

Institution des Chartreux

# PAPPE Léger

Epreuve E5

ADAM Emilien

## Contexte

L'entreprise pharmaceutique Galaxy Swiss Bourdin (GSB) est le leader de ce secteur industriel. Suite à la fusion avec le géant américain Galaxay le groupe cherche à optimiser son activité dans ses laboratoires. Pour cela un nouveau service a été créé pour l'entreprise.

## Application

### Description

L'application est une solution afin de pouvoir gérer le patrimoine informatique, d'un ou de plusieurs laboratoires. La solution apportée est une application web permettant de lister, modifier ou voir l'inventaire informatique physique et logiciel de l'entreprise ainsi que de gérer le personnel et les affectations de matériel. L'application est autant utilisable par les administrateurs, les techniciens de l'équipe informatique que par les employés de l'entreprise.

### Technologies

Cette application web utilise le Framework PHP Laravel en version 9.19 avec PHP version 8.0.2. Ce serveur Web communique donc avec un serveur de base de données MariaDB.

### Accès au code

[https://gitlab.com/sco-chartreux/slam-22-23/solo/adam-emilien/gsb\\_web](https://gitlab.com/sco-chartreux/slam-22-23/solo/adam-emilien/gsb_web)

### Tests

Pour tester l'application, plusieurs comptes sont mis à disposition (login:mdp) :

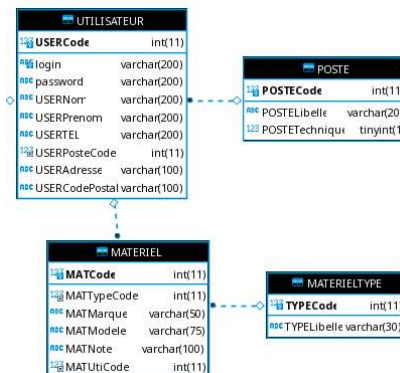
- Compte employé : testEmp:test
- Compte technicien : testTech:test
- Compte administrateur : interdum:aze

Adresse IP du serveur : 10.30.103.94

## Base de données

La base de données est composée de 8 tables et utilisée par deux applications créées pour GSB. Cette application de gestion de patrimoine informatique utilise 4 tables.

Voici ces tables :



La table « Utilisateur » est principale la table qui permet aux utilisateurs de se connecter mais est également la table qui répertorie tous les membres de l'entreprise. Les informations de base des employés y sont également stockées.

Chaque utilisateur à un poste, ces derniers étant stockés dans la table « Poste ». Le lien entre la table utilisateur et la table poste est un lien « 1 – n », un utilisateur à un poste et plusieurs utilisateurs peuvent avoir le même poste. Dans la table des postes est stocké l'intitulé du poste ainsi les permissions d'actions sur l'application associées au poste.

Ensuite, la seconde table importante est la table « Matériel » qui elle permet donc de réaliser et de suivre l'inventaire informatique du laboratoire. Chaque matériel à des informations basiques mais à également un lien « 1 – n » avec un utilisateur. En effet, un matériel est donc attribué à un utilisateur mais un utilisateur peut avoir plusieurs matériaux. De plus, la table « matériel » à un lien « 1 – n » avec une autre table, qui est la table « Matériel Type ». Un matériel à un type de matériel.

Les mots de passes sont hachés dans la base données avec l'algorithme argon2. Argon2 est un algorithme de hachage créée en 2015 qui a gagné la « Password Hashing Competition ». Argon2 est particulièrement résistant face aux attaques de force brute par GPU et il est également optimisée contre les attaques par canal auxiliaire. Le choix a été par la possibilité d'utiliser argon2 directement dans le Framework Laravel et le nombre d'analyses de sécurité réalisées sur l'algorithme.

## Fonctionnalités

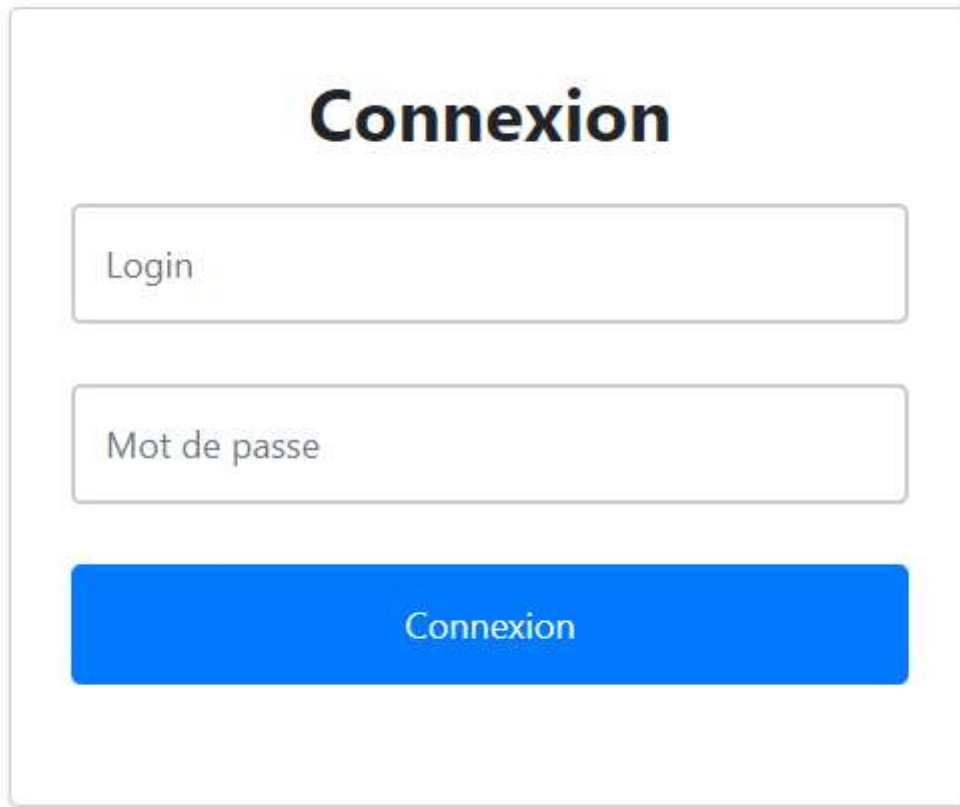
Le but principal de la solution est donc de pouvoir gérer le patrimoine informatique de l'entreprise ou du laboratoire dans lequel est implémenté l'application. Dans cette application, on peut ajouter le matériel informatique de l'entreprise ou du laboratoire. Le matériel fonctionne de la manière suivante : chaque matériel a un type (Route, Ordinateur...). Le matériel est regroupé et affiché par type de matériel. Avec cette application il est également possible de lister tous les utilisateurs, leurs informations et de leur attribuer du matériel qui est inventorié.

A terme l'application permettra d'intégrer un système de ticket qui permettra aux utilisateurs de signaler des dysfonctionnements sur le matériel qu'ils possèdent. Bien que la solution offre la possibilité de créer des nouveaux utilisateurs, dans le futur l'application permettra de lier un Active Directory et d'importer des utilisateurs grâce au protocole LDAP.

## Interface de connexion

Le « début » de l'application est au niveau de la page de connexion. Tous les utilisateurs utilisent la même. Cette interface de connexion se situe à la racine du site web.

Interface de connexion :



The image shows a web form for user login. At the top, the word "Connexion" is displayed in a large, bold, black font. Below it, there are two rectangular input fields. The first field is labeled "Login" in a light gray font. The second field is labeled "Mot de passe" (Password) in a light gray font. Below these fields is a prominent blue button with the word "Connexion" written in white text.

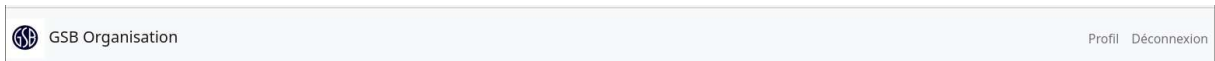
Les utilisateurs rentrent leurs informations dans le formulaire puis accèdent à l'application selon ces données rentrées. L'authentification a été paramétrée dans Laravel pour être utilisée par la table « Utilisateurs » de la base données.

/config/auth.php

```
62 'providers' => [  
63     'users' => [  
64         'driver' => 'eloquent',  
65         'model' => \App\Models\Utilisateur::class,  
66     ],  
]
```

C'est ici le modèle Utilisateur de ma base de données qui est utilisé pour l'authentification de Laravel.

Les utilisateurs peuvent se déconnecter à tout moment à l'aide du bouton de déconnexion présent sur la barre de navigation en haut de page.



```
46 public function logout(Request $request) {  
47     Auth::logout();  
48     return redirect(to: '/');  
49 }
```

Déconnexion de l'utilisateur grâce à la méthode logout de Laravel.

Si les informations de connexion de l'utilisateur sont erronées une erreur apparaît à l'écran.

Les informations de connexion fournies ne correspondent pas à nos enregistrements.

### Connexion

Login

Mot de passe

Connexion

La gestion des erreurs est la même pour tout le code, chaque vue importe la vue 'errors'. Cette vue regarde si une erreur existe dans la session, si oui alors elle est affichée en haut de page.

Contenu de la vue 'errors' :

```
1 @if (session('error'))  
2     <div class="alert alert-danger">  
3         {{ session('error') }}  
4     </div>  
5 @endif
```

Import de la vue 'errors' dans une autre vue :

```
1 @include('header')  
2 @include('errors')  
3 @if($type == "type")  
4  
5     <div class="container">
```

Pour retourner une erreur dans la vue on utilise la façon suivante :

```
return back()->with([
    'error' => 'Les informations de connexion fournies ne correspondent pas à nos enregistrements.',
])->onlyInput('error');
```

On retourne dans le Contrôleur à la page précédente avec comme message le message d'erreur et comme index 'erreur'.

### Barre de navigation

Sur toutes les pages du site se situe en haut une barre de navigation qui permet de naviguer rapidement entre les différentes parties du site. Le logo GSB à la gauche de la barre permet de revenir à la page principale, les boutons à droite permettent donc de se déconnecter, d'accéder à son profil et accéder aux utilisateurs et au matériel.























### Action sur le contenu

Pour tous les différents contenus (utilisateurs, matériel, matériel type) les mêmes options sont proposées. Les captures d'écran de description prennent donc un seul des types de contenu mais la fonctionnalité est la même pour les autres types.

Pour tout le matériel, la première page permet de les tous les lister, avec une pagination.

## Utilisateurs



#	Nom	Prénom	Téléphone	Poste	Action
1	Reiglar	Raimondootetee	146-436-8135	Admin	 
4	Fawckner	Conrade	997-119-8252	Aucun poste	 
6	Carsey	Miguel	216-906-1018	Aucun poste	 
8	Adamou	Aurelie	319-149-1487	Admin	 
9	Duligall	Lindi	799-934-9859	Aucun poste	 
10	Flinn	Hester	849-980-6672	Admin	 
11	Pickervance	Krishnah	869-890-5201	Employé	 
12	Langley	Donavon	941-248-7030	Employé	 
14	Dorbon	Merle	618-269-3787	Technicien	 
15	MacElholm	Melita	316-256-9523	Employé	 

Gestion de la pagination :

```
$users = Utilisateur::paginate(10);

$page = request()->input( key: "page") ?? 1;

if ($page < 1) {
    $page = 1;
}

$maxPages = $users->lastPage();
//ddd($page, $maxPages);
if ($page > $maxPages) {
    $page = $maxPages;
}

return view( view: 'utilisateurs', ["utilisateurs" => $users, "page" => $page, "maxPages" => $maxPages]);
```

Pour la pagination la méthode `paginate` de Laravel est utilisé. Ici, il y a donc 10 utilisateurs par page. Ensuite, on gère l'index de la page selon si on se situe au début de la page ou si le nombre est supérieur au nombre maximum de pages disponibles. On passe ensuite le numéro de la page à la vue pour l'affichage.

Depuis cette interface ou tout le contenu est listé on peut ajouter du matériel, modifier ou supprimer un matériel existant.

Interface de création d'un utilisateur :

### Création d'un utilisateur

Login	<input type="text" value="login"/>
Mot de passe	<input type="password" value="Mot de passe"/>
Nom	<input type="text" value="Nom"/>
Prénom	<input type="text" value="Prénom"/>
Téléphone	<input type="text" value="Téléphone"/>
Poste	<input type="text" value="0 - Aucun poste"/>

CréerAnnuler

Interface de modification :

#### Modification de l'utilisateur : ID: 1 - REIGLAR - Raimondootetee

Code	<input type="text" value="1"/>
Login	<input type="text" value="interdum"/>
Nom	<input type="text" value="Reiglar"/>
Prénom	<input type="text" value="Raimondootetee"/>
Téléphone	<input type="text" value="146-436-8135"/>
Adresse	<input type="text" value="test adresse"/>
Code postal	<input type="text" value="012110"/>
Poste	<input type="text" value="1 - Admin"/>
<input type="button" value="Valider"/>	
<input type="button" value="Annuler"/>	

Si on clique sur la gauche sur l'ID du contenu on accède alors au détail de cette unique composant. Depuis cette interface on peut également accéder aux interfaces de modification et d'édition du composant.



Exemple d'interface de détail :

# 1 - Reiglar

Login:	interdum
Nom:	Reiglar
Prénom:	Raimond
Téléphone:	146-436-8135
Poste occupé:	Admin
Adresse:	test adresse
Code Postal:	012110




## Matériel attribué

Tout désaffecter

#	Type	Marque	Modèle
<a href="#">1</a>	Souris	Dell	
<a href="#">2</a>	Ordinateur portable	Dell	
<a href="#">3</a>	Clavier	Lenovo	
<a href="#">5</a>	Ecran	Dell	
<a href="#">7</a>	Ordinateur fixe	Samsung	
<a href="#">9</a>	Ordinateur portable	Samsung	

## Profil personnel

Les utilisateurs peuvent accéder à la page 'profil', sur laquelle ils voient les informations de leur compte ainsi que tout le matériel qui leur est attribué.

 GSB Organisation

Utilisateurs Matériel Profil Déconnexion

### Profil

Admin

Raimondootete Reiglar

Téléphone: 146-436-8135

Adresse: test adresse

Code postal: 012110

### Votre matériel

#	Type	Marque	Modèle
1	Souris	Dell	
2	Ordinateur portable	Dell	
3	Sans type	Lenovo	
5	Ecran	Dell	
7	Ordinateur fixe	Samsung	
9	Ordinateur portable	Samsung	
11	Ecran	Samsung	
12	Clavier	Fujitsu	

## Rôles et permissions

Comme dis précédemment lors de la description de l'application, différents types d'employés peuvent se connecter à l'application. Cette dernière utilise des permissions selon le poste pour donner accès ou non à certaines données. Les employés « basiques », les techniciens et les administrateurs sont les différents rôles pouvant accéder à la solution à travers la même interface.

Les employés sans permissions particulières peuvent se connecter à l'application à travers la page de connexion basique. Ensuite, ces utilisateurs n'ont accès qu'à leur page de profil qui montre leurs informations basiques et peuvent voir le matériel informatique qui leur est attribué. Les employés ne peuvent donc pas accéder à la liste complète du matériel, ils ne peuvent pas non plus l'altérer et ne peuvent pas accéder aux informations d'autres employés. L'idée derrière cela est que les employés peuvent s'assurer que le matériel qui est listé dans l'application est correct et cela leur permet également d'avoir une preuve de quel matériel et le leur. A terme l'application offrirait aux employés la possibilité de demander d'emprunter certain matériel nécessaire à leur fonction ou de signaler un dysfonctionnement nécessitant l'intervention d'un technicien.

Les techniciens eux ont accès à la liste de tous les utilisateurs, des types de matériel et de tout le matériel. Ils peuvent également lister les informations d'un unique utilisateur ou d'un unique matériel, mais ils n'ont cependant pas de permissions de modifications, éditions ou suppression sur ces éléments. Les techniciens ont également accès leur profil avec leurs informations personnelles ainsi que le matériel informatique qui leur est attribué. L'idée derrière cela est de pouvoir permettre aux techniciens d'être rapidement au courant de l'état du parc informatique mais également de pouvoir facilement intervenir sur un matériel en cas de problème rapporté par un utilisateur. A terme l'application proposerait un système de ticket qui permettrait aux techniciens de connaître les problèmes en cours sur certaines parties du parc informatique ou pour certains employés.

Vu d'un technicien :

### Liste Routeurs

#	Marque	Modèle	Note
30	NetGear		Pellentesque in dolor id lacus.
37	NetGear		Pellentesque in dolor id lacus.
44	D-Link		Pellentesque in dolor id lacus.
51	D-Link		Pellentesque in dolor id lacus.
58	TP-Link		Pellentesque in dolor id lacus.

Enfin, les administrateurs eux ont les pleins droits sur l'application, c'est-à-dire qu'ils peuvent accéder à tous les utilisateurs, tout le matériel, tous les types de matériel et peuvent agir dessus, en les modifiant, en en créant, en en supprimant ou encore en affectant ou désaffectant du matériel un utilisateur. Les captures précédentes ont été prises depuis un compte administrateur.

## Middlewares

L'application Laravel utilise des Middlewares pour gérer l'authentification et les permissions. Ces middlewares sont des fonctions qui sont appelées automatiquement lorsque des actions sont réalisées sur l'application.

Pour l'authentification, le middleware est appliqué sur chaque route. Sur chacune de ces dernières, à chaque accès sur cette route, avant d'autoriser l'accès le middleware vérifie que l'utilisateur est bien authentifié et que sa session est encore valide. Si elle ne l'est pas alors il le redirige vers la page

d'authentification. Ce middleware est un middleware de base de Laravel qui permet de gérer l'authentification de l'application.

Exemple d'application du middleware d'authentification sur la route « /overview ».

```
21 Route::get( uri: '/overview', [\App\Http\Controllers\MaterieltypeController::class, "showAll"])
22     ->middleware( middleware: 'auth')
23     ->middleware( middleware: 'permissions');
```

Redirection si l'utilisateur n'est pas authentifié dans la session

```
15 protected function redirectTo($request)
16 {
17     if (! $request->expectsJson()) {
18         return route( name: 'main');
19     }
20 }
```

Le second middleware lui permet de gérer les permissions des utilisateurs sur les différentes parties de l'application. Ce middleware est également appliqué à toutes les routes du site web, pour qu'à chaque requête il vérifie si l'utilisateur a les permissions d'accéder à cette page ou non (permissions expliquées précédemment). Voici le code vérifiant les accès :

```
18 public function handle(Request $request, Closure $next, $roles)
19 {
20     if ($roles == "") {
21         return $next($request);
22     }
23
24     $rolesArray = explode( separator: ';', $roles);
25     $user = Auth::user();
26     $role = $user->POSTE->POSTELibelle;
27
28     foreach($rolesArray as $oneRole) {
29         if ($role == $oneRole) {
30             return $next($request);
31         }
32     }
33
34     return redirect()->back()->with('error', 'Vous n'êtes pas autorisé à accéder à cette page');
35 }
36 }
```

La fonction attend en paramètre les postes ayant l'autorisation d'accéder à la route donnée. On compare alors ces rôles autorisés avec le rôle du compte authentifié en session et l'on redirige ou non l'utilisateur. Voici un exemple de comment appliquer le middleware à une route. Ici, les rôles Admin et Technicien sont autorisés à accéder à la route '/overview' :

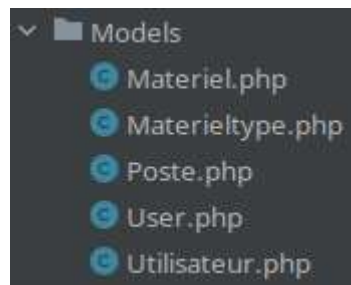
```
20 Route::get( uri: '/overview', [\App\Http\Controllers\MaterieltypeController::class, "showAll"])
21     ->middleware( middleware: 'auth')
22     ->middleware( middleware: 'permissions:Admin;Technicien');
```

Ensuite, à l'intérieur des vues même pour voir si les utilisateurs ont le droit de réaliser des actions ou non sur le contenu, on regarde simplement la valeur du chiffre associé à leur poste.

```
<tr>
    <th scope="col" class="text-center">#</th>
    <th scope="col" class="text-center">Type</th>
    <th scope="col" class="text-center">Nombre de matériel</th>
    @if(Auth::getUser()->POSTE->POSTETechnique > 1)
        <th scope="col" class="text-center">Action</th>
    @endif
</tr>
```

## VMC

Toute l'application utilise le principe de Vue – Modèle – Controller. Pour donc chaque table, un modèle lui est associé



Les Controller permettent de réaliser des actions et de récolter les données à afficher sur la page, la vue. Chaque route a donc un Controller associé qui traite les données qui vont ensuite être affichés ou ajoutés ou altérées dans la base de données.

```
Route::get( uri: '/overview', [\App\Http\Controllers\MaterieltypeController::class, "showAll"])
->middleware( middleware: 'auth')
->middleware( middleware: 'permissions');
```

Ici, la après une requête sur la route /overview, c'est la fonction « showAll » du Controller MaterieltypeController qui sera appelée et qui agira.

La capture suivante est un exemple de contenu de fonction de Controller, qui ici supprime un matériel selon son ID et redirige ensuite l'utilisateur.

```
96 function delete($id) {
97
98     $materiel = Materiel::find($id);
99
100     if ( ! isset($materiel->materielType) ) {
101
102         $materiel->delete();
103
104         return redirect( to: '/overview/sansType');
105
106     }
107
108     $materiel->delete();
109
110     return redirect( to: '/overview/' . $materiel->materielType->TYPELibelle);
111 }
```

## Gestion des erreurs

A travers toute l'application la gestion des erreurs est la même, dans chaque vue on importe une vue nommée erreur qui gère et affiche de potentielles erreurs.

```
@if (session('error'))
    <div class="alert alert-danger">
        {{ session('error') }}
    </div>
@endif
```

Pour retourner les erreurs dans la session on utilise dans les Controller la méthode back() de Laravel qui renvoie à la page précédente avec dans la session l'erreur et le message de notre choix.

```
return back()->with([
    'error' => 'Les informations de connexion fournies ne correspondent pas à nos enregistrements.',
])->onlyInput('error');
```

## Hachage des mots de passe

Le hachage des mots de passe est une fonctionnalité obligatoire de l'application afin de garantir la confidentialité des mots de passe des utilisateurs. Dans l'application l'algorithme de hachage utilisé est argon2 préféré à bcrypt. Le fichier /config/hashing.php a été modifier pour cela.

```
'driver' => 'argon',
```

Lors de la création d'utilisateurs, on utilise alors la méthode Hash::make de Laravel pour hacher le mot de passe.

```
$newUser->password = Hash::make( $password );
```

## Eloquent

A travers l'application Laravel, toute la communication avec la base de données est réalisée par Eloquent. Grâce aux 'models' expliqués précédemment, Laravel accède aux tables de la base de données et construit directement les liens entre les différents 'models' et crée donc lui-même les collections nécessaires par exemple. Ici, le lien est fait entre la table utilisateur et la table matériel grâce à la méthode 'hasOne' dans laquelle on passe en argument les noms des clés qui font la relation, un matériel est relié à un seul utilisateur à qui il est affecté.

```
class Materiel extends Model
{
    use HasFactory;

    protected $table = "MATERIEL";
    protected $primaryKey = "MATCode";

    public $timestamps = false;

    function materielType() {
        return $this->hasOne(related: Materieltype::class, foreignKey: "TYPECode", localKey: "MATTypeCode");
    }

    function utilisateur() {
        return $this->hasOne(related: Utilisateur::class, foreignKey: "USERCode", localKey: "MATUtiCode");
    }
}
```