categoryErrorMsg.innerText
    === "" && stockErrorMsg.innerText === "")
{
    // Renvoie vrai, la saisie est valide
    return true;
}
else
{
    // Efface le message d'erreur générer avec php
    errorMsg.innerHTML = "";
}
```

```

        // Renvoie faux, il y a des erreurs de saisie
        return false;
    }

}

// Regarde la valeur de la check box et change la valeur de featured
function toggleCheckboxValue(featured)
{
    if (featured.checked)
    {
        featured.value = "1";
    }
    else
    {
        featured.value = "0";
    }
}

```

## src/assets/js/validateAddCategory.js

```

// Met les lettres en majuscule
function toUpperCase()
{
    let name = document.getElementById("name");

    // Converti en majuscule
    name.value = name.value.toUpperCase();
}

// Permet de valider les entrer dans le formulaire
function validateForm()
{
    let name = document.getElementById("name");
    let nameErrorMsg = document.getElementById("nameHelp");
    let errorMsg = document.getElementById("errorMsg");

    // Si le champ est vide on affiche un message d'erreur
    if (name.value === "")
    {
        // Affiche un message d'erreur
        nameErrorMsg.innerText = "Le nom est requis.";
        name.focus();
    }
    else
    {
        // Efface le message d'erreur
        nameErrorMsg.innerText = "";
    }

    // Si les messages d'erreur sont vides on retourne true sinon false
    if (nameErrorMsg.innerText === "")
    {
        // Renvoie vrai, la saisie est valide
        return true;
    }
    else
    {
        // Efface le message d'erreur générer avec php

```

```

        errorMsg.innerHTML = "";

        // Renvoie faux, il y a des erreurs de saisie
        return false;
    }
}

```

## src/assets/js/validateEmail.js

```

/**
 * Vérifie si l'adresse email est valide en utilisant une expression régulière.
 *
 * @param {string} email - L'adresse email à vérifier.
 * @returns {boolean} - true si l'adresse email est valide, false sinon.
 */
function isValidEmail(email)
{
    // Expression régulière pour vérifier le format de l'email
    let emailRegex = /^[^\w-\.]+\@([^\w-]+\.)+[\w-]{2,4})?$/;

    // Renvoie vrai si le format est valide, faux sinon
    return emailRegex.test(email);
}

```

## src/assets/js/validateLogin.js

```

/**
 * Fonction qui valide la saisie de l'utilisateur lors de la connexion
 * et affiche les messages d'erreur appropriés.
 *
 * @return {boolean} True si la validation est réussie, False sinon.
 */
function validateLogin()
{
    // Saisie utilisateur
    let email = document.getElementById("email"); // Récupère l'élément input pour l'email
    let password = document.getElementById("password"); // Récupère l'élément input pour le
mot de passe

    // Message d'erreur
    let emailErrorMsg = document.getElementById("emailHelp"); // Récupère l'élément div pour
afficher le message d'erreur de l'email
    let passwordErrorMsg = document.getElementById("passwordHelp"); // Récupère l'élément div
pour afficher le message d'erreur du mot de passe
    let errorMsg = document.getElementById("errorMsg");

    // Vérification du mot de passe
    if (password.value === "")
    {
        // Affiche un message d'erreur
        passwordErrorMsg.innerText = "Le mot de passe est requis.";

        // Met le focus sur l'input de l'email
        password.focus();
    }
}

```

```

else
{
    // Efface le message d'erreur du mot de passe
    passwordErrorMsg.innerText = "";
}

// Vérification de l'email
if (email.value === "")
{
    // Affiche un message d'erreur
    emailErrorMsg.innerText = "L'email est requis.";

    // Met le focus sur l'input de l'email
    email.focus();
}
else if (!isValidEmail(email.value))
{
    // Affiche un message d'erreur
    emailErrorMsg.innerText = "Veuillez saisir un email valide.";

    // Met le focus sur l'input de l'email
    email.focus();
}
else
{
    // Efface le message d'erreur de l'email
    emailErrorMsg.innerText = "";
}

// Si les messages d'erreur sont vides on retourne true sinon false
if (emailErrorMsg.innerText === "" && passwordErrorMsg.innerText === "")
{
    // Renvoie vrai, la saisie est valide
    return true;
}
else
{
    // Efface le message d'erreur générer avec php
    errorMsg.innerHTML = "";

    // Renvoie faux, il y a des erreurs de saisie
    return false;
}
}

```

## src/assets/js/validateRegister.js

```

/**
 * Fonction qui valide la saisie de l'utilisateur lors de l'inscription
 * et affiche les messages d'erreur appropriés.
 *
 * @return {boolean} True si la validation est réussie, False sinon.
 */
function validateRegister()
{
    // Saisie de l'utilisateur
    let name = document.getElementById("name");           // Récupère le nom
    let surname = document.getElementById("surname");     // Récupère le prénom

```

```

let gender = document.getElementById("gender");           // Récupère le genre
let address = document.getElementById("address1");       // Récupère l'adresse 1
let city = document.getElementById("cities");            // Récupère la ville
let zipCode = document.getElementById("zipCode");        // Récupère le code postal
let email = document.getElementById("email");            // Récupère l'email
let password = document.getElementById("password");       // Récupère le mot de passe

// Message d'erreur afficher dans une div
let nameErrorMsg = document.getElementById("nameHelp");
let surnameErrorMsg = document.getElementById("surnameHelp");
let genderErrorMsg = document.getElementById("genderHelp");
let addressErrorMsg = document.getElementById("address1Help");
let cityErrorMsg = document.getElementById("citiesHelp");
let zipCodeErrorMsg = document.getElementById("zipCodeHelp");
let emailErrorMsg = document.getElementById("emailHelp");
let passwordErrorMsg = document.getElementById("passwordHelp");
let errorMsg = document.getElementById("errorMsg");

// Vérification du mot de passe
if (password.value === "") {
    // Affiche un message d'erreur
    passwordErrorMsg.innerText = "Le mot de passe est requis.";

    // Met le focus sur l'input de l'email
    password.focus();
}
else if (!validatePasswordLength(password)) {
    // Affiche un message d'erreur
    passwordErrorMsg.innerText = "Le mot de passe doit faire 8 caractères minimum.';

    // Met le focus sur l'input de l'email
    password.focus();
}
else if (!validatePasswordComplexity(password)) {
    // Affiche un message d'erreur
    passwordErrorMsg.innerText = "Le mot de passe doit contenir au minimum une majuscule, une minuscule et un chiffre.";

    // Met le focus sur l'input de l'email
    password.focus();
}
else {
    // Efface le message d'erreur du mot de passe
    passwordErrorMsg.innerText = "";
}

// Vérification de l'email
if (email.value === "") {
    // Affiche un message d'erreur
    emailErrorMsg.innerText = "L'email est requis.";

    // Met le focus sur l'input de l'email
}

```

```
        email.focus();
    }
    else if (!isValidEmail(email.value))
    {
        // Affiche un message d'erreur
        emailErrorMsg.innerText = "Veuillez saisir un email valide.";

        // Met le focus sur l'input de l'email
        email.focus();
    }
else
{
    // Efface le message d'erreur de l'email
    emailErrorMsg.innerText = "";
}

// Vérification du code postal
if (zipCode.value === "")
{
    // Affiche un message d'erreur
    zipCodeErrorMsg.innerText = "Le code postal doit être sélectionné.";
    zipCode.focus();
}
else if (!/\d{4}"/.test(zipCode.value))
{
    // Affiche un message d'erreur
    zipCodeErrorMsg.innerText = "Le code postal doit être composé de 4 chiffres
uniquement.";
    zipCode.focus();
}
else
{
    // Efface le message d'erreur
    zipCodeErrorMsg.innerText = "";
}

// Vérification de la ville
if (city.value === 0)
{
    // Affiche un message d'erreur
    cityErrorMsg.innerText = "La ville doit être sélectionnée.";
    city.focus();
}
else
{
    // Efface le message d'erreur
    cityErrorMsg.innerText = "";
}

// Vérification de l'adresse
if (address.value === "")
{
    // Affiche un message d'erreur
    addressErrorMsg.innerText = "L'adresse est requise.";
```

```
        address.focus();
    }
    else
    {
        // Efface le message d'erreur
        addressErrorMsg.innerText = "";
    }

    // Vérification du genre
    if (gender.value === 0)
    {
        // Affiche un message d'erreur
        genderErrorMsg.innerText = "Le genre doit être sélectionné.";
        gender.focus();
    }
    else
    {
        // Efface le message d'erreur
        genderErrorMsg.innerText = "";
    }

    // Vérification du prénom
    if (surname.value === "")
    {
        // Affiche un message d'erreur
        surnameErrorMsg.innerText = "Le prénom est requis.";
        surname.focus();
    }
    else if (!/^[\w\-\s-]+$/ .test(surname.value))
    {
        // Affiche un message d'erreur
        surnameErrorMsg.innerText = "Le prénom ne doit contenir que des lettres, des espaces et des traits d'union.";
        surname.focus();
    }
    else
    {
        // Efface le message d'erreur
        surnameErrorMsg.innerText = "";
    }

    // Vérification du nom
    if (name.value === "")
    {
        // Affiche un message d'erreur
        nameErrorMsg.innerText = "Le nom est requis.";
        name.focus();
    }
    else if (!/^[\w\-\s-]+$/ .test(name.value))
    {
        // Affiche un message d'erreur
        nameErrorMsg.innerText = "Le nom ne doit contenir que des lettres, des espaces et des traits d'union.";
        name.focus();
    }
}
```

```

        }
    else
    {
        // Efface le message d'erreur
        nameErrorMsg.innerText = "";
    }

    // Si les messages d'erreur sont vides on retourne true sinon false
    if (emailErrorMsg.innerText === "" && passwordErrorMsg.innerText === "" &&
nameErrorMsg.innerHTML === "" && surnameErrorMsg.innerText === "" && genderErrorMsg.innerText
 === "" && addressErrorMsg.innerText === "" && cityErrorMsg.innerText === "" &&
zipCodeErrorMsg.innerText === "")
    {
        // Renvoie vrai, la saisie est valide
        return true;
    }
    else
    {
        // Efface le message d'erreur générer avec php
        errorMsg.innerHTML = "";

        // Renvoie faux, il y a des erreurs de saisie
        return false;
    }
}

// Vérifie la longueur minimale du mot de passe
function validatePasswordLength(password)
{
    return password.value.length >= 8;
}

// Vérifie que le mot de passe contient une majuscule, une minuscule et un chiffre
function validatePasswordComplexity(password)
{
    // Expression régulière pour vérifier la présence d'une majuscule, d'une minuscule et d'un
chiffre
    const passwordRegex = /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d).*/;

    // Teste si le mot de passe correspond à l'expression régulière
    return passwordRegex.test(password.value);
}

```

## src/assets/js/validateUpdateArticle.js

```

let featured = document.getElementById("featured");

// Permet de valider les entrer dans le formulaire
function validateForm()
{
    // Saisie de l'utilisateur
    let name = document.getElementById("name");                      // Récupère le nom
    let description = editor ? editor.getContent() : "";           // Récupère la description
    let price = document.getElementById("price");                   // Récupère le prix
    let image = document.getElementById("image");                   // Récupère les images
}

```

```
let category = document.getElementById("categories");           // Récupère la catégorie
let stock = document.getElementById("stock");                  // Récupère la quantité en
stock

// Message d'erreur afficher dans une div
let nameErrorMsg = document.getElementById("nameHelp");
let descriptionErrorMsg = document.getElementById("descriptionHelp");
let priceErrorMsg = document.getElementById("priceHelp");
let imageErrorMsg = document.getElementById("imageHelp");
let categoryErrorMsg = document.getElementById("categoryHelp");
let stockErrorMsg = document.getElementById("stockHelp");
let errorMsg = document.getElementById("errorMsg");

// Vérification de la quantité en stock
if (stock.value.length <= 0)
{
    // Affiche un message d'erreur
    stockErrorMsg.innerText = "Veuillez saisir une quantité.";
    stock.focus();
}
else
{
    // Efface le message d'erreur
    stockErrorMsg.innerText = "";
}

// Vérification de la catégorie
if (category.value === "0")
{
    // Affiche un message d'erreur
    categoryErrorMsg.innerText = "Veuillez choisir une catégorie.";
    category.focus();
}
else
{
    // Efface le message d'erreur
    categoryErrorMsg.innerText = "";
}

// Vérification des images
if (image.files.length == 0)
{
    // Affiche un message d'erreur
    imageErrorMsg.innerText = "Veuillez saisir une image.";
    image.focus();
}
else
{
    // Efface le message d'erreur
    imageErrorMsg.innerText = "";
}

// Vérification du prix
if (price.value.length <= 0)
{
    // Affiche un message d'erreur
    priceErrorMsg.innerText = "Le prix doit être plus grand que 0.";
    price.focus();
}
else
```

```

{
    // Efface le message d'erreur
    priceErrorMsg.innerText = "";
}

// Vérification de la description
if (description.length < 10) {
    // Affiche un message d'erreur
    descriptionErrorMsg.innerText = "La description doit contenir au moins 10
caractères.";
    if (editor) {
        editor.focus(); // Définit le focus sur l'éditeur TinyMCE
    }
} else {
    // Efface le message d'erreur
    descriptionErrorMsg.innerText = "";
}

// Vérification du nom
if (name.value.length < 10 || name.value.length > 200)
{
    // Affiche un message d'erreur
    nameErrorMsg.innerText = "Le nom de l'article doit contenir entre 10 et 200
caractères.";
    name.focus();
}
else
{
    // Efface le message d'erreur
    nameErrorMsg.innerText = "";
}

// Si les messages d'erreur sont vides, on retourne true, sinon false
if (nameErrorMsg.innerText === "" && descriptionErrorMsg.innerText === "" &&
priceErrorMsg.innerHTML === "" && imageErrorMsg.innerText === "" && categoryErrorMsg.innerText
==="" && stockErrorMsg.innerText === "") {
    // Renvoie vrai, la saisie est valide
    return true;
} else {
    // Efface le message d'erreur généré avec php
    errorMsg.innerHTML = "";
}

// Renvoie faux, il y a des erreurs de saisie
return false;
}

// Regarde la valeur de la check box et change la valeur de featured
function toggle(featured)
{
    let checkbox = document.getElementById(featured);
    checkbox.checked = !checkbox.checked;
}

```

## src/assets/js/validateUpdateCategory.js

```

// Met les lettres en majuscule
function toUpperCase()

```

```
{  
    let name = document.getElementById("name");  
  
    // Converti en majuscule  
    name.value = name.value.toUpperCase();  
}  
  
// Permet de valider les entrer dans le formulaire  
function validateForm()  
{  
    let name = document.getElementById("name");  
    let categories = document.getElementById("categories");  
  
    let nameErrorMsg = document.getElementById("nameHelp");  
    let categoriesErrorMsg = document.getElementById("categoriesHelp");  
    let errorMsg = document.getElementById("errorMsg");  
  
    // Si le champ est vide on affiche un message d'erreur  
    if (name.value === "")  
    {  
        // Affiche un message d'erreur  
        nameErrorMsg.innerText = "Veuillez mettre un nouveau nom";  
        name.focus();  
    }  
    else  
    {  
        // Efface le message d'erreur  
        nameErrorMsg.innerText = "";  
    }  
  
    // Si le champ est vide on affiche un message d'erreur  
    if (categories.value === "0")  
    {  
        // Affiche un message d'erreur  
        categoriesErrorMsg.innerText = "Veuillez saisir une catégorie à modifier";  
        categories.focus();  
    }  
    else  
    {  
        // Efface le message d'erreur  
        categoriesErrorMsg.innerText = "";  
    }  
  
    // Si les messages d'erreur sont vides on retourne true sinon false  
    if (nameErrorMsg.innerText === "" && categoriesErrorMsg.innerText === "")  
    {  
        // Renvoie vrai, la saisie est valide  
        return true;  
    }  
    else  
    {  
        // Efface le message d'erreur générer avec php  
        errorMsg.innerHTML = "";  
  
        // Renvoie faux, il y a des erreurs de saisie  
        return false;  
    }  
}
```

**src/containers/Article.php**

```

<?php

class Article
{
    /**
     * @var int $id L'identifiant unique de l'article
     */
    public $id;

    /**
     * @var string $name Le nom de l'article
     */
    public $name;

    /**
     * @var string $description La description de l'article
     */
    public $description;

    /**
     * @var double $price Le prix de l'article
     */
    public $price;

    /**
     * @var int $stock La quantité en stock de l'article
     */
    public $stock;

    /**
     * @var bool $featured Indique si l'article est mis en avant
     */
    public $featured;

    /**
     * @var string $creationDate La date de création de l'article au format YYYY-MM-DD
     */
    public $creationDate;

    /**
     * @var string $updateDate La date de dernière mise à jour de l'article au format YYYY-MM-
    DD
     */
    public $updateDate;

    /**
     * @var int $categoryId L'identifiant de la catégorie à laquelle l'article appartient
     */
    public $categoryId;

    /**
     * Constructeur appelé lors de la création de l'objet.
     *
     * @param int $idParam L'identifiant unique de l'article (optionnel, -1 par défaut)
     * @param string $nameParam Le nom de l'article
     * @param string $descriptionParam La description de l'article
     * @param float $priceParam Le prix de l'article
    
```

```

 * @param int $stockParam La quantité en stock de l'article
 * @param bool $featuredParam Indique si l'article est mis en avant (optionnel, false par
défaut)
 * @param string $creationDateParam La date de création de l'article au format YYYY-MM-DD
 * @param string $updateDateParam La date de dernière mise à jour de l'article au format
YYYY-MM-DD
 * @param int $categoryIdParam L'identifiant de la catégorie à laquelle l'article
appartient
 */
public function __construct($idParam = -1, $nameParam, $descriptionParam, $priceParam,
$stockParam, $featuredParam = false, $creationDateParam, $updateDateParam = null,
$categoryIdParam)
{
    $this->id = $idParam;
    $this->name = $nameParam;
    $this->description = $descriptionParam;
    $this->price = $priceParam;
    $this->stock = $stockParam;
    $this->featured = $featuredParam;
    $this->creationDate = $creationDateParam;
    $this->updateDate = $updateDateParam;
    $this->categoryId = $categoryIdParam;
}
}

```

## src/containers/CartItem.php

```

<?php

class CartItem
{
    /**
     * @var int $userId L'identifiant unique de l'utilisateur
     */
    public $userId;

    /**
     * @var int $articleId L'identifiant unique de l'article
     */
    public $articleId;

    /**
     * @var int $quantity La quantité d'articles
     */
    public $quantity;

    /**
     * Constructeur appelé lors de la création de l'objet.
     *
     * @param int $userIdParam L'identifiant unique de l'utilisateur
     * @param int $articleIdParam L'identifiant unique de l'article
     * @param int $quantityParam La quantité d'articles
     */
    public function __construct($userIdParam, $articleIdParam, $quantityParam)
    {
        $this->userId = $userIdParam;
        $this->articleId = $articleIdParam;
        $this->quantity = $quantityParam;
    }
}

```

}

**src/containers/Category.php**

```
<?php

class Category
{
    /**
     * @var int $id L'identifiant unique de la catégorie
     */
    public $id;

    /**
     * @var string $name Le nom de la catégorie
     */
    public $name;

    /**
     * Constructeur appelé lors de la création de l'objet.
     *
     * @param int $idParam L'identifiant unique de la catégorie (optionnel, -1 par défaut)
     * @param string $nameParam Le nom de la catégorie
     */
    public function __construct($idParam = -1, $nameParam)
    {
        $this->id = $idParam;
        $this->name = $nameParam;
    }
}
```

**src/containers/City.php**

```
<?php

class City
{
    /**
     * @var int $id L'identifiant unique de la ville
     */
    public $id;

    /**
     * @var string $name Le nom de la ville
     */
    public $name;

    /**
     * Constructeur appelé lors de la création de l'objet.
     *
     * @param int $idParam L'identifiant unique de la ville (optionnel, -1 par défaut)
     * @param string $nameParam Le nom de la ville
     */
    public function __construct($idParam = -1, $nameParam)
    {
```

```

    $this->id = $idParam;
    $this->name = $nameParam;
}
}

```

## src/containers/Image.php

```

<?php

class Image
{
    /**
     * @var int $id L'identifiant unique de l'image
     */
    public $id;

    /**
     * @var string $content Le contenu de l'image (en base64, par exemple)
     */
    public $content;

    /**
     * @var string $name Le nom de l'image
     */
    public $name;

    /**
     * @var string $type Le type de l'image (par exemple, "jpeg" ou "png")
     */
    public $type;

    /**
     * @var bool $mainImage Indique si l'image est principale pour un article
     */
    public $mainImage;

    /**
     * @var int $articleID L'identifiant de l'article associé à l'image
     */
    public $articleId;

    /**
     * Constructeur appelé lors de la création de l'objet.
     *
     * @param int $idParam L'identifiant unique de l'image (optionnel, -1 par défaut)
     * @param string $contentParam Le contenu de l'image
     * @param string $nameParam Le nom de l'image
     * @param string $typeParam Le type de l'image
     * @param bool $mainImageParam Indique si l'image est principale pour un article
     * @param int $articleIDParam L'identifiant de l'article associé à l'image
     */
    public function __construct($idParam = -1, $contentParam = "", $nameParam = "", $typeParam
= "", $mainImageParam = "", $articleIDParam = "")
    {
        $this->id = $idParam;
        $this->content = $contentParam;
        $this->name = $nameParam;
        $this->type = $typeParam;
    }
}

```

```

    $this->mainImage = $mainImageParam;
    $this->articleId = $articleIdParam;
}
}

```

## src/containers/User.php

```

<?php

class User
{
    /**
     * @var int $id L'identifiant unique de l'utilisateur
     */
    public $id;

    /**
     * @var string $name Le nom d'utilisateur
     */
    public $name;

    /**
     * @var string $surname Le prénom d'utilisateur
     */
    public $surname;

    /**
     * @var string $email L'email de l'utilisateur
     */
    public $email;

    /**
     * @var string $password Le mot de passe haché de l'utilisateur
     */
    public $password;

    /**
     * @var string $gender Le genre de l'utilisateur
     */
    public $gender;

    /**
     * @var string $address1 La première adresse de l'utilisateur
     */
    public $address1;

    /**
     * @var string $address2 La deuxième adresse de l'utilisateur (optionnel)
     */
    public $address2;

    /**
     * @var string $city La ville de l'utilisateur
     */
    public $city;

    /**
     * @var string $zipCode Le code postal de l'utilisateur
     */
    public $zipCode;
}

```

```

*/
public $zipCode;

/**
 * @var bool $isAdmin Indique si l'utilisateur est administrateur ou non
 */
public $isAdmin;

/**
 * Constructeur appelé au moment de la création de l'objet. new User();
 *
 * @param int $idParam L'id unique provenant de la base de données. (Optionel) Défaut -1
 * @param string $nameParam Le nom de l'utilisateur
 * @param string $surnameParam Le prénom de l'utilisateur
 * @param string $emailParam L'email de l'utilisateur
 * @param string $passwordParam Le mot de passe de l'utilisateur
 * @param string $genderParam Le genre de l'utilisateur (homme ou femme)
 * @param string $adress1Param La première ligne de l'adresse de l'utilisateur
 * @param string $adress2Param La deuxième ligne de l'adresse de l'utilisateur
 * @param string $cityParam La ville de l'utilisateur
 * @param string $zipCodeParam Le code postal de l'utilisateur
 * @param bool $isAdminParam Détermine si l'utilisateur est un administrateur (true) ou
non (false)
*/
public function __construct($idParam = -1, $nameParam = "", $surnameParam = "",
$emailParam = "", $passwordParam = "", $genderParam = "", $adress1Param = "", $adress2Param =
 "", $cityParam = "", $zipCodeParam = "", $isAdminParam = "")
{
    $this->id = $idParam;
    $this->name = $nameParam;
    $this->surname = $surnameParam;
    $this->email = $emailParam;
    $this->password = $passwordParam;
    $this->gender = $genderParam;
    $this->address1 = $adress1Param;
    $this->address2 = $adress2Param;
    $this->city = $cityParam;
    $this->zipCode = $zipCodeParam;
    $this->isAdmin = $isAdminParam;
}
}
}

```

## src/includes/checkAll.php

```

<?php
// Constante pour avoir le chemin absolu
define('ROOT', $_SERVER['DOCUMENT_ROOT']."/Flavio_Soares_Rodrigues_TPI_2023/src/");

require_once ROOT.'includes/web.inc.all.php';

// Si il faut être connecté on appelle checkSessionHeader qui gère les redirections
if (isset($REQUIREDLOGIN) && $REQUIREDLOGIN === true)
{
    include_once ROOT.'/includes/checkSessionHeader.php';
}
?>

```

**src/includes/checkSessionHeader.php**

```
<?php

// Si la session n'est pas valide (utilisateur pas connecté) on le renvoie à la page de login
if (ESessionManager::IsValid() === false)
{
    header("Location: login.php");
    exit();
}

// Si pour accéder à la page il faut être admin
if (isset($REQUIREDADMIN) && $REQUIREDADMIN === true)
{
    // Si l'utilisateur n'est pas admin on le redirige
    if (ESessionManager::IsConnectedUserAdmin() === false)
    {
        header("Location: unauthorized.php");
        exit();
    }
}
?>
```

**src/includes/mailer.php**

```
<?php

// Constantes requise pour l'envois de mail
require_once ROOT. 'config/phpmailerparam.php';

// Importe les classes de PHPMailer
use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\SMTP;
use PHPMailer\PHPMailer\Exception;

require_once ROOT. 'assets/libraries/PHPMailer/src/PHPMailer.php';
require_once ROOT. 'assets/libraries/PHPMailer/src/SMTP.php';
require_once ROOT. 'assets/libraries/PHPMailer/src/Exception.php';

/**
 * Envoyer un e-mail à l'adresse de l'utilisateur
 *
 * @param string $email      L'adresse e-mail de l'utilisateur
 * @param string $subject    Le sujet de l'e-mail
 * @param string $body       Le contenu de l'e-mail
 *
 * @return bool Vrai si envoyé, Faux en cas d'erreur.
*/
function sendEmail($email, $subject, $body)
{
    // Créer une instance ; passer `true` active les exceptions
    $mail = new PHPMailer(true);

    try
    {
```

```

/* Paramètres du serveur

    - Activer la sortie de débogage détaillée
    - DEBUG_OFF pour aucune sortie de débogage
    - DEBUG_SERVER pour une sortie de débogage
*/
$mail->SMTPDebug = SMTP::DEBUG_OFF;

// Configuration du serveur SMTP
$mail->isSMTP();                                // Utiliser SMTP pour l'envoi
$mail->Host          = MAIL_SERVER;             // Définir le serveur SMTP pour l'envoi
$mail->SMTPAuth     = true;                     // Activer l'authentification SMTP
$mail->Username      = MAIL_USERNAME;           // Nom d'utilisateur SMTP
$mail->Password      = MAIL_PASSWORD;           // Mot de passe SMTP
$mail->SMTPSecure   = MAIL_TRANS;               // Activer le chiffrement TLS implicite
$mail->Port          = MAIL_PORT;                // Port TCP pour la connexion ; utilisez 587 si
vous avez défini `SMTPSecure = PHPMailer::ENCRYPTION_STARTTLS`


// Destinataires
$mail->setFrom(MAIL_USERNAME);
$mail->addAddress(strtolower($email));
$mail->addReplyTo(MAIL_USERNAME);

// Contenu
$mail->isHTML(true);                          // Définir le format de l'e-mail en HTML
$mail->CharSet = MAIL_ENCODE;                 // Encodage du mail
$mail->Subject = $subject;
$mail->Body   = $body;
$mail->send();

}

catch (Exception $e)
{
    return false;
}
// Terminé
return true;
}

```

## src/includes/nav.php

```

<?php

function ShowNavbar()
{
    $searchFilter = "";
    $categoriesFilter = 0;
    $minPriceFilter = "";
    $maxPriceFilter = "";

    // Récupère toutes les catégories
    $categories = GetCategories();

    if (isset($_GET['filterSubmit']) && $_GET['filterSubmit'] == "submit")
    {
        $searchFilter = filter_input(INPUT_GET, "searchFilter",
FILTER_SANITIZE_SPECIAL_CHARS);
    }
}

```

```

$categoriesFilter = intval(filter_input(INPUT_GET, "categoriesFilter",
FILTER_VALIDATE_INT));
$minPriceFilter = filter_input(INPUT_GET, "minPriceFilter", FILTER_VALIDATE_INT);
$maxPriceFilter = filter_input(INPUT_GET, "maxPriceFilter", FILTER_VALIDATE_INT);
}
else if (isset($_GET['filterSubmit'])) && $_GET['filterSubmit'] == "eraser")
{
    $_GET['searchFilter'] = "";
    $_GET['categoriesFilter'] = 0;
    $_GET['minPriceFilter'] = "";
    $_GET['maxPriceFilter'] = "";
}

$result = "
<header>
    <nav class='navbar navbar-dark bg-dark navbar-expand-xxl py-0'>
        <div class='container-fluid'>
            <a class='navbar-brand' href='index.php'><img style='width:203px; height:130px' class='my-0' src='./assets/images/LogoGYM.png' alt='Logo'></a>
            <button class='navbar-toggler' type='button' data-bs-toggle='collapse' data-bs-target='#navbarSupportedContent' aria-controls='navbarSupportedContent' aria-expanded='false' aria-label='Toggle navigation'>
                <span class='navbar-toggler-icon'></span>
            </button>
            <div class='d-flex flex-column w-100'>
                <div class='collapse navbar-collapse my-2 w-100' id='navbarSupportedContent'>

                    if (!ESessionManager::IsValid())
                    {
                        $result .= "
                            <ul class='navbar-nav ms-auto mb-lg-0 d-flex w-100'>
                                <li class='nav-item mx-2 my-1 filterContainer'>
                                    <a class='btn btn-primary navigationBtn' aria-current='page' href='login.php'><i class='fa-solid fa-lg fa-arrow-right-to-bracket'></i> Se connecter</a>
                                </li>
                                <li class='nav-item mx-2 my-1 filterContainer'>
                                    <a class='btn btn-primary navigationBtn' href='register.php'><i class='fa-solid fa-lg fa-user-plus'></i> S'enregister</a>
                                </li>
                            </ul>";
                    }
                    else
                    if (ESessionManager::IsConnectedUserAdmin() === false)
                    {
                        $result .= "
                            <ul class='navbar-nav ms-auto mb-lg-0 d-flex w-100'>
                                <li class='nav-item mx-2 my-1 filterContainer'>
                                    <a class='btn btn-primary navigationBtn' aria-current='page' href='cart.php'><i class='fa-solid fa-lg fa-cart-shopping'></i> Mon panier</a>
                                </li>
                                <li class='nav-item mx-2 my-1 filterContainer'>
                                    <a class='btn btn-primary navigationBtn' href='logout.php'><i class='fa-solid fa-lg fa-arrow-right-from-bracket'></i> Se déconnecter</a>
                                </li>
                            </ul>";
                    }
                    else
                    {
                        $result .= "
                            <ul class='navbar-nav ms-auto mb-lg-0 d-flex w-100'>

```

```

<li class='nav-item mx-2 my-1 filterContainer'>
    <a class='btn btn-primary navigationBtn' aria-
current='page' href='addArticle.php'><i class='fa-solid fa-lg fa-plus'></i> Ajouter un
article</a>
</li>
<li class='nav-item mx-2 my-1 filterContainer'>
    <a class='btn btn-primary navigationBtn' aria-
current='page' href='addCategory.php'><i class='fa-solid fa-lg fa-plus'></i> Ajouter une
catégorie</a>
</li>
<li class='nav-item mx-2 my-1 filterContainer'>
    <a class='btn btn-primary navigationBtn' aria-
current='page' href='updateCategory.php'><i class='fa-solid fa-lg fa-pencil'></i> Modifier une
catégorie</a>
</li>
<li class='nav-item mx-2 my-1 filterContainer'>
    <a class='btn btn-primary navigationBtn'
href='logout.php'><i class='fa-solid fa-lg fa-arrow-right-from-bracket'></i> Se
déconnecter</a>
</li>
</ul>";
}

$result .= "</div>
<div class='collapse navbar-collapse border-top border-3 border-white ms-2
me-2 rounded my-1' id='navbarSupportedContent'></div>
<div class='collapse navbar-collapse my-2 w-100'
id='navbarSupportedContent'>
    <form method='get' action='search.php' class='form-inline w-100'>
        <ul class='navbar-nav me-auto mb-lg-0 w-100'>
            <li class='nav-item mx-2 d-flex flex-row filterContainer'>
                <input class='form-control search' name='searchFilter'
type='text' placeholder='Rechercher un article...' value='$searchFilter'>
            </li>
            <li class='nav-item mx-2 filterContainer'>
                <select name='categoriesFilter' class='form-select
filterControl' value='$categoriesFilter'>
                    <option value='0' selected>Catégories</option>";

                // Affiche chaque catégorie dans la combo box
                <foreach ($categories as $category)>
                {
                    $result .= "<option value='".$category->id'">;
                    <if ($categoriesFilter == $category->id) >
                        $result .= " selected";
                    </if>
                    $result .= ">$category->name</option>";
                }

                $result .= "</select>
            </li>
            <li class='nav-item mx-2 filterContainer'>
                <input class='form-control filterControl'
name='minPriceFilter' type='number' placeholder='Prix min' min='0' value=''$minPriceFilter'>
            </li>
            <li class='nav-item mx-2 filterContainer'>
                <input class='form-control filterControl'
name='maxPriceFilter' type='number' placeholder='Prix max' min='0' value=''$maxPriceFilter'>
            </li>
            <li class='nav-item mx-2 filterControl filterContainer'>
                <button name='filterSubmit' class='btn btn-primary
filterBtn' value='submit' type='submit'><i class='fa-solid fa-magnifying-glass fa-lg'></i>
Rechercher</button>
            </li>
        </foreach>
    </ul>
</form>
</div>

```

```

        <li class='nav-item mx-2 filterControl filterContainer'>
            <button name='filterSubmit' class='btn btn-primary
filterBtn' id='eraserFilterBtn' value='eraser' type='submit'><i class='fa-solid fa-eraser fa-
lg'></i> Effacer les filtres</button>
        </li>
    </ul>
</form>
</div>
</div>
</nav>
</header>";

echo $result;
}
?>

```

**src/includes/web.inc.all.php**

```

<?php

// Le fichier de gestion des sessions
require_once ROOT.'session/SessionManager.php';

// Navbar
require_once ROOT.'includes/nav.php';

// phpmailer
require_once ROOT.'includes/mail.php';

// Les fichiers concernant les appels à la base de données
require_once ROOT.'model/user.php';
require_once ROOT.'model/article.php';
require_once ROOT.'model/category.php';
require_once ROOT.'model/image.php';
require_once ROOT.'model/cartItem.php';
require_once ROOT.'model/city.php';

```

**src/model/article.php**

```

<?php
require_once ROOT.'db/database.php';
require_once ROOT.'containers/Article.php';

/**
 * Ajoute un nouvel article dans la base de données
 * @param string $name Le nom de l'article
 * @param string $description La description de l'article
 * @param double $price Le prix de l'article
 * @param int $stock Le stock disponible pour l'article
 * @param bool $featured Indique si l'article est à la une ou non
 * @param int $categoryId L'ID de la catégorie à laquelle l'article appartient
 * @return bool Renvoie true si l'opération a réussi, false sinon
 */
function AddArticle($name, $description, $price, $stock, $featured, $categoryId)
{

```

```

// Requête SQL pour insérer un article dans la base de données
$sql = "INSERT INTO `ARTICLES`(`NAME`, `DESCRIPTION`, `PRICE`, `STOCK`, `FEATURED`,
`CATEGORIES_ID`)
VALUES(:name, :description, :price, :stock, :featured, :categoryId)";

// Prépare la requête SQL
$statement = EDatabase::prepare($sql);

try
{
    // Exécute la requête SQL avec les paramètres nécessaires
    $statement->execute(array(
        ":name" => $name,
        ":description" => $description,
        ":price" => $price,
        ":stock" => $stock,
        ":featured" => $featured,
        ":categoryId" => $categoryId
    ));
}
catch (PDOException $e)
{
    // En cas d'erreur, retourne false
    return false;
}

// Retourne true si tout s'est bien passé
return true;
}

/**
 * Récupère un article grâce à son nom donné en paramètre
 * @param string $name le nom de l'article
 * @return mixed L'article récupéré sous forme d'objet Article, ou false si une erreur est
survenue
 */
function GetArticle($name)
{
    // Requête SQL qui récupère les données de l'article
    $sql = "SELECT `ID`, `NAME`, `DESCRIPTION`, `PRICE`, `STOCK`, `FEATURED`, `CREATION_DATE`,
`UPDATE_DATE`, `CATEGORIES_ID`
FROM `ARTICLES`
WHERE `NAME` = :name";

    // Prépare la requête SQL
    $statement = EDatabase::prepare($sql);

    try
    {
        // Exécute la requête SQL en utilisant l'ID passé en paramètre
        $statement->execute(array(":name" => $name));

        // Récupère la première ligne de résultat
        $row = $statement->fetch(PDO::FETCH_ASSOC,PDO::FETCH_ORI_NEXT);

        // Crée un objet Article à partir des données récupérées
        return new Article(
            intval($row['ID']), // ID : l'identifiant unique de
l'article (entier)
            $row['NAME'], // NAME : le nom de l'article (chaîne
de caractères)
    }
}
```

```

        $row['DESCRIPTION'],
        (chaîne de caractères)
            doubleval($row['PRICE']),
            (nombre à virgule flottante)
                intval($row['STOCK']),
            l'article (entier)
                    intval($row['FEATURED']),
            l'article est à la une ou non (booléen)
                        $row['CREATION_DATE'],
            l'article (objet DateTime)
                            $row['UPDATE_DATE'],
            l'article (objet DateTime)
                                intval($row['CATEGORIES_ID'])
laquelle appartient l'article (entier)
);
}

catch (PDOException $e)
{
    // En cas d'erreur, retourne false
    return false;
}

/**
 * Récupère un article grâce à son ID donné en paramètre
 * @param int $id L'identifiant unique de l'article
 * @return mixed L'article récupéré sous forme d'objet Article, ou false si une erreur est
survenue
*/
function GetArticleById($id)
{
    // Requête SQL qui récupère les données de l'article
    $sql = "SELECT `ID`, `NAME`, `DESCRIPTION`, `PRICE`, `STOCK`, `FEATURED`, `CREATION_DATE`,
`UPDATE_DATE`, `CATEGORIES_ID`
FROM `ARTICLES`
WHERE `ID` = :id";

    // Prépare la requête SQL
    $statement = EDatabase:::prepare($sql);

    try
    {
        // Exécute la requête SQL en utilisant l'ID passé en paramètre
        $statement->execute(array(":id" => $id));

        // Récupère la première ligne de résultat
        $row = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT);

        // Crée un objet Article à partir des données récupérées
        return new Article(
            intval($row['ID']), // ID : l'identifiant unique de
l'article (entier)
            $row['NAME'], // NAME : le nom de l'article (chaîne
de caractères)
            $row['DESCRIPTION'], // DESCRIPTION : la description de l'article
(chaîne de caractères)
            doubleval($row['PRICE']), // PRICE : le prix de l'article
            (nombre à virgule flottante)
                intval($row['STOCK']), // STOCK : la quantité en stock de
            l'article (entier)
                    intval($row['FEATURED']), // FEATURED : un booléen qui indique si
            l'article est à la une ou non (entier converti en booléen)
                        new DateTime($row['CREATION_DATE']), // CREATION_DATE : la date de création de
            l'article (objet DateTime)
        );
    }
}

```

## &lt;h2&gt;Selected files&lt;/h2&gt;

// DESCRIPTION	: la description de l'article
// PRICE	: le prix de l'article
// STOCK	: la quantité en stock de
// FEATURED	: un booléen qui indique si
// CREATION_DATE	: la date de création de
// UPDATE_DATE	: la date de mise à jour de
// CATEGORIES_ID	: l'ID de la catégorie à

```

        new DateTime($row['UPDATE_DATE']), // UPDATE_DATE : la date de mise à jour de
l'article (objet DateTime)
        intval($row['CATEGORIES_ID']) // CATEGORIES_ID : l'ID de la catégorie à
laquelle appartient l'article (entier)
    );
}
catch (PDOException $e)
{
    // En cas d'erreur, retourne false
    return false;
}

}

/***
 * Cette fonction récupère les 10 articles mis à la une les plus récent
 * dans la base de données et retourne un tableau d'objets Article.
 *
 * @return array|false Un tableau d'objets Article ou false si une erreur est survenue.
 */
function GetFeaturedArticles()
{
    // Initialise un tableau d'articles vide
    $arrayArticles = array();

    // Requête SQL qui récupère les articles à la une
    $sql = "SELECT `ID`, `NAME`, `DESCRIPTION`, `PRICE`, `STOCK`, `FEATURED`, `CREATION_DATE`,
`UPDATE_DATE`, `CATEGORIES_ID`
FROM `ARTICLES`
WHERE `FEATURED` = 1
ORDER BY `CREATION_DATE` DESC LIMIT 10";

    // Prépare la requête SQL
    $statement = EDatabase::prepare($sql);

    try
    {
        // Exécute la requête SQL
        $statement->execute();

        // Parcourt tous les résultats de la requête pour créer un tableau d'objets Article
        while ($row = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT))
        {
            // Crée un objet Article avec les données récupérées
            $article = new Article(
                intval($row['ID']), // ID : Identifiant de l'article
                (entier) $row['NAME'], // NAME : Nom de l'article (chaîne de
                caractères) $row['DESCRIPTION'], // DESCRIPTION : Description de l'article
                (chaîne de caractères) doubleval($row['PRICE']), // PRICE : Prix de l'article (nombre à
                virgule flottante) intval($row['STOCK']), // STOCK : Stock disponible de
                l'article (entier) boolval($row['FEATURED']), // FEATURED : Indique si l'article est à
                la une (booléen) $row['CREATION_DATE'], // CREATION_DATE: Date de création de
                l'article (chaîne de caractères) $row['UPDATE_DATE'], // UPDATE_DATE : Date de mise à jour de
                l'article (chaîne de caractères) intval($row['CATEGORIES_ID']) // CATEGORIES_ID: Identifiant de la catégorie
                associée à l'article (entier)
            );
        }
    }
}

```

```

        // Ajoute l'objet Article au tableau
        array_push($arrayArticles, $article);
    }

    // Retourne le tableau d'objets Article
    return $arrayArticles;
}

catch(PDOException $e)
{
    // En cas d'erreur, retourne false
    return false;
}

}

/***
 * Récupère les articles en fonction des filtres spécifiés.
 *
 * @param string|null $text Chaîne de caractères à rechercher dans les noms et descriptions
des articles. Null si aucun filtre n'est appliqué.
 * @param int|null $categoryId Identifiant de la catégorie à laquelle appartiennent les
articles. Null si aucun filtre n'est appliqué.
 * @param float|null $minPrice Prix minimum des articles. Null si aucun filtre n'est appliqué.
 * @param float|null $maxPrice Prix maximum des articles. Null si aucun filtre n'est appliqué.
 * @return array|false Tableau d'objets Article filtrés ou false en cas d'erreur
*/
function GetFilteredArticles($text, $categoryId, $minPrice, $maxPrice)
{
    // Initialise un tableau d'articles vide
    $arrayArticles = array();

    // Requête SQL qui récupère les données de l'article avec les filtres appliqués
    $sql = "SELECT `ID`, `NAME`, `DESCRIPTION`, `PRICE`, `STOCK`, `FEATURED`, `CREATION_DATE`,
`UPDATE_DATE`, `CATEGORIES_ID` FROM `ARTICLES` WHERE 1=1";

    // Initialise un tableau de paramètres vide
    $params = array();

    // Si la variable $text n'est pas vide, ajoute une condition à la requête SQL et un
paramètre dans le tableau des paramètres à envoyer à la requête
    if (!empty($text))
    {
        $sql .= " AND (`NAME` LIKE :t OR `DESCRIPTION` LIKE :t)";
        $queryParam = "%" . $text . "%";

        $params[":t"] = $queryParam;
    }

    // Si la variable $categoryId est supérieure à 0, ajoute une condition à la requête SQL et
un paramètre dans le tableau des paramètres à envoyer à la requête
    if ($categoryId > 0)
    {
        $sql .= " AND `CATEGORIES_ID` = :category_id";
        $params[":category_id"] = $categoryId;
    }

    // Si la variable $minPrice est supérieure à 0, ajoute une condition à la requête SQL et
un paramètre dans le tableau des paramètres à envoyer à la requête
    if ($minPrice > 0)
    {
        $sql .= " AND `PRICE` >= :min_price";
    }
}

```

```

$params[":min_price"] = $minPrice;
}

// Si la variable $maxPrice est supérieure à 0, ajoute une condition à la requête SQL et
un paramètre dans le tableau des paramètres à envoyer à la requête
if ($maxPrice > 0)
{
    $sql .= " AND `PRICE` <= :max_price";
    $params[":max_price"] = $maxPrice;
}

// Prépare la requête SQL
$statement = EDatabase::prepare($sql);

try
{
    // Exécute la requête SQL en envoyant le tableau des paramètres
    $statement->execute($params);

    // Parcourt tous les résultats de la requête pour créer un tableau d'objets Article
    while ($row = $statement->fetch(PDO::FETCH_ASSOC,PDO::FETCH_ORI_NEXT))
    {
        // Crée un objet Article avec les données récupérées
        $article = new Article(
            intval($row['ID']),           // ID          : Identifiant de l'article
(entier)            $row['NAME'],           // NAME        : Nom de l'article (chaîne de
caractères)        $row['DESCRIPTION'], // DESCRIPTION : Description de l'article
(chaîne de caractères)    doubleval($row['PRICE']), // PRICE       : Prix de l'article (nombre à
virgule flottante)      intval($row['STOCK']), // STOCK       : Stock disponible de
l'article (entier)        boolval($row['FEATURED']), // FEATURED    : Indique si l'article est à
la une (booléen)        $row['CREATION_DATE'], // CREATION_DATE: Date de création de
l'article (chaîne de caractères)    $row['UPDATE_DATE'], // UPDATE_DATE : Date de mise à jour de
l'article (chaîne de caractères)    intval($row['CATEGORIES_ID']) // CATEGORIES_ID: Identifiant de la catégorie
associée à l'article (entier)
        );
        // Ajoute l'objet Article au tableau
        array_push($arrayArticles, $article);
    }
    // Retourne le tableau d'objets Article
    return $arrayArticles;
}
catch(PDOException $e)
{
    // En cas d'erreur, retourne false
    return false;
}
}

/**
 * Vérifie si un article avec un nom donné existe dans la table `ARTICLES`.
 *
 * @param string $name Le nom de l'article à rechercher.
 * @return bool True si l'article existe, False sinon.
 */

```

```

function ArticleExists($name)
{
    $sql = "SELECT `ID` FROM `ARTICLES` WHERE `NAME` = :n";
    $statement = EDatabase::prepare($sql);

    try
    {
        $statement->execute(array(":n" => $name));
        return boolval($statement->rowCount() > 0); // retourne true si l'article existe sinon
    }
    catch(PDOException $e)
    {
        return false;
    }
}

////
////
/////
/////
////

function UpdateArticle($articleId, $name, $description, $price, $stock, $featured,
$category)
{
    // Requête SQL qui met à jour l'article
    $sql = "UPDATE `ARTICLES`
            SET `NAME` = :name,
                `DESCRIPTION` = :description,
                `PRICE` = :price,
                `STOCK` = :stock,
                `FEATURED` = :featured,
                `CATEGORIES_ID` = :category
            WHERE `ID` = :articleId";

    // Prépare la requête SQL
    $statement = EDatabase::prepare($sql);

    try
    {
        // Exécute la requête SQL pour mettre à jour l'article dans la base de données
        $statement->execute(array(
            ":name" => $name,
            ":description" => $description,
            ":price" => $price,
            ":stock" => $stock,
            ":featured" => $featured,
            ":category" => $category,
            ":articleId" => $articleId
        ));
    }
    catch (PDOException $e)
    {
        // En cas d'erreur, retourne false
        return false;
    }

    // Retourne true si la mise à jour a été effectuée avec succès
    return true;
}

```

```

}

/** 
 * Diminue le stock d'un article.
 *
 * @param int $id Identifiant de l'article.
 * @param int $stock Quantité à déduire du stock.
 * @return bool Retourne true si la mise à jour a été effectuée avec succès, sinon false.
 */
function DecreaseStock($id, $stock)
{
    // Requête SQL qui met à jour la quantité de l'article
    $sql = "UPDATE ARTICLES
        SET STOCK = STOCK - :stock
        WHERE ID = :id";

    // Prépare la requête SQL
    $statement = EDatabase::prepare($sql);

    try
    {
        // Exécute la requête SQL pour mettre à jour la quantité de l'article dans le panier
        $statement->execute(array(':id' => $id, ':stock' => $stock));
    }
    catch (PDOException $e)
    {
        // En cas d'erreur, retourne false
        return false;
    }

    // Retourne true si la mise à jour a été effectuée avec succès
    return true;
}

```

**src/model/cartItem.php**

```

<?php
require_once ROOT. 'db/database.php';
require_once ROOT. 'containers/CartItem.php';

/** 
 * Ajoute un article au panier d'un utilisateur ou augmente sa quantité de 1 s'il existe déjà.
 *
 * @param int $userId L'ID de l'utilisateur.
 * @param int $articleId L'ID de l'article.
 * @return bool Retourne true si l'article a été ajouté ou mis à jour avec succès, sinon false.
 */
function AddCartItem($userId, $articleId)
{
    // Si l'article n'existe pas dans le panier on l'ajoute sinon on augmente sa quantité de 1
    if(!CartItemExists($userId, $articleId))
    {
        // Requête SQL qui insert un article dans le panier
        $sql = "INSERT INTO `CART_ITEMS` (`USERS_ID`, `ARTICLES_ID`, `QUANTITY`) VALUES
        (:userId, :articleId, 1)";
    }
    else

```

```

{
    // Requête SQL qui augmente la quantité de 1
    $sql = "UPDATE CART_ITEMS
    SET QUANTITY = QUANTITY + 1
    WHERE USERS_ID = :userId
    AND ARTICLES_ID = :articleId";
}

// Prépare la requête SQL
$statement = EDatabase::prepare($sql);

try
{
    // Exécute la requête SQL pour ajouter un article dans le panier
    $statement->execute(array(':userId' => $userId, ':articleId' => $articleId));
}
catch (PDOException $e)
{
    // En cas d'erreur, retourne false
    return false;
}

// Retourne true si la création a été effectuée avec succès
return true;
}

/**
 * Récupère les articles présents dans le panier d'un utilisateur.
 *
 * @param int $userId L'ID de l'utilisateur
 * @return array|false Le tableau d'objets CartItem représentant les articles du panier, ou
 * false en cas d'erreur.
 */
function GetCartItems($userId)
{
    // Initialise le tableau d'articles du panier
    $cartItemArray = array();

    // Requête SQL pour récupérer les données des articles du panier
    $sql = "SELECT `USERS_ID`, `ARTICLES_ID`, `QUANTITY`
    FROM `CART_ITEMS`
    WHERE `USERS_ID` = :userId";

    // Prépare la requête SQL
    $statement = EDatabase::prepare($sql);

    try {
        // Exécute la requête SQL pour récupérer les articles du panier
        $statement->execute(array(':userId' => $userId));

        // Parcourt tous les résultats de la requête pour créer un tableau d'objets CartItem
        while ($row = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT))
        {
            // Crée un objet CartItem à partir des données récupérées
            $cartItem = new CartItem(
                intval($row['USERS_ID']),           // USERS_ID : l'identifiant unique de
                l'utilisateur (entier)              // ARTICLES_ID : l'identifiant unique de
                intval($row['ARTICLES_ID']),       // QUANTITY : la quantité de l'article dans
                l'article (entier)                 // le panier (entier)
                intval($row['QUANTITY']));
        }
    }
}

```

```

    );
    // Ajoute l'objet CartItem créé au tableau d'articles du panier
    array_push($cartItemArray, $cartItem);
}
} catch (PDOException $e) {
    // En cas d'erreur, retourne false
    return false;
}

// Retourne le tableau d'objets CartItem représentant les articles du panier
return $cartItemArray;
}

/**
 * Met à jour la quantité d'un article dans le panier d'un utilisateur.
 *
 * @param int $userId L'ID de l'utilisateur.
 * @param int $articleId L'ID de l'article.
 * @param int $quantity La nouvelle quantité de l'article.
 * @return bool Retourne true si la mise à jour a été effectuée avec succès, sinon false.
 */
function UpdateQuantity($userId, $articleId, $quantity)
{
    // Requête SQL qui met à jour la quantité de l'article
    $sql = "UPDATE CART_ITEMS
        SET QUANTITY = :quantity
        WHERE USERS_ID = :userId
        AND ARTICLES_ID = :articleId";

    // Prépare la requête SQL
    $statement = EDatabase:::prepare($sql);

    try
    {
        // Exécute la requête SQL pour mettre à jour la quantité de l'article dans le panier
        $statement->execute(array(':userId' => $userId, ':articleId' => $articleId,
        ':quantity' => $quantity));
    }
    catch (PDOException $e)
    {
        // En cas d'erreur, retourne false
        return false;
    }

    // Retourne true si la mise à jour a été effectuée avec succès
    return true;
}

/**
 * Supprime un article du panier en fonction de son ID.
 *
 * @param int $userId L'ID de l'utilisateur.
 * @param int $articleId L'ID de l'article à supprimer.
 * @return bool Retourne true si la suppression a été effectuée avec succès, sinon false.
 */
function DeleteCartItem($userId, $articleId)
{
    // Requête SQL qui supprime un article du panier en fonction de son ID
}

```

```

$sql = "DELETE FROM `CART_ITEMS` WHERE `ARTICLES_ID` = :articleId AND `USERS_ID` = :userId";

// Prépare la requête SQL
$statement = EDatabase::prepare($sql);

try
{
    // Exécute la requête en passant l'ID de l'article en paramètre
    $statement->execute(array(":articleId" => $articleId, ":userId" => $userId));

    // Retourne true pour indiquer que la suppression a réussi
    return true;
}
catch (PDOException $e)
{
    // Retourne false pour indiquer que la suppression a échoué
    return false;
}

/**
 * Vérifie si un article est déjà présent dans le panier d'un utilisateur.
 *
 * @param int $userId L'ID de l'utilisateur
 * @param int $articleId L'ID de l'article
 * @return bool True si l'article est présent dans le panier, sinon False.
 */
function CartItemExists($userId, $articleId)
{
    // Requête SQL pour vérifier si l'article est présent dans le panier
    $sql = "SELECT `USERS_ID`, `ARTICLES_ID` FROM CART_ITEMS WHERE `USERS_ID` = :userId AND `ARTICLES_ID` = :articleId";

    // Prépare la requête SQL
    $statement = EDatabase::prepare($sql);

    try
    {
        // Exécute la requête avec l'ID utilisateur et l'ID article fournis
        $statement->execute(array(":userId" => $userId, ":articleId" => $articleId));

        // Vérifie s'il y a des lignes correspondantes
        return boolval($statement->rowCount() > 0);
    }
    catch (PDOException $e)
    {
        // En cas d'erreur, retourne False
        return false;
    }
}
}

```

## src/model/category.php

```

<?php
require_once ROOT. 'db/database.php';
require_once ROOT. 'containers/Category.php';

```

```

/**
 * Récupère la liste des catégories sous forme d'objet Category
 * @return array|false Le tableau d'objets Category représentant les catégories, ou false en
cas d'erreur
 */
function GetCategories()
{
    // Initialise le tableau de catégories
    $arrayCategory = array();

    // Requête SQL qui récupère les données des catégories
    $sql = "SELECT `ID`, `NAME` FROM `CATEGORIES`";

    // Prépare la requête SQL
    $statement = EDatabase::prepare($sql);

    try
    {
        // Exécute la requête SQL pour récupérer les catégories
        $statement->execute();

        // Parcourt tous les résultats de la requête pour créer un tableau d'objets Category
        while ($row = $statement->fetch(PDO::FETCH_ASSOC,PDO::FETCH_ORI_NEXT))
        {
            // Crée un objet Category à partir des données récupérées
            $category = new Category(
                intval($row['ID']),           // ID          : l'identifiant unique de la
catégorie (entier)
                $row['NAME']                 // NAME       : le nom de la catégorie (chaîne de
caractères)
            );

            // Ajoute l'objet Category créé au tableau de catégories
            array_push($arrayCategory, $category);
        }
    }
    catch (PDOException $e)
    {
        // En cas d'erreur, retourne false
        return false;
    }

    // Retourne le tableau d'objets Category représentant les catégories
    return $arrayCategory;
}

/**
 * Récupère le nom de la catégorie correspondant à l'ID spécifié
 * @param int $id L'ID de la catégorie à rechercher
 * @return Category|false L'objet Category correspondant à l'ID spécifié, ou false en cas
d'erreur
 */
function GetCategory($id)
{
    // Requête SQL qui récupère le nom de la catégorie correspondant à l'ID spécifié
    $sql = "SELECT `ID`, `NAME` FROM `CATEGORIES` WHERE `ID` = :id";

    // Prépare la requête SQL
    $statement = EDatabase::prepare($sql);

    try

```

```

{
    // Exécute la requête SQL pour récupérer le nom de la catégorie
    $statement->execute(array(":id" => $id));

    // Récupère la première ligne de résultat
    $row = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT);

    // Crée un objet Category à partir des données récupérées
    return new Category(
        intval($row['ID']),           // ID          : l'identifiant unique de la catégorie
        $row['NAME']                  // NAME       : le nom de la catégorie (chaîne de
        caractères)                 );
}

catch (PDOException $e)
{
    // En cas d'erreur, retourne false
    return false;
}
}

/***
 * Modifie le nom d'une catégorie dans la table `CATEGORIES`
 * @param int $id L'ID de la catégorie à modifier
 * @param string $name Le nouveau nom de la catégorie
 * @return bool True si la mise à jour a été effectuée avec succès, False sinon
 */
function UpdateCategory($id, $name)
{
    // Requête SQL qui modifie le nom d'une catégorie
    $sql = "UPDATE `CATEGORIES` SET `NAME` = :n WHERE `ID` = :id";

    // Prépare la requête SQL
    $statement = EDatabase::prepare($sql);

    try
    {
        // Exécute la requête SQL pour modifier le nom de la catégorie
        $statement->execute(array(':n' => $name, ':id' => $id));
    }
    catch (PDOException $e)
    {
        // En cas d'erreur, retourne false
        return false;
    }

    // Retourne true si la mise à jour a été effectuée avec succès
    return true;
}

/***
 * Vérifie si une catégorie avec un nom donné existe dans la table `CATEGORIES`.
 *
 * @param string $name Le nom de la catégorie à rechercher.
 * @return bool True si la catégorie existe, False sinon.
 */
function CategoryExists($name)
{
    // Requête SQL pour vérifier si la catégorie existe
}

```

```

$sql = "SELECT `ID` FROM `CATEGORIES` WHERE `NAME` = :n";

// Prépare la requête SQL
$statement = EDatabase::prepare($sql);

try
{
    // Exécute la requête SQL avec le nom de la catégorie en paramètre
    $statement->execute(array(":n" => $name));

    // Retourne true si la catégorie existe (si la requête renvoie au moins une ligne),
false sinon
    return boolval($statement->rowCount() > 0);
}
catch(PDOException $e)
{
    // En cas d'erreur, retourne false
    return false;
}

/**
 * Crée une nouvelle catégorie dans la table `CATEGORIES`
 * @param string $name Le nom de la nouvelle catégorie
 * @return bool True si la création a été effectuée avec succès, False sinon
 */
function AddCategory($name)
{
    // Requête SQL qui insère une nouvelle catégorie dans la table `CATEGORIES`
    $sql = "INSERT INTO `CATEGORIES` (`NAME`) VALUES (:n)";

    // Prépare la requête SQL
    $statement = EDatabase::prepare($sql);

    try
    {
        // Exécute la requête SQL pour créer la nouvelle catégorie
        $statement->execute(array(':n' => $name));
    }
    catch (PDOException $e)
    {
        // En cas d'erreur, retourne false
        return false;
    }

    // Retourne true si la création a été effectuée avec succès
    return true;
}

```

## src/model/city.php

```

<?php
require_once ROOT. 'db/database.php';
require_once ROOT. 'containers/City.php';

/**
 * Récupère la liste des villes sous forme d'objet City

```

```

 * @return array|false Le tableau d'objets City représentant les villes, ou false en cas
d'erreur
 */
function GetCities()
{
    // Initialise le tableau de villes
    $arrayCity = array();

    // Requête SQL qui récupère les données des villes
    $sql = "SELECT `ID`, `NAME` FROM `CITIES`";

    // Prépare la requête SQL
    $statement = EDatabase::prepare($sql);

    try
    {
        // Exécute la requête SQL pour récupérer les villes
        $statement->execute();

        // Parcourt tous les résultats de la requête pour créer un tableau d'objets City
        while ($row = $statement->fetch(PDO::FETCH_ASSOC,PDO::FETCH_ORI_NEXT))
        {
            // Crée un objet City à partir des données récupérées
            $city = new City(
                intval($row['ID']),           // ID          : l'identifiant unique de la ville
(entier)                $row['NAME']           // NAME       : le nom de la ville (chaîne de
caractères)
            );

            // Ajoute l'objet City créé au tableau de villes
            array_push($arrayCity, $city);
        }
    }
    catch (PDOException $e)
    {
        // En cas d'erreur, retourne false
        return false;
    }

    // Retourne le tableau d'objets City représentant les villes
    return $arrayCity;
}

/**
 * Récupère une ville à partir de son identifiant unique
 * dans la base de données et retourne un objet City correspondant.
 *
 * @param int $id L'identifiant unique de la ville à récupérer (entier)
 *
 * @return City|false Un objet City correspondant à la ville demandée ou false si une erreur
est survenue.
 */
function GetCity($id)
{
    // Requête SQL qui sélectionne une ville en fonction de son ID
    $sql = "SELECT `ID`, `NAME` FROM `CITIES` WHERE `ID` = :id";

    // Prépare la requête SQL
    $statement = EDatabase::prepare($sql);
}

```

```

try
{
    // Exécute la requête SQL en passant l'ID en paramètre
    $statement->execute(array(":id" => $id));

    // Récupère la première ligne de résultat
    $row = $statement->fetch(PDO::FETCH_ASSOC,PDO::FETCH_ORI_NEXT);

    // Crée un objet City avec les données récupérées
    return new City(
        intval($row['ID']),           // ID          : l'identifiant unique de la ville
(entier)
        $row['NAME']                 // NAME       : le nom de la ville (chaîne de
caractères)
    );
}
catch (PDOException $e)
{
    // En cas d'erreur, retourne false
    return false;
}
}

```

## src/model/image.php

```

<?php
require_once ROOT.'db/database.php';
require_once ROOT.'containers/Image.php';

/**
 * Insère une image encodée en base64 dans la base de données.
 *
 * @param string $content Contenu de l'image encodé en base64.
 * @param string $fileName Nom de fichier de l'image.
 * @param string $fileType Type de fichier de l'image.
 * @param bool $mainImage Indique si l'image est une image principale ou non.
 * @param int $articlesId ID de l'article auquel l'image est associée.
 * @return bool Retourne true si l'insertion a réussi, sinon retourne false.
 */
function AddEnc64Image($content, $fileName, $fileType, $mainImage, $articlesId)
{
    // Encodage de l'image en base64.
    $encoded64Content = 'data:'.$fileType.';base64, '.base64_encode($content);

    // Requête SQL pour insérer une image dans la base de données.
    $sql = "INSERT INTO `IMAGES` (`CONTENT`, `NAME`, `TYPE`, `MAIN_IMAGE`, `ARTICLES_ID`)
VALUES(:encoded64Content, :fileName, :fileType, :mainImage, :articlesId)";

    // Prépare la requête SQL.
    $statement = EDatabase::prepare($sql);

    try
{
    // Exécute la requête SQL avec les paramètres nécessaires.
    $statement->execute(array(
        ":encoded64Content" => $encoded64Content,
        ":fileName" => $fileName,
        ":fileType" => $fileType,
    ));
}

```

```

        ":mainImage" => $mainImage,
        ":articlesId" => $articlesId
    )));
}
catch (PDOException $e)
{
    // En cas d'erreur, retourne false.
    return false;
}

// Retourne true si tout s'est bien passé.
return true;
}

/**
 * Récupère toutes les images associées à un article donné.
 * @param int $articlesId L'identifiant de l'article
 * @return array|bool Un tableau d'objets Image représentant les images associées à l'article,
ou false en cas d'erreur
 */
function GetImages($articlesId)
{
    // Initialise le tableau d'images
    $arrayImage = array();

    // Requête SQL qui récupère les images d'un article
    $sql = "SELECT `ID`, `CONTENT`, `NAME`, `TYPE`, `MAIN_IMAGE`, `ARTICLES_ID` FROM `IMAGES`"
    WHERE `ARTICLES_ID` = :articlesId";

    // Prépare la requête SQL
    $statement = EDatabase::prepare($sql);

    try
    {
        // Exécute la requête SQL en utilisant l'email passé en paramètre
        $statement->execute(array(":articlesId" => $articlesId));

        // Parcourt tous les résultats de la requête pour créer un tableau d'objets Image
        while ($row = $statement->fetch(PDO::FETCH_ASSOC,PDO::FETCH_ORI_NEXT))
        {
            // Crée un objet Image à partir des données de la ligne courante
            $img = new Image(
                intval($row['ID']), // ID : Identifiant de l'image
                (entier) base64 (chaîne de caractères) $row['CONTENT'], // CONTENT : Contenu de l'image encodé en
                caractères) $row['NAME'], // NAME : Nom de l'image (chaîne de
                de caractères) $row['TYPE'], // TYPE : Type MIME de l'image (chaîne
                principale pour l'article associé (booléen) boolval($row['MAIN_IMAGE']), // MAIN_IMAGE : Indique si l'image est la
                associé (entier) intval($row['ARTICLES_ID']) // ARTICLES_ID : Identifiant de l'article
            );

            // Ajoute l'objet Image créé au tableau d'images
            array_push($arrayImage, $img);
        }
    }
    catch(PDOException $e)
    {
}

```

```

        return false;
    }
    // retourne le tableau d'images
    return $arrayImage;
}

/***
 * Récupère l'image principale de l'article (l'image qui est mise en avant)
 * @param int $articlesId L'identifiant de l'article
 * @return Image|bool une Image représentant l' image principale associée à l'article, ou
false en cas d'erreur
 */
function GetMainImage($articlesId)
{
    // Requête SQL qui récupère l' image principale d'un article
    $sql = "SELECT `ID`, `CONTENT`, `NAME`, `TYPE`, `MAIN_IMAGE`, `ARTICLES_ID` FROM `IMAGES`"
    WHERE `ARTICLES_ID` = :articlesId
    AND `MAIN_IMAGE` = 1";

    // Prépare la requête SQL
    $statement = EDatabase:::prepare($sql);

    try
    {
        $statement->execute(array(":articlesId" => $articlesId));

        // Récupère la première ligne de résultat
        $row = $statement->fetch(PDO::FETCH_ASSOC,PDO::FETCH_ORI_NEXT);

        // Créer un objet Image à partir des données récupérées
        return new Image(
            intval($row['ID']),           // ID          : Identifiant de l'image (entier)
            $row['CONTENT'],              // CONTENT    : Contenu de l'image encodé en
            base64 (chaîne de caractères)
            $row['NAME'],                // NAME       : Nom de l'image (chaîne de
            caractères)
            $row['TYPE'],                // TYPE       : Type MIME de l'image (chaîne de
            caractères)
            boolval($row['MAIN_IMAGE']), // MAIN_IMAGE : Indique si l'image est la
            principale pour l'article associé (booléen)
            intval($row['ARTICLES_ID']) // ARTICLES_ID : Identifiant de l'article associé
            (entier)
        );
    }
    catch(PDOException $e)
    {
        return false;
    }
}

/***
 * Supprime une image de la base de données.
 *
 * @param int $imageId L'identifiant de l'image à supprimer
 * @return bool true si la suppression a réussi, false sinon
 */
function DeleteImage($imageId)
{
    // Requête SQL qui supprime une image en fonction de son ID
    $sql = "DELETE FROM `IMAGES` WHERE `ID` = :imageId";
}

```

```
// Prépare la requête SQL
$statement = EDatabase::prepare($sql);

try
{
    // Exécute la requête en passant l'ID de l'image en paramètre
    $statement->execute(array(":imageId" => $imageId));

    // Retourne true pour indiquer que la suppression a réussi
    return true;
}
catch(PDOException $e)
{
    // Retourne false pour indiquer que la suppression a échoué
    return false;
}
}
```

**src/model/user.php**

```
<?php
require_once ROOT. 'db/database.php';
require_once ROOT. 'containers/User.php';
require_once ROOT. 'session/SessionManager.php';

/**
 * Insère un nouvel utilisateur dans la table USERS de la base de données.
 * @param string $name Nom de l'utilisateur.
 * @param string $surname Prénom de l'utilisateur.
 * @param string $email Adresse e-mail de l'utilisateur.
 * @param string $password Mot de passe de l'utilisateur.
 * @param string $gender Genre de l'utilisateur.
 * @param string $address1 Adresse de l'utilisateur (ligne 1).
 * @param string $address2 Adresse de l'utilisateur (ligne 2).
 * @param int $city ID de la ville de l'utilisateur.
 * @param string $zipCode Code postal de l'utilisateur.
 * @return bool Retourne TRUE si l'inscription réussit, sinon FALSE.
*/
function RegisterUser($name, $surname, $email, $password, $gender, $address1, $address2,
$cityId, $zipCode)
{
    // Insère un nouvel utilisateur dans la table "USERS" de la base de données
    $sql = "INSERT INTO `USERS` (`NAME`, `SURNAME`, `EMAIL`, `PASSWORD`, `GENDER`, `ADDRESS1`,
`ADDRESS2`, `CITIES_ID`, `ZIP_CODE`)
VALUES(:name, :surname, :email, :pw, :gender, :address1, :address2, :cityID, :zipCode)";

    // Prépare la requête SQL
    $statement = EDatabase::prepare($sql);

    try
    {
        // Crée un tableau des valeurs à exécuter
        $params = array(
            ":name" => $name,
            ":surname" => $surname,
            ":email" => $email,
            ":pw" => $password,
            ":gender" => $gender,
```

```

        ":address1" => $address1,
        ":cityID" => $cityId,
        ":zipCode" => $zipCode
    );

    // Vérifie si $address2 est vide
    if (!empty($address2))
    {
        $params[":address2"] = $address2;
    }
    else
    {
        // Si $address2 est vide, assigne une valeur nulle à :address2 dans le tableau des
        valeurs
        $params[":address2"] = null;
    }

    // Exécute la requête en utilisant les valeurs passées en paramètre
    $statement->execute($params);
}

catch (PDOException $e)
{
    // Retourne false en cas d'erreur
    return false;
}

// Retourne true si tout s'est bien passé
return true;
}

/**
 * Récupère un utilisateur grâce à son email donné en paramètre
 * @param string $email L'email de l'utilisateur à récupérer
 * @return mixed L'utilisateur récupéré sous forme d'objet User, ou false si une erreur est
survenue
 */
function GetUser($email)
{
    // Requête SQL qui récupère les données de l'utilisateur
    $sql = "SELECT `USERS`.`ID`, `USERS`.`NAME`, `SURNAME`, `EMAIL`, `PASSWORD`, `GENDER`,
`ADDRESS1`, `ADDRESS2`, CITIES.NAME as CITY, `ZIP_CODE`, `IS_ADMIN`
FROM `USERS`
INNER JOIN CITIES ON USERS.CITIES_ID = CITIES.ID
WHERE EMAIL = :email";

    // Prépare la requête SQL
    $statement = EDatabase::prepare($sql);

    try
    {
        // Exécute la requête SQL en utilisant l'email passé en paramètre
        $statement->execute(array(":email" => $email));

        // Récupère la première ligne de résultat
        $row = $statement->fetch(PDO::FETCH_ASSOC,PDO::FETCH_ORI_NEXT);

        // Créer un objet User à partir des données récupérées
        return new User(
            intval($row['ID']),           // ID          : l'identifiant unique de l'utilisateur
(entier)           $row['NAME'],           // NAME        : le nom de l'utilisateur (chaîne de
caractères)
    
```

```

        $row['SURNAME'],
caractères) // SURNAME : le prénom de l'utilisateur (chaîne de
        $row['EMAIL'], // EMAIL : l'adresse email de l'utilisateur
(chaîne de caractères)
        $row['PASSWORD'], // PASSWORD : le mot de passe haché de l'utilisateur
(chaîne de caractères)
        intval($row['GENDER']), // GENDER : le genre de l'utilisateur (entier)
        $row['ADDRESS1'], // ADDRESS1 : la première adresse de l'utilisateur
(chaîne de caractères)
        $row['ADDRESS2'], // ADDRESS2 : la deuxième adresse de l'utilisateur
(optionnel) (chaîne de caractères)
        $row['CITY'], // CITY : le nom de la ville où habite
l'utilisateur (chaîne de caractères)
        intval($row['ZIP_CODE']), // ZIP_CODE : le code postal de l'utilisateur
(entier)
        boolval($row['IS_ADMIN']) // IS_ADMIN : un booléen qui indique si l'utilisateur
est un administrateur ou non (booléen)
    );
}
catch (PDOException $e)
{
    // En cas d'erreur, retourne false
    return false;
}
}

/**
 * Récupère un utilisateur à partir de son identifiant donné en paramètre.
 *
 * @param int $id L'identifiant de l'utilisateur à récupérer.
 * @return mixed L'utilisateur récupéré sous forme d'objet User, ou false si une erreur est
survenue.
 */
function GetUserById($id)
{
    // Requête SQL qui récupère les données de l'utilisateur.
    $sql = "SELECT `USERS`.`ID`, `USERS`.`NAME`, `SURNAME`, `EMAIL`, `PASSWORD`, `GENDER`,
`ADDRESS1`, `ADDRESS2`, CITIES.NAME as CITY, `ZIP_CODE`, `IS_ADMIN`
FROM `USERS`
INNER JOIN CITIES ON USERS.CITIES_ID = CITIES.ID
WHERE `USERS`.`ID` = :id";

    // Prépare la requête SQL.
    $statement = EDatabase:::prepare($sql);

    try
    {
        // Exécute la requête SQL en utilisant l'identifiant passé en paramètre.
        $statement->execute(array(":id" => $id));

        // Récupère la première ligne de résultat.
        $row = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT);

        // Crée un objet User à partir des données récupérées.
        return new User(
            intval($row['ID']), // ID : l'identifiant unique de l'utilisateur
(entier)
            $row['NAME'], // NAME : le nom de l'utilisateur (chaîne de
caractères)
            $row['SURNAME'], // SURNAME : le prénom de l'utilisateur (chaîne de
caractères)
            $row['EMAIL'], // EMAIL : l'adresse email de l'utilisateur
(chaîne de caractères)
    );
}

```

```

        $row['PASSWORD'],
        (chaîne de caractères)
            intval($row['GENDER']),
            $row['ADDRESS1'],
        (chaîne de caractères)
            $row['ADDRESS2'],
        (optionnelle) (chaîne de caractères)
            $row['CITY'],
        l'utilisateur (chaîne de caractères)
                intval($row['ZIP_CODE']),
        (entier)
            boolval($row['IS_ADMIN'])
        est un administrateur ou non (booléen)
        );
    }
    catch (PDOException $e)
    {
        // En cas d'erreur, retourne false.
        return false;
    }
}

/**
 * Vérifie si l'email est déjà présent dans la base de données.
 *
 * @param string $email L'adresse email à rechercher.
 * @return bool True si l'email existe, False sinon.
 */
function EmailExists($email)
{
    // Requête SQL pour récupérer l'email dans la table USERS
    $sql = "SELECT `EMAIL` FROM USERS WHERE `EMAIL` = :email";

    // Prépare la requête SQL
    $statement = EDatabase:::prepare($sql);

    try
    {
        // Exécute la requête SQL avec l'email en paramètre
        $statement->execute(array(":email" => $email));

        // Vérifie le nombre de lignes de résultats pour déterminer si l'email existe
        return boolval($statement->rowCount() > 0);
    }
    catch(PDOException $e)
    {
        // En cas d'erreur, retourne False
        return false;
    }
}

/**
 * Vérifie les informations de connexion d'un utilisateur.
 * Si les informations sont valides, enregistre l'utilisateur dans une session.
 *
 * @param string $email L'adresse email de l'utilisateur.
 * @param string $password Le mot de passe de l'utilisateur.
 * @return bool Retourne true si l'utilisateur est connecté avec succès, sinon retourne false.
 */
function LoginUser($email, $password)
{

```

```

// Vérifie que l'email existe dans la base de données
if (EmailExists($email))
{
    // Récupère l'utilisateur correspondant à l'email
    $user = GetUser($email);

    // Vérifie que le mot de passe est correct en utilisant la fonction password_verify()
de PHP
    if (password_verify($password, $user->password))
    {
        // Crée une session avec l'utilisateur
        ESessionManager::SetUser($user->id, $user->isAdmin);

        // Retourne true si tout c'est bien passé
        return true;
    }
    else
    {
        // Retourne false si il y a une erreur
        return false;
    }
}
else
{
    // Retourne false si il y a une erreur
    return false;
}
}

```

## src/tests/testArticle.php

```

<?php
require_once '../model/article.php';

$articleName = "Hack squat";
$msg = "";

// Permet de vérifier si le nom de l'article existe ou non dans la base de données
if (ArticleExists($articleName) == true)
{
    $msg = "existe";
}
else
{
    $msg = "n'existe pas alors on peut créer l'article";
    AddArticle($articleName, "L'équipement parfait pour le développement du bas du corps.
Maximisez les bénéfices des squats avec Hack Squat", 6999.95, 3, 0, 1); // Créer un article
}

echo($msg);

var_dump(GetFilteredArticles("bell", 0, 0, 0));
var_dump(GetFilteredArticles("bell", 0, 10, 0));
var_dump(GetFilteredArticles("bell", 0, 10, 100));

```

```
echo "-----";
var_dump (GetArticle(1));
echo "-----";
var_dump(GetFeaturedArticles());
```

**src/tests/testCartItem.php**

```
<?php

define('ROOT', $_SERVER['DOCUMENT_ROOT']."/Flavio_Soares_Rodrigues_TPI_2023/src/");
require_once '../model/cartItem.php';

// UpdateQuantity(9, 25, 3);
// DeleteCartItem(9, 25);
```

**src/tests/testCategory.php**

```
<?php
require_once '../model/category.php';

$categoryName = "APPAREILS";
$msg = "";

var_dump(GetCategories()); // Récupère l'intégralité des catégories
var_dump(GetCategory(1)); // Récupère la catégorie correspondant à l'id donnée en paramètre
UpdateCategory(1, "APPAREILS"); // Met à jour le nom de la catégorie liée à l'id 1

// Permet de vérifier si le nom de la catégorie existe ou non dans la base de données
if (CategoryExists($categoryName) == true)
{
    $msg = "existe";
}
else
{
    $msg = "n'existe pas alors on peut créer la catégorie";
    AddCategory($categoryName);
}

echo($msg);
```

**src/tests/testCity.php**

```
<?php
require_once '../model/city.php';

var_dump(GetCity(1)); // Récupère la ville de zurich
echo "-----";
var_dump(GetCities()); // Récupère toutes les villes
```

**src/tests/testImage.php**

```
<?php
require_once '../model/image.php';

$imageURL = "../images/LogoGYM.png"; // Récupère le chemin (local)
$imageContent = file_get_contents($imageURL); // Récupère le contenu de l'image
$imageName = uniqid() . "-" . basename($imageURL); // Récupère le nom de l'image et créer un nom unique
$imageType = getimagesize($imageURL)['mime']; // récupère le type d'image

var_dump($imageContent);

// AddEnc64Image($imageContent, $imageName, $imageType, 1, 2);
// GetImages($articlesId);
// GetMainImage($articlesId);
// DeleteImage($imageId);
```

**src/tests/testsessions.php**

```
<?php
define('ROOT', $_SERVER['DOCUMENT_ROOT']."/Flavio_Soares_Rodrigues_TPI_2023/src/");

require_once ROOT.'session/SessionManager.php';

// Test si la session est valide
if (ESessionManager::IsValid() === false)
{
    echo 'Session not valid';
}

// Mettre un utilisateur dans la session
ESessionManager::SetUser(250, false);

$userId = ESessionManager::GetConnectedUserId();
if ($userId === false)
{
    echo 'pas connecté';
}
if (ESessionManager::IsConnectedUserAdmin())
{
    echo 'Utilisateur est admin';
}
else
{
    echo 'Utilisateur est pas admin';
}

ESessionManager::Clear();

$userId = ESessionManager::GetConnectedUserId();
if ($userId === false)
{
    echo 'pas connecté';
}
```

**src/tests/testUser.php**

```
<?php
require_once '../model/user.php';
require_once '../model/tools.php';

// Test la fonction RegisterUser dans le cas ou tout est correcte (email unique non vérifier ici)
// RegisterUser("Du-pont", "Marcel", "MarcelDp@hotmail.ch",
'$2y$10$fvm40sn0.vBP8Wen8xt6qeRMSEU7.kqqHyPeP8WxEggx.pleBK0zS', "homme", "Av. des Grandes-Communes 2", "", 53, 1213); // retourne : true

// Test la fonction RegisterUser dans le cas ou les information ne sont pas toute correcte avec un champ vide qui ne peut pas être vide et un autre qui est un int remplacer par un string (email unique non vérifier ici)
// RegisterUser("De la porte", "Pierrot", "PierrotDLP@hotmail.ch",
'$2y$10$fvm40sn0.vBP8Wen8xt6qeRMSEU7.kqqHyPeP8WxEggx.pleBK0zS', "homme", "", "", "je suis du text", 1213); // retourne : false

EmailExists("MarcelDp@hotmail.ch");
LoginUser("MarcelDp@hotmail.ch", "Super");
//session_destroy();
var_dump($_SESSION);
```

**src/tools/addArticleTools.php**

```
<?php
/*
 * Cette page contient les fonctions d'affichage liée à la page addArticle.php
 */

// Affiche les catégories dans le combo box
function ShowCategories($value)
{
    // Récupère toutes les catégories
    $categories = GetCategories();

    // Affiche chaque catégorie dans la combo box
    foreach ($categories as $category)
    {
        echo "<option value='".$category->id."'";
        if ($value == $category->id) {
            echo " selected";
        }
        echo ">$category->name</option>";
    }
}
```

**src/tools/articleDetailsTools.php**

```
<?php

function ShowArticleDetails($article)
{
    $result = "";
```

```

$formattedPrice = number_format($article->price, 2, '.', " ");
setDescription = html_entity_decode($article->description);

$result .= "
<div class='d-flex flex-column w-75 mx-1 my-2 p-2'>
    ".ShowImages($article)."
    <h2 class='text-center mb-4'><strong>$article->name</strong></h2>
    <p class='fs-5 text-center my-0'><strong>$formattedPrice</strong><span> CHF</span></p>
    <div class='mt-3 mx-2 text-center'>
        $description
    </div>";

if (!ESessionManager::IsValid())
{
    $result.= "<a class='btn btn-primary my-2 mx-auto detailsBtn' href='login.php'>Ajouter au panier</a>";
}
elseif(ESessionManager::IsConnectedUserAdmin() === false)
{
    $result .= "
        <div class='d-flex my-2'>
            <form method='post' class='w-100 d-flex flex-wrap justify-content-center mx-auto'>
                <button name='addToCart' type='submit' class='btn btn-primary my-2 mx-auto detailsBtn' value='addToCart'>Ajouter au panier</button>
            </form>
        </div>";
}
else
{
    $result.= "<a class='btn btn-primary my-2 mx-auto detailsBtn' href='updateArticle.php?id=$article->id'>Modifier l'article</a>";
}

$result.= "</div>";
echo $result;
}

function ShowImages($article)
{
    $images = GetImages($article->id);
    $mainImage = GetMainImage($article->id);

    $result = "<div id='carouselImages' class='carousel carousel-dark slide' data-bs-interval='false'>
        <div class='carousel-indicators'>

        // Parcours les images pour affichager la navigation dans le carrousel
        <foreach ($images as $key => $image)>
            {
                $isMainImage = $image->mainImage ? "class='active' aria-current='true'" : "";
                $result .= "<button type='button' data-bs-target='#carouselImages' style='width:60px; height:6px' data-bs-slide-to='$key' $isMainImage aria-label='Slide $key'>
                </button>";
            }

        $result .= "
        </div>
        <div class='carousel-inner'>

        // Parcours les images pour les afficher dans le carrousel

```

```

foreach ($images as $image)
{
    $isMainImage = $image->mainImage ? "active" : "";
    $result .= "
        <div class='carousel-item mb-5 $isMainImage'>
            <img class='rounded mx-auto d-block detailsImg' src='$image->content'
alt='$image->name'>
        </div>";
}

$result .="
</div>
<div>
    <button class='carousel-control-prev' type='button' data-bs-
target='#carouselImages' data-bs-slide='prev'>
        <span class='carousel-control-prev-icon' aria-hidden='true'></span>
        <span class='visually-hidden'>Previous</span>
    </button>
</div>

<div>
    <button class='carousel-control-next' type='button' data-bs-
target='#carouselImages' data-bs-slide='next'>
        <span class='carousel-control-next-icon' aria-hidden='true'></span>
        <span class='visually-hidden'>Next</span>
    </button>
</div>
</div>";
return $result;
}

```

## src/tools/indexTools.php

```

<?php

function ShowFeaturedArticles()
{
    // Récupère tous les articles mis en vedette
    $featuredArticles = GetFeaturedArticles();

    $result = "";

    foreach ($featuredArticles as $article)
    {
        $mainImage = GetMainImage($article->id);
        $formattedPrice = number_format($article->price, 2, '.', " ");

        $result .= "
            <div class='mx-1 my-2 d-flex flex-column justify-content-end border border-2 p-2
filteredArticle'>
                <img class='rounded mx-auto d-block' style='width: 300px' src='$mainImage-
content' alt='$mainImage->name'>
                    <p class='fs-4 text-center'><strong>$article->name</strong></p>
                    <p class='fs-5 text-center my-0'><strong>$formattedPrice</strong> <span>CHF</span>
                </p>
                <a class='btn btn-primary mt-2' href='articleDetails.php?articleName=$article-
name'>Voir les détails...</a>
            </div>";
    }
}

```

```
echo $result;
}
```

**src/tools/registerTools.php**

```
<?php
/*
* Cette page contient les fonctions d'affichage liée à la page register.php
*/

// Affiche les villes dans le combo box
function ShowCities($value)
{
    // Récupère toutes les villes
    $cities = GetCities();

    // Affiche chaque ville dans la combo box
    foreach ($cities as $city)
    {
        echo "<option value='".$city->id."'";
        if ($value == $city->id) {
            echo " selected";
        }
        echo ">$city->name</option>";
    }
}
```

**src/tools/saveCart.php**

```
<?php

define('ROOT', $_SERVER['DOCUMENT_ROOT']."/Flavio_Soares_Rodrigues_TPI_2023/src/");
require_once ROOT.'includes/web.inc.all.php';

$data = file_get_contents("php://input");

if ($data === false)
{
    echo '{"ReturnCode": 1}';
    exit;
}

// Récupère le contenu de l'objet json
$object = json_decode($data, true);

// Affiche un message d'erreur si on ne reçoit pas de json
if ($object === null)
{
    echo '{"ReturnCode": 1}';
    exit;
}

// Récupère l'id de l'utilisateur
```

```
$userId = $object['id'];

// Parcours l'objet json
foreach($object['cart'] as $item)
{
    // Récupère l'id de l'article et sa quantité
    $articleId = $item['id'];
    $articleQuantity = $item['quantity'];

    // Met à jour la quantité
    UpdateQuantity($userId, $articleId, $articleQuantity);
}

echo '{"ReturnCode": 0}';
```

## src/addArticle.php

```
<?php
// Pour accéder à cette page l'utilisateur doit être authentifié entant qu'administrateur
$REQUIREDLOGIN = true;
$REQUIREDADMIN = true;

require_once './includes/checkAll.php';
require_once ROOT. 'tools/addArticleTools.php';

// Initialisation des variables
$name = "";
setDescription = "";
$price = 0;
$category = 0;
$stock = 0;
$featured = 0;
$msg = "";

// Si le bouton du formulaire a été cliqué
if (isset($_POST['submit']))
{
    $name = filter_input(INPUT_POST, "name", FILTER_SANITIZE_SPECIAL_CHARS);
    // Nom
    $description = filter_input(INPUT_POST, "description", FILTER_SANITIZE_SPECIAL_CHARS);
    // Description
    $price = doubleval(filter_input(INPUT_POST, "price", FILTER_VALIDATE_FLOAT));
    // Prix
    $category = intval(filter_input(INPUT_POST, "categories", FILTER_VALIDATE_INT));
    // Catégorie
    $stock = intval(filter_input(INPUT_POST, "stock", FILTER_VALIDATE_INT));
    // Stock
    $featured = isset($_POST['featured']) ? 1 : 0;
    // Mis en avant

    // Si l'article existe on affiche un message d'erreur
    if (ArticleExists($name))
    {
        // Affiche un message d'erreur
        $msg = "<p id='error'><i class='fa-solid fa-triangle-exclamation fa-xl me-2'></i> Cet article existe déjà</p>";
    }
    else
    {
```

```

$files = $_FILES['files']; // Récupère les images

// Ajoute un article
if (AddArticle($name, $description, $price, $stock, $featured, $category))
{
    $articleId = GetArticle($name)->id; // Récupère l'id de l'article qui vien d'être
créer

    // Permet de vérifier que $_FILES contient un fichier
    if (!empty($files['name'][0]))
    {
        // Parcours l'ensemble des fichiers
        for ($i = 0; $i < count($files['name']); $i++)
        {
            $fileName = uniqid(); // Nom du
fichier (unique)
            $fileContent = file_get_contents($files["tmp_name"][$i]); // Contenu de
l'image
            $fileType = $files['type'][$i]; // Type de
fichier

            // Image principale (true pour la première insertion) puis false
            $mainImage = ($i === 0);

            // Ajoute les images
            AddEnc64Image($fileContent, $fileName, $fileType, intval($mainImage),
$articleId);
        }
        $msg = "<p id='success'>L'article à été ajouté avec success</p>";
    }
}
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ajouter un article</title>

    <!-- Fontawesome -->
    <link href="assets/libraries/fontawesome/css/fontawesome.css" rel="stylesheet">
    <link href="assets/libraries/fontawesome/css/brands.css" rel="stylesheet">
    <link href="assets/libraries/fontawesome/css/solid.css" rel="stylesheet">
    <link href="assets/libraries/fontawesome/css/regular.css" rel="stylesheet">

    <!-- Bootstrap -->
    <link rel="stylesheet" href="assets/libraries/bootstrap/bootstrap.css">

    <!-- Custom CSS -->
    <link rel="stylesheet" href="assets/css/style.css">
</head>
<body>
    <?=ShowNavbar()?>
    <main class="mx-auto mt-5">
        <h2 class="mb-5">Ajouter un article</h2>
        <div class="my-3" id='errorMsg' role='alert'><?= $msg ?></div>

        <form method="post" onsubmit="return validateForm()" enctype="multipart/form-data">

```

```


<label for="name" class="form-label">Nom de l'article <i class="fa-sharp fa-solid fa-star-of-life text-primary"></i></label>
    <input type="text" name="name" id="name" class="form-control">
    <div id="nameHelp" class="form-text text-danger"></div>
</div>



<label for="description" class="form-label">Description <i class="fa-sharp fa-solid fa-star-of-life text-primary"></i></label>
    <textarea id="description" name="description"></textarea>
    <div id="descriptionHelp" class="form-text text-danger"></div>
</div>



<label for="price" class="form-label">PrixA <i class="fa-sharp fa-solid fa-star-of-life text-primary"></i></label>
    <input type="number" id="price" class="form-control" name="price" step="0.05" min="0">
    <div id="priceHelp" class="form-text text-danger"></div>
</div>



<label for="image" class="form-label">Image <i class="fa-sharp fa-solid fa-star-of-life text-primary"></i></label>
    <input type="file" id="image" name="files[]" class="form-control" accept="image/*" multiple>
    <div id="imageHelp" class="form-text text-danger"></div>
</div>



<label for="categories" class="form-label">Catégorie <i class="fa-sharp fa-solid fa-star-of-life text-primary"></i></label>
    <select id="categories" class="form-select" name="categories">
        <option value="0" selected>---</option>
        <?=$category?>
    </select>
    <div id="categoryHelp" class="form-text text-danger"></div>
</div>



<label for="stock" class="form-label">Quantité en stock <i class="fa-sharp fa-solid fa-star-of-life text-primary"></i></label>
    <input type="number" id="stock" class="form-control" name="stock" min="0">
    <div id="stockHelp" class="form-text text-danger"></div>
</div>



<label for="featured" class="form-check-label">Mis en avant <i class="fa-sharp fa-solid fa-star-of-life text-primary"></i></label>
    <input type="checkbox" id="featured" class="form-check-input" name="featured">
</div>



<input name="submit" type="submit" class="btn btn-primary submitBtn mb-3" value="Valider">
    </div>
</form>
</main>

<script src=".//assets/libraries/tinymce/tinymce.min.js"></script>
<script src=".//assets/libraries/tinymce/parametreTinymce.js"></script>


```

```

<script src=".//assets/js/validateAddArticle.js"></script>
<script src=".//assets/libraries/bootstrap/bootstrap.js"></script>

</body>
</html>

```

## src/validateOrder.php

```

<?php
// Inclusion des fichiers nécessaires
require_once './includes/checkAll.php';
require_once ROOT.'tools/indexTools.php';

// Constantes
define("HOLDER_NAME", "GYM SA");
define("RESIDENCE", "Rue de la Caille 1");
define("ZIP_CODE", "1213 Onex ");
define("COUNTRY", "Suisse");
define("IBAN", "CH35 0023 0241 9371 8929 9");

// Récupère l'utilisateur
$user = GetUserById(ESessionManager::GetConnectedUserId());

// Si il appuie sur le bouton
if (isset($_POST['validateOrder']))
{
    // Récupère les articles du panier pour l'utilisateur connecté
    $cartItems = GetCartItems(ESessionManager::GetConnectedUserId());

    $subject = "Confirmation de votre commande GYM";
    $body = "
<div style='font-size: 16px; color: black'>
    <p>Votre commande sera traitée dans les plus brefs délais.</p>
    <p style='font-size: 18px'><b>Résumé de la commande :</b></p>";
}

$totalPrice = 0;

// Parcourt chaque élément du panier
foreach ($cartItems as $cartItem)
{
    // Récupère les informations de l'article
    $article = GetArticleById($cartItem->articlesId);

    $formattedPrice = number_format($article->price, 2, '.', ' ') . ' CHF';

    // Calcul du prix total pour cet article
    $totalPriceForOneArticle = $article->price * $cartItem->quantity;

    // Formatage du prix avec 2 décimales et un séparateur de milliers
    $formattedPriceForOneArticle = number_format($totalPriceForOneArticle, 2, '.', ' ') . ' CHF';

    // Mise à jour du prix total et de la quantité totale
    $totalPrice += $totalPriceForOneArticle;

    $formattedTotalPrice = number_format($totalPrice, 2, '.', ' ') . ' CHF';

    // Construction du code HTML pour l'article du panier
}

```

```

$body .= "<p> <b>$article->name</b> - Prix : <b>$formatedPrice</b> - Quantité :  

<b>$cartItem->quantity</b> - Total : <b>$formattedPriceForOneArticle</b></p>";

// Modifie la quantité en stock
DecreaseStock($cartItem->articlesId, $cartItem->quantity);

// Supprime les articles du panier
DeleteCartItem(ESessionManager::GetConnectedUserId(), $cartItem->articlesId);
}

$body.= "<p>Total : <b>$formattedTotalPrice</b></p>";

$body .= "<p style='margin-top:20px;'>Heureux de vous avoir comme client,</p>
<p style='font-weight: bold; color: #0B5ED7; font-size: 22px;'>L'équipe GYM</p>
</div>
";

$email = $user->email;

// Envois un mail
sendEmail($email, $suject, $body);

header('location: index.php');
exit();
}

?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Accueil</title>

    <!-- Fontawesome -->
    <link href="assets/libraries/fontawesome/css/fontawesome.css" rel="stylesheet">
    <link href="assets/libraries/fontawesome/css/brands.css" rel="stylesheet">
    <link href="assets/libraries/fontawesome/css/solid.css" rel="stylesheet">
    <link href="assets/libraries/fontawesome/css/regular.css" rel="stylesheet">

    <!-- Bootstrap -->
    <link rel="stylesheet" href="assets/libraries/bootstrap/bootstrap.css">

    <!-- Custom CSS -->
    <link rel="stylesheet" href="assets/css/style.css">
</head>
<body>
    <?=ShowNavbar()?>
    <main class="w-100 mt-5">
        <h2 class="mb-4">Valider votre commande</h2>
        <div class="text-center my-5 mx-3">
            <p class="my-3 fs-5">Titulaire : <strong class="text-primary"><?=HOLDER_NAME?>
        </strong></p>
            <p class="my-3 fs-5">Domiciliation : <br>
                <?=RESIDENCE?> <br>
                <?=ZIP_CODE?><br>
                <?=COUNTRY?>
            </p>
            <p class="my-3 fs-5">IBAN : <strong><?=IBAN?></strong></p>
</body>

```

```

</div>
<div class="text-center my-5">
    <form method="post">
        <input type="submit" class="btn btn-primary" style="width: 300px;" name="validateOrder" value="Valider la commande">
    </form>
</div>
</main>
<script src=".//assets/libraries/bootstrap/bootstrap.js"></script>
</body>
</html>

```

## src/updateCategory.php

```

<?php
// Pour accéder à cette page l'utilisateur doit être authentifié entant qu'administrateur
$REQUIREDLOGIN = true;
$REQUIREDADMIN = true;

// Inclusion des fichiers nécessaires
require_once './includes/checkAll.php';

// Déclare les variables
$categoriesInput = '';
$name = "";
$msg = "";

// Si le bouton du formulaire a été cliqué
if (isset($_POST['submit']))
{
    // Filtre le nom
    $name = filter_input(INPUT_POST, "name", FILTER_SANITIZE_SPECIAL_CHARS);
    $categoriesInput = intval(filter_input(INPUT_POST, "categories",
FILTER_VALIDATE_INT));

    // Affiche une erreur si la catégorie existe sinon on crée l'article
    if (CategoryExists($name))
    {
        $msg = "<p id='error'><i class='fa-solid fa-triangle-exclamation fa-xl me-2'></i>
Cette catégorie existe déjà.</p>";
    }
    else
    {
        // Récupère la catégorie qu'on a sélectionné (utilisé pour afficher le message de
succes)
        $category = GetCategory(intval($categoriesInput));

        // Met à jour la catégorie
        UpdateCategory(intval($categoriesInput), $name);

        $msg = "<p id='success'> La catégorie <strong> $category->name </strong> à été
modifier en <strong> $name </strong>. </p>";
        $name = '';
    }
}

function ShowCategories($categoriesInput)
{
    // Récupère toutes les catégories

```

```

$categories = GetCategories();

$result = "";

// Affiche chaque catégorie dans la combo box
foreach ($categories as $category)
{
    $result .= "<option value='".$category->id."'";
    if ($categoriesInput == $category->id)
    {
        $result .= " selected";
    }
    $result .= ">$category->name</option>";
}
$result .= "</select>";
echo $result;
}

?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Modifier une catégorie</title>

    <!-- Fontawesome -->
    <link href="assets/libraries/fontawesome/css/fontawesome.css" rel="stylesheet">
    <link href="assets/libraries/fontawesome/css/brands.css" rel="stylesheet">
    <link href="assets/libraries/fontawesome/css/solid.css" rel="stylesheet">
    <link href="assets/libraries/fontawesome/css/regular.css" rel="stylesheet">

    <!-- Bootstrap -->
    <link rel="stylesheet" href="assets/libraries/bootstrap/bootstrap.css">

    <!-- Custom CSS -->
    <link rel="stylesheet" href="assets/css/style.css">
</head>
<body>
    <?=ShowNavbar()?>
    <main class="mx-auto mt-5">
        <h2 class="mb-5">Modifier une catégorie</h2>
        <div class="my-3" id='errorMsg' role='alert'><?=+$msg?></div>

        <form method="post" onsubmit="return validateForm()">

            <div class="my-4">
                <label for="categories" class="form-label">Choisissez une catégorie à modifier
                <i class="fa-sharp fa-solid fa-star-of-life text-primary"></i></label>
                <select name='categories' id="categories" class='form-select filterControl' value='$categoriesInput'>
                    <option value='0' selected>---</option>
                    <?=ShowCategories($categoriesInput)?>
                    <div id="categoriesHelp" class="form-text text-danger"></div>
                </div>

                <div class="my-4">
                    <label for="name" class="form-label">Nouveau nom de la catégorie <i class="fa-sharp fa-solid fa-star-of-life text-primary"></i></label>

```

```

<input name="name" type="name" class="form-control" id="name"
onkeyup="toUpperCase()" value="<?=$name?>">
    <div id="nameHelp" class="form-text text-danger"></div>
</div>

<div class="d-flex flex-row-reverse my-3">
    <button name="submit" type="submit" class="btn btn-primary submitBtn mb-3"
value="submit">Modifier le nom de la catégorie</button>
</div>
</form>
</main>
<script src=".//assets/libraries/bootstrap/bootstrap.js"></script>
<script src=".//assets/js/validateUpdateCategory.js"></script>
</body>
</html>

```

**src/updateArticle.php**

```

<?php
// Pour accéder à cette page l'utilisateur doit être authentifié entant qu'administrateur
$REQUIREDLOGIN = true;
$REQUIREDADMIN = true;

require_once './includes/checkAll.php';
require_once ROOT. 'tools/addArticleTools.php';

// Empêche l'utilisateur d'accéder à la page via l'url
if (!isset($_GET['id']))
{
    header("Location: login.php");
    exit();
}

$article = GetArticleById(intval($_GET['id']));
$articleId = $article->id;

//var_dump($images);

$description = $article->description;
$featuredValue = $article->featured ? 'true' : 'false';

// Initialisation des variables
/*$name = "";
$description = "";
$price = 0;
$category = 0;
$stock = 0;*/

$msg = "";

// Si le bouton du formulaire a été cliqué
if (isset($_POST['submit']) && $_POST['submit'] == "Valider")
{
    $name = filter_input(INPUT_POST, "name", FILTER_SANITIZE_SPECIAL_CHARS);
    // Nom
    $description = filter_input(INPUT_POST, "description", FILTER_SANITIZE_SPECIAL_CHARS);
    // Description
    $price = doubleval(filter_input(INPUT_POST, "price", FILTER_VALIDATE_FLOAT));
    // Prix
}

```

```

$categorie = intval(filter_input(INPUT_POST, "categories", FILTER_VALIDATE_INT));
// Catégorie
$stock = intval(filter_input(INPUT_POST, "stock", FILTER_VALIDATE_INT));
// Stock
$featured = isset($_POST['featured']) ? 1 : 0;
// Mis en avant

// Si l'article existe on affiche un message d'erreur
if (ArticleExists($name))
{
    // Affiche un message d'erreur
    $msg = "<p id='error'><i class='fa-solid fa-triangle-exclamation fa-xl me-2'></i> Cet
article existe déjà</p>";
}
else
{
    $files = $_FILES['files']; // Récupère les images

    // Ajoute un article
    if (UpdateArticle($articleId, $name, $description, $price, $stock, $featured,
$categorie))
    {
        $articleId = GetArticle($name)->id; // Récupère l'id de l'article qui vien d'être
créer

        // Permet de vérifier que $_FILES contient un fichier
        if (!empty($files['name'][0]))
        {
            // Parcours l'ensemble des fichiers
            for ($i = 0; $i < count($files['name']); $i++)
            {
                $fileName = uniqid(); // Nom du
fichier (unique)
                $fileContent = file_get_contents($files["tmp_name"][$i]); // Contenu de
l'image
                $fileType = $files['type'][$i]; // Type de
fichier

                // Image principale (true pour la première insertion) puis false
                $mainImage = ($i === 0);

                // Ajoute les images
                AddEnc64Image($fileContent, $fileName, $fileType, intval($mainImage),
$articleId);
            }
            $msg = "<p id='success'>L'article à été ajouté avec success</p>";
        }
    }
}

// Affiche les images
function ShowImages($articleId)
{
    $images = GetImages($articleId);

    $result = "";

    foreach ($images as $key => $image)
    {
        // Supprime une image lors ce que l'on clique sur le bouton supprimer
        if (isset($_POST['deleteImage']) && intval($_POST['deleteImage']) == $image->id)

```

```

    {
        DeleteImage($image->id);
    }

$result .= "
<div class='d-flex flex-row align-items-center'>
    <img src='$image->content' alt='$image->name' style='width:200px'>
    <form method='post' class='text-center'>
        <button name='deleteImage' type='submit' class='btn btn-danger mb-3 mx-4' style='width:200px; height:60px' value='$image->id'><i class='fa-solid fa-trash fa-lg'></i> Supprimer</button>
    </form>
</div>";
}

echo $result;
}

?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Modifier un article</title>

    <!-- Fontawesome -->
    <link href="assets/libraries/fontawesome/css/fontawesome.css" rel="stylesheet">
    <link href="assets/libraries/fontawesome/css/brands.css" rel="stylesheet">
    <link href="assets/libraries/fontawesome/css/solid.css" rel="stylesheet">
    <link href="assets/libraries/fontawesome/css/regular.css" rel="stylesheet">

    <!-- Bootstrap -->
    <link rel="stylesheet" href="assets/libraries/bootstrap/bootstrap.css">

    <!-- Custom CSS -->
    <link rel="stylesheet" href="assets/css/style.css">
</head>
<body>
<!--body onload="toggle(document.getElementById('featured'))"-->
<?=ShowNavbar()?>
<main class="mx-auto mt-5">
    <h2 class="mb-5">Modifier un article</h2>
    <div class="my-3" id='errorMsg' role='alert'><?= $msg ?></div>

    <form method="post" onsubmit="return validateForm()" enctype="multipart/form-data">

        <div class="my-4">
            <label for="name" class="form-label">Nom de l'article <i class="fa-sharp fa-solid fa-star-of-life text-primary"></i></label>
            <input type="text" name="name" id="name" class="form-control" value="<?
                = $article->name ?>">
            <div id="nameHelp" class="form-text text-danger"></div>
        </div>

        <div class="my-4">
            <label for="description" class="form-label">Description <i class="fa-sharp fa-solid fa-star-of-life text-primary"></i></label>
            <textarea id="description" name="description" value=""></textarea>
            <div id="descriptionHelp" class="form-text text-danger"></div>
        </div>
    </form>
</body>

```

```

        <div class="my-4">
            <label for="price" class="form-label">Prix <i class="fa-sharp fa-solid fa-star-of-life text-primary"></i></label>
            <input type="number" id="price" class="form-control" name="price" step="0.05" min="0" value="<?=$article->price?>">
            <div id="priceHelp" class="form-text text-danger"></div>
        </div>

        <div >
            <?=ShowImages($articleId)?>
        </div>

        <div class="my-4">
            <label for="image" class="form-label">Image <i class="fa-sharp fa-solid fa-star-of-life text-primary"></i></label>
            <input type="file" id="image" name="files[]" class="form-control" accept="image/*" multiple>
            <div id="imageHelp" class="form-text text-danger"></div>
        </div>

        <div class="my-4">
            <label for="categories" class="form-label">Catégorie <i class="fa-sharp fa-solid fa-star-of-life text-primary"></i></label>
            <select id="categories" class="form-select" name="categories">
                <option value="0" selected>---</option>
                <?=ShowCategories($article->categoryId)?>
            </select>
            <div id="categoryHelp" class="form-text text-danger"></div>
        </div>

        <div class="my-4">
            <label for="stock" class="form-label">Quantité en stock <i class="fa-sharp fa-solid fa-star-of-life text-primary"></i></label>
            <input type="number" id="stock" class="form-control" name="stock" min="0" value="<?=$article->stock?>">
            <div id="stockHelp" class="form-text text-danger"></div>
        </div>

        <div class="my-4">
            <label for="featured" class="form-check-label">Mise en avant <i></i></label>
            <input type="checkbox" id="featured" class="form-check-input" name="featured" value="<?=$article->featured?>" <?= $featuredValue === 'true' ? 'checked' : '' ?>>
        </div>

        <div class="my-4">
            <input name="submit" type="submit" class="btn btn-primary submitBtn mb-3" value="Valider">
        </div>
    </form>
</main>

<script src=".//assets/libraries/tinymce/tinymce.min.js"></script>
<script src=".//assets/libraries/tinymce/parametreTinymce.js"></script>
<script src=".//assets/js/validateUpdateArticle.js"></script>
<script src=".//assets/libraries/bootstrap/bootstrap.js"></script>

</body>
<script>
    let descriptionContent = "<?php echo htmlspecialchars($description); ?>"
    window.addEventListener("load", function(){
        let el = tinymce.get('description');

```

```

        if (el)
            el.setContent(descriptionContent);
    });

</script>
</html>

```

**src/search.php**

```

<?php

// Empêche l'utilisateur d'accéder à la page via l'url
if (!isset($_GET['searchFilter']) && !isset($_GET['categoriesFilter']) &&
!isset($_GET['minPriceFilter']) && !isset($_GET['maxPriceFilter']))
{
    header("Location: login.php");
    exit();
}

// Inclusion des fichiers nécessaires
require_once './includes/checkAll.php';

function ShowFilteredArticles()
{
    // Récupères les filtres
    $searchFilter = filter_var($_GET['searchFilter'], FILTER_SANITIZE_SPECIAL_CHARS);
    $categoriesFilter = filter_var($_GET['categoriesFilter'], FILTER_SANITIZE_NUMBER_INT);
    $minPriceFilter = filter_var($_GET['minPriceFilter'], FILTER_SANITIZE_NUMBER_INT);
    $maxPriceFilter = filter_var($_GET['maxPriceFilter'], FILTER_SANITIZE_NUMBER_INT);

    // Récupères les articles filtré
    $filteredArticles = GetFilteredArticles($searchFilter, intval($categoriesFilter),
    intval($minPriceFilter), intval($maxPriceFilter));

    $result = "";

    foreach ($filteredArticles as $article)
    {
        $mainImage = GetMainImage($article->id);
        $formattedPrice = number_format($article->price, 2, '.', " ");

        $result .= "
            <div class='mx-1 my-2 d-flex flex-column justify-content-end border border-2 p-2
filteredArticle'>
                <img class='rounded mx-auto d-block' style='width: 300px' src='$mainImage-
content' alt='$mainImage->name'>
                    <p class='fs-4 text-center'><strong>$article->name</strong></p>
                    <p class='fs-5 text-center my-0'><strong>$formattedPrice</strong> <span>CHF</span>
                </p>
                <a class='btn btn-primary mt-2' href='articleDetails.php?articleName=$article-
>name'>Voir les détails...</a>
            </div>";
    }
    echo $result;
}

?>

```

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Accueil</title>

    <!-- Fontawesome -->
    <link href="assets/libraries/fontawesome/css/fontawesome.css" rel="stylesheet">
    <link href="assets/libraries/fontawesome/css/brands.css" rel="stylesheet">
    <link href="assets/libraries/fontawesome/css/solid.css" rel="stylesheet">
    <link href="assets/libraries/fontawesome/css/regular.css" rel="stylesheet">

    <!-- Bootstrap -->
    <link rel="stylesheet" href="assets/libraries/bootstrap/bootstrap.css">

    <!-- Custom CSS -->
    <link rel="stylesheet" href="assets/css/style.css">
</head>
<body>
    <?=ShowNavbar()?>
    <main class="w-100 mt-3">
        <div class="d-flex flex-wrap justify-content-center mx-auto">
            <?=ShowFilteredArticles()?>
        </div>
    </main>
    <script src=".//assets/libraries/bootstrap/bootstrap.js"></script>
</body>
</html>

```

## src/register.php

```

<?php
    // Inclusion des fichiers nécessaires
    require_once './includes/checkAll.php';
    require_once ROOT.'tools/registerTools.php';

    // Vérifie si la session de l'utilisateur est valide, si oui, redirige vers la page
    d'accueil
    if (ESessionManager::IsValid() === true)
    {
        header("Location: index.php");
        exit();
    }

    $name = "";
    $surname = "";
    $gender = 0;
    $address1 = "";
    $address2 = "";
    $city = 0;
    $zipCode = "";
    $email = "";
    $password = "";

    // Variable pour le message d'erreur généré avec php

```

```

$msg = "";

// Si le bouton du formulaire a été cliqué
if (isset($_POST['submit']))
{
    $name = filter_input(INPUT_POST, "name", FILTER_SANITIZE_SPECIAL_CHARS); // Nom
    $surname = filter_input(INPUT_POST, "surname", FILTER_SANITIZE_SPECIAL_CHARS); // Prénom
    $gender = intval(filter_input(INPUT_POST, "gender", FILTER_VALIDATE_INT)); // Genre
    $address1 = filter_input(INPUT_POST, "address1", FILTER_SANITIZE_SPECIAL_CHARS); // Adresse 1
    $address2 = filter_input(INPUT_POST, "address2", FILTER_SANITIZE_SPECIAL_CHARS); // Adresse 2
    $city = intval(filter_input(INPUT_POST, "cities", FILTER_VALIDATE_INT)); // Ville
    $zipCode = filter_input(INPUT_POST, "zipCode", FILTER_SANITIZE_SPECIAL_CHARS); // Code postal
    $email = filter_input(INPUT_POST, "email", FILTER_VALIDATE_EMAIL); // Email
    $password = filter_input(INPUT_POST, "password", FILTER_SANITIZE_SPECIAL_CHARS); // Mot de passe

    if (EmailExists($email))
    {
        // Affiche un message d'erreur
        $msg = "<p id='error'><i class='fa-solid fa-triangle-exclamation fa-xl me-2'></i>
L'email est déjà utilisé.<a class='card-link text-decoration-none'
href='login.php'> Connectez-vous</a></p>";
    }
    else
    {
        // Hache le mot de passe avec un coût plus élevé
        $passwordHashed = password_hash($password, PASSWORD_BCRYPT, ['cost' => 12]);

        // Crée l'utilisateur
        RegisterUser($name, $surname, $email, $passwordHashed, $gender, $address1,
$address2, $city, $zipCode);

        // Envoi d'un mail à l'utilisateur
        $mailTitle = "Confirmation de la création de votre compte GYM";
        $mailBody =
<div style='font-size: 16px; color: black'>
    <p>Vous êtes bien enregistré sur le site GYM.</p>
    <p>Vous pouvez dès maintenant commander les articles que vous souhaitez.</p>
    <p>Heureux de vous avoir comme utilisateur,</p>
    <p style='font-weight: bold; color: #0B5ED7; font-size: 18px;'>L'équipe
GYM</p>
</div>";

        sendEmail($email, $mailTitle, $mailBody);

        $msg = "<p id='success'> Votre compte a été créé. Un email de confirmation vous a
été envoyé.
            <a class='card-link text-decoration-none' href='login.php'> Connectez-vous</a>
</p>";

        // Vide toutes les valeurs du formulaire
        $name = "";
        $surname = "";
        $gender = 0;
        $address1 = "";
    }
}

```

```

        $address2 = "";
        $city = 0;
        $zipCode = "";
        $email = "";
        $password = "";
    }
}

?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Inscription</title>

    <!-- Fontawesome -->
    <link href="assets/libraries/fontawesome/css/fontawesome.css" rel="stylesheet">
    <link href="assets/libraries/fontawesome/css/brands.css" rel="stylesheet">
    <link href="assets/libraries/fontawesome/css/solid.css" rel="stylesheet">
    <link href="assets/libraries/fontawesome/css/regular.css" rel="stylesheet">

    <!-- Bootstrap -->
    <link rel="stylesheet" href="assets/libraries/bootstrap/bootstrap.css">

    <!-- Custom CSS -->
    <link rel="stylesheet" href="assets/css/style.css">
</head>
<body>
<?=ShowNavbar()?>
<main class="mx-auto mt-5">
    <h2 class="mb-5">Inscription</h2>
    <div class="my-3" id='errorMsg' role='alert'><?=$msg?></div>

    <!-- Formulaire d'inscription -->
    <form method="post" onsubmit="return validateRegister()">

        <!-- Nom -->
        <div class="my-4">
            <label for="name" class="form-label">Nom <i class="fa-sharp fa-solid fa-star-of-life text-primary"></i></label>
            <input name="name" type="text" class="form-control" id="name" aria-describedby="nameHelp" value="<?=$name?>">
            <div id="nameHelp" class="form-text text-danger"></div>
        </div>

        <!-- Prénom -->
        <div class="my-4">
            <label for="surname" class="form-label">Prénom <i class="fa-sharp fa-solid fa-star-of-life text-primary"></i></label>
            <input name="surname" type="text" class="form-control" id="surname" aria-describedby="surnameHelp" value="<?=$surname?>">
            <div id="surnameHelp" class="form-text text-danger"></div>
        </div>

        <!-- Genre -->
        <div class="my-4">
            <label for="gender" class="form-label">Genre <i class="fa-sharp fa-solid fa-star-of-life text-primary"></i></label>

```

```

        <select name="gender" id="gender" class="form-select" aria-
describedby="genderHelp">
            <option value="0" <?php if ($gender == "0") echo 'selected'; ?>>---
</option>
            <option value="1" <?php if ($gender == "1") echo 'selected'; ?>
>Homme</option>
            <option value="2" <?php if ($gender == "2") echo 'selected'; ?>
>Femme</option>
            <option value="3" <?php if ($gender == "3") echo 'selected'; ?>
>Autre</option>
        </select>
        <div id="genderHelp" class="form-text text-danger"></div>
    </div>

    <!-- Adresse 1 -->
    <div class="my-4">
        <label for="address1" class="form-label">Adresse 1 <i class="fa-sharp fa-solid fa-star-of-life text-primary"></i></label>
        <input name="address1" type="text" class="form-control" id="address1" aria-
describedby="address1Help" value="<?=$address1?>">
        <div id="address1Help" class="form-text text-danger"></div>
    </div>

    <!-- Adresse 2 -->
    <div class="my-4">
        <label for="address2" class="form-label">Adresse 2</label>
        <input name="address2" type="text" class="form-control" id="address2" value="
<?=$address2?>">
    </div>

    <!-- Ville -->
    <div class="my-4">
        <label for="cities" class="form-label">Ville <i class="fa-sharp fa-solid fa-
star-of-life text-primary"></i></label>
        <select name="cities" id="cities" class="form-select" aria-
describedby="citiesHelp" aria-selected="<?=$cities?>">
            <option value="0" selected>---</option>
            <?=$city?>
        </select>
        <div id="citiesHelp" class="form-text text-danger"></div>
    </div>

    <!-- Code postal -->
    <div class="my-4">
        <label for="zipCode" class="form-label">Code postal <i class="fa-sharp fa-
solid fa-star-of-life text-primary"></i></label>
        <input name="zipCode" type="text" class="form-control" id="zipCode" aria-
describedby="zipCode" value="<?=$zipCode?>">
        <div id="zipCodeHelp" class="form-text text-danger"></div>
    </div>

    <!-- Email -->
    <div class="my-4">
        <label for="email" class="form-label">Email <i class="fa-sharp fa-solid fa-
star-of-life text-primary"></i></label>
        <input name="email" type="email" class="form-control" id="email" aria-
describedby="emailHelp" value="<?=$email?>">
        <div id="emailHelp" class="form-text text-danger"></div>
    </div>

    <!-- Mot de passe -->
    <div class="my-4">
        <label for="password" class="form-label">Mot de passe <i class="fa-sharp fa-

```

```

solid fa-star-of-life text-primary"></i></label>
    <input name="password" type="password" class="form-control" id="password"
aria-describedby="passwordHelp">
    <div id="passwordHelp" class="form-text text-danger"></div>
</div>

<!-- Bouton de validation --&gt;
&lt;div class="d-flex flex-row-reverse my-3"&gt;
    &lt;button name="submit" type="submit" class="btn btn-primary submitBtn mb-3"
value="register"&gt;S'inscrire&lt;/button&gt;
&lt;/div&gt;
&lt;/form&gt;

&lt;div class="text-center mt-2 mb-4 card card-body"&gt;
    &lt;h6 class="card-subtitle mb-2"&gt; Vous possédez déjà un compte ?&lt;/h6&gt;
    &lt;a class="card-link text-decoration-none" href="login.php"&gt;Connectez-vous&lt;/a&gt;
&lt;/div&gt;
&lt;/main&gt;
&lt;script src=".//assets/js/validateEmail.js"&gt;&lt;/script&gt;
&lt;script src=".//assets/js/validateRegister.js"&gt;&lt;/script&gt;
&lt;script src=".//assets/libraries/bootstrap/bootstrap.js"&gt;&lt;/script&gt;
&lt;/body&gt;
&lt;/html&gt;
</pre>

```

## src/logout.php

```

<?php
/*Page qui déconnecte l'utilisateur de la session*/

// Inclue le fichiers nécessaire
require_once './includes/checkAll.php';

// Supprime la session
ESessionManager::Clear();

// Redirection vers la page de login
header("Location: login.php");
exit();

```

## src/login.php

```

<?php
// Inclusion des fichiers nécessaires
require_once './includes/checkAll.php';

// Vérifie si la session de l'utilisateur est valide, si oui, redirige vers la page
d'accueil
if (ESessionManager::IsValid() == true)
{
    header("Location: index.php");
    exit();
}

// Filtre les valeurs des inputs
$email = filter_input(INPUT_POST, "email", FILTER_VALIDATE_EMAIL);
$password = filter_input(INPUT_POST, "password", FILTER_SANITIZE_SPECIAL_CHARS);

```

```

$submit = filter_input(INPUT_POST,"submit", FILTER_SANITIZE_SPECIAL_CHARS);

// Variable pour le message d'erreur générée avec php
$msg = "";

// Si le bouton du formulaire a été cliqué
if ($submit == "login")
{
    // Vérifie si les identifiants de connexion sont valides, si oui, redirige vers la
page d'accueil
    if (LoginUser($email, $password))
    {
        header("Location: index.php");
        exit();
    }
    else
    {
        // Affiche un message d'erreur
        $msg = "<p id='error'><i class='fa-solid fa-triangle-exclamation fa-xl me-2'></i>
Email ou mot de passe incorrecte.</p>";
    }
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login</title>

    <!-- Fontawesome -->
    <link href="assets/libraries/fontawesome/css/fontawesome.css" rel="stylesheet">
    <link href="assets/libraries/fontawesome/css/brands.css" rel="stylesheet">
    <link href="assets/libraries/fontawesome/css/solid.css" rel="stylesheet">
    <link href="assets/libraries/fontawesome/css/regular.css" rel="stylesheet">

    <!-- Bootstrap -->
    <link rel="stylesheet" href="assets/libraries/bootstrap/bootstrap.css">

    <!-- Custom CSS -->
    <link rel="stylesheet" href="assets/css/style.css">
</head>
<body>
    <?=ShowNavbar()?>
    <main class="mx-auto mt-5">
        <h2 class="mb-5">Connexion</h2>
        <div class="my-3" id='errorMsg' role='alert'><?= $msg ?></div>

        <!-- Formulaire de connexion -->
        <form method="post" onsubmit="return validateLogin()">

            <!-- Email -->
            <div class="my-4">
                <label for="email" class="form-label">Email</label>
                <input name="email" type="email" class="form-control" id="email" aria-
describedby="emailHelp" value="<?=$email?>">
                    <div id="emailHelp" class="form-text text-danger"></div>
            </div>
        </form>
    </main>
</body>

```

```

<!-- Mot de passe -->
<div class="my-4">
    <label for="password" class="form-label">Mot de passe</label>
    <input name="password" type="password" class="form-control" id="password"
aria-describedby="emailHelp">
        <div id="passwordHelp" class="form-text text-danger"></div>
    </div>

    <!-- Bouton de validation -->
    <div class="d-flex flex-row-reverse my-3">
        <button name="submit" type="submit" class="btn btn-primary submitBtn mb-3"
value="login">Se connecter</button>
    </div>
</form>
<div class="text-center mt-2 mb-4 card card-body">
    <h6 class="card-subtitle mb-2"> Vous êtes nouveau sur GYM ?</h6>
    <a class="card-link text-decoration-none" href="register.php">Inscrivez-
vous</a>
</div>
</main>
<script src=".//assets/js/validateEmail.js"></script>
<script src=".//assets/js/validateLogin.js"></script>
<script src=".//assets/libraries/bootstrap/bootstrap.js"></script>
</body>
</html>

```

## src/index.php

```

<?php
// Inclusion des fichiers nécessaires
require_once './includes/checkAll.php';
require_once ROOT.'tools/indexTools.php';

?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Accueil</title>

    <!-- Fontawesome -->
    <link href="assets/libraries/fontawesome/css/fontawesome.css" rel="stylesheet">
    <link href="assets/libraries/fontawesome/css/brands.css" rel="stylesheet">
    <link href="assets/libraries/fontawesome/css/solid.css" rel="stylesheet">
    <link href="assets/libraries/fontawesome/css/regular.css" rel="stylesheet">

    <!-- Bootstrap -->
    <link rel="stylesheet" href="assets/libraries/bootstrap/bootstrap.css">

    <!-- Custom CSS -->
    <link rel="stylesheet" href="assets/css/style.css">
</head>
<body>
    <?=ShowNavbar()?>
    <main class="w-100 mt-3">

```

```

<h2 class="mb-4">Nos articles à la une</h2>
<div class="d-flex flex-wrap justify-content-center mx-auto">
    <?=ShowFeaturedArticles()?>
</div>
</main>
<script src=".//assets/libraries/bootstrap/bootstrap.js"></script>
</body>
</html>

```

**src/cart.php**

```

<?php

// This page requires to be authenticated
$REQUIREDLOGIN = true;

require_once './includes/checkAll.php';

// Récupère l'id de l'utilisateur connecté
$userId = ESessionManager::GetConnectedUserId();

// Récupère les articles du panier pour l'utilisateur connecté
$cartItems = GetCartItems($userId);

/**
 * Affiche les articles du panier avec leurs informations.
 * @return string Le code HTML contenant les éléments du panier.
 */
function ShowCartItems($cartItems)
{
    // Variables pour le calcul du prix total et de la quantité totale
    $totalPrice = 0;
    $totalQuantity = 0;

    $result = "";

    // Parcourt chaque élément du panier
    foreach ($cartItems as $cartItem)
    {
        // Récupère les informations de l'article
        $article = GetArticleById($cartItem->articlesId);

        // Récupère l'image principale de l'article
        $mainImage = GetMainImage($cartItem->articlesId);

        $formattedPrice = number_format($article->price, 2, '.', ' ');

        // Calcul du prix total pour cet article
        $totalPriceForOneArticle = $article->price * $cartItem->quantity;

        // Formatage du prix avec 2 décimales et un séparateur de milliers
        $formattedPriceForOneArticle = number_format($totalPriceForOneArticle, 2, '.', ' ');

        // Mise à jour du prix total et de la quantité totale
        $totalPrice += $totalPriceForOneArticle;
        $totalQuantity += $cartItem->quantity;
    }
}

```

```

// Construction du code HTML pour l'article du panier
$result .= "
<div class='mx-1 d-flex flex-row align-items-end justify-content-center p-3 w-100'
      id='article$cartItem->articlesId'
      data-price='$article->price'
      <img class='rounded mx-auto d-block w-25' src='$mainImage->content'
alt='$mainImage->name'>
    <div class='w-50 mx-auto text-center'>
        <p class='fs-4 text-center'><strong>$article->name</strong></p>
        <p class='fs-5 text-center my-3'>$formattedPrice <span>CHF</span></p>
        <input type='number' class='form-control my-3 text-center mx-auto'
style='width:100px' id='articleQuantity$cartItem->articlesId' data-Id='$cartItem->articlesId'
name='quantity' min='1' max='$article->stock' step='1' value='$cartItem->quantity'>
    </div>
    <div class='d-flex flex-row flex-wrap text-center mx-auto w-25'>
        <p class='fs-5 text-center w-100'>Total : <strong><span
id='articleTotal$cartItem->articlesId'>$formattedPriceForOneArticle</span></strong></p>
        <form method='post' class='text-center w-100'>
            <button name='deleteCartItem' type='submit' class='btn btn-danger mb-3 w-
100' value='$cartItem->articlesId'><i class='fa-solid fa-trash fa-lg'></i> Supprimer</button>
        </form>
    </div>
</div>
<div class='border border-2 rounded w-100 my-2'></div>";

// Enlève l'objet du panier si on clique sur le bouton supprimer
if (isset($_POST['deleteCartItem']) && intval($_POST['deleteCartItem']) == $cartItem->articlesId)
{
    DeleteCartItem(ESessionManager::GetConnectedUserId(), $cartItem->articlesId);
}

// Formatage du prix avec 2 décimales et un séparateur de milliers
$formattedTotalPrice = number_format($totalPrice, 2, '.', " ").' CHF';
// Construction du code HTML pour le total du panier
if($cartItems != null)
{
    $result.= "
<div class='d-flex flex-column justify-content-center w-100'>
    <p class='fs-5 text-center'>Total <strong>".count($cartItems)."
</strong>article(s) : <strong><span id='totalPrice' data-
totalPrice='\$totalPrice'>$formattedTotalPrice</span></strong></p>
    <form method='get' action='validateOrder.php' class='text-center w-100'>
        <button name='order' id='order' type='submit' class='btn btn-primary submitBtn
mb-3 my-2 w-50 mx-auto' value='order'><i class='fa-solid fa-lg fa-truck'></i>
Commander</button>
    </form>
    <button name='save' type='submit' class='btn btn-success submitBtn mb-5 mx-
auto' id='save'><i class='fa-solid fa-lg fa-floppy-disk'></i> Sauvegarder</button>
</div>";
}
else
{
    $result .= "<div class='fs-5 alert alert-danger mt-4 w-100 text-center'
role='alert'>Votre panier est vide</div>";
}

// Retourne le code HTML contenant les éléments du panier
return $result;
}

?>

```

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Accueil</title>

    <!-- Fontawesome -->
    <link href="assets/libraries/fontawesome/css/fontawesome.css" rel="stylesheet">
    <link href="assets/libraries/fontawesome/css/brands.css" rel="stylesheet">
    <link href="assets/libraries/fontawesome/css/solid.css" rel="stylesheet">
    <link href="assets/libraries/fontawesome/css/regular.css" rel="stylesheet">

    <!-- Bootstrap -->
    <link rel="stylesheet" href="assets/libraries/bootstrap/bootstrap.css">

    <!-- Custom CSS -->
    <link rel="stylesheet" href="assets/css/style.css">
</head>
<body>
    <?=ShowNavbar()?>
    <main class="mt-5 mx-auto">
        <h2>Mon panier</h2>
        <div class="d-flex flex-wrap justify-content-center mx-auto">
            <?=ShowCartItems($cartItems)?>
        </div>
        <script src=".//assets/libraries/bootstrap/bootstrap.js"></script>
</body>
<script>
    var id = <?php echo ESessionManager::GetConnectedUserId();?>;
    var articles = [];
<?php
    foreach ($cartItems as $cartItem)
    {
?>
        article = {
            "id": <?php echo $cartItem->articlesId;?>, // id de l'article
            "quantity": <?php echo $cartItem->quantity;?>, // quantité de l'article
        };
        articles.push(article);
<?php
    }
?>

<?php
    // Parcourir les articles du panier
    foreach ($cartItems as $cartItem)
    {
?>
        // Sélectionne l'élément de quantité de l'article
        element = document.querySelector("#articleQuantity<?php echo $cartItem->articlesId;?>");
        element.addEventListener("change", (event) => {

            // Obtient la quantité précédente de l'article
            let previousQuantity = parseInt(event.target.attributes.value.value);

            // Obtient la nouvelle quantité de l'article

```

```

let quantity = parseInt(event.target.value);

// Obtient l'ID de l'article
let articleId = parseInt(event.target.getAttribute("data-Id"));

// Parcourt les articles et met à jour la quantité correspondante
articles.forEach((item) => {
    if (item.id == articleId)
        item.quantity = quantity;
});

// Récupère l'élément d'article pour obtenir le prix
let el = document.querySelector("#article<?php echo $cartItem->articlesId;?>");
let price = parseFloat(el.getAttribute("data-price"));

// Calcule le total précédent avant le changement de quantité
let previousTotal = previousQuantity * price;

// Calcule le nouveau total en multipliant la quantité par le prix
let total = quantity * price;

// Calcule la variation de total
let deltaTotal = total - previousTotal;

// Récupère l'élément d'affichage du total et met à jour sa valeur formatée
el = document.querySelector("#articleTotal<?php echo $cartItem->articlesId;?>");
let tot = new Intl.NumberFormat('fr-CH', { style: 'currency', currency: 'CHF' }).format(total);
el.innerHTML = tot; // Met à jour le contenu HTML de l'élément avec le total formaté

// Met à jour la valeur de l'attribut "value" avec la nouvelle quantité
event.target.attributes.value.value = quantity;

// Récupère l'élément du total global
el = document.getElementById('totalPrice');
let currentTotal = parseFloat(el.getAttribute("data-totalPrice"));

// Calcule le nouveau total global en ajoutant la variation de total
let newTotal = currentTotal + deltaTotal;

// Met à jour l'affichage du nouveau total formaté
tot = new Intl.NumberFormat('fr-CH', { style: 'currency', currency: 'CHF' }).format(newTotal);
el.innerHTML = tot; // Met à jour le contenu HTML de l'élément avec le total formaté

// Met à jour l'attribut "data-totalPrice" avec le nouveau total global
el.setAttribute("data-totalPrice", newTotal);
});

<?php
}
?>

// Event pour éviter de fermer la page ou changer
window.addEventListener('beforeunload', (event) => {
    event.returnValue = 'Les modifications que vous avez apportées ne seront peut-être pas enregistrées.';
});

// Sauvegarde la quantité si le bouton sauvegarder est clicker

```

```

document.querySelector("#save").addEventListener("click", (event) => {
    saveCart(id, articles);
});

// Sauvegarde la quantité si le bouton commander est clicker
document.querySelector("#order").addEventListener("click", (event) => {
    saveCart(id, articles);
});

/** 
 * Exécuter la requête ajax pour sauvegarder le panier de l'utilisateur
 *
 * @param int id      L'id de l'utilisateur
 * @param array cart  Le tableau des articles du panier
 */
function saveCart(id, cart)
{
    let jsonObj = {};

    jsonObj.id = id;
    jsonObj.cart = cart;

    fetch('./tools/saveCart.php', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json; charset=utf-8'
        },
        body: JSON.stringify(jsonObj)
    })

    .then(function (response){
        return response.json();
    })
    .then(function (data) {
        switch (data.ReturnCode)
        {
            case 0: // C'est tout bon
                //window.location = "./index.php";
                break;

            case 1: // Les champs ne sont pas tous remplis
                //document.getElementById("errorMsg").style.display = "block";
                //document.getElementById("errorMsg").innerHTML = data.Message;
                break;
        }
    })
    .catch((error) => {
        console.error('Error:', error);
    });
}
</script>
</html>

```

## src/addCategory.php

```

<?php
    // Pour accéder à cette page l'utilisateur doit être authentifié en tant qu'administrateur
    $REQUIREDLOGIN = true;

```

```
$REQUIREDADMIN = true;

// Inclusion des fichiers nécessaires
require_once './includes/checkAll.php';

// Déclare les variables
$name = "";
$msg = "";

// Si le bouton du formulaire a été cliqué
if (isset($_POST['submit']))
{
    // Filtre le nom
    $name = filter_input(INPUT_POST, "name", FILTER_SANITIZE_SPECIAL_CHARS);

    // Affiche une erreur si la catégorie existe sinon on crée l'article
    if (CategoryExists($name))
    {
        $msg = "<p id='error'><i class='fa-solid fa-triangle-exclamation fa-xl me-2'></i>
Cette catégorie existe déjà.</p>";
    }
    else
    {
        AddCategory($name);
        $msg = "<p id='success'> La catégorie $name a été ajoutée. </p>";
    }
}
?>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ajouter une catégorie</title>

    <!-- Fontawesome -->
    <link href="assets/libraries/fontawesome/css/fontawesome.css" rel="stylesheet">
    <link href="assets/libraries/fontawesome/css/brands.css" rel="stylesheet">
    <link href="assets/libraries/fontawesome/css/solid.css" rel="stylesheet">
    <link href="assets/libraries/fontawesome/css/regular.css" rel="stylesheet">

    <!-- Bootstrap -->
    <link rel="stylesheet" href="assets/libraries/bootstrap/bootstrap.css">

    <!-- Custom CSS -->
    <link rel="stylesheet" href="assets/css/style.css">
</head>
<body>
    <?=ShowNavbar()?>
    <main class="mx-auto mt-5">
        <h2 class="mb-5">Ajouter une nouvelle catégorie</h2>
        <div class="my-3" id='errorMsg' role='alert'><?= $msg ?></div>

        <form method="post" onsubmit="return validateForm()">
            <div class="my-4">
                <label for="name" class="form-label">Nom de la catégorie <i class="fa-sharp fa-solid fa-star-of-life text-primary"></i></label>
                <input name="name" type="name" class="form-control" id="name" onkeyup="toUpperCase()" value="<?= $name ?>">
            </div>
        </form>
    </main>
</body>
```

```

        <div id="nameHelp" class="form-text text-danger"></div>
    </div>

    <div class="d-flex flex-row-reverse my-3">
        <button name="submit" type="submit" class="btn btn-primary submitBtn mb-3"
value="submit">Ajouter la catégorie</button>
    </div>
</form>
</main>
<script src=".//assets/libraries/bootstrap/bootstrap.js"></script>
<script src=".//assets/js/validateAddCategory.js"></script>
</body>
</html>

```

**src/articleDetails.php**

```

<?php

if (!isset($_GET['articleName']))
{
    header("Location: index.php");
    exit();
}

// Inclusion des fichiers nécessaires
require_once './includes/checkAll.php';
require_once ROOT.'tools/articleDetailsTools.php';

// Récupère tous les articles mis en vedette
$article = GetArticle($_GET['articleName']);
$msg = "";

if (isset($_POST['addToCart']))
{
    if(AddCartItem(ESessionManager::GetConnectedUserId(), $article->id))
    {
        $msg = "<p id='success'>L'article à été ajouter au panier</p>";
    }
}
else
{
    $msg = "";
}

?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Accueil</title>

    <!-- Fontawesome -->
    <link href=".//assets/libraries/fontawesome/css/fontawesome.css" rel="stylesheet">
    <link href=".//assets/libraries/fontawesome/css/brands.css" rel="stylesheet">
    <link href=".//assets/libraries/fontawesome/css/solid.css" rel="stylesheet">
    <link href=".//assets/libraries/fontawesome/css/regular.css" rel="stylesheet">

```

```

<!-- Bootstrap -->
<link rel="stylesheet" href="assets/libraries/bootstrap/bootstrap.css">

<!-- Custom CSS -->
<link rel="stylesheet" href="assets/css/style.css">
</head>
<body>
    <?=ShowNavbar()?>
    <main class="w-100 mt-5">
        <div class="my-3" id='errorMsg' role='alert'><?=$msg?></div>
        <div class="d-flex justify-content-center mx-auto">
            <?=ShowArticleDetails($article)?>
        </div>
    </main>
    <script src=".//assets/libraries/bootstrap/bootstrap.js"></script>
</body>
</html>

```

## src/addArticle.php

```

<?php
// Pour accéder à cette page l'utilisateur doit être authentifié entant qu'administrateur
$REQUIREDLOGIN = true;
$REQUIREDADMIN = true;

require_once './includes/checkAll.php';
require_once ROOT. 'tools/addArticleTools.php';

// Initialisation des variables
$name = "";
getDescription = "";
$price = 0;
$category = 0;
$stock = 0;
$featured = 0;
$msg = "";

// Si le bouton du formulaire a été cliqué
if (isset($_POST['submit']))
{
    $name = filter_input(INPUT_POST, "name", FILTER_SANITIZE_SPECIAL_CHARS);
// Nom
    $description = filter_input(INPUT_POST, "description", FILTER_SANITIZE_SPECIAL_CHARS);
// Description
    $price = doubleval(filter_input(INPUT_POST, "price", FILTER_VALIDATE_FLOAT));
// Prix
    $category = intval(filter_input(INPUT_POST, "categories", FILTER_VALIDATE_INT));
// Catégorie
    $stock = intval(filter_input(INPUT_POST, "stock", FILTER_VALIDATE_INT));
// Stock
    $featured = isset($_POST['featured']) ? 1 : 0;
// Mis en avant

    // Si l'article existe on affiche un message d'erreur
    if (ArticleExists($name))
    {
        // Affiche un message d'erreur

```

```

$msg = "<p id='error'><i class='fa-solid fa-triangle-exclamation fa-xl me-2'></i> Cet
article existe déjà</p>";
}

else
{
    $files = $_FILES['files']; // Récupère les images

    // Ajoute un article
    if (AddArticle($name, $description, $price, $stock, $featured, $category))
    {
        $articleId = GetArticle($name)->id; // Récupère l'id de l'article qui vien d'être
        créer

        // Permet de vérifier que $_FILES contient un fichier
        if (!empty($files['name'][0]))
        {
            // Parcours l'ensemble des fichiers
            for ($i = 0; $i < count($files['name']); $i++)
            {
                $fileName = uniqid(); // Nom du
                fichier (unique)
                $fileContent = file_get_contents($files["tmp_name"][$i]); // Contenu de
                l'image
                $fileType = $files['type'][$i]; // Type de
                fichier

                // Image principale (true pour la première insertion) puis false
                $mainImage = ($i === 0);

                // Ajoute les images
                AddEnc64Image($fileContent, $fileName, $fileType, intval($mainImage),
                $articleId);
            }
            $msg = "<p id='success'>L'article à été ajouté avec success</p>";
        }
    }
}
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ajouter un article</title>

    <!-- Fontawesome -->
    <link href="assets/libraries/fontawesome/css/fontawesome.css" rel="stylesheet">
    <link href="assets/libraries/fontawesome/css/brands.css" rel="stylesheet">
    <link href="assets/libraries/fontawesome/css/solid.css" rel="stylesheet">
    <link href="assets/libraries/fontawesome/css/regular.css" rel="stylesheet">

    <!-- Bootstrap -->
    <link rel="stylesheet" href="assets/libraries/bootstrap/bootstrap.css">

    <!-- Custom CSS -->
    <link rel="stylesheet" href="assets/css/style.css">
</head>
<body>
    <?=ShowNavbar()?>

```

```

<main class="mx-auto mt-5">
    <h2 class="mb-5">Ajouter un article</h2>
    <div class="my-3" id='errorMsg' role='alert'><?= $msg?></div>

    <form method="post" onsubmit="return validateForm()" enctype="multipart/form-data">

        <div class="my-4">
            <label for="name" class="form-label">Nom de l'article <i class="fa-sharp fa-solid fa-star-of-life text-primary"></i></label>
            <input type="text" name="name" id="name" class="form-control">
            <div id="nameHelp" class="form-text text-danger"></div>
        </div>

        <div class="my-4">
            <label for="description" class="form-label">Description <i class="fa-sharp fa-solid fa-star-of-life text-primary"></i></label>
            <textarea id="description" name="description"></textarea>
            <div id="descriptionHelp" class="form-text text-danger"></div>
        </div>

        <div class="my-4">
            <label for="price" class="form-label">Price <i class="fa-sharp fa-solid fa-star-of-life text-primary"></i></label>
            <input type="number" id="price" class="form-control" name="price" step="0.05" min="0">
            <div id="priceHelp" class="form-text text-danger"></div>
        </div>

        <div class="my-4">
            <label for="image" class="form-label">Image <i class="fa-sharp fa-solid fa-star-of-life text-primary"></i></label>
            <input type="file" id="image" name="files[]" class="form-control" accept="image/*" multiple>
            <div id="imageHelp" class="form-text text-danger"></div>
        </div>

        <div class="my-4">
            <label for="categories" class="form-label">Catégorie <i class="fa-sharp fa-solid fa-star-of-life text-primary"></i></label>
            <select id="categories" class="form-select" name="categories">
                <option value="0" selected>---</option>
                <?= ShowCategories($category)?>
            </select>
            <div id="categoryHelp" class="form-text text-danger"></div>
        </div>

        <div class="my-4">
            <label for="stock" class="form-label">Quantité en stock <i class="fa-sharp fa-solid fa-star-of-life text-primary"></i></label>
            <input type="number" id="stock" class="form-control" name="stock" min="0">
            <div id="stockHelp" class="form-text text-danger"></div>
        </div>

        <div class="my-4">
            <label for="featured" class="form-check-label">Mis en avant <i></i></label>
            <input type="checkbox" id="featured" class="form-check-input" name="featured">
        </div>

        <div class="my-4">
            <input name="submit" type="submit" class="btn btn-primary submitBtn mb-3" value="Valider">
        </div>
    </form>

```

```

</form>
</main>

<script src=".//assets/libraries/tinymce/tinymce.min.js"></script>
<script src=".//assets/libraries/tinymce/parametreTinymce.js"></script>
<script src=".//assets/js/validateAddArticle.js"></script>
<script src=".//assets/libraries/bootstrap/bootstrap.js"></script>

</body>
</html>

```

## src/session/SessionManager.php

```

<?php
/**
 * @copyright dominique@aigroz.com 2003-2016
 */
require_once ROOT.'session/SecureSession.php';

/**
 * @brief Helper class for managing session
 * @author dominique.aigroz@kadeo.net
 * @remark
 */
class ESessionManager {
    private static $objInstance;
    private $sessionInstance;
    /**
     * @brief Class Constructor - Create a new session if one doesn't exist
     *        Set to private so no-one can create a new instance via ' = new
    ESession();'
     */
    private function __construct() {}
    /**
     * @brief Like the constructor, we make __clone private so nobody can clone the instance
     */
    private function __clone() {}
    /**
     * @brief Returns the session instance
     * @return $objInstance;
     */
    public static function getInstance()
    {
        if(!self::$objInstance)
        {
            self::$objInstance = new ESessionManager();
            ini_set('session.save_handler', 'files');
            session_save_path(ROOT.ESES_DATAFOLDER);

            self::$objInstance->sessionInstance = new ESession(ESES_KEY);
            // Test if valid
            if (!self::$objInstance->sessionInstance->isValid()) {
                self::$objInstance->sessionInstance->forget();
            }
        }
        return self::$objInstance;
    } # end method
}

```

```

/**
 * Is the session valid
 * @return boolean True if valid, otherwise false.
 */
public static function IsValid(){
    $Instance = self::getInstance();

    if ($Instance && $Instance->sessionInstance && $Instance->sessionInstance->isValid()){
        if ($Instance->sessionInstance->get(ESES_USERID) !== null &&
            $Instance->sessionInstance->get(ESES_ISADMIN) !== null)
            return true;
    }
    return false;
} # end method

/**
 * Clear the session
 * @return boolean True if cleared, otherwise false.
 */
public static function Clear(){
    $Instance = self::getInstance();

    if ($Instance && $Instance->sessionInstance && $Instance->sessionInstance->isValid()){
        return $Instance->sessionInstance->forget();
    }
    return false;
} # end method

/**
 * Get the user email of the connected user
 * @return string|boolean The email, otherwise false.
 */
public static function GetConnectedUserId(){
    $Instance = self::getInstance();

    if ($Instance && $Instance->sessionInstance && $Instance->sessionInstance->isValid())
    {
        $id = $Instance->sessionInstance->get(ESES_USERID);
        if ($id)
            return $id;
    }
    return false;
} # end method

/**
 * Is the connected user admin
 * @return boolean True if admin, otherwise false.
 */
public static function IsConnectedUserAdmin()
{
    $Instance = self::getInstance();

    if ($Instance && $Instance->sessionInstance && $Instance->sessionInstance->isValid())
    {
        $isAdmin = $Instance->sessionInstance->get(ESES_ISADMIN);

        if ($isAdmin)
        {
            return $isAdmin;
        }
    }
}

```

```

        }
    }
    return false;
} # end method

/**
 * Set the user for this session
 * @param integer $userid    The user id
 * @param boolean $admin If admin
 * @return boolean True if successfully set, otherwise false
 */
public static function SetUser($userid, $admin){
    $Instance = self::getInstance();

    if ($Instance && $Instance->sessionInstance && $Instance->sessionInstance->isValid()){
        $Instance->sessionInstance->put(ESES_USERID, $userid);
        $Instance->sessionInstance->put(ESES_ISADMIN, $admin);
        return true;
    }
    return false;
} # end method

} # end class

```

## src/session/SecureSession.php

```

<?php
/**
 * @copyright dominique@aigroz.com 2003-2016
 */
require_once ROOT.'config/sessionparam.php';

/**
 * @brief Session class for managing session
 * @author dominique.aigroz@kadeo.net
 * @see https://www.php.net/manual/en/class.sessionhandler.php
 */
class ESession extends SessionHandler {

    protected $key;
    protected $name;
    protected $cookie;

    /**
     * Constructor
     * @param unknown $key    The session key
     * @param string $name   The session name
     * @param array $cookie The cookie array
     */
    public function __construct($key, $name = 'SCHOOL_SESSION', $cookie = [])
    {
        $this->key = $key;
        $this->name = $name;
        $this->cookie = $cookie;

        $this->cookie += [
            'lifetime' => 0,

```

```

'path'      => ini_get('session.cookie_path'),
'domain'   => ini_get('session.cookie_domain'),
'secure'    => isset($_SERVER['HTTPS']),
'httponly'  => true
];
$this->setup();
// Start the session
session_start();
} # end method

/***
 * Setup the session using its name
 */
private function setup()
{
    ini_set('session.use_cookies', 1);
    ini_set('session.use_only_cookies', 1);

    session_name($this->name);
    session_set_cookie_params(
        $this->cookie['lifetime'],
        $this->cookie['path'],
        $this->cookie['domain'],
        $this->cookie['secure'],
        $this->cookie['httponly']
    );
} # end method

/***
 * Starts the session
 * @return boolean True if successfully started, otherwise false is returned
 */
public function start()
{
    if (session_id() === '') {
        if (session_start()) {
            return mt_rand(0, 4) === 0 ? $this->refresh() : true; // 1/5
        }
    }
    return false;
} # end method

/***
 * Clear the session
 * @return boolean True if successfully destroyed, otherwise false is returned
 */
public function forget()
{
    if (session_id() === '') {
        return false;
    }
    // Destroy the session
    $_SESSION = [];
    setcookie(
        $this->name,
        '',
        time() - 42000,

```

```

        $this->cookie['path'],
        $this->cookie['domain'],
        $this->cookie['secure'],
        $this->cookie['httponly']
    );
    return session_destroy();
} # end method

/**
 * Refresh the session by regenerate its id
 * @return boolean
 */
public function refresh()
{
    return session_regenerate_id(true);
} # end method

/**
 * Read the session data using an id
 * {@inheritDoc}
 * @see SessionHandler::read()
 */
public function read($id)
{
    return mcrypt_decrypt(MCRYPT_3DES, $this->key, parent::read($id), MCRYPT_MODE_ECB);
} # end method

/**
 * Write data to the session using an id
 * {@inheritDoc}
 * @see SessionHandler::write()
 */
public function write($id, $data)
{
    return parent::write($id, mcrypt_encrypt(MCRYPT_3DES, $this->key, $data,
MCRYPT_MODE_ECB));
} # end method

/**
 * Has the session expired
 * @param number $ttl The time. Default is 30
 * @return boolean True if expired, otherwise false.
 */
public function isExpired($ttl = 120)
{
    $last = isset($_SESSION['_last_activity']) ? $_SESSION['_last_activity'] : false;
    if ($last !== false && time() - $last > $ttl * 60) {
        return true;
    }

    $_SESSION['_last_activity'] = time();
    return false;
} # end method

/**
 * Is the fingerprint corresponding to the one stored within the session

```

```

 * @return boolean True if corresponding, otherwise false.
 */
public function isFingerprint()
{
    $hash = md5(
        $_SERVER['HTTP_USER_AGENT'] .
        (ip2long($_SERVER['REMOTE_ADDR']) & ip2long('255.255.0.0'))
    );

    if (isset($_SESSION['_fingerprint'])) {
        return $_SESSION['_fingerprint'] === $hash;
    }

    $_SESSION['_fingerprint'] = $hash;
    return true;
} # end method

/***
 * Is the session valid
 * @return boolean True if valid, False if not.
 */
public function isValid()
{
    // prevent the session is started
    if (session_id() === '')
        $this->start();

    return ! $this->isExpired() && $this->isFingerprint();
} # end method

/***
 * Get session value
 * @param unknown $name The value key
 * @return NULL|unknown The value or null if not found
 */
public function get($name)
{
    // prevent the session is started
    if (session_id() === '')
        $this->start();

    $parsed = explode('.', $name);
    $result = $_SESSION;

    while ($parsed) {
        $next = array_shift($parsed);
        if (isset($result[$next])) {
            $result = $result[$next];
        } else {
            return null;
        }
    }
    return $result;
} # end method

/***
 * Put a value for a key name
*/

```

```

 * @param unknown $name The value key
 * @param unknown $value The value to set
 */
public function put($name, $value)
{
    // prevent the session is started
    if (session_id() === '')
        $this->start();

    $parsed = explode('.', $name);
    $session =& $_SESSION;

    while (count($parsed) > 1) {
        $next = array_shift($parsed);
        if (!isset($session[$next]) || !is_array($session[$next])) {
            $session[$next] = [];
        }
        $session =& $session[$next];
    }

    $session[array_shift($parsed)] = $value;
} # end method
} # end class

```

## src/model/user.php

```

<?php
require_once ROOT.'db/database.php';
require_once ROOT.'containers/User.php';
require_once ROOT.'session/SessionManager.php';

/**
 * Insère un nouvel utilisateur dans la table USERS de la base de données.
 * @param string $name Nom de l'utilisateur.
 * @param string $surname Prénom de l'utilisateur.
 * @param string $email Adresse e-mail de l'utilisateur.
 * @param string $password Mot de passe de l'utilisateur.
 * @param string $gender Genre de l'utilisateur.
 * @param string $address1 Adresse de l'utilisateur (ligne 1).
 * @param string $address2 Adresse de l'utilisateur (ligne 2).
 * @param int $city ID de la ville de l'utilisateur.
 * @param string $zipCode Code postal de l'utilisateur.
 * @return bool Retourne TRUE si l'inscription réussit, sinon FALSE.
*/
function RegisterUser($name, $surname, $email, $password, $gender, $address1, $address2,
$cityId, $zipCode)
{
    // Insère un nouvel utilisateur dans la table "USERS" de la base de données
    $sql = "INSERT INTO `USERS`(`NAME`, `SURNAME`, `EMAIL`, `PASSWORD`, `GENDER`, `ADDRESS1`,
`ADDRESS2`, `CITIES_ID`, `ZIP_CODE`)
VALUES(:name, :surname, :email, :pw, :gender, :address1, :address2, :cityID, :zipCode)";

    // Prépare la requête SQL
    $statement = EDatabase::prepare($sql);

    try
    {

```

```

// Crée un tableau des valeurs à exécuter
$params = array(
    ":name" => $name,
    ":surname" => $surname,
    ":email" => $email,
    ":pw" => $password,
    ":gender" => $gender,
    ":address1" => $address1,
    ":cityID" => $cityId,
    ":zipCode" => $zipCode
);

// Vérifie si $address2 est vide
if (!empty($address2))
{
    $params[":address2"] = $address2;
}
else
{
    // Si $address2 est vide, assigne une valeur nulle à :address2 dans le tableau des
valeurs
    $params[":address2"] = null;
}

// Exécute la requête en utilisant les valeurs passées en paramètre
$statement->execute($params);
}

catch (PDOException $e)
{
    // Retourne false en cas d'erreur
    return false;
}

// Retourne true si tout s'est bien passé
return true;
}

/**
 * Récupère un utilisateur grâce à son email donné en paramètre
 * @param string $email L'email de l'utilisateur à récupérer
 * @return mixed L'utilisateur récupéré sous forme d'objet User, ou false si une erreur est
survenue
 */
function GetUser($email)
{
    // Requête SQL qui récupère les données de l'utilisateur
    $sql = "SELECT `USERS`.`ID`, `USERS`.`NAME`, `SURNAME`, `EMAIL`, `PASSWORD`, `GENDER`,
`ADDRESS1`, `ADDRESS2`, CITIES.NAME as CITY, `ZIP_CODE`, `IS_ADMIN`
FROM `USERS`
INNER JOIN CITIES ON USERS.CITIES_ID = CITIES.ID
WHERE EMAIL = :email";

    // Prépare la requête SQL
    $statement = EDatabase:::prepare($sql);

    try
    {
        // Exécute la requête SQL en utilisant l'email passé en paramètre
        $statement->execute(array(":email" => $email));

        // Récupère la première ligne de résultat
        $row = $statement->fetch(PDO::FETCH_ASSOC,PDO::FETCH_ORI_NEXT);
    }
}

```

```

// Crée un objet User à partir des données récupérées
return new User(
    intval($row['ID']),           // ID      : l'identifiant unique de l'utilisateur
(entier)    $row['NAME'],          // NAME   : le nom de l'utilisateur (chaîne de
caractères) $row['SURNAME'],       // SURNAME : le prénom de l'utilisateur (chaîne de
caractères) $row['EMAIL'],        // EMAIL   : l'adresse email de l'utilisateur
(chaîne de caractères) $row['PASSWORD'], // PASSWORD : le mot de passe haché de l'utilisateur
(chaîne de caractères)           // GENDER  : le genre de l'utilisateur (entier)
(intervalle) $row['ADDRESS1'],     // ADDRESS1 : la première adresse de l'utilisateur
(chaîne de caractères)           // ADDRESS2 : la deuxième adresse de l'utilisateur
(optionnel)  $row['CITY'],          // CITY    : le nom de la ville où habite
l'utilisateur (chaîne de caractères) intval($row['ZIP_CODE']), // ZIP_CODE : le code postal de l'utilisateur
(entier)    boolval($row['IS_ADMIN']) // IS_ADMIN : un booléen qui indique si l'utilisateur
est un administrateur ou non (booléen)
);
}

catch (PDOException $e)
{
    // En cas d'erreur, retourne false
    return false;
}

/**
 * Récupère un utilisateur à partir de son identifiant donné en paramètre.
 *
 * @param int $id L'identifiant de l'utilisateur à récupérer.
 * @return mixed L'utilisateur récupéré sous forme d'objet User, ou false si une erreur est
survenue.
 */
function GetUserById($id)
{
    // Requête SQL qui récupère les données de l'utilisateur.
    $sql = "SELECT `USERS`.`ID`, `USERS`.`NAME`, `SURNAME`, `EMAIL`, `PASSWORD`, `GENDER`,
`ADDRESS1`, `ADDRESS2`, CITIES.NAME as CITY, `ZIP_CODE`, `IS_ADMIN`
FROM `USERS`
INNER JOIN CITIES ON USERS.CITIES_ID = CITIES.ID
WHERE `USERS`.`ID` = :id";

    // Prépare la requête SQL.
    $statement = EDatabase::prepare($sql);

    try
    {
        // Exécute la requête SQL en utilisant l'identifiant passé en paramètre.
        $statement->execute(array(":id" => $id));

        // Récupère la première ligne de résultat.
        $row = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT);

        // Crée un objet User à partir des données récupérées.
        return new User(

```

```

        intval($row['ID']),           // ID      : l'identifiant unique de l'utilisateur
(entier)    $row['NAME'],          // NAME    : le nom de l'utilisateur (chaîne de
caractères) $row['SURNAME'],     // SURNAME : le prénom de l'utilisateur (chaîne de
caractères) $row['EMAIL'],       // EMAIL   : l'adresse email de l'utilisateur
(chaîne de caractères) $row['PASSWORD'], // PASSWORD : le mot de passe haché de l'utilisateur
(chaîne de caractères)         intval($row['GENDER']), // GENDER   : le genre de l'utilisateur (entier)
                                $row['ADDRESS1'],    // ADDRESS1 : la première adresse de l'utilisateur
(chaîne de caractères)         $row['ADDRESS2'],    // ADDRESS2 : la deuxième adresse de l'utilisateur
(optionnelle) (chaîne de caractères) $row['CITY'],      // CITY     : le nom de la ville où habite
l'utilisateur (chaîne de caractères) intval($row['ZIP_CODE']), // ZIP_CODE : le code postal de l'utilisateur
(entier)      boolval($row['IS_ADMIN']) // IS_ADMIN : un booléen qui indique si l'utilisateur
est un administrateur ou non (booléen)
);
}

catch (PDOException $e)
{
    // En cas d'erreur, retourne false.
    return false;
}
}

/**
 * Vérifie si l'email est déjà présent dans la base de données.
 *
 * @param string $email L'adresse email à rechercher.
 * @return bool True si l'email existe, False sinon.
 */
function EmailExists($email)
{
    // Requête SQL pour récupérer l'email dans la table USERS
    $sql = "SELECT `EMAIL` FROM USERS WHERE `EMAIL` = :email";

    // Prépare la requête SQL
    $statement = EDatabase::prepare($sql);

    try
    {
        // Exécute la requête SQL avec l'email en paramètre
        $statement->execute(array(":email" => $email));

        // Vérifie le nombre de lignes de résultats pour déterminer si l'email existe
        return boolval($statement->rowCount() > 0);
    }
    catch(PDOException $e)
    {
        // En cas d'erreur, retourne False
        return false;
    }
}

/**
 * Vérifie les informations de connexion d'un utilisateur.
 * Si les informations sont valides, enregistre l'utilisateur dans une session.

```

```

*
* @param string $email L'adresse email de l'utilisateur.
* @param string $password Le mot de passe de l'utilisateur.
* @return bool Retourne true si l'utilisateur est connecté avec succès, sinon retourne false.
*/
function LoginUser($email, $password)
{
    // Vérifie que l'email existe dans la base de données
    if (EmailExists($email))
    {
        // Récupère l'utilisateur correspondant à l'email
        $user = GetUser($email);

        // Vérifie que le mot de passe est correct en utilisant la fonction password_verify()
        // de PHP
        if (password_verify($password, $user->password))
        {
            // Créer une session avec l'utilisateur
            ESessionManager::SetUser($user->id, $user->isAdmin);

            // Retourne true si tout c'est bien passé
            return true;
        }
        else
        {
            // Retourne false si il y a une erreur
            return false;
        }
    }
    else
    {
        // Retourne false si il y a une erreur
        return false;
    }
}
}

```

**src/includes/web.inc.all.php**

```

<?php

// Le fichier de gestion des sessions
require_once ROOT.'session/SessionManager.php';

// Navbar
require_once ROOT.'includes/nav.php';

// phpmailer
require_once ROOT.'includes/mail.php';

// Les fichiers concernant les appels à la base de données
require_once ROOT.'model/user.php';
require_once ROOT.'model/article.php';
require_once ROOT.'model/category.php';
require_once ROOT.'model/image.php';
require_once ROOT.'model/cartItem.php';
require_once ROOT.'model/city.php';

```

**src/db/database.php**

```

<?php
// @remark Mettre le bon chemin d'accès à votre fichier contenant les constantes

/*
 * @remark Si vous voulez travailler en lien absolu, utilisez la notation ci-après
 * require_once $_SERVER['DOCUMENT_ROOT'].'/config/conparam.php';
*/

require_once ROOT.'config/conparam.php';

/** @brief Helper class encapsulating the PDO object
 * @author dominique@aigroz.com
 * @remark
 */
class EDatabase
{
    // @var PDO The static PDO object instance created within getInstance()
    private static $objInstance;

    /** @brief Class Constructor - Create a new database connection if one doesn't exist
     * Set to private so no-one can create a new instance via ' = new EDatabase();'*/
    private function __construct() {}

    // @brief Like the constructor, we make __clone private so nobody can clone the instance
    private function __clone() {}

    /** @brief Returns DB instance or create initial connection
     * @return $objInstance;
     */
    private static function getInstance()
    {
        if(!self::$objInstance)
        {
            try
            {
                $dsn = EDB_DBTYPE.':host='.EDB_HOST.';port='.EDB_PORT.';dbname='.EDB_DBNAME;
                self::$objInstance = new PDO($dsn, EDB_USER, EDB_PASS,
array('charset'=>'utf8'));
                self::$objInstance->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
            }
            catch(PDOException $e )
            {
                echo "EDatabase Error: ".$e;
            }
        }
        return self::$objInstance;
    } # end method

    /** @brief Passes on any static calls to this class onto the singleton PDO instance
     * @param      $chrMethod      The method to call
     * @param      $arrArguments   The method's parameters
     * @return     $mix             The method's return value
     */
    final public static function __callStatic( $chrMethod, $arrArguments )
    {
        $objInstance = self::getInstance();
        return call_user_func_array(array($objInstance, $chrMethod), $arrArguments);
    }
}

```

```
} # end method
}
```

## src/containers/User.php

```
<?php

class User
{
    /**
     * @var int $id L'identifiant unique de l'utilisateur
     */
    public $id;

    /**
     * @var string $name Le nom d'utilisateur
     */
    public $name;

    /**
     * @var string $surname Le prénom d'utilisateur
     */
    public $surname;

    /**
     * @var string $email L'email de l'utilisateur
     */
    public $email;

    /**
     * @var string $password Le mot de passe haché de l'utilisateur
     */
    public $password;

    /**
     * @var string $gender Le genre de l'utilisateur
     */
    public $gender;

    /**
     * @var string $address1 La première adresse de l'utilisateur
     */
    public $address1;

    /**
     * @var string $address2 La deuxième adresse de l'utilisateur (optionnel)
     */
    public $address2;

    /**
     * @var string $city La ville de l'utilisateur
     */
    public $city;

    /**
     * @var string $zipCode Le code postal de l'utilisateur
     */
}
```

```

*/
public $zipCode;

/**
 * @var bool $isAdmin Indique si l'utilisateur est administrateur ou non
 */
public $isAdmin;

/**
 * Constructeur appelé au moment de la création de l'objet. new User();
 *
 * @param int $idParam L'id unique provenant de la base de données. (Optionel) Défaut -1
 * @param string $nameParam Le nom de l'utilisateur
 * @param string $surnameParam Le prénom de l'utilisateur
 * @param string $emailParam L'email de l'utilisateur
 * @param string $passwordParam Le mot de passe de l'utilisateur
 * @param string $genderParam Le genre de l'utilisateur (homme ou femme)
 * @param string $adress1Param La première ligne de l'adresse de l'utilisateur
 * @param string $adress2Param La deuxième ligne de l'adresse de l'utilisateur
 * @param string $cityParam La ville de l'utilisateur
 * @param string $zipCodeParam Le code postal de l'utilisateur
 * @param bool $isAdminParam Détermine si l'utilisateur est un administrateur (true) ou
non (false)
*/
public function __construct($idParam = -1, $nameParam = "", $surnameParam = "",
$emailParam = "", $passwordParam = "", $genderParam = "", $adress1Param = "", $adress2Param =
 "", $cityParam = "", $zipCodeParam = "", $isAdminParam = "")
{
    $this->id = $idParam;
    $this->name = $nameParam;
    $this->surname = $surnameParam;
    $this->email = $emailParam;
    $this->password = $passwordParam;
    $this->gender = $genderParam;
    $this->address1 = $adress1Param;
    $this->address2 = $adress2Param;
    $this->city = $cityParam;
    $this->zipCode = $zipCodeParam;
    $this->isAdmin = $isAdminParam;
}
}

```

## src/config/sessionparam.php

```

<?php
/**
 * @copyright dominique@aigroz.com 2003-2016
 */

/*
 * @brief Session Constants
 */
define('ESES_KEY', "gym.soares");
define('ESES_USERID', "user.id");
define('ESES_ISADMIN', "user.isadmin");

```

```
// Folder from the root to store session data
define('ESES_DATAFOLDER', "/SessionData");
```

## src/config/phpmailerparam.php

```
<?php
/*
    Constantes requise pour l'envois de mail avec PHPMailer
*/

define('MAIL_SERVER', "smtp.office365.com");           // Adresse de serveur SMTP
define('MAIL_PORT', 587);                                // Port 587 pour TLS ou 465 pour SSL
define('MAIL_TRANS', "STARTTLS");                        // Protocole de sécurisation des
échanges de données
define('MAIL_USERNAME', "SuperUserGym@hotmail.com");     // Adresse mail de l'envoyeur
define('MAIL_PASSWORD', "TpI2022-2023");                 // Mot de passe
define('MAIL_ENCODE', "UTF-8");                          // Encodage du mail
```

## src/config/conparam.php

```
<?php
// @remark Remplir correctement les constantes ci-dessous en fonction de votre base de
données
// @brief Connection constants

/*
define('EDB_DBTYPE', "mysql");
define('EDB_DBNAME', "le nom de la base de données");
define('EDB_HOST', "127.0.0.1");
define('EDB_PORT', "3306");
define('EDB_USER', "l'utilisateur qui établit la connexion");
define('EDB_PASS', "le mot de passe de l'utilisateur");
*/
define('EDB_DBTYPE', "mysql");
define('EDB_DBNAME', "GYM");
define('EDB_HOST', "127.0.0.1");
define('EDB_PORT', "3306");
define('EDB_USER', "GYM_ADMIN");
define('EDB_PASS', "TpI2022-2023");
```

## src/assets/js/validateUpdateCategory.js

```
// Met les lettres en majuscule
function toUpperCase()
{
    let name = document.getElementById("name");

    // Converti en majuscule
    name.value = name.value.toUpperCase();
}

// Permet de valider les entrer dans le formulaire
```

```

function validateForm()
{
    let name = document.getElementById("name");
    let categories = document.getElementById("categories");

    let nameErrorMsg = document.getElementById("nameHelp");
    let categoriesErrorMsg = document.getElementById("categoriesHelp");
    let errorMsg = document.getElementById("errorMsg");

    // Si le champ est vide on affiche un message d'erreur
    if (name.value === "")
    {
        // Affiche un message d'erreur
        nameErrorMsg.innerText = "Veuillez mettre un nouveau nom";
        name.focus();
    }
    else
    {
        // Efface le message d'erreur
        nameErrorMsg.innerText = "";
    }

    // Si le champ est vide on affiche un message d'erreur
    if (categories.value === "0")
    {
        // Affiche un message d'erreur
        categoriesErrorMsg.innerText = "Veuillez saisir une catégorie à modifier";
        categories.focus();
    }
    else
    {
        // Efface le message d'erreur
        categoriesErrorMsg.innerText = "";
    }

    // Si les messages d'erreur sont vides on retourne true sinon false
    if (nameErrorMsg.innerText === "" && categoriesErrorMsg.innerText === "")
    {
        // Renvoie vrai, la saisie est valide
        return true;
    }
    else
    {
        // Efface le message d'erreur générer avec php
        errorMsg.innerHTML = "";

        // Renvoie faux, il y a des erreurs de saisie
        return false;
    }
}

```

## src/assets/css/style.css

```

main
{
    width: 50%;
}

```

```
#errorMsg
{
    font-weight: 600;
    font-size: large;
    text-align: center;
}

#error
{
    color: █ #FFC107;
}

#success
{
    color: █ #28A745;
}

h2
{
    text-align: center;
}

.submitBtn
{
    width: 100%;
}

.fa-star-of-life
{
    font-size: 12px;
}

#carouselFeatured
{
    width: 90%;
    margin: 0 auto 0 auto;
}

.carouselImages
{
    height: 250px;
    width: 350px;
}

.filterControl
{
    width: 100%;
}

.search
{
    width: 100%;
}

.filterContainer
{
    width: 100%;
}

.filterBtn
```

```
{  
    width: 100%;  
}  
  
.navigationBtn  
{  
    width: 100%;  
}  
  
.detailsBtn  
{  
    width: 25%;  
}  
  
.detailsImg  
{  
    width: 500px;  
}  
  
/*xxl*/  
@media screen and (max-width: 1400px)  
{  
    .nav-item  
    {  
        padding-right: 15px;  
    }  
  
    .filterControl  
    {  
  
        margin-top: 8px;  
    }  
  
    #eraserFilterBtn  
    {  
        margin-bottom: 10px;  
    }  
  
    .detailsBtn  
    {  
        width: 40%;  
    }  
}  
  
@media screen and (max-width: 1200px)  
{  
    main  
    {  
        width: 65%;  
    }  
  
    .detailsImg  
    {  
        width: 400px;  
    }  
}  
  
@media screen and (max-width: 992px)  
{  
    main  
}
```

```
{  
    width: 90%;  
}  
  
.filteredArticle  
{  
    width: 90%;  
}  
  
.detailsBtn  
{  
    width: 65%;  
}  
}  
  
@media screen and (max-width: 600px)  
{  
    main  
{  
        width: 95%;  
    }  
  
.detailsImg  
{  
    width: 300px;  
}  
}
```