

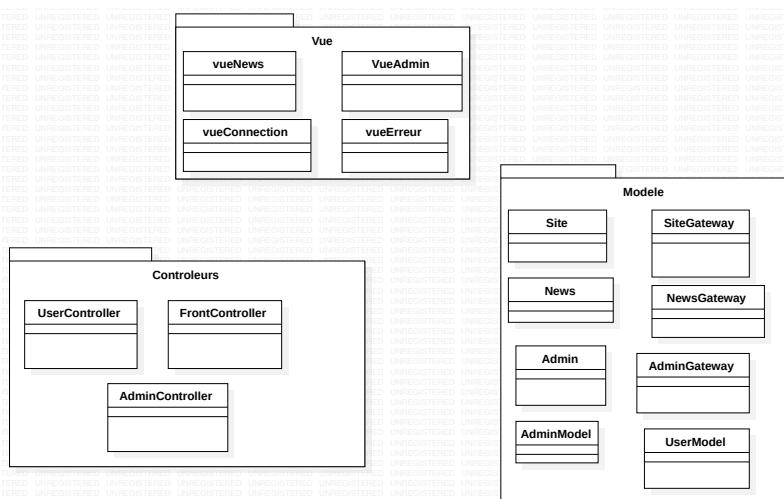
Compte-Rendu projet PHP : Site de news par lecture de flux RSS

1. Présentation du projet
2. Architecture
 - 2.1. Organisation des fichiers
 - 2.1.1. Config
 - 2.1.2. Vues/CSS
 - 2.1.3 Contrôleurs
 - 2.1.4. Modèles
 - 2.1.5 Index
 - 2.2. Organisation de la base
 - 2.3. Plan d'exécution de requêtes
3. Réponses aux attentes du module
4. Les bonus

1. Présentation du projet

Ce projet est un site web qui récapitule l'actualité de plusieurs sites de news par lecture de flux RSS. Il permet à n'importe quel visiteur d'avoir accès au contenu.

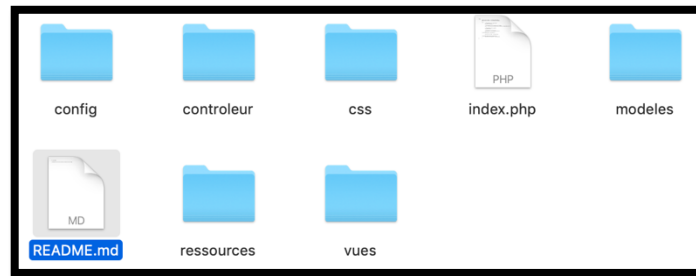
Les utilisateurs connectés en tant qu'administrateurs peuvent modifier les sites référencés.



Insérer le diagramme de cas
d'utilisation

2. Architecture

2.1. Organisation des fichiers



L'architecture suit le pattern «MVC »

2.1.1. Config

Nom du dossier : config

Rôle : Contient les fichiers utiles pour le reste du projet tel que l'initialisation de variables globales, la validation de données...

Contenu :



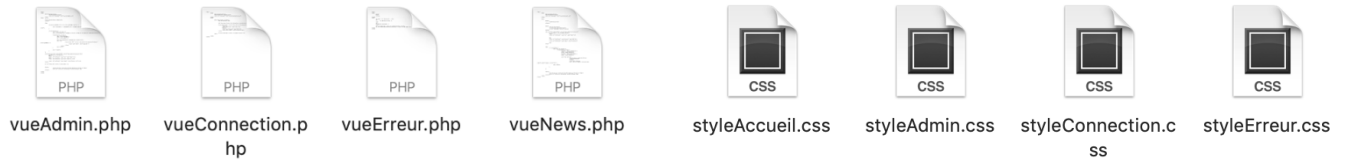
Nom	Rôle	Stratégie
Autoload.php	Permet de charger automatiquement les classes lors de l'initialisation d'un objet sans avoir à passer par un require() .	Réécrit la méthode : spl_autoload_register() Lors de l'initialisation d'une instance de classe inconnue, cette méthode est appelée. Elle : <ul style="list-style-type: none">• parcourt toute l'arborescence de fichier jusqu'à trouver le fichier ayant le même nom.• Elle effectue dès lors un require() de ce fichier.
Config.php	Initialise des variables globales utiles pour le reste du projet.	Contient : <ul style="list-style-type: none">• les données de connexion à la base• les chemins d'accès vers les vues (dans un dictionnaire \$vues)• Les variables contenant les données que les vues doivent afficher (\$dVue, \$dErreur)
Connection.php	Classe encapsulant les méthodes d'accès à la base de données pour fournir un unique accès à celle-ci.	Hérite de la classe PDO, permet l'exécution des requêtes en base et la récupération des résultats.
Validation.php	Permet d'effectuer les filtrages et nettoyages de données récupérées par des variables \$_REQUEST et autre. Permet d'éviter notamment les injections de code.	Utilisation entre autre de filter_var() pour : <ul style="list-style-type: none">• la confirmation de données de connexion en admin• des actions• Et autres (trie des news)
XMLParser.php	Classe permettant la récupération des news des sites référencés grâce à la méthode simplexml_load_file() .	<ul style="list-style-type: none">• Parcours tous les url de la table TSites• Récupère le flux RSS en XML• Le parse• Retourne les derniers articles.

2.1.2 Vues/CSS

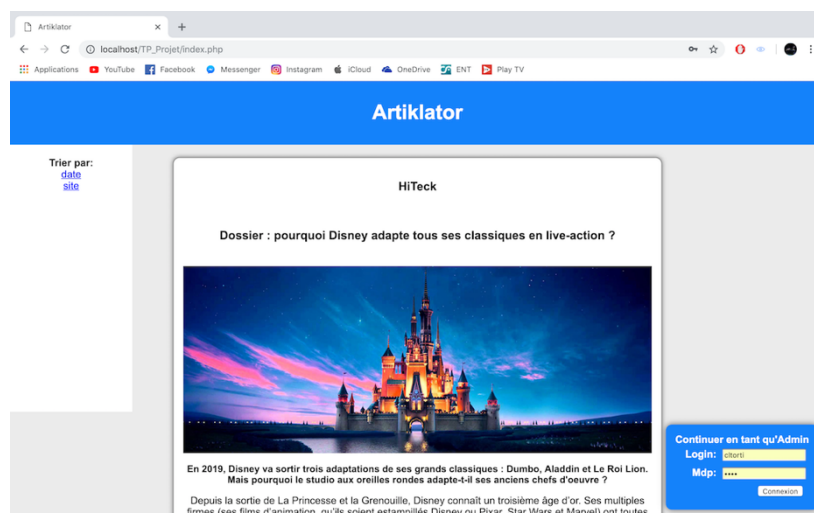
Nom des dossiers : vues et css

Rôle : Contient les pages html qui affichent les informations aux utilisateurs et administrateurs.

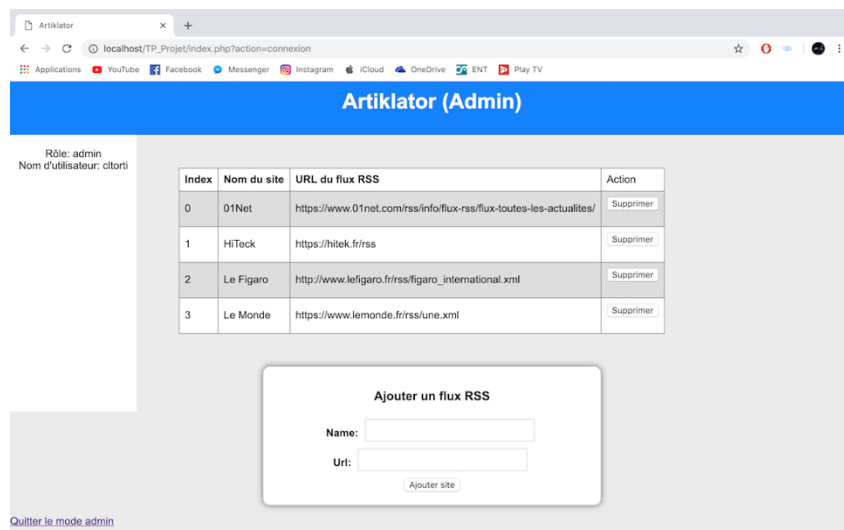
Contenu :



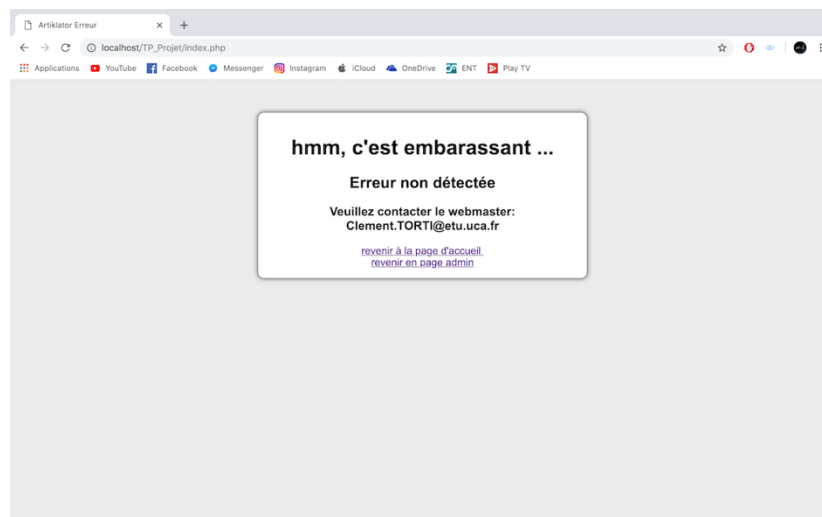
Nom	Contenu
vueNews.php + styleAccueil.css	Première page affichée, contient : <ul style="list-style-type: none"> • Les articles les plus récents • Un formulaire de connexion • Des boutons de tri des articles par date ou site
vueAdmin.php + styleAdmin.css	Page par défaut pour les administrateurs connectés, elle contient : <ul style="list-style-type: none"> • Un tableau des sites référencées • Un formulaire d'ajout de flux RSS • Un bouton de redirection vers la page d'accueil
vueErreur.php + styleErreur.css	Page qui s'affiche en cas d'erreur, en cas d'injection de code par exemple. Elle contient : <ul style="list-style-type: none"> • Le titre de l'erreur • La description de l'erreur • Les boutons de redirection
vueConnection.php + styleConnection.css	Page qui s'affiche lorsque l'on tente d'effectuer une action administrateur sans être connecté. Elle contient : <ul style="list-style-type: none"> • Le formulaire de connexion • Les boutons de redirection



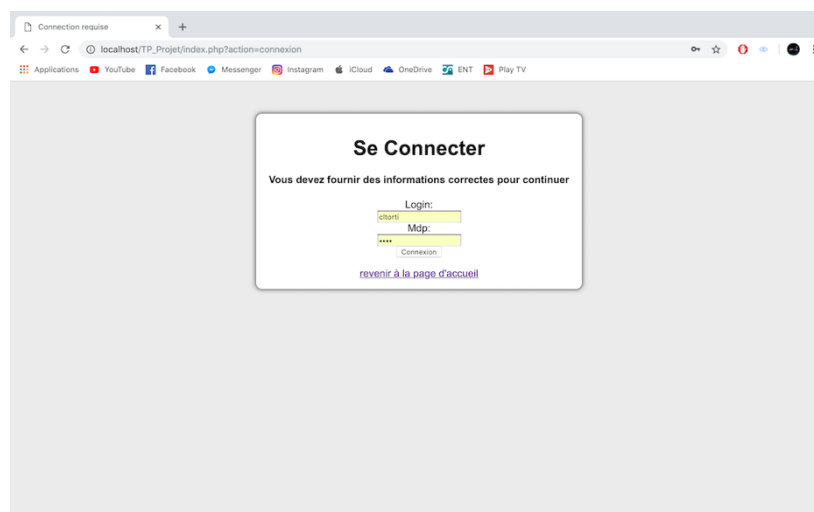
Page de News



Page Admin



Page erreur



Page de connexion

2.1.3. Contrôleurs

Nom du dossier : controleur

Rôle : Coordinateur des fichiers du projet, permet la récupération d'une action, son traitement et l'affichage de la vue adaptée.

Contenu :



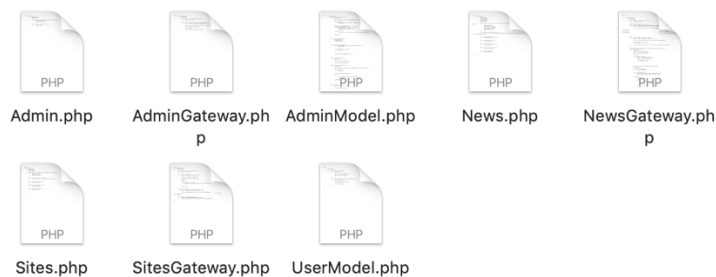
Nom	Rôle	Stratégie
FrontController.php	Reçoit toutes les actions la redirige vers le controleur adapté.	<ul style="list-style-type: none">• Récupère l'action avec <code>\$_REQUEST</code>• Si c'est une action admin, vérifie que l'utilisateur est connecté et le transmet à AdminController.php sinon renvoi vers la page de connexion.• Sinon, transmet l'action à UserController.php
UserController.php	Permet l'affichage de news et la connexion en mode admin.	<ul style="list-style-type: none">• Lit l'action• Effectue la méthode adaptée.
AdminController.php	Permet l'ajout/suppression de site et la déconnexion.	

2.1.4. Modèles

Nom du dossier : modeles

Rôle : Permet de gérer les actions utilisateur et admin, communiquer avec la base de données et formater les résultats grâce aux classes métier.

Contenu :



Nom	Rôle	Stratégie
UserModel.php AdminModel.php	Énumère les actions possible d'un acteur. Fait la jonction entre le controleur et les Gateway.	<ul style="list-style-type: none"> Vérifie les données. Appel les Gateway pour accéder à la base. Récupère le résultat et le retourne au controleur.
NewsGateway.php AdminGateway.php SitesGateway.php	Permet un accès contrôlé à la base.	<ul style="list-style-type: none"> Génère les requêtes. Fait appel à la classe Connection. Récupère le résultat. Le formate à l'aide des classes métier et le retourne.
News.php Admin.php Site.php	Classe conteneur de données venant de la base.	

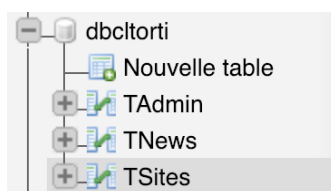
2.1.5 Index




Nom : index.php

Rôle : Fichier appelé par défaut.

Stratégie : Charge le fichier config, lance l'Autoloader, établie la connexion à la base, démarre la session et instancie le frontController.

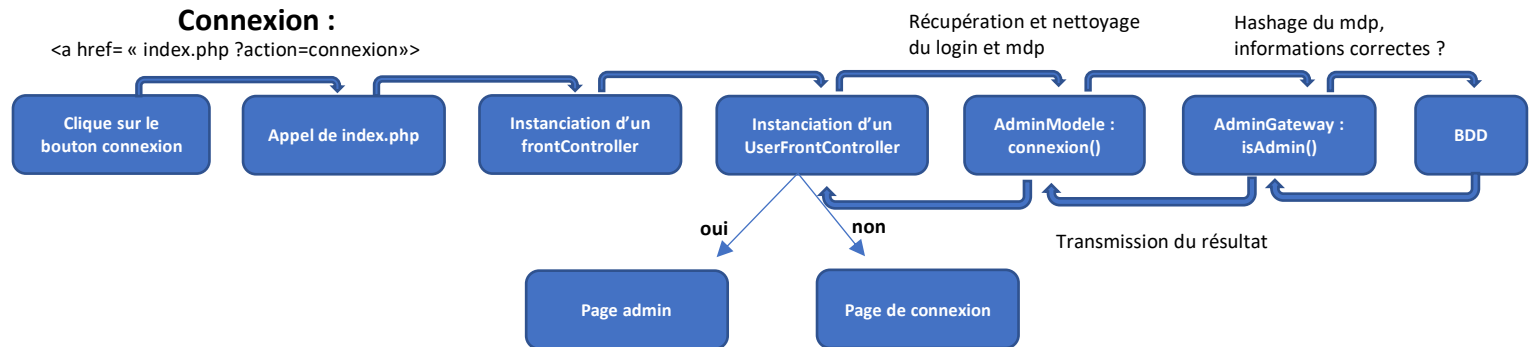
2.2 Organisation de la base



TSites	TAdmin	TNews																																							
<table> <tr> <th>#</th><th>Nom</th><th>Type</th></tr> <tr> <td><input type="checkbox"/> 1</td><td>url</td><td>varchar(500)</td></tr> <tr> <td><input type="checkbox"/> 2</td><td>name </td><td>varchar(150)</td></tr> </table> <p>Contient les sites dont on lit le flux RSS. url : URL du flux RSS name : Nom du site</p>	#	Nom	Type	<input type="checkbox"/> 1	url	varchar(500)	<input type="checkbox"/> 2	name 	varchar(150)	<table> <tr> <th>#</th><th>Nom</th><th>Type</th></tr> <tr> <td><input type="checkbox"/> 1</td><td>id</td><td>int(11)</td></tr> <tr> <td><input type="checkbox"/> 2</td><td>login</td><td>varchar(200)</td></tr> <tr> <td><input type="checkbox"/> 3</td><td>mdp</td><td>varchar(200)</td></tr> </table> <p>Contient les utilisateurs administrateur. Id : Unique pour chaque admin. Login : Nom d'administrateur Mdp : Mot de passe.</p>	#	Nom	Type	<input type="checkbox"/> 1	id	int(11)	<input type="checkbox"/> 2	login	varchar(200)	<input type="checkbox"/> 3	mdp	varchar(200)	<table> <tr> <th>#</th><th>Nom</th><th>Type</th></tr> <tr> <td><input type="checkbox"/> 1</td><td>id</td><td>int(11)</td></tr> <tr> <td><input type="checkbox"/> 2</td><td>title</td><td>varchar(150)</td></tr> <tr> <td><input type="checkbox"/> 3</td><td>content</td><td>varchar(500)</td></tr> <tr> <td><input type="checkbox"/> 4</td><td>website</td><td>varchar(150)</td></tr> <tr> <td><input type="checkbox"/> 5</td><td>time</td><td>date</td></tr> </table> <p>Attention : Nous ne conservons pas les news en base dans la dernière version. Ceux-ci sont générés par le ParserXML et non écrit en dur.</p>	#	Nom	Type	<input type="checkbox"/> 1	id	int(11)	<input type="checkbox"/> 2	title	varchar(150)	<input type="checkbox"/> 3	content	varchar(500)	<input type="checkbox"/> 4	website	varchar(150)	<input type="checkbox"/> 5	time	date
#	Nom	Type																																							
<input type="checkbox"/> 1	url	varchar(500)																																							
<input type="checkbox"/> 2	name 	varchar(150)																																							
#	Nom	Type																																							
<input type="checkbox"/> 1	id	int(11)																																							
<input type="checkbox"/> 2	login	varchar(200)																																							
<input type="checkbox"/> 3	mdp	varchar(200)																																							
#	Nom	Type																																							
<input type="checkbox"/> 1	id	int(11)																																							
<input type="checkbox"/> 2	title	varchar(150)																																							
<input type="checkbox"/> 3	content	varchar(500)																																							
<input type="checkbox"/> 4	website	varchar(150)																																							
<input type="checkbox"/> 5	time	date																																							

2.3. Plan d'exécution des requêtes

Connexion :



Résumé : Toutes les autres requêtes ont un plan d'exécution similaire.

La vue déclenche une action, le `frontController` redirige vers le contrôleur adéquat qui appelle le modèle.

Le modèle appelle à son tour la Gateway qui communique avec la base de données.

Les résultats remontent jusqu'au contrôleur qui met à jour la vue en fonction de ceux-ci.

3. Réponses aux attentes du module

- Organisation en MVC et utilisation du `FrontController`
- Utilisation des cookies (Type de trie des articles)
- Utilisation de la session (Login et rôle admin)
- Autoloader
- Validation et nettoyage des données systématique (données de connexion, ajout de flux RSS)
- Vérification du rôle à chaque action admin
- Classe `Connection` qui étend `PDO` et `try {} catch()` des exceptions `PDO`

4. Les bonus

- Lecture du flux RSS avec la méthode `simplexml_load_file()` et parsage des données.
- Hashage des mots de passe en base pour plus de sécurité (méthodes `password_hash()` et `password_verify()`)
- Trier les articles par date ou site en sauvegardant la préférence à l'aide d'un cookie.