

Site Web association sportive

Étude de l'existant

Table des matières

1.	La base de données.....	1
1.1.	La gestion des membres et des administrateurs.....	1
1.2.	Les données informatives.....	1
1.3.	Les épreuves des 4 saisons	2
1.1.	Les classements.....	2
2.	Source et structure de l'application	2
3.	La structure de chaque fonctionnalité	4
4.	La déclaration de type	5
5.	L'accès aux données	5
6.	Les classes techniques	6
7.	Les composants utilisés côté client	7
8.	Les composants utilisés côté serveur chargés via composer	7
9.	Le traitement des erreurs	8
10.	Le contrôle d'accès.....	9
11.	Les classes métiers.....	11
12.	La page d'accueil.....	15
13.	Les pages consultables par tous les visiteurs.....	15
13.1.	La connexion.....	17
14.	Les pages consultables uniquement par les membres.....	18
15.	Les pages consultables par des administrateurs.....	20
15.1.	La modification des mentions légales.....	20
15.2.	La gestion des classements	21
15.4.	La gestion des membres	22
15.5.	L'ajout d'un membre	22
15.6.	La planification des 4 saisons.....	23
15.7.	La photothèque.....	25
16.	Le répertoire backoffice	26
16.1.	Contrôle d'accès sur le répertoire backoffice.....	26
16.2.	Principe de la gestion des droits des administrateurs.....	27
16.3.	L'interface d'accueil	27
16.4.	La gestion des administrateurs.....	28
16.5.	La gestion des fonctions d'administration	28
16.6.	L'attribution des droits.....	29
16.7.	Le journal des erreurs.....	29

1. La base de données

1.1. La gestion des membres et des administrateurs

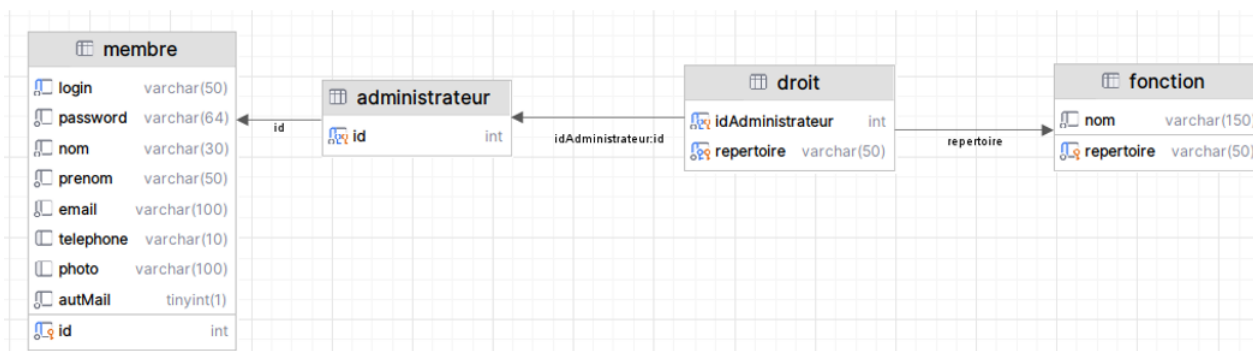
Table	Membre
Rôle	Contient les coordonnées sur les membres.
Champs	id , login, password, nom, prenom, email, telephone, photo, autMail
Remarque	index unique sur nom, prenom index unique sur login fixe, mobile, photo sont optionnels. Ils pourront être renseignés par le membre autMail permet de savoir si le membre autorise l'affichage de son email dans l'annuaire des membre

Parmi les membres se trouve des administrateurs qui peuvent réaliser un certain nombre de tâches administratives.

Table	administrateur(id)
Rôle	Contient les id des membres définis comme administrateur
Liaison	Avec la table membre : suppression et modification en cascade

Table	droit(idAdministrateur , repertoire)
Rôle	Associe les fonctions à un administrateur
Liaison	Avec la table administrateur et la table Fonction : suppression et modification en cascade pour les 2

Table	fonction(repertoire nom)
Rôle	Contient les fonctions d'administration
Liaison	Avec la table droit : suppression en cascade



1.2. Les données informatives

Table	page(id , nom, contenu)
Rôle	Contient les textes affichés dans les pages modifiables du site au nombre de 4 : Présentation club, Présentation 4 saisons, Mentions légales et Politique de confidentialité
Remarque	L'ajout n'est pas possible depuis le site

1.3. Les épreuves des 4 saisons

Table	cpreuve(saison description, date)
Rôle	Contient les informations sur les 4 éditions des 4 saisons
Remarque	Seule la modification des informations est possible.

1.1. Les classements

Table	classement(id , date, titre, fichier, nb)
Rôle	Contient les informations sur les derniers classements disponibles sur le site
Remarque	La colonne 'fichier' contient le nom du fichier pdf contenant le classement. Les fichiers PDF sont stockés dans le répertoire data/classement La colonne 'nb' comptabilise le nombre de téléchargement du fichier. Pour cela le téléchargement passe par le script afficherclassement.php.

2. Source et structure de l'application

Répertoire	Contenu
.admin	restaurer.php : met à jour le dépôt local à partir du dépôt Github sauvegarder.php : met à jour le dépôt Github à partir du dépôt local
.config	database.php : Contient les paramètres de connexion à la base de données L'accès direct est interdit à l'aide d'un fichier .htaccess
.sql	Scripts pour la mise en place de la base de données ppe
4saisons	Module affichant la page de présentation des 4 saisons (un enregistrement de la table page, colonne contenu)
administration	Module affichant les fonctions d'administration accessibles par le membre connecté Ce répertoire contient un sous-répertoire pour chaque fonction d'administration déjà réalisée : classement : ajout, modification, suppression et remplacement du fichier pdf epreuve : modification membre : liste et ajout mention et politique : modification photoinformation : ajout ou suppression de photo qui pourront être utilisées dans le site
ajax	Scripts PHP standards permettant l'ajout, la modification et la suppression dans une table qui possède une clé primaire non composée. Ces scripts sont appelés depuis un appel côté client réalisé en Ajax
allure	Module affichant la vitesse et l'allure en fonction d'un temps et d'une distance.
backoffice	Module permettant l'administration technique du site. Ce répertoire contient un sous répertoire pour chaque fonctionnalité administrative technique : administrateur : ajout ou suppression d'un administrateur fonction : ajout ou suppression d'une fonction d'administration droit : attribution des droits aux administrateurs journal : affichage des journaux (ici le journal des erreurs) Ce module dispose de son propre répertoire include pour l'instant seule la gestion des droits est réalisée/ Son accès doit être protégé par un fichier .htaccess associé à un fichier .htpassword

categorie	Module affichant les catégories sportives avec possibilité de générer côté client un fichier PDF en utilisant le composant javascript html2pdf Les données sont stockées 'en dur' dans le code : Cf. classe Categorie
classemetier	Contient les scripts PHP des classes réalisant l'interface entre la base de données et l'application. En règle générale, il existe une classe pour chaque table. Lorsque la table contient un champ dont la valeur pointe une ressource externe (document PDF ou image), la classe comporte une constante de type Tableau regroupant l'ensemble des paramètres concernant cette ressource (extension, type mime, taille, etc.)
classetechnique	Classes PHP utilisables dans tous les projets (erreur, std, input et classes dérivées, table, select, fichier, etc.).
club	Page affichant les informations sur le club stockées dans la table page
composant	Fichier source des composants Javascript utilisés afin de pouvoir travailler sans accès internet. Chaque sous répertoire correspond à un composant. Exemple : autocomplete, bootstrap Les composants personnels sont stockés dans le sous répertoire fonction
connexion	Interface de connexion pour les membres de l'association.
css	Contient la feuille de style standard utilisée dans tous les projets
data	Contient les données (document PDF, image) gérées par l'application. Les données sont organisées dans des sous répertoires. Un fichier .htaccess interdit l'accès direct pour les documents. Les photos des membres sont conservées dans le sous répertoire photomembre
erreur	Scripts gérant l'affichage de toutes les erreurs détectées par le serveur Apache (401, 403, 404, 410) et les erreurs détectées par l'application
img	Images utilisées par l'application
include	Scripts PHP à inclure dans toutes les pages : autoload.php : Accès au session, chargement dynamique des classes interface.php : Squelette de chaque page du site avec intégration entête + pied header.php : personnalisation de l'entête : différente si un membre est connecté. footer.php : personnalisation du pied de page comprenant deux fenêtres modales pour l'affichage des mentions légales et de la politique de sécurité.
membre	Ensemble des modules accessibles uniquement aux membres : connexion : assure la connexion (vérification couple login/mot de passe) deconnexion : assure la déconnexion en supprimant la session annuaire : affichage de l'annuaire des membres profil : affichage du profil et modification de sa photo : toutes les opérations de gestion du profil ne sont pas implémentées dans cette version)
mentionslegales	Script pdf.php permettant de générer un fichier PDF à partir du contenu HTML stockée dans la table page concernant les mentions légales
politique	Script pdf.php permettant de générer un fichier PDF à partir du contenu HTML stockée dans la table page concernant la politique de sécurité
/ (racine)	index.php : chargement des données alimentant la page d'accueil index.html : corps HTML de la page d'accueil indes.js : Alimentation de la page d'accueil à partir des données transmises afficherclassement.php : affiche un classement avec comptabilisation de la demande. .htaccess : redirection vers les pages d'erreur et protection des fichiers html

3. La structure de chaque fonctionnalité

Elle se compose de trois fichiers :

- Un fichier PHP qui fait office de contrôleur : il prend en charge le contrôle d'accès, le chargement des données et le transfert de ces données côté client dans des variables Javascript
- Un fichier HTML qui représente la vue statique (l'interface de la fonctionnalité)
- Un fichier .js qui représente la vue dynamique (affichage des données, procédures événementielles etc)

Le fichier PHP respecte la structure suivante :

```
// activation du chargement dynamique des ressources
require $_SERVER['DOCUMENT_ROOT'] . "/include/autoload.php";

// chargement des données
$ligne = Page::getPage(1);
$titreFonction = $ligne['nom'];
$data = json_encode($ligne);

// transmission des données à l'interface
$head = <<<EOD
    <script>
        const data = $data
    </script>
EOD;

// chargement de l'interface
require RACINE . "/include/interface.php";
```

Pour les fonctionnalités ayant besoin de récupérer dynamiquement des données, un sous répertoire ajax contient les scripts PHP réalisant la récupération des données et l'envoi de la réponse dans le format JSON.

Le chargement des données s'effectue toujours par l'appel d'une méthode d'une classe.

Ces classes sont stockées dans le répertoire classemetier.

Il n'y a donc aucune instruction d'accès aux données et donc de requête SQL dans les fichiers PHP d'un module. Ces instructions se trouvent uniquement dans les instructions des méthodes des classes métiers.

L'utilisation de la classe technique Select facilite l'interrogation de la base de données en proposant trois méthodes pour récupérer les données :

- `getRows` : pour récupérer plusieurs enregistrements
- `getRow` : pour récupérer un seul enregistrement
- `getValue` : pour récupérer une valeur

Version	Date	Auteur	Document	Page
2025.1	04/09/2025	Guy Verghote	Étude de l'existant	4 sur 29

La constante RACINE définit dans le script `/include/autoload.php` permet d'accéder aux ressources (classes, scripts) en utilisant un chemin absolu. Cela permet d'appeler une classe de la même façon quelle que soit la position du script appelant dans l'arborescence du site.

En effet il n'est pas possible d'utiliser l'adressage relatif pour accéder à une ressource tel que l'affichage d'un menu lorsque cet appel peut se faire à différents endroits de l'arborescence du site.

Le fichier `autoload.php` active le chargement dynamique des classes ce qui évite au développeur de devoir charger lui-même au début des scripts, les classes dont il a besoin.

Les classes sont ainsi chargées à la construction d'un l'objet (`new`) ou lors du premier appel d'une méthode statique de la classe.

Cela évite le chargement de classes non utilisées ou le double chargement en cas d'inclusion de script

4. La déclaration de type

Les déclarations de types peuvent être ajoutées aux arguments des fonctions, valeurs de retour, et, à partir de PHP 7.4.0, aux propriétés de classes. Cela permet de vérifier que les valeurs transmises ou retournées correspondent bien au type spécifié au moment de l'appel. Dans le cas contraire une `TypeError` est lancée.

Un problème se pose néanmoins lorsqu'un paramètre peut avoir plusieurs types. Dans ce cas depuis la version 8 de PHP il est possible d'utiliser le type `mixed` qui signifie 'tout type de données'

Nouveauté aussi apportée par la version 8.0, la déclaration des types de paramètre et de valeur de retour peut désormais être marquée en tant que nullable en préfixant le nom du type avec un point d'interrogation. Ceci signifie que le type spécifié aussi bien que `null` peuvent être passés comme argument, ou retournés en tant que valeur, respectivement.

Dernière nouveauté, les Union Types qui permettent de déclarer plusieurs types pour les arguments, les types de retour et les propriétés de classe.

```
/**
 * @param string $login
 * @return array|null
 */
static function getMembreByLogin(string $login) : object|bool
```

5. L'accès aux données

Tous les accès aux données de la base de données sont centralisés dans les méthodes des classes métier.

L'accès aux données s'effectue techniquement avec l'API PDO.

Les paramètres de connexion à la base sont stockés dans le fichier `./config/databape.php`

Les méthodes de type consultation instancie un objet `Select` pour assurer la connexion à la base.

Les méthodes de type mise à jour utilise la méthode `Database::getInstance()` qui retourne un objet PDO pour se connecter à la base;

Version	Date	Auteur	Document	Page
2025.1	04/09/2025	Guy Verghote	Étude de l'existant	5 sur 29

6. Les classes techniques

Classe	Rôle
Database	Assure la connexion à la base de données Les paramètres de connexion sont conservés dans le fichier de configuration <code>/.config/database.php</code>
Erreur	Propose 3 méthodes statiques pour traiter les erreurs : envoyerReponse() qui retourne dans le format JSON le message d'erreur et son type. Cette méthode s'utilise dans les scripts appelé en Ajax donc sur les scripts contenus dans les répertoires 'ajax'. afficherReponse() qui va rediriger vers la page d'erreur pour afficher le message d'erreur. Cette méthode s'utilise sur un script appelé directement. traiterErreur() qui va chercher à savoir comment le script est appelé pour appeler <code>envoyerReponse</code> ou <code>afficherReponse</code> . Cette méthode s'utilise dans les scripts qui peuvent utiliser les deux modes d'appel.
Input et classes dérivées	Ensemble des classes permettant de définir des objets représentant une donnée associée à ses règles de vérification (min max, required, pattern). Il existe une classe dérivée pour chaque type de données (texte, entier, date, liste, url, ail). Les objets sont utilisés dans les classes métiers pour définir la structure de la table gérée.
Journal	Permet d'enregistrer des événements dans des fichiers de journalisations. Chaque événement comporte 4 informations : la date et l'heure, la description de l'événement, le script PHP concerné et l'adresse IP du visiteur Par défaut l'application utilise le journal <code>error.log</code> pour conserver une trace des erreurs de type système. Ce fichier est stocké dans <code>/backoffice/.log/error.log</code> Son contenu peut être visualisé dans le module backoffice
Select	Classe permettant de lancer des requêtes SQL de consultation.
Std	Cette classe PHP comprend un ensemble de méthodes statiques offrant des fonctionnalités facilitant le contrôle des données, les conversions ou la récupération des fichiers dans un répertoire. L'application utilise les méthodes suivantes : existe : pour vérifier la transmission des paramètres attendus. emailValide , fixeValide , mobileValide : pour vérifier la validité des formats passwordValide : pour vérifier la conformité du mot de passe par rapport aux règles
Table	Classe abstraite dont la plupart des classes métier dérivent afin de rendre totalement standard les opérations d'ajout, de modification et de suppression sur une table de la base de données possédant une clé primaire simple. Les classes métier vont définir la structure de la table en représentant chaque colonne non calculée par un objet Input afin de lui appliquer automatiquement tous les contrôles de validité. La réalisation d'une opération de mise à jour s'effectue en appelant par un appel Ajax un des scripts du répertoire <code>/ajax/</code> :

7. Les composants utilisés côté client

L'application utilise de nombreux composants standards (Bootstrap, autocomplete, tinyMce, purify, html2pdf) et aussi des composants personnels (menu vertical, menu horizontal, fonction)

Afin d'éviter les éventuels problèmes d'accès aux CDN (Content Delivery Network ou Réseau de distribution de contenu en français), les fichiers de ces composants sont stockés sur le site dans le répertoire /composant

Seul le composant Bootstrap est chargé sur l'ensemble des pages du site (chargement depuis le fichier /include/interface.php)

L'intégration d'un composant spécifique à une page s'effectue dans le fichier index.php de la page.

Par exemple pour utiliser les composants Javascript autocomplete (tarekraafat-autocomplete.js)

```
$head = <<<HTML
<script src="/composant/autocomplete.min.js"></script>
<link rel="stylesheet" href="/composant/autocomplete.css">
...
HTML;
```

Le sous répertoire fonction comporte un ensemble de scripts développés par les étudiants permettant de réaliser des opérations de base. Les principaux sont les suivants :

afficher.js	Ensemble de fonctions permettant d'afficher un message sur l'interface
ajax.js	Ensemble de fonctions assurant un appel Ajax à partir de l'API Fetch.
date.js	Ensemble de fonctions permettant de manipuler des dates dans différents formats.
etape.js	Permet de découper une interface de saisie en plusieurs étapes
format.js	Ensemble de fonctions permettant la mise en forme des données
formulaire.js	Ensemble des fonctions assurant le contrôle des données saisies sur une interface
openclose.js	Mise en place d'un système d'ouverture/fermeture sur des cadres
password.js	Permet d'ajouter au niveau d'un champ de type 'password' la possibilité de visualiser/masquer la valeur saisie
tableau.js	Ce module fournit des fonctionnalités prêtes à l'emploi pour trier dynamiquement un tableau HTML, mélanger des données, et les exporter facilement en CSV ou JSON.
vitesse.js	Ensemble de fonctions permettant le calcul d'une vitesse ou d'une allure

8. Les composants utilisés côté serveur chargés via composer

php-image-resize	permet de redimensionner une image. Il est utilisé dans la classe technique InputFileImg.
mpdf	permet de générer des fichiers PDF à partir de code HTML et CSS. Il est utilisé pour générer une version PDF des mentions légales et de la politique de confidentialité.

9. Le traitement des erreurs

Le traitement des erreurs http (401, 403, 404 et 410) est mise en place à l'aide du fichier .htaccess qui les redirigent vers les scripts personnalisés placés dans le répertoire erreur.

ErrorDocument 404 /erreur/404.php ErrorDocument 403 /erreur/403.php ErrorDocument 401 /erreur/401.php ErrorDocument 410 /erreur/410.php	<div style="background-color: #FFD700; text-align: center; padding: 5px;">Avertissement</div> <div style="text-align: center; padding: 10px;"> <p style="color: red;">La page demandée n'existe pas.</p> <hr/> <p>Revenir à la page d'accueil</p> </div>
--	---

Rappel : L'erreur 410 indique que la ressource demandée a été intentionnellement supprimée du serveur et n'est plus disponible

Le traitement des erreurs détectées par l'application est réalisé par l'intermédiaire de la classe technique Erreur

Si l'erreur doit être affichée (cas d'un appel direct), la méthode `afficherReponse` est utilisée.

Si l'erreur doit être retournée dans un format JSON (cas d'un appel ajax), la méthode `envoyerReponse` est utilisée.

Si le script peut être appelé directement ou par un appel Ajax, la fonction `traiterReponse()` est utilisée.

En cas d'erreur sur une appel direct d'un script PHP, le message d'erreur est stocké dans une variable de session, et une redirection est réalisé vers le script `erreur/index.php`, chargé d'afficher l'erreur et de supprimer cette variable de session.

En cas d'erreur sur un appel Ajax d'un script PHP, la réponse est retournée dans le format JSON

Exemple : Sur l'interface de connexion, le script `ajax/connecter.php` va contrôler la validité du mot de passe transmis :

```
// vérification du mot de passe
if (!Membre::verifierPassword($membre['id'], $password)) {
    Erreur::envoyerReponse('Nom d'utilisateur et/ou mot de passe incorrect.', 'global');
}
```

Réponse retournée :

```
{'error':
  {'global': 'Nom d'utilisateur et/ou mot de passe
            incorrect.'}
}
```

Traitement automatique de la réponse côté client par la fonction `appelAjax()`

Le message de type global est automatiquement affiché en haut de l'interface

The screenshot shows a login interface. At the top, there is a red error message box that says "Nom d'utilisateur et/ou mot de passe incorrect." with a close button (X). Below this, there are two input fields: "Login *" containing the text "test" and "Mot de passe *" which is currently masked with dots. There is an eye icon to toggle password visibility. At the bottom, there is a blue button labeled "Connexion".

10. Le contrôle d'accès

Trois cas sont possibles :

- La fonctionnalité (répertoire) est accessible à tous : aucun contrôle à prévoir
- La fonctionnalité est réservée aux membres : il faut donc être connecté
- La fonctionnalité est réservée aux membres possédant les droits nécessaires sur cette fonctionnalité (ce répertoire)

Certains membres se voient attribuer des fonctions d'administration au niveau du site.

Les fonctions d'administration suivantes sont actuellement mises en place :

classement	permet d'ajouter un classement, d'en modifier son titre, de remplacer le fichier PDF associé en cas d'erreur sur et finalement de le supprimer au bout d'un certain temps
epreuve	permet de modifier les informations sur chacune des 4 épreuves programmées dans l'année) savoir la description, les dates et les url d'inscription
membre	permet de visualiser la liste des membres et d'en ajouter
mention	permet de modifier le contenu des mentions légales
photoinformation	permet d'ajouter et de supprimer des photos qui pourront être utilisées dans les informations. Ces photos sont stockées dans le répertoire /data/phjotoinformation

Techniquement les scripts gérant ces fonctions sont placés dans un sous répertoire de même nom au niveau du répertoire administration.

La vérification de l'accès aux différentes pages du site est prise en charge par les méthodes de la classe `ControleAcces`.

La méthode **`verifierAccesMembre()`** vérifie que le membre est bien connecté.

La méthode **`verifierAccesAdministration()`** vérifie que le membre est connecté, qu'il est bien administrateur et qu'il possède le droit d'accès sur cette fonctionnalité (ce répertoire)

Toutes les pages réservées aux membres doivent donc appeler `verifierAccesMembre()`

Toutes les pages réservées aux membres administrateur doivent donc appeler `verifierAccesAdministration()`

Afin d'éviter cette redondance de code, voir la possibilité d'avoir une faille de sécurité si le programmeur oublie de placer le contrôle d'accès au début de la fonctionnalité, il est souhaitable de centraliser l'appel en plaçant le contrôle d'accès dans le fichier `autoload.php` qui est exécuté au début de chaque page.

Mais cela pose le problème suivant : comment savoir quelle méthode de la classe `ControlerAcces` faut-il appeler ? aucune ? `verifierAccesMembre()` ? `verifierAccesAdministration()` ?

C'est le rôle de la méthode `ControleAcess::verifier()` qui est appelée dans le script `autoload.php`

Toutes les fonctionnalités administratives sont regroupées dans le répertoire `administration` du site

Toutes les fonctionnalités accessibles aux membres sont regroupées dans le répertoire `membre` du site

La méthode `verifier()` va analyser l'url appelé pour en extraire le répertoire et ainsi rediriger vers la bonne méthode de contrôle

Si l'utilisateur n'est pas connecté, il est redirigé vers la page de connexion et l'url demandée est conservée dans une variable de session afin de pouvoir directement sur la page initialement demandée une fois la connexion réussie.

Si le membre connecté ne dispose pas des droits sur la fonctionnalité administrative demandé, il est redirigé vers la page d'erreur : <http://ppe-vds/erreur/>

Avertissement

Vous devez être administrateur pour accéder à cette fonctionnalité

[Revenir à la page d'accueil](#)

Version	Date	Auteur	Document	Page
2025.1	04/09/2025	Guy Verghote	Étude de l'existant	10 sur 29

11. Les classes métiers

La classe Administrateur

getLesAdministrateurs()	Select membre.id, concat(nom, ' ', prenom) as nom from membre join administrateur on membre.id = administrateur.id order by nom;
getLesFonctionsAutorisees(\$idadministrateur)	Select nom, fonction.repertoire from fonction join droit on fonction.repertoire = droit.repertoire where idAdministrateur = :idAdministrateur order by nom;
getLesMembres()	Select id, concat(nom, ' ', prenom) as nom from membre where id not in (select id from administrateur) order by nom;
estUnAdministrateur(\$id)	select 1 from administrateur where id = :id
peutAdministrer(\$id, \$repertoire)	select 1 from droit where idAdministrateur = :id and repertoire = :repertoire;
ajouterAdministrateur(\$id)	insert into administrateur (id) values (:id)
supprimerAdministrateur(\$id)	delete from administrateur where id = :id;
ajouterDroit(\$idAdministrateur, repertoire)	insert into droit (repertoire, idAdministrateur) values (:repertoire, :idAdministrateur)
supprimerDroit(\$idadministrateur, repertoire)	delete from droit where repertoire = :repertoire and idAdministrateur = :idAdministrateur;
supprimerTousLesDroits(\$idAdministrateur)	delete from droit where idAdministrateur = :id;
ajouterTousLesDroits(\$idAdministrateur)	delete from droit where idAdministrateur = :id; insert into droit(idAdministrateur, repertoire) select :id, repertoire from fonction;

La classe Categorie

getAll()	Retourne le tableau des catégories stockée 'en dur' dans la classe
----------	--

La classe Fonction

getAll()	Select repertoire, nom from fonction order by repertoire;
----------	---

La classe Page

Méthode	Rôle ou requête SQL
getMentions()	select contenu from page where id = 5;
modifierMentions(\$contenu)	update page set contenu = :contenu where id = 5

D'autres méthodes 'get' permettent d'accès aux autres enregistrements

La classe Classement

getAll()	SELECT id, titre, date, DATE_FORMAT(date, '%d/%m/%Y') AS dateFr, fichier, nbDemande FROM classement Ajout d'une clé 'present' permettant de savoir si le fichier PDF existe bien.
getLast()	select id, titre, date_format(date, '%d/%m/%y') as datefr from classement where date between curdate() - interval 2 week and curdate() order by date desc;
getById(\$id)	select id, date, date_format(date, '%d/%m/%y') as datefr, fichier, titre from classement where id = :id;
comptabiliserDemande(\$id)	update classement set nbDemande = nbDemande + 1 where id = :id;
getNbDemande()	select fichier as nom, nbDemande as nb from classement order by date desc;
getConfig()	[// Répertoire de stockage 'repertoire' => '/data/classement', // Extensions autorisées pour les fichiers téléversés 'extensions' => ['pdf'], // Types MIME acceptés pour l'upload 'types' => ['application/force-download', 'application/pdf'], // Taille maximale autorisée (en octets) 'maxSize' => 2000 * 1024, // Le fichier est-il obligatoire ? 'require' => true, // Le fichier doit-il être renommé automatiquement si doublon 'rename' => false, // Supprimer les accents dans le nom de fichier 'sansAccent' => true, // Forcer la casse des noms de fichiers 'casse' => 'L', // valeur de l'attribut accept 'accept' => '.pdf', // Label pour l'input de fichier, utilisé dans les formulaires 'label' => 'Fichier PDF (2 Mo max)',];

La classe ControlerAcces

verifier()	Dirige vers la bonne fonction de contrôle en fonction du répertoire appelé et déclenche la gestion du jeton CRCF protégeant l'accès des scripts présents dans les répertoires ajax
verifierAccesAdministration()	Vérifie l'accès sur un module d'administration L'utilisateur doit être connecté, faire partie des administrateurs et posséder le droit sur la fonctionnalité (le répertoire) demandé
verifierAccesMembre()	L'utilisateur doit être connecté
gererJeton()	Utilise la classe Jeton pour créer ou vérifier le jeton CSRF Si le script appelé se trouve dans un répertoire ajax, le jeton est vérifié. Si le répertoire contenant le script appelé contient un sous répertoire ajax, le jeton est créé.

La classe Epreuve

getAll()	Select saison, date, description From epreuve
getProchaineEpreuve()	Select date, description, saison From epreuve where date >= curdate() order by date limit 1;

La classe FichierImage

Cette classe permet de gérer une bibliothèque d'images

getAll()	Fichier::getLesFichiers(self::DIR, self::CONFIG['extensions']);
ajouter()	Ajoute une nouvelle image dans la bibliothèque
supprimer(\$nomFichier)	Supprimer une image de la bibliothèque
getConfig()	['repertoire' => '/data/photoinformation', 'extensions' => ["jpg", "png", "webp", "avif"], 'types' => ["image/pjpeg", "image/jpeg", "x-png", "image/png", "image/webp", "image/avif", "image/heif"], 'maxSize' => 300 * 1024, 'require' => true, 'rename' => true, 'sansAccent' => true, 'accept' => '.jpg, .png, .webp, .avif', 'redimensionner' => true, 'height' => 0, // 0 pour ne pas redimensionner 'width' => 350, 'label' => 'Fichiers jpg, png, webp et avif acceptés (300 Ko max)',];


La classe Membre

getConfig()	Retourne les paramètres de configuration pour la photo du membre ['repertoire' => '/data/photomembre', 'extensions' => ['jpg', 'png'], 'types' => ["image/pjpeg", "image/jpeg", "x-png", "image/png"], 'maxSize' => 200 * 1024, 'require' => false, 'rename' => true, 'sansAccent' => true, 'redimensionner' => true, 'height' => 200, 'width' => 200, 'accept' => '.jpg, .png',]
getAll()	Select id, concat(nom, ' ', prenom) as nomPrenom, nom, prenom, email, ifnull(photo, 'Non renseignée') as photo, login, if(autMail, 'Oui', 'Non') as afficherMail, ifnull(telephone, 'Non renseigné') as telephone, if(SHA2('0000', 256) = password, 'Non', 'Oui') as actif From membre Order by nom, prenom; # actif permet de détecter l'obligation de personnaliser
getLesMembres()	Select nom, prenom, if(autMail, email, 'Non communiqué') as mail, ifnull(telephone, 'Non renseigné') as telephone, ifnull(photo, 'Non renseignée') as photo From membre order by nom, prenom;
getByLogin(\$login)	Select id, login, email, nom, prenom from membre where login = :login;
getById(\$id)	Select id, nom, prenom, email, login, telephone, autMail From membre where id = :id;
getPhoto(\$id)	Select photo from membre where id = :id; ⇒ retourne le chemin et le nom de la photo ou null si elle n'existe pas
modifierPhoto(\$id, \$nomFichier)	update membre set photo = :nomFichier where id = :id
supprimerPhoto(\$id)	update membre set photo = null where id = :id;
verifierPassword(\$id, \$password)	Select password from membre where id = :id;
connexion(\$membre)	Création d'une variable de session
deconnexion()	Suppression de la variable de session

12. La page d'accueil

Scripts: index.php, index.js et index.html

La page d'accueil contient les points d'entrée de chaque fonctionnalité, organisés dans des cadres.


[Bienvenue, Guy](#)
[Mon espace](#)
[Annuaire membre](#)
[Mes fonctions](#)
[Se déconnecter](#)
[Site du VDS](#)

Amicale du Val de Somme – Club de course à pied à Amiens
 Tout ouvrir
Tout fermer

Derniers Résultats ▼

Le Club ▼

Informations utiles ▼

Les 4 saisons ▼

[Mentions légales](#)
[Politique de confidentialité](#)

Les cadres peuvent s'ouvrir ou se fermer en cliquant sur l'icône placée à droite dans l'entête.

Le système est totalement standard. Il repose sur le composant 'composant/fonction/openclose.js'


13. Les pages consultables par tous les visiteurs.

Actuellement, il s'agit essentiellement de pages dont le contenu HTML est sauvegardé dans la table page : Présentation (club), Calendrier, horaire et autres informations à connaître (4 saisons), mentions légales et politique de confidentialité.

Les scripts de ces fonctionnalités sont respectivement stockés dans les répertoires club, 4saisons, mentions et politique.

Le contenu de ses 4 modules est pratiquement identique, seule la méthode de la classe Page, appelé dans le fichier index.php change afin de récupérer le contenu de la page.

Exemple :


[Se connecter](#)
[Les 4 saisons](#)

Les courses (5 et 10 Km) des 4 saisons ont lieu à Amiens parc du Grand Marais **4 fois par an**.
 L'édition d'**hiver** et l'édition d'**automne** ont le **Label régional FFA qualificatif au championnat de France**
 L'**édition d'automne** fait office de finale avec plus de 2000 € de prime distribués et un lot de grande qualité offert à chaque arrivant.

Horaires et distances des courses

09h20 5 km découverte : Épreuve ouverte à partir de la catégorie Minimes, coureurs licenciés ou non licenciés.
10h05 10 km des joggers : Épreuve ouverte à partir de la catégorie Cadets, coureurs licenciés ou non licenciés. Joggers du dimanche, débutants ou confirmés en 45 minutes et plus
 Chronométrage à partir de 44 minutes pour les hommes : **toute performance inférieure à 44' sera ramenée à 44' dans le classement officiel, le temps réel sera pris en compte pour les féminines**
11h20 10 km des as
 Coureurs en moins de 45 minutes. Tous les temps (même supérieurs à 45'00) sont pris en compte. Épreuve ouverte à partir de la catégorie Cadets, coureurs licenciés ou non licenciés.

Version	Date	Auteur	Document	Page
2025.1	04/09/2025	Guy Verghote	Étude de l'existant	15 sur 29

Le cadre 'Information utiles pour tous les coureurs' présente des fonctionnalités consultables par tous les visiteurs

Informations utiles

Traduire son temps de course en vitesse et en allure

Connaitre sa catégorie le jour de la course

Les catégories

Vitesse et allure

Temps (h:mm:ss)

0:43:30

Distance en km

10

Calculer votre vitesse et votre allure

Vitesse (km/h)

13.79

Allure (min/km)

4:21

Ce module utilise les fonctions 'getAllure' et 'getVitesse' du composant /fonction/vitesse.js";
Les scripts du module sont stockés dans le répertoire 'allure'.

Sa catégorie le jour de la course

Date de la course *

04/09/2025

Votre année de naissance :

1969

Votre catégorie le jour de la course

Master 4 (M4)


Distance maximale autorisée

Illimitée

Ce module utilise la classe 'Categorie' pour récupérer les catégories, les fonctions 'getDateCourante' et 'getSaison' du module 'fonction/date.js' pour déterminer la saison.
Les scripts du module sont stockés dans le répertoire 'macategorie'.

Version	Date	Auteur	Document	Page
2025.1	04/09/2025	Guy Verghote	Étude de l'existant	16 sur 29

Les catégories


[Se connecter](#)

Categorie

Télécharger le tableau en PDF

Tableau des catégories pour la saison en cours

Catégorie	Code	Âge entre	Né(e) entre	Distance maximale
Minime	MI	14 - 15 ans	2010 - 2011	5 km
Cadet	CA	16 - 17 ans	2008 - 2009	15 km
Junior	JU	18 - 19 ans	2006 - 2007	25 km
Espoir	ES	20 - 22 ans	2003 - 2005	Illimitée
Senior	SE	23 - 34 ans	1991 - 2002	Illimitée
Master 0	M0	35 - 39 ans	1986 - 1990	Illimitée
Master 1	M1	40 - 44 ans	1981 - 1985	Illimitée
Master 2	M2	45 - 49 ans	1976 - 1980	Illimitée
Master 3	M3	50 - 54 ans	1971 - 1975	Illimitée
Master 4	M4	55 - 59 ans	1966 - 1970	Illimitée
Master 5	M5	60 - 64 ans	1961 - 1965	Illimitée
Master 6	M6	65 - 69 ans	1956 - 1960	Illimitée
Master 7	M7	70 - 74 ans	1951 - 1955	Illimitée
Master 8	M8	75 - 79 ans	1946 - 1950	Illimitée
Master 9	M9	80 - 84 ans	1941 - 1945	Illimitée
Master 10	M10	85 - 89 ans	1936 - 1940	Illimitée
Master 11	M11	90 - 99 ans	1926 - 1935	Illimitée

Mentions légales
Politique de confidentialité

Les données n'évoluant pas, elle sont stockées 'en dur' dans le code de la classe Categorie.
L'intervalle exprimé en année de naissance est calculé car il dépend de la date de changement de catégorie fixée au premier septembre.

Le composant client html2pdf est utilisé pour générer une version PDF du corps de la page.
Les scripts du module sont stockés dans le répertoire 'categorie'.

13.1. La connexion

Initialement chaque membre possède un login composé de la première lettre du prénom suivie du nom en minuscules avec un suffixe en cas de doublon et un mot de passe '0000'.

id	login	password	nom	prenom	email	telephone	photo	autMail
1	admin	9af15b336e6a9619928537df30b2e6a2376569fcf9d7e773ec...	VERGHOTE	GUY	guy.verghote@saint-remi.net	NULL	ac.png	0
2	sbakassi	9af15b336e6a9619928537df30b2e6a2376569fcf9d7e773ec...	BAKASSI	SOULAIMANE	soulaimane.bakassi@saint-remi.net	NULL	bakassi_soulaimane.jpg	0
3	jbernard	9af15b336e6a9619928537df30b2e6a2376569fcf9d7e773ec...	BERNARD	JULIEN	julien.bernard@saint-remi.net	NULL	bernard_julien.jpg	0

<http://ppe-vds/connexion/>

Login *

Mot de passe *

Connexion

L'entête prend l'allure suivante si le membre n'est pas un administrateur :

Bienvenue, Julien
[Mon espace](#)
[Annuaire membre](#)
[Se déconnecter](#)
Site du VDS

Amicale du Val de Somme – Club de course à pied à Amiens
Tout ouvrir
Tout fermer

L'entête prend l'allure suivante pour un administrateur :

Bienvenue, Guy
[Mon espace](#)
[Annuaire membre](#)
[Mes fonctions](#)
[Se déconnecter](#)
Site du VDS

Amicale du Val de Somme – Club de course à pied à Amiens
Tout ouvrir
Tout fermer

14. Les pages consultables uniquement par les membres

Les fonctionnalités réservées aux membres du club sont accessibles depuis les liens affichés dans l'entête une fois le membre connecté.

Rappel : Le contrôle d'accès à ces fonctionnalités est réalisé par l'appel de la méthode `ControleAccess::vérifier()` dans le fichier `autoload.php`.

Si le visiteur n'est pas connecté, il est redirigé vers la page de connexion et l'url qu'il demandait est mémorisée dans une variable de session afin de pouvoir s'y rendre automatiquement après s'être connecté.

Lien 'Annuaire membre' : <http://ppe-vds/membre/annuaire/>

Bienvenue, Guy
[Mon espace](#)
[Annuaire membre](#)
[Mes fonctions](#)
[Se déconnecter](#)
Annuaire des membres

Nombre d'adhérents : **42**

BAKASSI SOULAIMANE
Adresse mail masquée
Téléphone non renseigné

BALDE AISSATOU
Adresse mail masquée
Téléphone non renseigné

BERNARD JULIEN
Adresse mail masquée
Téléphone non renseigné

BOILET KAMERON
Adresse mail masquée
Téléphone non renseigné

BOULARBI MEDDHY
Adresse mail masquée
Téléphone non renseigné

BOULLY ALEXANDRE
Adresse mail masquée
Téléphone non renseigné

CARON ADAM
Adresse mail masquée
Téléphone non renseigné

CAZIN TOM
Adresse mail masquée
Téléphone non renseigné

CHARKAOUI RAYANE
Adresse mail masquée
Téléphone non renseigné

Version	Date	Auteur	Document	Page
2025.1	04/09/2025	Guy Verghote	Étude de l'existant	18 sur 29

Lien 'Mon espace' : <http://ppe-vds/membre/profil/>



 **Profil**  Photo

Nom et prénom

BERNARD JULIEN

Email

julien.bernard@saint-remi.net

La sélection d'une fonctionnalité s'effectue par un menu horizontal : Profil – Photo

Les options du menu horizontal sont stockées dans le fichier /membre/.config/menuhorizontal.json

Un membre peut ajouter ou modifier leur photo : <http://ppe-vds/membre/photo/>



 Profil  **Photo**



Les paramètres à respecter pour la photo sont définies dans la classe Membre et accessibles par la méthode getConfig :

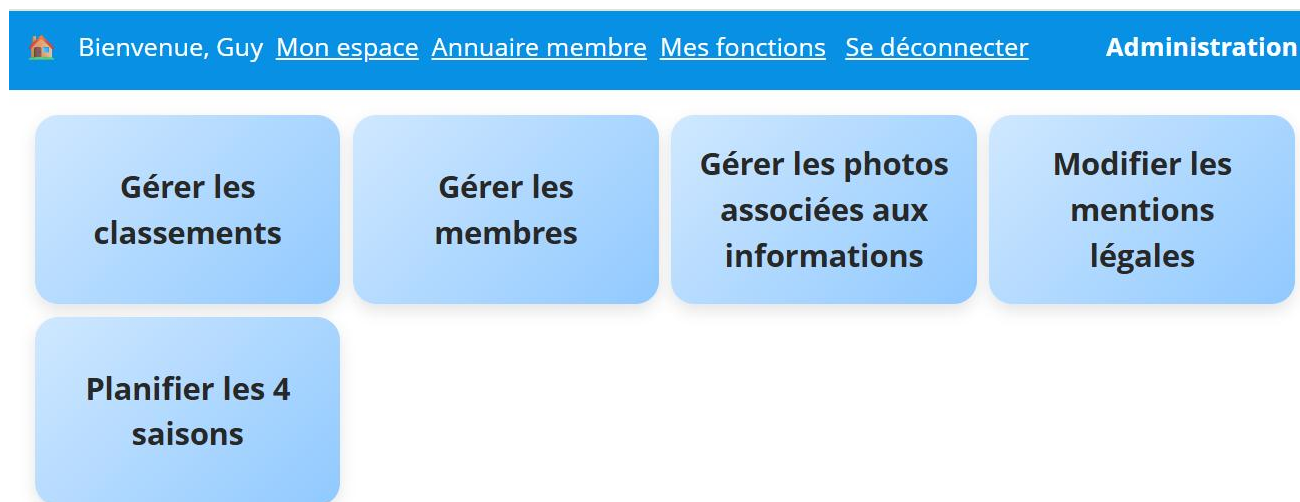
```
'repertoire' => '/data/photomembre',
'extensions' => ['jpg', 'png'],
'types' => ['image/pjpeg', 'image/jpeg', 'x-png', 'image/png'],
'maxSize' => 200 * 1024,
'require' => false,
'rename' => true,
'sansAccent' => true,
'redimensionner' => true,
'height' => 200,
'width' => 200,
'accept' => '.jpg, .png',
```

Elles sont chargées par le fichier index.php de la fonctionnalité afin de les transmettre côté client

Version	Date	Auteur	Document	Page
2025.1	04/09/2025	Guy Verghote	Étude de l'existant	19 sur 29

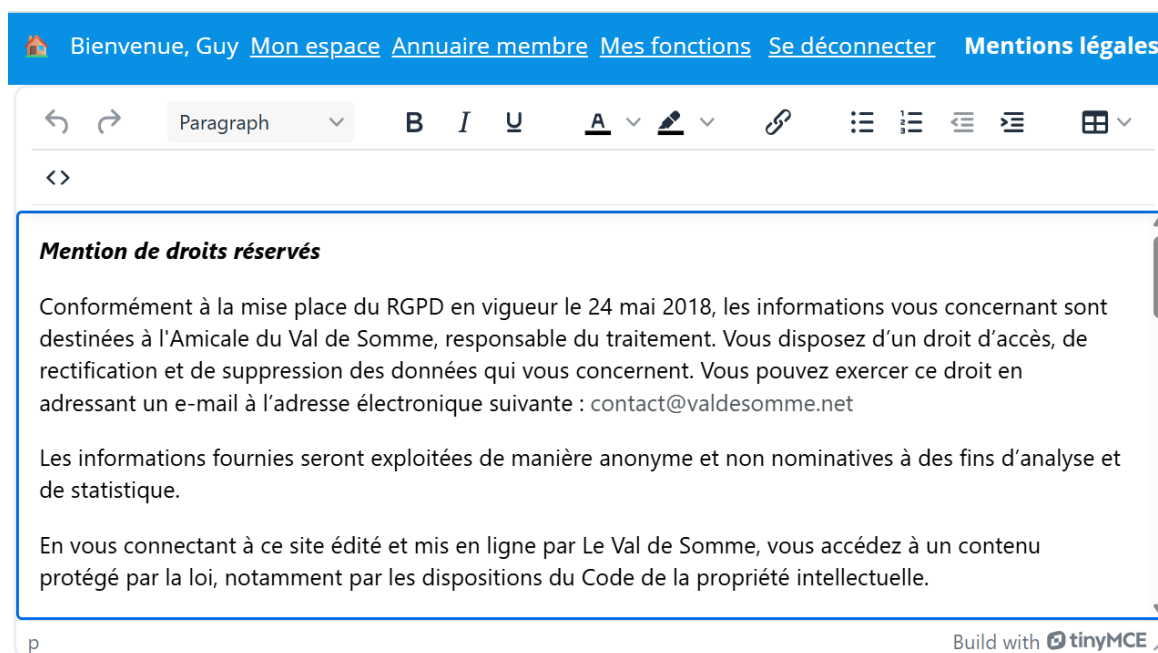
15. Les pages consultables par des administrateurs

Pour un membre connecté disposant de droits d'administration, un lien 'Mes fonctions' apparaît dans l'entête. Ce lien affiche les fonctions d'administration accessible par ce membre



15.1. La modification des mentions légales

<http://ppe-vds/administration/mention/>



[Modifier](#)

Le contenu de la page est géré par le composant tinyMce qui transforme un champ 'textarea' en un mini éditeur de texte.

Les scripts du module sont stockés dans le répertoire 'administration/mention'.

Le module comporte un fichier `ajax/modifier.php` chargé d'appeler la méthode de la classe Page permettant d'enregistrer les modifications.

Version	Date	Auteur	Document	Page
2025.1	04/09/2025	Guy Verghote	Étude de l'existant	20 sur 29

15.2. La gestion des classements

<http://ppe-vds/administration/classement/>

Bienvenue, Guy
 [Mon espace](#)
[Annuaire membre](#)
[Mes fonctions](#)
[Se déconnecter](#)
Gestions des classements

Nombre de classements : 3 Ajouter un classement

	Date		Titre	Nb. Dem.
	27/04/2025		4 saisons printemps 5 km	598
	27/04/2025		4 saisons printemps 10 km populaire	682
	27/04/2025		4 saisons printemps 10 km as	312

Les classements au format PDF sont stockés dans le répertoire /data/classement
 La table classement stocke les informations sur ces différents classements

id	date	titre	fichier	nbDemande
1	2025-04-27	4 saisons printemps 5 km	5 km.pdf	598
2	2025-04-27	4 saisons printemps 10 km populaire	10 km populaire.pdf	682
3	2025-04-27	4 saisons printemps 10 km as	10 km as.pdf	312

L'interface de gestion permet :

- ✓ La consultation du classement
- ✓ Le remplacement du fichier PDF
- ✓ La suppression du classement
- ✓ La modification de la date ou du titre (en mode colonne)
- ✓ L'ajout d'un classement : <http://ppe-vds/administration/classement/ajout.php>

Date de l'épreuve *(La date ne peut excéder le 31/08/2025)* *

30/08/2025

Nom de l'épreuve *(lettres, chiffres, espace de séparation accepté)* *

Les boucles rubempréennes 5 km

Fichier PDF (2 Mo max) *

Choisir un fichier

2025-08-30 Boucles Rubempréenne Pierre Stannard 5km.pdf

Ajouter

15.4. La gestion des membres

<http://ppe-vds/administration/membre/>

Bienvenue, Guy Mon espace Annuaire membre Mes fonctions Se déconnecter Liste des membres					
Ajouter un membre					
Login	Nom et prénom ▲	Mail	Aff.	Photo	Téléphone
sbakassi	BAKASSI SOULAIMANE	soulaimane.bakassi@saint-remi.net	Non	bakassi soulaimane.jpg	Non renseigné
abalde	BALDE AISSATOU	aissatou.balde@saint-remi.net	Non	balde aissatou.jpg	Non renseigné
jbernard	BERNARD JULIEN	julien.bernard@saint-remi.net	Non	bernard julien.jpg	Non renseigné
kboilet	BOILET KAMERON	kameron.boilet@saint-remi.net	Non	boilet kameron.jpg	Non renseigné

La fonction `activerTri` du composant `tableau.js` est utiliser pour permettre le tri du tableau sur la colonne nom et prénom

```
import {activerTri} from "/composant/fonction/tableau.js";

activerTri({
  idTable: "leTableau",
  getData: () => data,
  afficher: afficher,
  triInitial: { colonne: "nomPrenom", sens: "asc" }
});
```

Un administrateur peut simplement ajouter un membre.

La modification du login, du nom et du prénom n'est pas possible ni dans l'application, ni dans la base de données (mise en place par un trigger)

15.5. L'ajout d'un membre

<http://ppe-vds/administration/membre/ajout.php>

Bienvenue, Guy Mon espace Annuaire membre Mes fonctions Se déconnecter Ajouter un membre
--

Nom

Prénom

Adresse e-mail

[+ Ajouter](#)

L'ajout utilise le script standard `/ajax/ajouter.php` pour réaliser automatiquement l'ajout dans la table membre.

Version	Date	Auteur	Document	Page
2025.1	04/09/2025	Guy Verghote	Étude de l'existant	22 sur 29

membre	
login	varchar(50)
password	varchar(64)
nom	varchar(30)
prenom	varchar(50)
email	varchar(100)
telephone	varchar(10)
photo	varchar(100)
autMail	tinyint(1)
id	int

Le déclencheur avant ajout permet d'alimenter automatiquement les champs suivants :

login : première lettre du prénom suivie du nom

password : 0000 (il devra être personnalisé dès la première connexion de l'utilisateur)

Les champs telephone et photo seront éventuellement complétés par le membre par la suite. Si ces champs sont complétés, ils seront affichés dans l'annuaire des membres consultable uniquement par les membres.

Le champ autMail prend sa valeur par défaut : false

Ce champ permet à l'utilisateur d'autoriser l'affichage de son email dans l'annuaire des membres.

15.6. La planification des 4 saisons

<http://ppe-vds/administration/epreuve/>

Le club organise chaque année depuis 1983 4 éditions de sa course 'les 4 saisons', une par saison.

Cette fonctionnalité permet de modifier la description et la planification des 4 éditions

Elle utilise la classe métier Epreuve et le script standard /ajax/modifier.php pour enregistrer les modifications dans la table 'epreuve'

Bienvenue, Guy
 [Mon profil](#)
[Mes fonctions](#)
[Se déconnecter](#)
 Administration/epreuve

Étape 1 / 4
Suivant ▷

Sélectionnez la saison
 Hiver ▼

Étape 2 / 4
< Précédent
Suivant ▷

Épreuve : Hiver
 Description *

Formats ▼
 B
I
U
 A ▼
 ▼

 <>

Label régional FFA qualificatif au championnat de France sur l'ensemble des courses.

Les Courses

- 5 km découverte: 9H20
- 10 km Course des Joggers: 10H05
- 10 km Course des As: 11H20

Le tarif qui vous donne droit le droit de participer à l'ensemble des courses.

- Licenciés FFA : 8 euros
- Non licenciés FFA : 10 euros
- Gratuité pour les membres de l'Amicale du Val de Somme

ul > li
 Build with tinyMCE

Version	Date	Auteur	Document	Page
2025.1	04/09/2025	Guy Verghote	Étude de l'existant	23 sur 29

Étape 3 / 4

< Précédent

Suivant >

Épreuve : Hiver

Date de l'épreuve *

01/03/2026



Étape 4 / 4

< Précédent

Résumé avant modification

Épreuve : Hiver

Description :

Label régional FFA qualificatif au championnat de France sur l'ensemble des courses.

Les Courses

- 5 km découverte: 9H20
- 10 km Course des Joggers: 10H05
- 10 km Course des As: 11H20

Le tarif qui vous donne droit le droit de participer à l'ensemble des courses.

- Licenciés FFA : 8 euros
- Non licenciés FFA : 10 euros
- Gratuité pour les membres de l'Amicale du Val de Somme

Date de l'épreuve : 2026-03-01

Confirmer la modification

Le composant /composant/fonction/etape.js est utilisé pour découper l'interface de saisie en plusieurs étapes avec une étape finale résumant l'ensemble des données saisies

```
import {initialiserEtapes} from "/composant/fonction/etape.js";

// initialisation des étapes avec fonction de rappel sur la dernière étape
initialiserEtapes(majResume);
```

Le composant externe tinymce est utilisé pour saisir la description de l'épreuve

```
// transmission des données à l'interface
$head = <<<HTML
  <script src="/composant/tinymce/tinymce.min.js" referrerpolicy="origin"></script>
<script>
  const lesEpreuves = $data;
</script>
HTML;
```

Version	Date	Auteur	Document	Page
2025.1	04/09/2025	Guy Verghote	Étude de l'existant	24 sur 29

15.7. La photothèque

Ce module permet simplement l'ajout ou la suppression d'image qui serviront à alimenter les informations publiées sur le site.



Il utilise la classe métier FichierImage qui définit les paramètres des fichiers acceptés

```
'repertoire' => '/data/photoinformation',
'extensions' => ["jpg", "png", "webp", "avif"],
'types' => ["image/jpeg", "image/png", "image/webp", "image/avif",
"image/heif"],
'maxSize' => 300 * 1024,
'require' => true,
'rename' => true,
'sansAccent' => true,
'accept' => '.jpg, .png, .webp, .avif',
'redimensionner' => true,
'height' => 0, // 0 pour ne pas redimensionner
'width' => 350,
'label' => 'Fichiers jpg, png, webp et avif acceptés (300 Ko max)',
```

Version	Date	Auteur	Document	Page
2025.1	04/09/2025	Guy Verghote	Étude de l'existant	25 sur 29

16. Le répertoire backoffice

Le répertoire backoffice contient tous les modules d'administration du site accessible uniquement par l'administrateur du site.

Ce répertoire contient les répertoires suivants :

administrateur	Permet l'ajout d'un membre dans les administrateurs ou la suppression d'un membre des administrateurs
droit	Permet l'ajout ou la suppression de droits aux administrateurs
fonction	Permet l'ajout d'une fonction d'administration ou éventuellement sa suppression
journal	Permet la consultation u fichier de journalisation erreur.log
.log	Contient le fichier de journalisation erreur.log
.config	Contient le fichier menuvertical.json qui définit les options du menu vertical associé au backoffice
include	Contient

Le répertoire backoffice comporte son propre répertoire include contenant les fichiers suivants :

Le fichier interface.php intègre le chargement du menu vertical et ne comporte pas d'entête de page n i de pied de page puisque cela est incompatible avec un menu vertical.

16.1. Contrôle d'accès sur le répertoire backoffice

L'accès à ce répertoire doit être protégé par un fichier .htaccess associé à un fichier .htpasswd afin de n'autoriser l'accès qu'à l'administrateur

Documentation en ligne : Développement Web > Protection des accès > Protection de l'accès par un fichier htaccess

Contenu du fichier .htaccess :

```
AuthUserFile j:/ppe/backoffice/.htpasswd
AuthName "Accès administration"
AuthType Basic
require user admin

# protection du fichier .htpasswd
<Files ".htpasswd">
    Require all denied
</Files>
```

Le fichier .htpasswd contient la liste des utilisateurs autorisés, ici un seul.

```
admin :admin
```

Nous sommes sur un serveur Windows, nous pouvons donc laisser les mots de passe en clair et créer ce fichier à l'aide d'un simple éditeur de texte.

Version	Date	Auteur	Document	Page
2025.1	04/09/2025	Guy Verghote	Étude de l'existant	26 sur 29

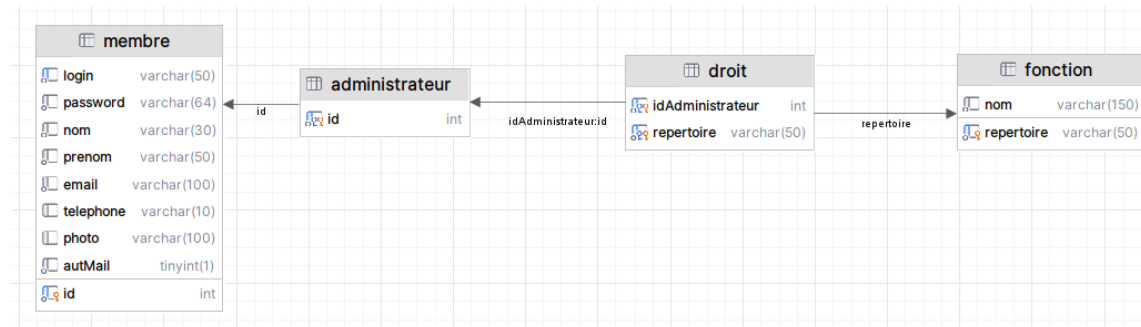
16.2. Principe de la gestion des droits des administrateurs

L'ensemble des informations concernant les membres de l'association sont stockées dans la table membre.

Parmi ces membres certains peuvent être autorisés à réaliser certaines fonctions administratives contenu dans la table fonction.

Les membres 'administrateur' sont stockés dans la table administrateur.

Les droits des administrateurs sont stockés dans la table droit.

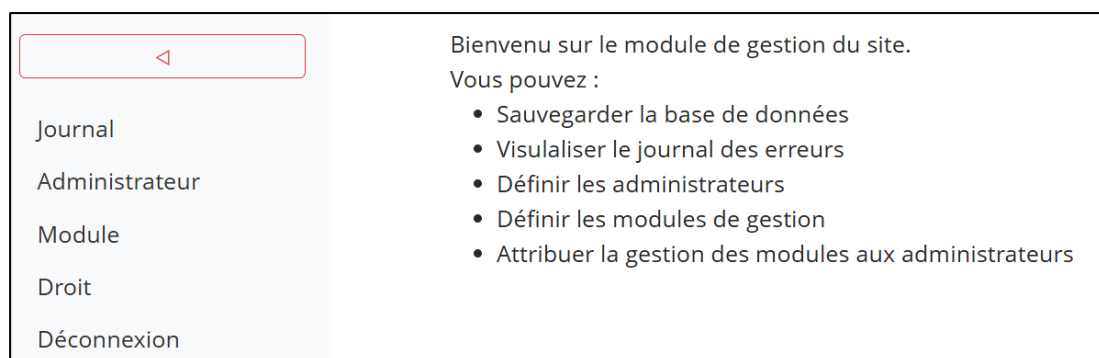


Au niveau physique, une fonction correspond à un répertoire.

Par exemple, les fichiers permettant de gérer les pages statiques se trouvent dans répertoire page.

Tous ces répertoires sont placés dans le répertoire administration du site.

16.3. L'interface d'accueil



Elle utilise le composant menuvertical pour proposer l'accès à l'ensemble des fonctionnalités

Les options du menu sont stockées dans le fichier /backoffice/.config/menuvertical.json

L'intégration de menu s'effectue au niveau du fichier include/interface.php

```

<?php
// Récupération des options du menu vertical
$lesOptions = file_get_contents(RACINE . '/backoffice/.config/menuvertical.json');
?>
<!DOCTYPE HTML>
...
<script type='module'>
  import {initialiserMenuVertical} from "/composant/menuvertical/menu.js";
  initialiserMenuVertical(<?=$lesOptions;?>, 230);
</script>
...
</html>
  
```

16.4. La gestion des administrateurs

<http://ppe-vds/backoffice/administrateur/>

Gestion des administrateurs

Nom : + Ajouter

Liste des administrateurs

X	VERGHOTE GUY
---	--------------

L'utilisateur sélectionne un membre dans la liste puis clique sur le bouton ajouter pour en faire un administrateur.

Les méthodes de la classe Administrateur sont utilisées pour alimenter la zone de liste et ajouter ou supprimer un administrateur.

Le script `/backoffice/administrateur/ajax/ajouter.php` permet l'ajout d'un administrateur.

Le script `/backoffice/administrateur/ajax/supprimer.php` permet la suppression d'un administrateur.

16.5. La gestion des fonctions d'administration

<http://ppe-vds/backoffice/fonction/>

Gestion des fonctions d'administration

Répertoire *

Nom *

+ Ajouter

Liste des fonctions d'administration

	Répertoire	Nom
X	classement	Gérer les classements
X	epreuve	Planifier les 4 saisons
X	membre	Gérer les membres
X	mention	Modifier les mentions légales
X	photoinformation	Gérer les photos associées aux informations
X	politique	Modifier la politique de confidentialité

La mise à jour dans la base de données est déclenchée immédiatement (clic sur la case).

Version	Date	Auteur	Document	Page
2025.1	04/09/2025	Guy Verghote	Étude de l'existant	28 sur 29

16.6. L'attribution des droits



Attribution des droits d'administration

Administrateur : VERGHOTE GUY

Retirer toutLes fonctions disponiblesAjouter tout

- ☒ Gérer les classements
- ☒ Planifier les 4 saisons
- ☒ Gérer les membres
- ☒ Modifier les mentions légales
- ☒ Gérer les photos associées aux informations
- ☒ Modifier la politique de confidentialité

16.7. Le journal des erreurs

<http://ppe-vds/backoffice/journal/>

Toutes les erreurs de type 'system' sont conservées dans le fichier `/backoffice/.log/erreur.log`



Le journal des erreurs

 Vider le journal

```
30/05/2025 11:08:01
insert into classement set date = :date, titre = :titre, fichier =
:fichier : SQLSTATE[45000]: <<Unknown error>>: 1644 L'URI du
fichier PDF n'est pas valide :la bibliotheque fonction.js.pdf
/ajax/ajouter.php
::1
```

Le fichier `index.php` appelle méthode `getLesEvenement('erreur')` de la classe `Journal` afin de récupérer et transmettre côté client le contenu du fichier `erreur.log`

Le fichier `ajax/effacer.php` appelle la méthode `Journal::supprimer('erreur')` afin de supprimer le fichier `erreur.log`.

A noter : la classe 'Journal' est programmée pour gérer plusieurs fichiers de journalisation.

Version	Date	Auteur	Document	Page
2025.1	04/09/2025	Guy Verghote	Étude de l'existant	29 sur 29