Spécification Fonctionnelle Détaillée

**Projet :** Site web pour le Club Taekwondo de St Priest

# Partie 1 : Informations Générales

**1.1. URL du Site Web**

* **URL du site en ligne :** <https://www.taekwondo-pise.worldlite.fr/>

**1.2. Accès aux Comptes**

**Compte Utilisateur (Membre) :**

* Login : admin@admin.fr
* Mot de passe : MOTdepasse123 !

**Compte Administrateur :**

* Login : membre@membre.com
* Mot de passe : MOTdepasse123 !

# Partie 2 : Description Générale du Projet

**2.1. Objet du Site**

Le site web "Taekwondo Club St Priest" est une plateforme dynamique conçue pour servir à la fois de vitrine numérique pour le club et d'outil de gestion interne. Son objectif principal est de renforcer la communication avec les membres actuels et potentiels, de promouvoir les activités du club et de faciliter l'administration quotidienne par les responsables. Il vise à offrir une expérience utilisateur intuitive et un environnement d'administration robuste pour la gestion des contenus et des utilisateurs.

**2.2. Fonctionnement Général**

Le site se compose de deux grandes parties distinctes, le "Front-Office" pour les visiteurs et membres, et le "Back-Office" pour les administrateurs du club.

Côté "Front-Office" (Utilisateurs et Visiteurs) :

* **Navigation Publique :** Les visiteurs peuvent consulter les informations générales du club, y compris la présentation du club, les actualités, les entraîneurs, et les informations de contact.
* **Espace Membre :** Les utilisateurs peuvent créer un compte et s'authentifier pour accéder à leur espace personnel. Dans cet espace, ils peuvent consulter et modifier leurs informations de profil (nom, prénom, email, mot de passe).
* **Interactions :** Le site propose des fonctionnalités de filtrage et de tri pour les actualités, et un formulaire de contact pour envoyer des messages au club.

Côté "Back-Office" (Administrateurs) :

Les utilisateurs avec le rôle 'admin' ont accès à des modules de gestion avancés via leur espace profil. Ils peuvent y gérer :

* **Utilisateurs :** Création, modification des informations (y compris le rôle d'administrateur/membre), et suppression de comptes utilisateurs.
* **Actualités :** Ajout, modification et suppression des articles d'actualité, y compris leur contenu, auteur et tags.
* **Cours :** Gestion des offres de cours du club (nom, niveau, prix, description, entraîneur associé).
* **Entraîneurs :** Ajout, modification et suppression des profils des entraîneurs (prénom, nom, biographie, photo).
* **Messages :** Consultation des messages envoyés via le formulaire de contact, marquage comme lu et suppression.

# PARTIE 3 : Architecture Technique et Organisation des Fichiers

**3.1. Architecture Générale**

Le projet suit une architecture simple, proche d'un modèle "Front Controller" ou "Model View Controller", où la logique est centralisée et les responsabilités sont séparées entre les fichiers PHP pour l'affichage, la logique métier, et la configuration.

**3.2. Structure des Répertoires**

* **`/` (Racine du Projet) :** Contient les principaux fichiers PHP d'entrée pour les différentes pages du site (`index.php`, `about.php`, `contact.php`, `news.php`, `news\_detail.php`, `login.php`, `logout.php`, `profile.php`, `register.php`, `team.php`, `erreur.php`).
* **`/css/` :** Contient la feuille de style principale (`styles.css`).
* **`/fonction/` :** Centralise toutes les fonctions PHP métier et utilitaires (`fonctions.php`).
* **`/img/` :** Stocke toutes les ressources images du site.
* **`/includes/` :** Regroupe les parties réutilisables du code PHP et contient les éléments communs à toutes les pages (`header.php`, `footer.php`, `head.php`).
* **`/includes/components/` :** Contient des composants d'interface génériques (`table.php`, `pagination.php`, `toolbar.php`).
* **`/includes/profile/` :** Contient les modules spécifiques à l'espace membre/admin (`overview.php`, `users.php`, `posts.php`, `classes.php`, `trainers.php`, `messages.php`).
* **`/js/` :** Contient le fichier JavaScript principal (`main.js`).
* **`/logs/` :** Stocke les fichiers de log des erreurs (`bdd\_erreurs.log`).
* **`/parametrage/` :** Contient le fichier de configuration (`param.php`).
* **`/document/` :** Contient le dump de la base de données (`tkd.sql`) ainsi que la Spécification Fonctionnelle Détaillée (ce-document).

**3.3. Base de Données**

Le site utilise une seule base de données relationnelle (`tkd`). Le schéma est structuré en plusieurs tables interconnectées via des clés étrangères pour organiser les données de manière cohérente :

* `users` : Gestion des comptes utilisateurs (membres et administrateurs).
* `team` : Informations sur les entraîneurs du club.
* `classes` : Détails des cours proposés.
* `schedules` : Planning des cours (liens `classes` et jours/heures).
* `posts` : Articles d'actualité.
* `tags` : Mots-clés pour catégoriser les actualités.
* `post\_tag` : Table de liaison entre `posts` et `tags`.
* `messages` : Messages envoyés via le formulaire de contact.

# PARTIE 4 : Fonctions Centrales

Le fichier `fonctions.php` est le cœur logique du projet. Il regroupe l'ensemble des fonctions PHP utilisées pour interagir avec la base de données, valider les données et gérer les aspects métier.

**4.1. Gestion de la Base de Données**

* `logErreur(string $function, string $message, array $context = array())` : Enregistre les erreurs rencontré dans un fichier de log dédié et doit être appeler dans un bloc `try {} catch {}`.
* `connexionBaseDeDonnees(): PDO` : Établit une connexion unique (singleton) à la base de données via la classe PDO, avec gestion des exceptions.

**4.2. Gestion des Utilisateurs (Authentification)**

Sont regroupées ci-dessous les fonctions permettant à un utilisateur de se connecter au site web :

* `estValideMail(string $email): bool` : Valide le format et la taille d'une adresse e-mail.
* `estValideMotdepasse(string $mdp): bool` : Vérifie la robustesse d'un mot de passe (longueur, majuscule, chiffre, caractère spécial).
* `authentification(string $email, string $mdp): bool` : Authentifie un utilisateur en vérifiant les identifiants (email) et le hachage du mot de passe.
* `connexionUtilisateur(string $email): array|null` : Renvoi les données d'un utilisateur après authentification.
* `isUtilisateur(string $email, ?int $excludeId = null): bool` : Vérifie si un e-mail est déjà utilisé, permet d’exclure un email dans le cas d’une mise à jour de données.

**4.3. Gestion des Utilisateurs (CRUD)**

Le modèle de gestion CRUD défini pour les utilisateurs est répliqué pour les autres sections métiers du Back-Office, assurant une cohérence et une robustesse dans la manipulation des données pour les actualités (posts), les entraîneurs (team), les cours (classes) et les messages (messages).

* `enregistrerUtilisateur(array $donnees): bool` : Crée un nouvel utilisateur.
* `getListeUtilisateurs(): array` : Récupère la liste de tous les utilisateurs.
* `getUtilisateurParId(int $id): array` : Récupère les données d'un utilisateur par son ID.
* `modifierUtilisateur(int $id, array $donnees): bool` : Met à jour les informations d'un utilisateur.
* `supprimerUtilisateur(int $id): bool` : Supprime un utilisateur.
* `validerDonneesUtilisateur(array $data, ?int $excludeId = null): array` : Valide les données soumises pour un utilisateur.

**4.4. Gestion des Actualités et Tags**

* `getListePosts(int $excerptLength = 200, ?string $filterTag = null, string $sort = 'desc'): array` : Récupère la liste des articles avec options de filtrage et tri.
* `syncPostTags(int $postId, string $tagsCsv): void` : Synchronise les tags associés à un article.
* `getTagsPourPost(int $id): array` : Récupère la liste des tags associés à un article.
* `getListeTags()` : Récupère la liste de tous les tags existants.

**4.5. Fonctions Utilitaires Générales**

* `displayFlash(): void` : Affiche les messages flash stockés en session.
* `setFlash(string $type, string $msg): void` : Définit un message flash.
* `requireConnexion(): void` : Redirige vers la page de connexion si l'utilisateur n'est pas connecté.
* `paginateArray(array $all, string $param, int $perPage = 10): array` : Gère la pagination en mémoire d'un tableau de données.

**4.6. Configuration**

Le fichier param.php centralise les constantes et paramètres essentiels au fonctionnement de l'application, tels que les identifiants de connexion à la base de données (`DB\_HOST`, `DB\_NAME`, `DB\_USER`, `DB\_PASS`), les contraintes de sécurité pour les mots de passe (`PSSWD\_MIN\_LEN`) et le chemin du fichier de log (`LOG\_PATH`).

**4.7. Succès techniques**

La robustesse et la maintenabilité du code sont assurées par l'implémentation de plusieurs fonctions techniques spécifiques :

* **Sécurité des données** : La fonction nettoyerDonnees() joue un rôle crucial dans la prévention des injections XSS en nettoyant les entrées utilisateur. De plus, l'utilisation systématique de password\_hash() et password\_verify() (intégrée dans authentification() et enregistrerUtilisateur()) garantit un hachage sécurisé des mots de passe.
* **Gestion des erreurs** : Le mécanisme centralisé de logErreur() permet un suivi efficace des incidents en production, en enregistrant les détails des exceptions dans un fichier dédié.
* **Optimisation et performance** : La connexion à la base de données via connexionBaseDeDonnees() utilise le pattern Singleton pour garantir une instance unique de PDO, optimisant ainsi les ressources. La pagination dynamique des données est gérée par paginateArray(), offrant une expérience utilisateur fluide sans surcharger le serveur.
* **Cohérence des données** : La fonction syncPostTags() illustre une gestion complexe des relations entre entités (posts et tags), assurant la cohérence des mots-clés associés aux articles.
* **Expérience utilisateur** : Le système de setFlash() et displayFlash() fournit un feedback clair et non intrusif à l'utilisateur après chaque action.