

Cahier technique

Table des matières

[CHAPITRE 1 : PRESENTATION GENERALE ET Principes de fonctionnement 4](#_Toc116309184)

[I. Conception 4](#_Toc116309185)

[II. Choix des technologies : 4](#_Toc116309186)

[III. Environnement technique : 5](#_Toc116309187)

[CHAPITRE 5 - RÉALISATIONS ET EXTRAITS DE CODES 7](#_Toc116309188)

[I. Maquettage et wireframe : 7](#_Toc116309189)

[II. Design des pages et composants : 8](#_Toc116309190)

[III. Graphe de dialogue 9](#_Toc116309191)

[IV. Dictionnaire de donnéés et arborescence 11](#_Toc116309192)

[V. MCD de l’application 12](#_Toc116309193)

[VI. MPD de l’application 16](#_Toc116309194)

[VII. Extraits de code js 17](#_Toc116309195)

[A. Code qui gère l’animation de la sidebar. 17](#_Toc116309196)

[B. Code qui gère le modal : 17](#_Toc116309197)

[C. Code qui gère la libraire chart.js : 17](#_Toc116309198)

[VIII. Extraits de code PHP 19](#_Toc116309199)

[A. Classe Autoload : 19](#_Toc116309200)

[B. Extrait de code de la classe Controller : 20](#_Toc116309201)

[C. Extrait de code de la classe Model : 21](#_Toc116309202)

[D. Extrait de la classe Article, fille de la classe Model  et présentation des transactions par PDO 21](#_Toc116309203)

[E. Extrait de code de la classe model statistique montrant les appels de procédures : 22](#_Toc116309204)

[F. Extrait du code de la view Layout : 22](#_Toc116309205)

[G. Méthode qui génère les notifications 23](#_Toc116309206)

[H. Extrait de code de la classe Admin : 23](#_Toc116309207)

[I. Description de la méthode principale qui gère les permissions dans la classe Security 25](#_Toc116309208)

[CHAPITRE 2 – PRÉSENTATION DES FONCTIONNALITÉS 27](#_Toc116309209)

[I. Gestion des permissions : 27](#_Toc116309210)

[II. Nouvelle sortie de marchandises 27](#_Toc116309211)

[CHAPITRE 3 – FONCTIONNEMENT ET DEBOGAGE 28](#_Toc116309212)

[I. Probleme de routage 28](#_Toc116309213)

[II. Débogage de routage 28](#_Toc116309214)

[A. Etape 1 : vérifier le fonctionnement du rewrite engine d’Apache 28](#_Toc116309215)

[B.Etape 2 : vérifier que le routeur PHP fonctionne correctement 28](#_Toc116309216)

[III. GESTION DES FEUILLES DE STYLES 29](#_Toc116309217)

[IV. Namespace et reference de fichier 29](#_Toc116309218)

[V. AUTOLOADER 29](#_Toc116309219)

[VI. mode debogage SQL 30](#_Toc116309220)

[VII. configuration de l’IMPORT DES IMAGES 30](#_Toc116309221)

[VIII. GESTION AUTOMATIQUE DU http ET HTTPS 30](#_Toc116309222)

[IX. creation et modification des pdf 30](#_Toc116309223)

[X. creation et modification des graphiques 31](#_Toc116309224)

[XI. personnalisation des urls 31](#_Toc116309225)

[XII. filtre des valeurs posts & session 31](#_Toc116309226)

[XIII. protection du htaccess 31](#_Toc116309227)

[XIV. probleme de retour a la page de connexion 32](#_Toc116309228)

[CHAPITRE 4 – TO DO LIST DE SECURITE: ce qui est fait et reste a faire 33](#_Toc116309229)

[XV. Les injections SQL 33](#_Toc116309230)

[XVI. Scripts intersites ou failles xss 33](#_Toc116309231)

[I. La fixation de session 29](#_Toc116309232)

[II. Faille include 29](#_Toc116309233)

[III. Falsification de demandes intersites ou failles csrf 29](#_Toc116309234)

[IV. Attaques par force brute 30](#_Toc116309235)

[V. Cryptage de mots de passe et recommandations de sécurité 30](#_Toc116309236)

[VI. Le détournement ou vol de session 30](#_Toc116309237)

[CHAPITRE 8 – INFORMATIONS ET REMARQUES 31](#_Toc116309238)

[I. Défaut de rewrite-engine sur le serveur linux : 31](#_Toc116309239)

[II. Défaut de rollback de PDO en cas d’échec d’une transaction : 1](#_Toc116309240)

[III. Défaut de permission pour l’utilisation de procédures stockées utilisant du LMD : 1](#_Toc116309241)

[IV. Liste des développeurs : 3](#_Toc116309242)

1. Le lien entre la bdd et php se fait dans la classe model. Renseigner les paramètres correctes de PDO pour le fonctionnement correcte à savoir l’hôte, le nom de la base de l’utilisateur et de son mot de passe.
2. Créer d’abord une bdd du nom de l’aati. Pour l’importation de la bdd, commencer à insérer dans phpmyadmin les requêtes sql de la procédure pour la création des tables, puis celle de la création du jeu d’essai. Exécutez à la suite ces procédures par une requête « call create\_tables1() » puis call jeu\_dessai\_1() pour tester le bon fonctionnement avec un jeu de données.
3. Une fois la bdd créé insérer progressivement l’ensemble des fonctions et procédures et vous pouvez utiliser la procédure de truncate\_table pour tronquer les tables de leurs données. Des fonctions et des procédures sont utilisées dans des requêtes de l’application.
4. Une fois les fonction et les requêtes importées vous pourrez importer les triggers au sein de l’application.

Pour plus de détails sur l’écriture de la création et la manipulation des requêtes bdd et des transactions consulter mysql sh.

Les identifiants de l’administrateur du jeu d’essai sont :

Id : [admin@admin.fr](mailto:admin@admin.fr)

mdp : admin

Si l’ensemble des procédures et fonctions ont bien été importés et exécutés, aucun message d’erreur ne devrait apparaitre sinon se fier aux instructions du message d’erreur.

Si l’ensemble des triggers n’est pas importé des erreurs peuvent survenir en base de données. Les triggers ayant pour rôle d’assurer la cohérence des données métiers de l’application.

# CHAPITRE 1 : PRESENTATION GENERALE ET Principes de fonctionnement

## Conception

**L’architecture : le pattern MVC et l’orienté Objet .**

L’application sous l’architecture MVC orientée objet  utilise deux couches/systèmes de routage indispensables à son fonctionnement. La première couche de routage se fait via apache et son moteur de réécriture d’url. Celui-ci redirige l’ensemble des requêtes utilisateurs tapées dans l’url vers un point d’entrée unique qu’est le fichier index.php à la racine du site.

**Intérêt de l’architecture :** Le MVC est l’architecture moderne pour la conception de sites et d’applications web. Cette architecture présente l’avantage de séparer la partie graphique et le moteur de l’application. Un designer peut ainsi mettre à jour graphiquement sans qu’il ait besoin d’accéder au source de l’application et des données peuvent être rajoutées avec de légers rajouts dans le code. Cette architecture présente l’avantage d’une séparation du code en objets métiers via des classes, on est dans la Programmation Orienté Objet (POO). Cette technique avec les principes d’encapsulation et d’héritage renforcent la sécurité des sites en encapsulant les blocs de code sous des couches avec des niveaux d’accès. L’utilisation de classes rend également le code plus maintenable.

## Choix des technologies :

.

**Serveur SGBD : le choix de MySQL**

Une des fonctionnalités attendues est la persistance des données. Notre solution se tournera sur une solution SQL et un SGBD relationnel. Les plus populaires parmi les hébergeurs sont généralement les solutions gratuites MySQL ou MariaDB.

**Le serveur http : le choix d’apache et le .htaccess**

Le choix a été libre. Il s’est porté sur Apache2 en raison de la solidité et de la popularité de cette solution. La fonctionnalité qui nous intéresse particulièrement est l’utilisation du rewrite engine, le module de réécriture d’url d’Apache. Il permet de protéger l’arborescence des sites en la masquant à l’utilisateur.

*Les autres technologies sont des technologies d’aide pour accélérer le développement comme :*

* *la bibliothèque* ***JQuery*** *qui apporte des fonctionnalités dans le codage du langage javascript*
* *ou le Framework* ***Bootstrap*** *qui apportent des composants et des outils de mises en page pour le responsive.*

*Certaines apportent des fonctionnalités avancées sur des besoins décrits dans le cahier des charges comme :*

* *la classe* ***PHP mailer*** *qui propose des fonctions avancées dans l’envoi de mails*
* *ou* ***FPDF*** *dans la génération de fichiers PDF. Ces classes se basent sur les classes et méthodes PHP PDF et mail et proposent un usage bien plus avancé.*

## Environnement technique :



**L’environnement de développement :** Il n’est pas imposé. J’ai choisi VSCODE comme environnement de développement. Il est performant, propose des extensions et est installé sur ma machine de travail.

**Serveur de développement :** J’ai essentiellement travaillé sur LARAGON qui propose un serveur http apache2, un serveur de base de données MySQL, l’interface PhpMyAdmin pour la gestion de la BDD. J’ai également travaillé sur KSWEB sur Android ou sur une machine virtuelle linux après avoir installé les serveurs HTTP et SGBD, et PHP.

**Les logiciels :** Il faudra installer sur la machine virtuelle les logiciels nécessaire au fonctionnement de l’application à savoir :

*▶* Apache 2 pour le serveur HTTP ▶ MySQL 5 pour le serveur de base de données

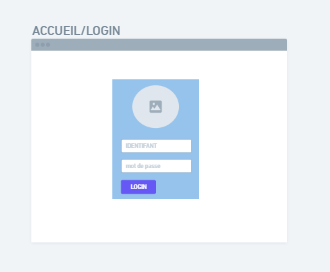
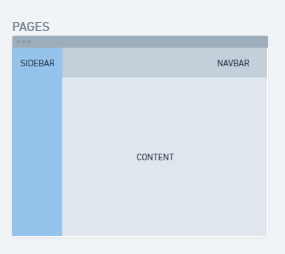
▶ PHP 7.4 pour le préprocesseur HTML ▶ PhpMyAdmin en interface de gestion de la BDD

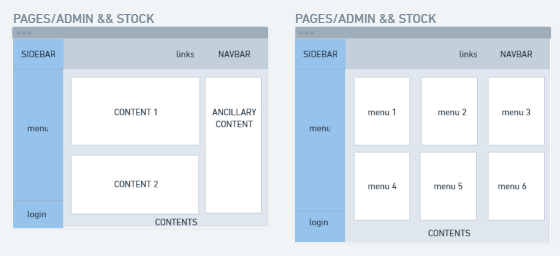
▶ PROFTPD pour le serveur FTP

Les instructions pour l’installation se feront en lignes de commande par accès sécurisé SSH via le logiciel PUTTY. L’upload des sources sur le serveur se fera via le logiciel Filezilla après l’installation du serveur ftp. Une recherche de documentation sera nécessaire dans la réalisation de cette étape du projet.

# **CHAPITRE 5 - RÉALISATIONS ET EXTRAITS DE CODES**

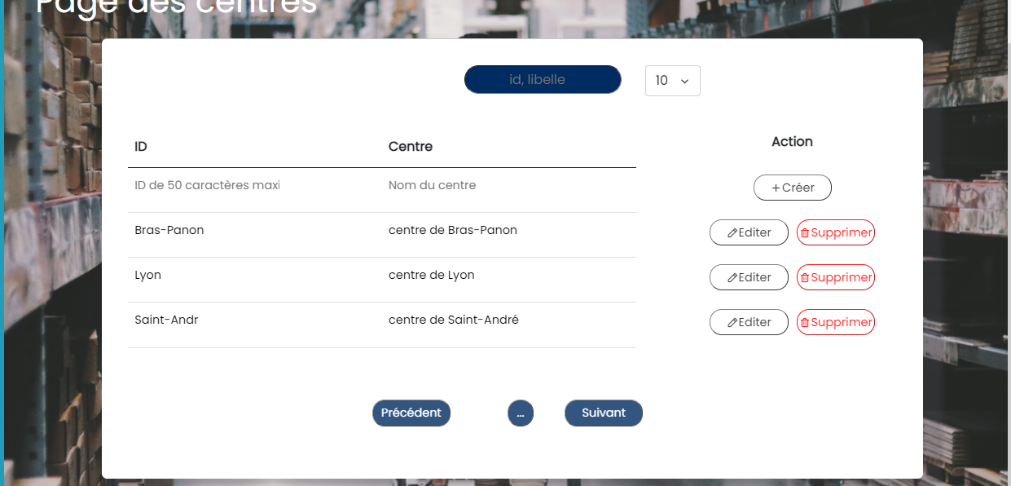
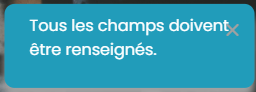
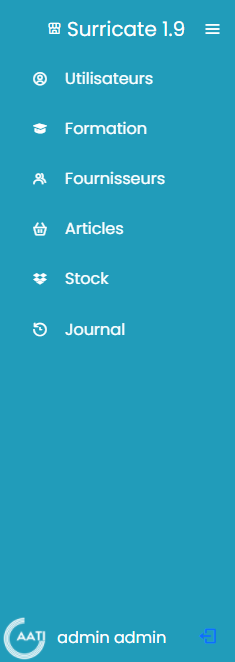
## Maquettage et wireframe :



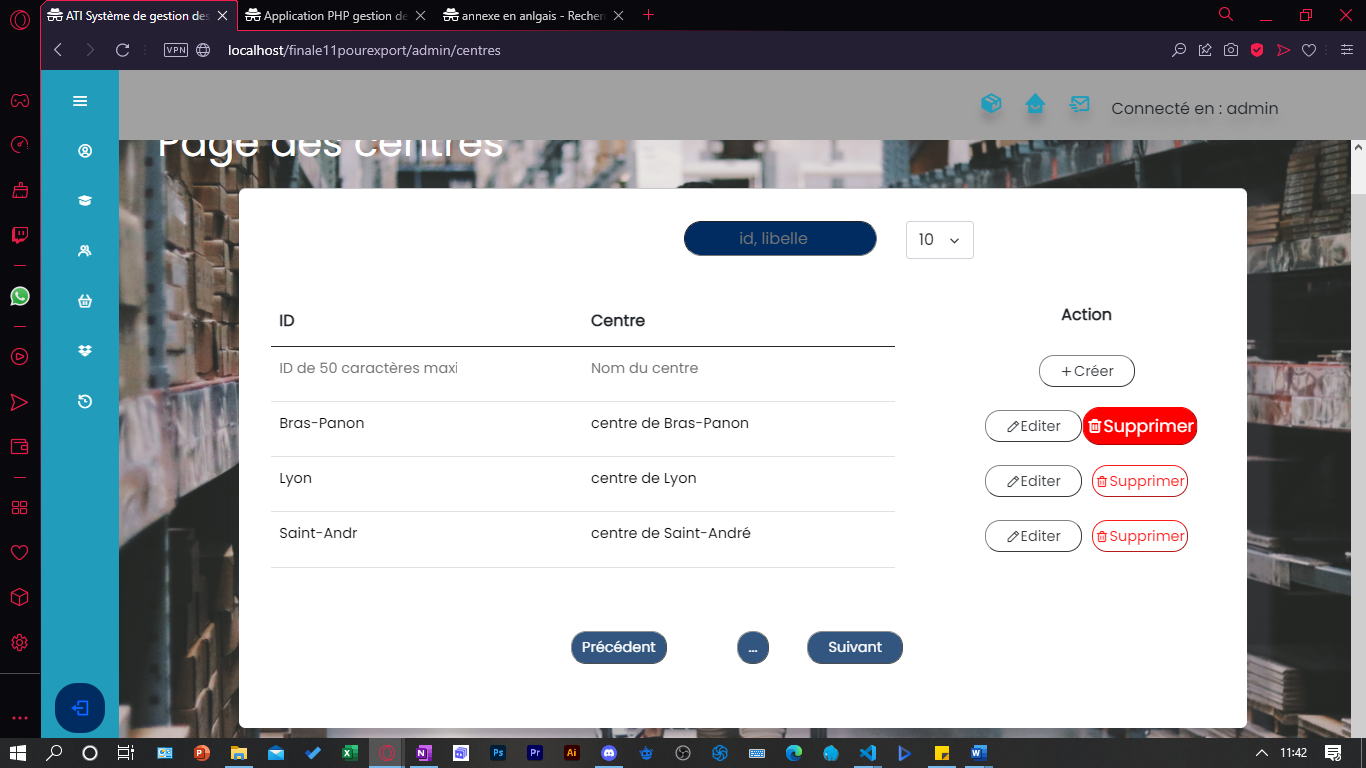
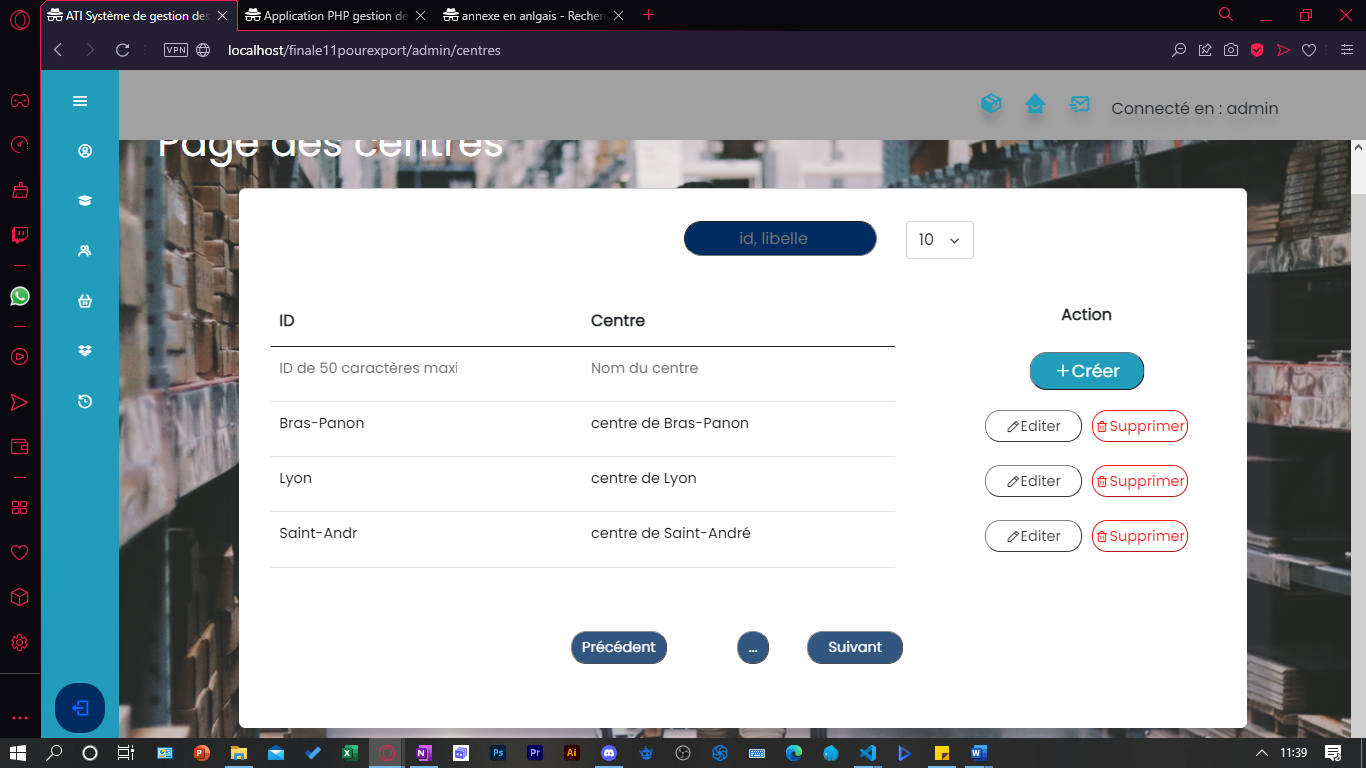
## Design des pages et composants :

**SIDEBAR NAVBAR**

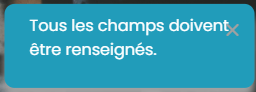
**BOUTONS :**

Etat 1 : Etat 2 :

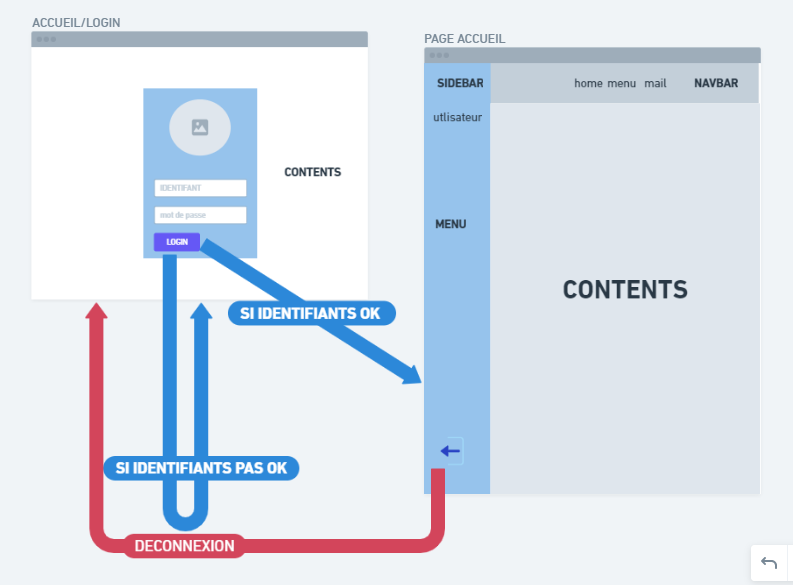
**SEARCH-BOX :**



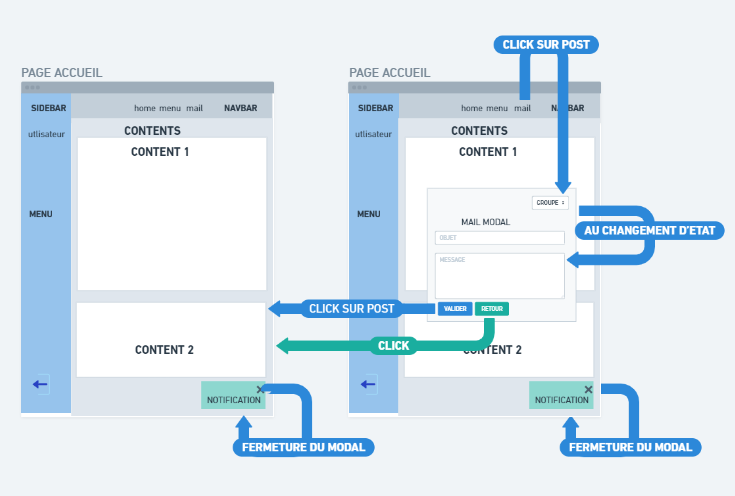
**NOTIFICATION- MODAL :**

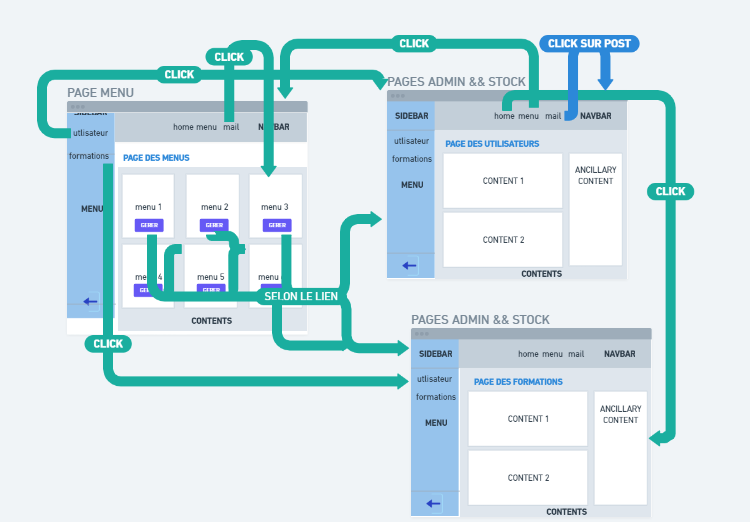
## Graphe de dialogue

**AUTHENTIFICATION/DECONNEXION**



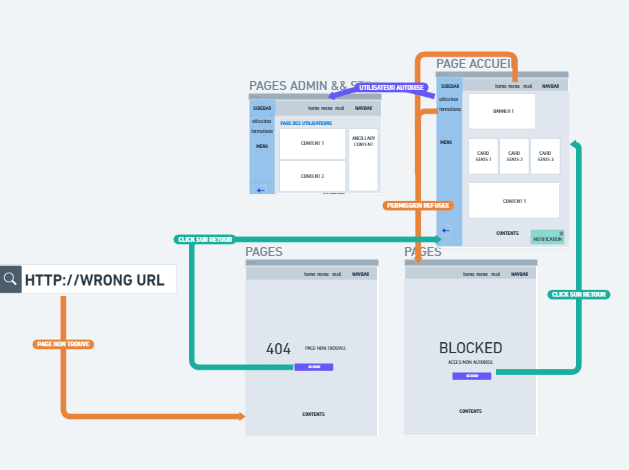
**MAILING/NOTIFICATION**



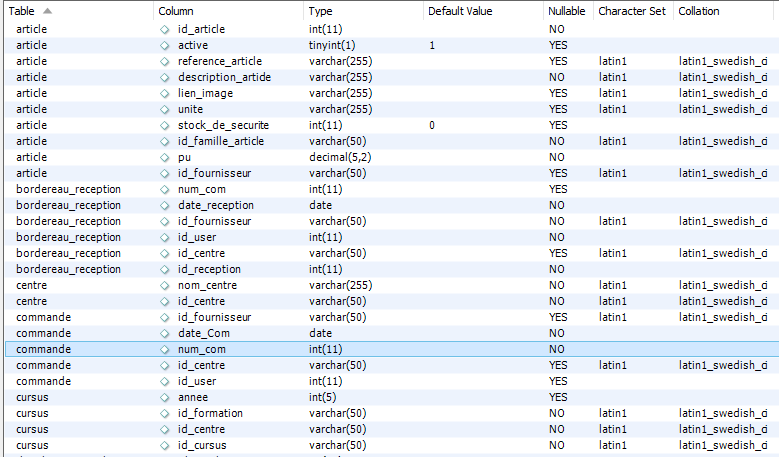
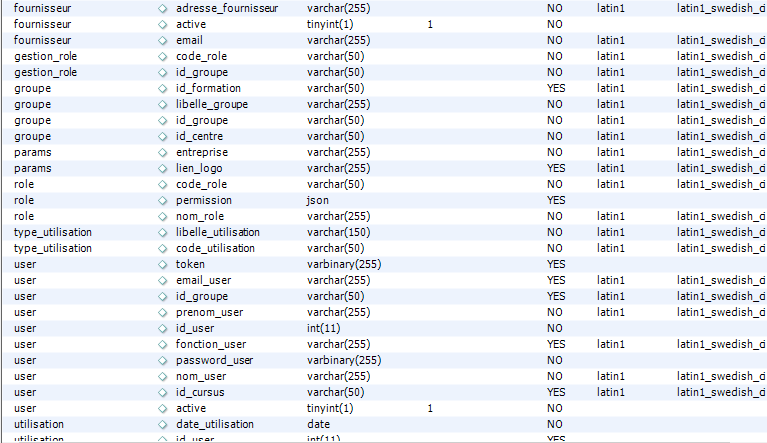
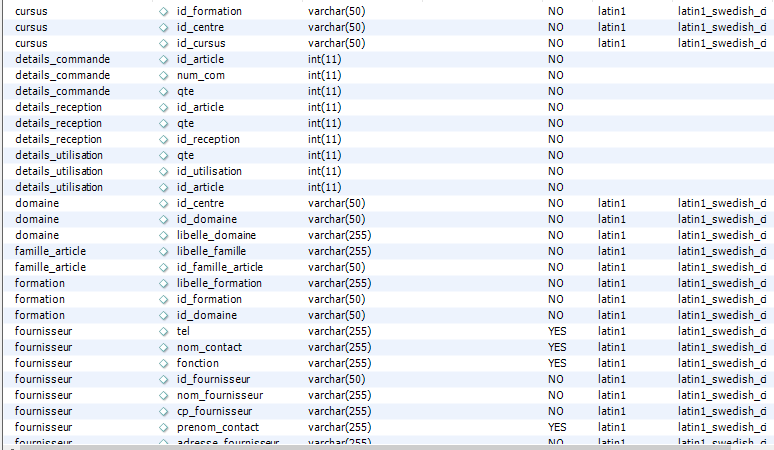


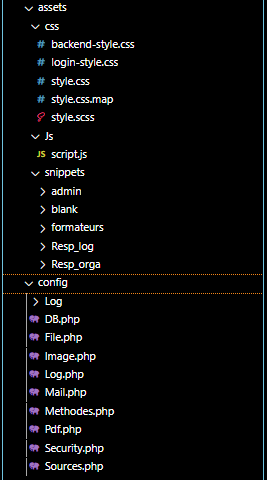
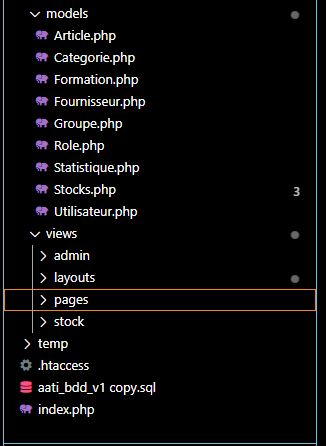
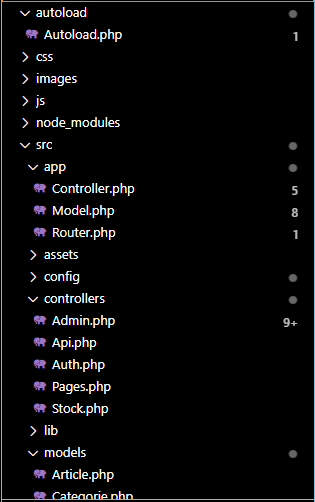
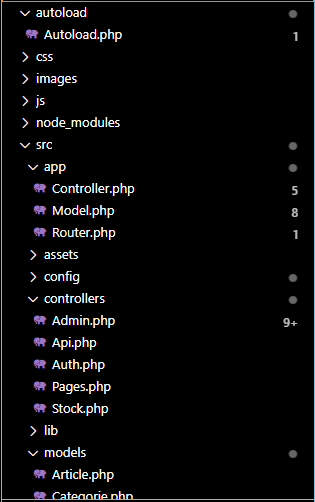
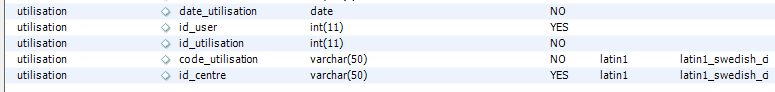
**ERREURS ET PERMISSIONS**

**MENUS**



## Dictionnaire de donnéés et arborescence





## MCD de l’application

## MPD de l’application

**{ 16 }**

## Extraits de code js

### Code qui gère l’animation de la sidebar.

document.querySelectorAll('.menu').forEach(item => {

    item.addEventListener('click', event => {

        sub\_menu.forEach(item =>{

            item.classList.remove('active');

        });

        item.parentElement.querySelectorAll('.sub-menu').forEach(item =>{

            item.classList.toggle('active');

            sidebar.classList.add("active");

            content.classList.add('active');

        });

    });

});

### Code qui gère le modal :

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Modal notification \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//(document.querySelector('.modal\_notification p').innerText.length >-1)?document.querySelector('.modal\_notification').classList.add('active'):'';

//document.querySelector('.modal\_notification i').addEventListener('click',function(){

//  document.querySelector('.modal\_notification').classList.remove('active') ;

//})

let y=50;

let x=0;

document.querySelectorAll('.modal\_notification').forEach(item =>{

    /\*console.log(item.querySelector('.modal\_text'));\*/

    (item.querySelector('.modal\_text').innerText.length >0)?item.classList.add('active'):'';

    item.style.transform = 'translate('+x+'%,'+y+'%)';

    y=y+20;

    x=x-7;

    console.log(item.style.transform);

    item.querySelector('i').addEventListener('click',function(){

        item.classList.remove('active');

    })

})

### Code qui gère la libraire chart.js :

let sorties =new Array();

document.querySelectorAll('.Sorties').forEach(item=>{

    sorties.push(item.innerText);

});

let securite =new Array();

document.querySelectorAll('.stock\_de\_securite').forEach(item=>{

    securite.push(item.innerText);

});

let marge =new Array();

document.querySelectorAll('.Marge\_beneficiaire').forEach(item=>{

    marge.push(item.innerText);

});

let entree =new Array();

document.querySelectorAll('.EntréesDivers').forEach(item=>{

    entree.push(item.innerText);

});

let labels =new Array();

document.querySelectorAll('.reference\_article').forEach(item=>{

    labels.push(item.innerText);

});

let livraisons =new Array();

document.querySelectorAll('.Livraison').forEach(item=>{

    livraisons.push(item.innerText);

});

const ctx = document.getElementById('myChart');

const myChart = new Chart(ctx, {

    type: 'bar',

    data: {

        labels: labels,

        datasets: [{

            label: 'Sorties',

            data: sorties,

            backgroundColor: [

                'marroon'

            ],

            borderColor: [

                'marroon',

            ],

            borderWidth: 1

        },

        {

            label: 'Stock disponible',

            data: marge,

            backgroundColor: [

                'green'

            ],

            borderColor: [

                'green',

            ],

            borderWidth: 1

        }]

    },

    options: {

        plugins: {

          title: {

            display: true,

            text: 'Répartition du volume de marchandises'

          },

        },

        responsive: true,

        scales: {

          x: {

            stacked: true,

          },

          y: {

            stacked: true

          }

        }

      }

    });

## Extraits de code PHP

### Classe Autoload :

On centralise toutes les inclusions de fichiers au travers de la méthode autoloadClass().

namespace Surricate;

/\*\*

 \* Autoload Classes

 \*/

class Autoload{

    /\*\*

     \* contient les noms de dossiers des classes à lancer via l'autoload

     \*/

    private const CLASSES\_SOURCES = [

        'app',

        'controllers',

        'models',

        'config',

        'lib'

    ];

     /\*\*

         \* (M) Méthode qui démarre l'autoload et s'occupe du démarrage automatique de méthodes au lancement

         \* (O) rien

         \* (I) rien

         \*/

    public static function init(){//initialise et démarre la classe autoload

       // define("ROOT\_INDEX",str\_replace('public/index.php','',$\_SERVER['SCRIPT\_FILENAME']));//on définit une constante PHP pour le ROOT du dossier

        define("ROOT",str\_replace('index.php','',$\_SERVER['SCRIPT\_FILENAME']));

        spl\_autoload\_register(function($class){

            self::namespaceToClass($class);//on traduit les namespaces en classe pour l'autoload des classes

            self::autoloadClass($class);

        });

        Log::init();// on lance d'abord le log

        Security::init();// puis la sécurité

        Router::init();//enfin on lance le routeur

    }

    /\*\*

     \* (M) Méthode qui fera les appels de fichier à chaque appel de classe

     \* (O) rien

     \* (I) le string du nom de classe à lancer

     \* @param string class string du nom de classe. Respecter la convention camelCase

    \*/

    private static function autoloadClass($class) {

        $sources = array\_map( function($sources) use($class) {

            return ROOT.'src/'. $sources . '/' . $class . '.php';

        }, self::CLASSES\_SOURCES );

        foreach($sources as $s){

            if(file\_exists($s)){

                require\_once $s;

            }

        }

    }

    /\*\*

     \* (M) Méthode qui transforme les namespace en nomde classe. Lors des appels de classes l'autoload appelle les classes suivant le systeme de namespace ce qui pose problème pour les appels de fichiers. Cette méthode résout ce problème.

    On adapte l’autoloader au système de namespace avec deux méthodes traductrices classe/Namespace.

 \* (O) rien

     \* (I) le string du namespace ou celui transmis par la méthode spl\_autoload\_register

     \* @param string class string du nom de classe.

    \*/

    public static function namespaceToClass(&$class){

            $class=preg\_replace('/(.)\*\\\/','',$class);//on enlève le namespace pour garder que le nom de la classe

    }

     /\*\*

     \* (M) Méthode qui transforme les nom de classe en namespace. L'instanciation d'une classe par l'utilisateur via la méthode des variables de string n'utilise pas le systeme de namespace. Cette méthode résout ce problème

     \* (O) rien

     \* (I) le string de la classe à passer

     \* @param string class string du nom de classe.

    \*/

    public static function classToNamespace($controller){

        return '\\'.\_\_NAMESPACE\_\_.'\\'.$controller;

    }

    private function autoloadStyle(){

        $params=explode('/',$\_GET['p']);

    }

}

### Extrait de code de la classe Controller :

La méthode loadModel() sert à l’appel des classes models et la méthode render() prépare et réalise l’affichage finale en procédant à l’inclusion des views et des composants (elle réalise les require nécessaires en lui passant le nom de la view).

/\*\*

         \* M permet d'initialiser les classe modèles choisies qui devra communiquer avec la bdd

         \* O rien

         \* I le string du nom du modèle ex: le model ;

         \*/

        protected function loadModel(string ...$model){//créé une instance de models transmis pour pouvoir faire appel à leurs méthodes

            foreach($model as $item){//sert uniquement à récupérer de la donnée

                $item=ucfirst($item);//On commence le nom de classe par une majuscule pour respecter le nom de fichier qui correspond à celui utilisé pour l'appel de classe donc impérativement en majuscule

                $itemNamespace=Autoload::classToNamespace($item);// on récupère le namespace correspondant

                $this->$item = new $itemNamespace();// on instancie le model

            }

        }

        protected function getDatas($methode,string ...$data){

        }

        /\*\*

         \* M permet d'initialiser les classe modèles choisies qui devra communiquer avec la bdd

         \* O rien

         \* I le string du nom du modèle ex: le model ;

         \*/

        protected function render(string $fichier,array $data = [],$class=null){// sert uniquement à l'affichage en faisant les require des vues

            extract($data);//extrait les données d'un tableau associatif sous forme de variables. On accède ainsi aux données unitaires recues de la bdd

            if($class==null){//si classe non défini on utilise la classe du contexte en cours ex: Admin, Auth...

                $class=get\_class($this);

            }

            Autoload::namespaceToClass($class);//on change les espaces de nom en classe pour les require

            try{

                    self::loadComponent($this->theme);//charge les composants ou éléments html headers... dans des constantes pour pouvoir les placer de facon structurer dans des layouts

            }catch(Exception $e){

                self::loadComponent($this->defaultTheme); //charge le theme par défaut en cas d'erreur

            }

            ob\_start();//enclenche la temporisation de sortie. Aucune donnée sauf les entetes n'est envoyé au navigateur

            require\_once(Sources::path('views').strtolower($class).'/'.$fichier.'.php');//commenter pour changer de méthodes

            //require\_once(ROOT.'src/views/'.strtolower($class).'/'.$fichier.'.php');// decommenter pour revenir à la méthode par défaut des require

            $content=ob\_get\_clean();//les echos et require de la vue ont été placé dans une variable qui est dans le layout

            require\_once(ROOT.'src/views/layouts/'.$this->layout.'.php');//une fois la variable déclarée on importe le layout

        }

### Extrait de code de la classe Model :

La plupart des méthodes des classes filles de Model utilisent pour les requêtes SQL les méthodes de Model à travers l’héritage. La connexion et ces méthodes sont encapsulées.

  protected function updateBy($id,$valeur\_id,$data,$isToEncrypt=null,$key=self::ENCRYPT\_SQL\_KEY){

        $datas=array($id,$valeur\_id,$data);

        $this->secure($datas);

        extract($datas);

        $i=0;

        $sql="UPDATE $this->table SET";

        foreach($data as $colonne => $valeur){

          $sql.=($isToEncrypt!=null && !empty($key) && in\_array($colonne,$isToEncrypt))?' '.$colonne.' = '.$this->enDecryptData('?',$key).',':' '.$colonne.' = ? ,';

          $valeurs[$i]= $valeur;

          $i++;

        }

        $valeurs[$i]= $valeur\_id;

        $sql=substr($sql,0,-1);

        $sql.=" WHERE $id = ?";

        $this->testSql($sql,$valeurs);

        $req=$this->connexion->prepare($sql);

        $this->setDataNull($valeurs);//[6] en param 2

        $req=$req->execute($valeurs);

        return $req;

    }

### Extrait de la classe Article, fille de la classe Model  et présentation des transactions par PDO

/\*\*

     \* (M) Méthode qui update un article

     \* (O) rien

     \* (I) 3 champs

     \* @param string champ ou nom de colonne de l'id

     \* @param string valeur du champs

     \* @param array file facultatif contenant l'ensemble des propiétés du fichier image uploadé.

     \*/

    public function updateOneArticle($id,$valeur\_id,$file=null){

        $this->table='article';// on choisit a table article

        $this->setData();// on configure les données à utiliser. Post non déclarés sont mis à vide et int on leur met une valeur 0

        ($file!=null)?$this->data['lien\_image']=$file['filename'].$file['extendedName'].'.'.$file['type']:'';// on définit l'adresse de l'image associé à l'article

        if($\_POST['pu']>=0){// Sécurité sur le prix si le prix n'est pas correcte on refuse la mise à jour

            (empty($this->data['active']))?$this->data['active']=0:'';

            $results= $this->updateBy($id,$valeur\_id,$this->data);// on met à jour les données

            if(!$results){// Si on ne peut pas update

                try{

                    $this->connexion->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);// on met le mode erreur de PDO en Exception pour qu'il retourne le message d'erreur

                    $this->connexion->beginTransaction();//     on commence une transaction

                        $this->insertOneArticle($this->data);// On insère la modfication comme nouvel article

                        $tables=['approvisionnement','details\_commande','details\_reception','details\_utilisation']; // on définit les tables à modifier avec les nouvelles valeurs

                        $lastId=$this->getLastId();// on récupère l'id de la nouvelle ligne insérée

                        foreach($tables as $table){// pour chaque table à mettre à jour

                            $this->table=$table;

                            $this->updateBy('id\_article',$\_POST['validating\_edit'],['id\_article'=>$lastId]);// on met à jour les données

                        }

                        $this->table='article';// on revient dans article

                        $this->deleteBy($id,$valeur\_id);//on supprime l'ancienne occurrence d'article

                    $this->connexion->commit();// on fait un commit

                }catch(Exception $e){// si la transaction echoue

                    $this->connexion->rollback();// on fait un rollback

                    $results=false;// on retourne un false pour echec

                }

            }

        }else{// si le prix est inférieur à 0

            $results=false;// on notifie un echec pour l'opération

        }

        return $results;//on retourne false ou true

    }

### Extrait de code de la classe model statistique montrant les appels de procédures :

  public function fluxComCurrYearByUser($idUser){

        $sql= 'call flux\_commande\_current\_year\_by\_user(?)';

        $req=$this->connexion->prepare($sql);

        $req->execute(array($idUser));

        return $req->fetchAll(); }

### Extrait du code de la view Layout :

Il recueille les différentes views à travers la variable $content ainsi que les différents composants d’une page (header,footer, sidebar, navbar…). Le layout permet de présenter les différentes views à travers des pages homogènes.

<!DOCTYPE html>

<html lang="en">

  <head>

    <?=HEAD?>

    <link rel="stylesheet" href="<?=BASE\_URI?>src/assets/css/backend-style.css">

  </head>

  <body>

    <?=(isset($\_POST['mail']) && !empty(MAIL))? MAIL:""?>

    <header>

      <?=NAVBAR?>

    </header>

      <?=SIDEBAR?>

    <main>

      <section>

        <div class="home\_content">

          <div class="text top\_title">

            <h1>

              Page des <?=PAGE?>

            </h1>

          </div>

          <div class="content">

            <div class="container">

              <?=$content?>

            </div>

          </div>

        </div>

      </section>

    </main>

    <?=(!empty(MODAL) && !empty($\_SESSION['message']))?MODAL:"";?>

    <?=(isset($\_POST['upload\_file']) && !empty(FILE))? FILE:""?>

    <script src="<?=BASE\_URI?>node\_modules/chart.js/dist/chart.min.js"></script>

    <script src="<?=BASE\_URI?>src/assets/Js/script.js"></script>

    <?=FOOTER?>

  </body>

</html>

### Méthode qui génère les notifications

 /\*\*

         \* M méthode qui adresse les messages

         \* O Rien

         \* I Rien

         \*/

        protected function message($message,$id\_message=1){

            if(!empty($\_SESSION['message'][0])){

                unset($\_SESSION['message'][0]);

            }

            (isset($\_SESSION['message'][$id\_message]))?$\_SESSION['message'][$id\_message]=$message :$\_SESSION['message'][$id\_message]=$message;

        }

### Extrait de code de la classe Admin :

#### Page d’accueil des formateurs :

  public function index(){//pour aller au menu de back-end

        define('PAGE','espaces membres');// on indique le nom de la page d'accueil admin comme "parametres"

        if($\_SESSION['login']=='logged'){//si on est loggé on affiche le menu

            (!$this->checkUserContenuPermission('INDEX',['LECTURE']))?$this->redirection('pages/blocked') :'';// on continue si on a les permissions ou on est redirigé

            $this->loadModel('Statistique');

            if(!empty($idUser=$\_SESSION['id'])){

                $commandesCurrentYear=$this->Statistique->fluxComCurrYearByUser($idUser);

                $comUserGrpByArt=$this->Statistique->fluxComCurrYearByUserGrpByArticle($idUser);

                $sortieCurrMonthUser= $this->Statistique->sortieCurMonthByUser($idUser);

                $comMonthUser = $this->Statistique-> comCurrentMonthByUser($idUser);

                $comMonthUserGrpByArt = $this->Statistique-> comCurrentMonthByUserGrpByArticle($idUser);

                $fluxMensuel = $this->Statistique->fluxUtilisationMensuelleUser($idUser);

                (!empty($preferedArticle =$this->Statistique->preferedArticleByUser($idUser)))?$\_POST['preferedArticle']=$preferedArticle:$\_POST['preferedArticle']=[];

            }

            //var\_dump($preferedArticle );die();

            $this->render('espaceperso',compact('fluxMensuel','preferedArticle','commandesCurrentYear','comUserGrpByArt','sortieCurrMonthUser','comMonthUser','comMonthUserGrpByArt'),'admin');

        }else{//sinon réaffiche la page de login

            $this->redirection('auth/login');

        }

    }

#### Extrait de la méthode centres de la classe Admin

public function centres(){//méthode qui gère les groupes

        define('PAGE',\_\_FUNCTION\_\_);// definit le nom de la page

        if($\_SESSION['login']=='logged'){

            (!$this->checkUserContenuPermission(PAGE,['LECTURE']))?$this->redirection('pages/blocked'):'';// on continue si on a les permissions ou on est redirigé

            $this->loadModel("Formation",);//on charge les modèles utilisés par la méthode groupes

            if(!empty($\_POST['delete'])){//si on valide la suppresion du groupe

                (!$this->checkUserContenuPermission(PAGE,['SUPPRESSION']))?$this->redirection('pages/blocked'):'';// on continue si on a les permissions ou on est redirigé

                if($this->Formation->deleteOneCentre('id\_centre',$\_POST['delete'])){//si la suppression réussit

                    $this->message("Le centre a bien été supprimé",\_\_FUNCTION\_\_); //on envoie une notification de succès

                    $this->redirection('admin/'.\_\_FUNCTION\_\_);// on redirige vers la page d'accueil des centres

                } else{

                    $this->message("Erreur lors de la suppression",\_\_FUNCTION\_\_); //on envoie une notification de succès

                }

            }elseif(isset($\_POST['create'])&&!empty(($\_POST['id\_centre']))&& !empty(($\_POST['nom\_centre']))){//si on valide la création d'une centre non vide

                (!$this->checkUserContenuPermission(PAGE,['CREATION']))?$this->redirection('pages/blocked'):'';// on continue si on a les permissions ou on est redirigé

                if($this->Formation->insertOnecentre()){// si le rajout du nouveau centre réussit

                    $this->message("La centre a été créé",\_\_FUNCTION\_\_); // on envoie une notification de succès

                    $this->redirection('admin/'.\_\_FUNCTION\_\_);

                }else{

                    $this->message("La centre n\'a pas été créé",\_\_FUNCTION\_\_); // on envoie une notification de succès

                }

            }elseif(!empty($\_POST['validating\_edit'])){//si on valide l'édition d'un centre

                (!$this->checkUserContenuPermission(PAGE,['MODIFICATION']))?$this->redirection('pages/blocked'):'';// on continue si on a les permissions ou on est redirigé

                $centres=$this->Formation->getAllcentres();// on récupère les informations sur les centres

                $data=[// on récupère les informations mises à jour du centre venant du formulaire

                    'id\_centre' =>$\_POST['id\_centre'.$\_POST['validating\_edit']],

                    'nom\_centre' =>$\_POST['nom\_centre'.$\_POST['validating\_edit']]

                ];

                if($this->Formation->updateOneCentre('id\_centre',$\_POST['validating\_edit'],$data)){// si la mise à jour réussit

                    $this->message('Le centre '.$data['nom\_centre'].' a été mis à jour',\_\_FUNCTION\_\_); // on envoie une notification de succès

                    $this->redirection('admin/'.\_\_FUNCTION\_\_);// on redirige vers l'accueil du centre

                }else{

                    $this->message('Le centre '.$data['nom\_centre'].' n\'a pas été mis à jour',\_\_FUNCTION\_\_); // on envoie une notification de succès

                }

            }

        }else{//si on n'est pas loggé

            $this->redirection('utilisateurs/login');// on est renvoyé à la page de login

        }

        $this->setPagination('Formation','countCentres',$search,$limit,$numberOfPage);

        $datas=$this->Formation->getAllCentres($search,$limit);// on récupère les infos actualisées des centres

        $this->render('centres',compact('datas','numberOfPage'));    // on affiche d'accueil des groupes avec les liste des groupes

    }

### Description de la méthode principale qui gère les permissions dans la classe Security

        /\*\*Description : fonction qui gère la vérification des permissions des users par page

         \* @param string nomPage string du nom de page

         \* @param array permString de string de nom des permissions nécessaires ex :

         \*  ['LECTURE',

            'CREATION',

            'MODIFICATION',

            'SUPPRESSION']

         \* @param boolean egaliteStrict facultatif si on veut définir des permissions strictes par exemple seul création est autorisé.Si une personne a un score supérieure elle sera quand meme refusée.

         \* peut servir a affiner les permissions sur des pages ou l'affichage ou non de blocs html

         \* @return boolean true si le score de permission de l'utilisateur est supérieure à celle demandée

         \*/

        protected function checkUserContenuPermission($nomPage,$permString,$egaliteStrict=false){

            $allowedScore=0;// on met un score autorisé de 0

            foreach($permString as $perm){// pour chaque string de permission entré

                if($this->translatePermStringToValue($perm)>=0){//on traduit le string en score et s'il est bon et >=0

                    $allowedScore=$allowedScore+$this->translatePermStringToValue($perm);// on le rajoute au score autorisé

                }

            }

            $idPage=$this->translatePermStringToValue(strtoupper($nomPage),Security::PERMISSION\_CONTENU);//on récupère l'id de page à partir du nom

            $results=false;

            $roleFromDb = new Role();// on initie une instance de role

            if(!empty($\_SESSION['role'])){//si la session n'est pas vide

                $roles= explode(',',$\_SESSION['role']);// on obtient tous les roles enrgistrés dans la session

                foreach($roles as $role){// pour chaque role

                    $score=0;// on définit un score de départ de 0

                    $userRole=$roleFromDb->getOneRole('code\_role',$role);// puis on récupère les infos du role

                    $userPermission=@json\_decode($userRole["permission"]);//on récupère plus précisememnt l'ensemble des permissions du role sous forme d'objet

                    if(gettype($userPermission)=='array'){

                        foreach($userPermission as $permission){//pour chaque permission de page

                            ($permission->id == $idPage && $permission->permissions >=$score)?$score=$permission->permissions:'';// si l'id de page correspond et si son score de permission est supérieure au score en cours on l'enregistre et obtient le niveau de permission accordé le plus élevé

                        }

                    }

                }

                if($egaliteStrict==true){// si on souhaite une valeur de permission égale

                    $results = $score==$allowedScore;// si le score de l'utilisateur est strictement égale à celui demandé on retourne true

                }else{// si on ne souhaite pas d'égalité stricte

                    $results = ($score>=$allowedScore);// si le score de l'utilisateur est suffisant on retourne true

                }

           }

          return $results;

        }

#### Fonction qui masque le nom des classes par des alias dans l’url. Elle peut être étendue aux méthodes.

 /\*\*(M) Fonction qui à partir d'un alias redonne le nom de classe telle que défini dans Security::CLASS\_ALIAS. La fonction sert à masquer le nom des classes dans les url en utilisant des alias. Le mode est personnalisable par Security::USE\_STRICT\_CLASS\_ALIAS qui à false autorise les alias et nom de classe

         \* (O) String du nom de classe correspondant à l'alias

         \* (I) prends 1 parametre 1 string de l'alias

         \* @param string alias nomde l'alias

         \*/

        public static function classAlias($alias){

           return @self::alias($alias,self::CLASS\_ALIAS,self::USE\_STRICT\_CLASS\_ALIAS);

        }

#### Méthode qui sécurise les données utilisateurs au travers de la fonction HtmlEntities(). D’autres fonctions peuvent être ajoutées.

/\*\*(M) Méthode qui applique des méthodes pour sécuriser les variables Post et SESSION

         \* (O) rien

         \* (I) rien

         \*/

        private static function sanitizePostAndSession(){

            @self::methodsToArrayValues($\_POST,self::SECURTY\_METHOD);

            @self::methodsToArrayValues($\_SESSION,self::SECURTY\_METHOD);

        }

        /\*\*(M) Méthode qui applique des méthodes pour sécuriser un array ou un string

         \* (O) rien

         \* (I) prends 1 parametre le string ou l'array de valeur à sécuriser

         \* @param string string ou array des valeurs à sécuriser

        \*/

        public static function sanitizeUserData($data){

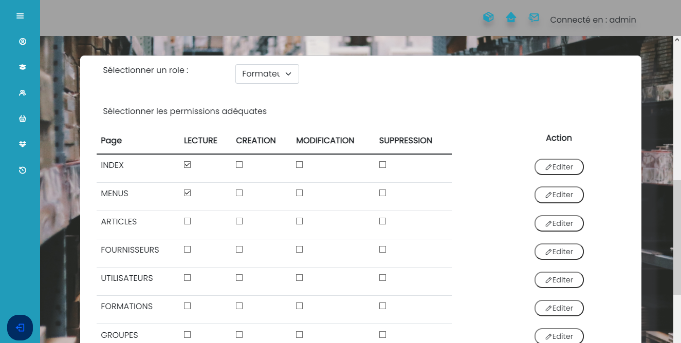
            @self::methodsToArrayValues($data,Security::SECURTY\_METHOD);

        }

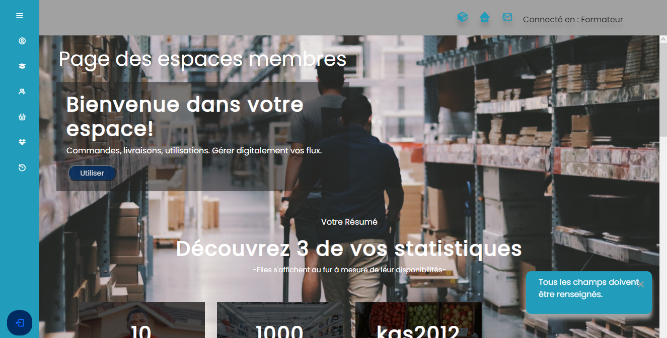
# **CHAPITRE 2 – PRÉSENTATION DES FONCTIONNALITÉS**

## Gestion des permissions :

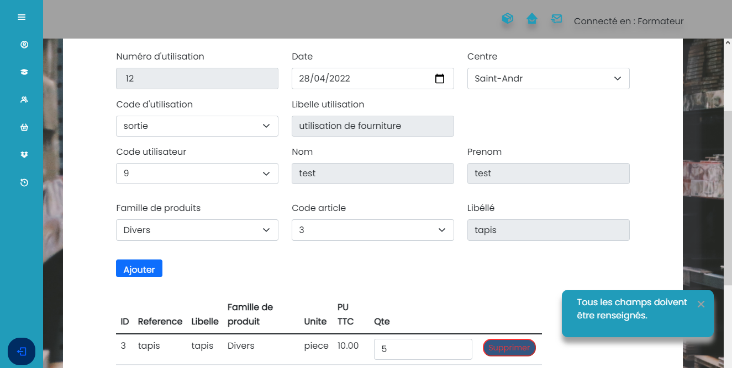
On est connecté en admin et on donne les permissions de lecture de la page index et menus au rôle formateur et lui refuse les autres permissions.



On est connecté sous le rôle formateur. Nous avons bien accès à la page index et non aux pages d’administration telle que la page de gestion des utilisateurs. A la tentative de connexion on est renvoyé vers une page d’erreur.

## Nouvelle sortie de marchandises



On valide une sortie de 5 articles tapis sur le bon 12.



On retrouve bien nos 5 tapies en sortie par l’utilisateur test qui est un formateur.

# **CHAPITRE 3 – FONCTIONNEMENT ET DEBOGAGE**

## Débogage de routage

### Etape 1 : vérifier le fonctionnement du rewrite engine d’Apache

Le routeur apache est localisé dans le fichier htaccess à la racine du projet.

Si le routeur ne fonctionne pas. Vérifier en premier qu’en tapant différentes requêtes qu’on accède bien et toujours au même fichier index.php situé à la racine du site selon différentes méthodes en mettant par exemple un fichier index.php avec du texte à afficher.

Si l’index n’est pas accessible, le problème vient alors de la réécriture d’url par le serveur web en l’occurrence ici Apache. Vérifier la configuration d’apache, que le module de réécriture d’url est bien disponible et activé.

Pour activer le moteur de réécriture taper en ligne de commande sur linux : **sudo a2enmode rewrite**. Vous pouvez également consulter les fichiers de configuration d’apache et peut être le fichier **mods-enabled**.

Attention la redirection réécrit toutes les url, les url en .php sauf les autres url contenant un point tel que le .css où aucune redirection n’est réalisée. Il est ainsi possible d’appeler normalement et classiquement les feuilles de styles.

Pour plus d’informations consulter la documentation en commentaire dans le fichier.

### B.Etape 2 : vérifier que le routeur PHP fonctionne correctement

Le fichier php du routeur php se trouve dans le dossier app, sous src qui se trouve à la racine de l’application.

Vérifier que le routeur, c’est-à-dire le fichier routeur.php soit accessible. Cela peut se faire par un echo dans la fonction init() du routeur qui indique alors si son appel se passe correctement.

Si l’appel du routeur se passe correctement mais que votre fichier n’est pas appelé. Le problème provient alors d’un problème de traitement de la méthode setRedirection()

Consulter la documentation de la méthode en commentaire dans le fichier.

## GESTION DES FEUILLES DE STYLES

Cette partie est en cours de développement et n’a pas encore été réalisée. La gestion des feuilles de styles se fait via des appels classiques de fichier dans par le head et n’est pas encore gérée par le php.

## Namespace et reference de fichier

L’application utilise le système de namespace. Pensez à vous familiariser avec le système d’espace de nom ainsi qu’à l’utiliser dans de futures modifications. Le namespace de base pour tous les fichiers est surricate qui fait office de dossier virtuel de base qui regroupe tous les fichiers. Le référencement de l’ensemble des fichiers de l’application se fait donc de la façon suivante : \Surricate\nom de fichier. Veiller à éviter les doublons dans vos fichiers ou pensez à redéfinir et complexifier les namespaces.

Liste des namespaces utilisés :

|  |  |
| --- | --- |
| Namespace | Bibliothèques |
| Surricate | Fichiers de l’application |
| Fpdf | Pdf\_Linegraph, FPDF |

## AUTOLOADER

L’application utilise le système de l’autoloader. Il se trouve dans le dossier autoload à la racine de l’application. C’est-à-dire que les fichiers sont automatiquement appelés par un simple appel de sa classe.

La constante CLASSES\_SOURCES liste dans un tableau les dossiers dont lesquels l’application va chercher automatiquement les fichiers. C’est-à-dire que les fichiers doivent être impérativement inclus dans ces dossiers pour utiliser le système d’autoload, ou le nom du nouveau dossier contenant les fichiers des classes à exécuter doit être rajouté à la constante.

5 dossiers sont listés : app, controllers,models,config,lib.

Etant donné que l’ensemble des fichiers utilisent le système de namespace. La classe autoloader contient deux méthodes pour faciliter leur appel et utilisation :

La méthode publique et statique à utiliser quand vous faites un appel à partir du nom de classe est : classToNamespace().

La méthode publique et statique à utiliser quand vous souhaiter récupérer le nom de la classe à partir d’un namespace est : namespaceToClass().

Une documentation du fonctionnement des méthodes sont inclus dans le fichier source Autoload.php qui est dans le dossier autoload à la racine.

## mode debogage SQL

Il est possible d’activer le mode débogage sql en allant dans le fichier db sous src/app/config. Pour cela modifier la constante ENV en « debug ».

## configuration de l’IMPORT DES IMAGES

Il est possible de modifier les extensions, la taille, le poids ainsi que le dossier dans lequel les images des articles vont être uploadés en modifier la constante ARTICLE\_IMAGE du fichier Sources.php dans config.

## GESTION AUTOMATIQUE DU http ET HTTPS

Pour utiliser la gestion automatique du basculement des liens en https si disponible utiliser la constante HTTPS en lieu et place du https dans vos liens absolus. Si la constante n’est pas accessible veiller à appeler le fichier Sources qui se trouve dans src/config.

## creation et modification des pdf

Pour créer ou modifier un pdf rendez vous dans le fichier Pdf.php dans le dossier /src/config/pdf.php.

Pour pourrez modifier les bons de commande et de livraisons en modifiant les paramètres de sa méthode publique createBon.

Attention les bons de commande et de livraison utilisent la même méthode createBon dans la génération des pdf et les données dynamiques du bon sont fournies à partir de requêtes sql transmis à la méthode à travers ses paramètres d’entrées qui comprend le titre du bon, les données du bon tels que le numéro de commandes et les informations fournisseurs et enfin les données des articles telles que la quantité, le prix et le nom des produits : createBon($name,$bon,$articles).

Les informations statiques tels que le nom du centre peut être modifié directement en dur dans la méthode.

Pour plus d’information sur la bibliothèque FPDF et sa licence consulter l’ensemble des fichiers de documentation sous src/lib/FPDF/.

## creation et modification des graphiques

Pour créer ou modifier des graphiques rendez-vous dans le fichier script.js sous src/assets/snippets/ et aller à la section charts. Modifier les constantes myChartXXX pour modifier les graphiques. Vous pouvez également réutiliser le code pour ajouter de nouveaux graphiques. Vous pouvez modifier le type de graphique, les couleurs, les légendes, etc… Les données sont récupérées directement de la page qui a été générée.

Les fichiers de la bibliothèque se trouve à la racine sous node\_modules\chart.js\. Pour plus d’informations sur le fonctionnement de la bibliothèque myChartJs, consulter le site de la bibliothèque.

## personnalisation des urls

Pour personnaliser les urls, veuillez modifier la constante/ le parametre CLASS\_ALIAS de la classe SECURITY qui est un array. La clef devant être le nom modifié de la classe. Ainsi « administration => admin » conduit quand on tape administration à la place de admin à être redirigé vers admin.

Pour éviter que deux urls ne pointent sur la même page, vous pouvez également empêcher que l’url SERVEUR/admin pointe vers admin en modifiant le booléen de la constante USE\_STRICT\_CLASS\_ALIAS à true. Ainsi dans notre exemple seul l’url administration reste valide.

## filtre des valeurs posts & session

Le filtre des valeurs POST en sécurité se fait automatiquement avant chaque traitement via la méthode sanitizePostAndSession de la classe SECURITY. Elle va appliquer sur les variables les méthodes de l’array SECURITY\_METHOD de la classe SECURITY.

On peut également modifier les filtres de caractères qui sont appliqués à la réception des variables POST ou SESSION en modifiant la méthode translateSpecialChars et le paramètre Security :NOT\_ALLOWED\_CHARS.

## protection du htaccess

Pour désactiver la protection du HTACCESS du dossier temporaire d’importation des images(src/temp), commenter le constructeur de la classe file du dossier config. Une vérification est lancée à chaque import de fichier si le htaccess a été modifié il revient avant toute action à sa valeur par défaut contenue dans le string du paramètre HTACCESS\_CONTENT de la classe File.

## probleme de retour a la page de connexion

En cas de problème de retour à la page de connexion. Veuillez réinitialiser les cookies pour un retour au fonctionnement initial.

# **CHAPITRE 4 – TO DO LIST DE SECURITE: ce qui est fait et reste a faire**

## Les injections SQL

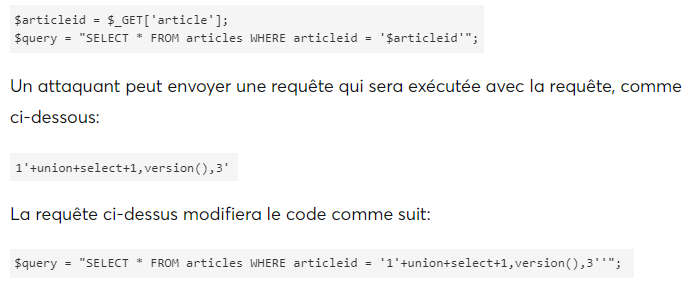
**Réalisation**

Les injections sql consistent à insérer du code dans les requêtes SQL au travers des inputs de formulaires pour en modifier les comportements et les faire exécuter côté serveur. Elle sont utilisées ainsi pour exécuter des LMD non voulus et voler des données de la BDD.

**Un premier correctif consiste à utiliser les requêtes paramétrées** soit en SQL dur avec « SET @variable = ‘’ » ou par l’intermédiaire des requêtes préparées de PDO qui utilise cette technique.

**La solution a été portée par PDO.**

**Description**



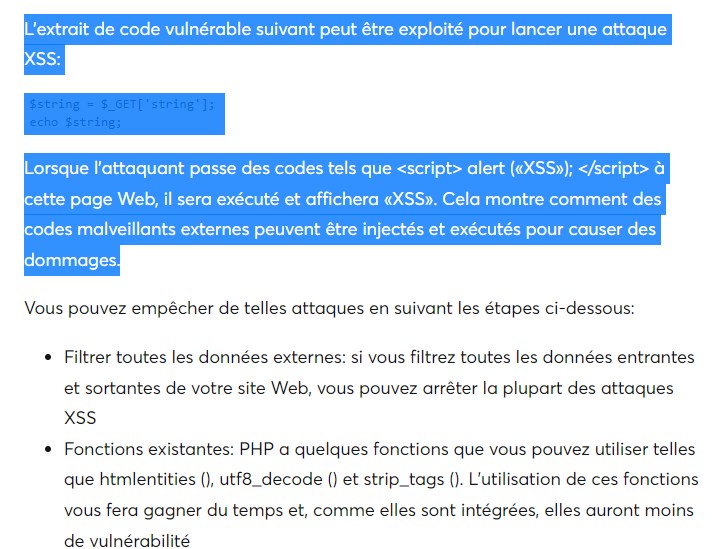
## Scripts intersites ou failles xss

**Réalisation**

Les failles XSS consistent à insérer un script étranger dans la page côté client afin de le faire exécuter par le système de l’utilisateur. Elles sont utilisées pour voler des informations d’indentification ou le contenu des cookies.

**Comme premier correctif on a appliqué la méthode htmlEntities().**

**Description**



## La fixation de session

Elle consiste pour l’attaquant à créer une session et à envoyer son id de session à sa victime qui lorsqu’elle se connectera se connectera sous cet id et validera la session. Une recommandation pour pouvoir authentifier les sessions valides implique l’usage parallèle de cookies afin d’identifier la machine locale. Cela demandera à l’attaquant un effort supplémentaire . **Comme premier correctif utilisé, à chaque connexion on détruit systématiquement la session** en générant une nouvelle.

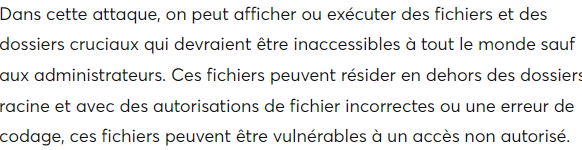
## Faille include

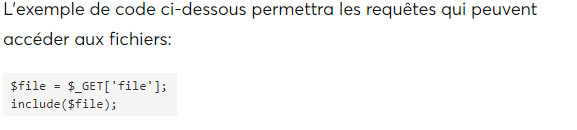
**Réalisation**

**Comme correctif :** aucun appel de fichier n’utilise de variables fournies par l’utilisateur. **Nous utilisons une classe Autoload** avec des méthodes et propriétés privées pour l’inclusion de fichiers. Les noms de classes sont fournis par la méthode spl\_autoload\_register et seul les classes des dossiers spécifiés sont appelées.

**Un système d’alias et un routeur ont également été mis en place** afin de masquer les noms de classes et dossiers.

**Description**





## Falsification de demandes intersites ou failles csrf

**Réalisation**

**Comme correctif : on a privilégié l’utilisation de POST** à la place des GET dans le formulaire, l’utilisation d’un triple token (SESSION, cookies,URL) obtenue à partir d’un token généré et sauvegardé en BDD par session est également en phase de développement, ainsi que le cryptage aléatoire par sessions des noms de variables POST.

**Description**

Il s’agit de faire exécuter à l’internaute des actions non désirées en profitant des privilèges utilisateurs dont il bénéficie. Cette pratique peut prendre la forme de GET envoyés via une url qui sera adressée et cliquée par l’utilisateur.

L’utilisation de HTTPS comme protocole pour le tunnel de transmission de données a également été recommandée.

## Attaques par force brute

**Réalisation**

Comme correctif  ultérieur le nombre de tentative peut être enregistré en bdd pour chaque compte pour pouvoir les limiter à 3.

**Description**

On teste chaque combinaison possible d'un mot de passe/identifiant donné afin se connecter au service ciblé.

## Cryptage de mots de passe et recommandations de sécurité

**Pour l’instant le procédé de cryptage utilise la méthode de cryptage AES()** proposée par MySQL où il est stocké en donnée de type VARBINARY(). **Le rajout de password\_hash() est prévu.**

Password\_hash() génère un hachage de soixante caractères basé sur BCRYPT (algorithme CRYPT\_BLOWFISH). Il prend 3 paramètres : le mot de passe, l’algorithme de hashage et le coût. Le mot de passe crypté ne peut plus être retrouvé mais vérifié via password\_verify().



## Le détournement ou vol de session

Le pirate réussit à obtenir les identifiants de session de l’utilisateur et les utilise pour tromper le site afin d’obtenir l’accès aux privilèges utilisateurs de la session piratée.

**Comme premier correctif, une vérification des variables d’environnement telle que l’adresse IP,** l’user -agent… sera apportée. Les variables en cours seront comparées à celles utilisées par la dernière session utilisateur qui auront été préalablement sauvegardées en BDD afin de les authentifier.

# **CHAPITRE 8 – INFORMATIONS ET REMARQUES**

## Défaut de rewrite-engine sur le serveur linux :

*https://ubiq.co/tech-blog/how-to-enable-mod\_rewrite-in-xampp-wamp/*

L’application et le pattern MVC se base sur l’utilisation du module de réécriture d’URL d’Apache afin de fixer un seul point d’entrée qui est le fichier index.php. Ce module est un composant essentiel de l’application. Cependant sur le serveur local sur lequel devait être installé l’application, le module de réécriture n’était pas activé.

**J’ai pu obtenir les commandes cli sur bash et les instructions de configuration de apache pour pouvoir activer le module rewrite :**



**#cd mods-enabled**

**#sudo a2enmod rewrite**

## Défaut de rollback de PDO en cas d’échec d’une transaction :

https://stackoverflow.com/questions/23851360/pdocommit-success-or-failure

Certains use-cases nécessitent la manipulation de deux tables ayant une relation cif. Pour respecter les contraintes d’intégrité fonctionnelle plusieurs requêtes SQL doivent être exécutées. Par exemple la mise à jour d’un rôle dont la clé primaire est absorbée par la table groupe nécessite de faire un insert dans role, un update dans groupe puis role et un delete dans role pour respecter les contraintes d’intégrité fonctionnelle à moins de désactiver la vérification des clés étrangères avec « SET FOREIGN\_KEY\_CHECKS = 0; ». L’utilisation d’une transaction est devenue nécessaire afin de s’assurer de la cohérence des données selon les propriétés ACID.

* Atomicité : une transaction se fait au complet
* Cohérence : l’état des données avant et après doivent être valides selon l’ensemble des règles définies(logique métier, cif…)
* Isolation : une transaction est indépendante d’une autre transaction
* Durabilité : une fois une transaction exécutée elle est enregistrée dans la BDD

Bien que la transaction puisse être écrite en dure dans le SQL , PDO propose des méthodes pour le faire qui constituent un choix valide. Cependant, l’absence de rollback par PDO lors d’erreurs mettait à mal les principes des propriétés ACID et donc la validité des données. **J’ai pu trouver la solution à ce problème en mettant PDO en mode ERRMODE\_EXCEPTION.** En effet pour que PDO déclenche le rollback en cas d’erreur, ses méthodes doivent pouvoir retourner une erreur.

## Défaut de permission pour l’utilisation de procédures stockées utilisant du LMD :

https://stackoverflow.com/questions/26015160/deterministic-no-sql-or-reads-sql-data-in-its-declaration-and-binary-logging-i

L’application utilise des triggers mais aussi des procédures et fonctions stockées dans sa conception. Cependant MySQL me renvoyait ce message d’erreur lors de leurs insertions en base de données :



Cette solution a été proposée sur le site Stackoverflow  pour la résolution du problème:

SET GLOBAL log\_bin\_trust\_function\_creators = 1;

## Liste des développeurs :

Christian marais [mywebjob@outlook.fr](mailto:mywebjob@outlook.fr) version 1.9