



Accompagnement – Formation – Audit des systèmes de management
Intégrateur de solution /Interpréariat - traduction / Librairie

DOCUMENTAIRE CODE SOURCE

Coherence Risk - CRM

Sommaire

- 1. Introduction**
- 2. Installation**
 - 2.1. Prérequis
 - 2.2. Installation locale
- 3. Explication des fichiers principaux du projet**
- 4. Modèles (Models)**
 - 4.1. Description des modèles
 - 4.2. Relations entre les modèles
- 5. Contrôleurs (Controller)**
 - 5.1. Description des contrôleurs
 - 5.2. Route associée
- 6. Vues (View)**
- 7. Evènements (Event)**
 - 7.1. Liste des évènement utilisés
 - 7.2. Fonctionnement
- 8. Base de données**

1. Introduction

Bienvenue dans le documentaire du code source de notre application dédiée à la gestion des risques, des non-conformités, des réclamations et des incidents. Dans ce document, nous allons explorer en détail les différentes parties de notre application, en mettant en lumière les fonctionnalités essentielles et la logique de programmation de notre solution logicielle.

- **Objectif de l'Application**

Notre application a pour objectif principal d'automatiser la gestion des risques associés à vos processus, tout en fournissant des outils puissants pour gérer efficacement les non-conformités, les réclamations et les incidents. En outre, elle vise à évaluer l'efficacité des actions mises en place pour faire face à ces risques, en offrant des fonctionnalités robustes de suivi et de rapport.

- **Gestion des Risques**

La gestion des risques est un aspect crucial de toute entreprise. Notre application fournit des fonctionnalités avancées pour identifier, évaluer et gérer les risques potentiels associés à vos processus. Grâce à des outils d'analyse intégrés et à des mécanismes de suivi, vous pouvez prendre des décisions éclairées pour atténuer les risques et renforcer la résilience de votre organisation.

- **Gestion des Non-conformités, Réclamations et Incidents**

En plus de la gestion des risques, notre application offre des fonctionnalités complètes pour gérer les non-conformités, les réclamations et les incidents. Que ce soit pour suivre les incidents en temps réel, gérer les réclamations des clients ou identifier les non-conformités dans vos processus, notre application vous permet de gérer efficacement ces aspects critiques de votre entreprise.

- **Évaluation de l'Efficacité des Actions**

Évaluer l'efficacité des actions mises en place face aux risques est essentiel pour une gestion proactive des risques. Notre application fournit des mécanismes intégrés pour suivre et évaluer l'impact des actions correctives et préventives prises pour atténuer les risques identifiés. Cela vous permet de prendre des décisions informées pour améliorer continuellement vos processus et renforcer la résilience de votre organisation.

- **Structure du Document**

Dans les sections suivantes, nous explorerons en détail chaque composant de notre application, en mettant en évidence la logique de programmation, les fonctionnalités clés et les meilleures pratiques utilisées pour garantir la robustesse et la fiabilité de notre solution logicielle.

2. Installation et configuration

2.1. Prérequis

Avant d'utiliser l'application Coherence Risk – CRM en locale, assurez-vous d'avoir les prérequis nécessaires en place tels que :

- PHP
- Composer
- Node.js ou NPM
- Serveur WEB (WampServer ou Xamp)
- MySQL
- Git

2.2. Installation locale

Installer localement L'application Coherence Risk – CRM implique plusieurs étapes. Assurez-vous d'avoir Git, Composer et PHP installés sur votre machine avant de commencer. Voici les étapes générales :

- **Cloner le projet depuis Git**

Ouvrez votre invite de commande et taper :

```
git clone https://github.com/Davidkouachi/coherence.git
```

- **Accéder au répertoire du projet**

Toujours dans l'invite de commande taper :

```
Cd coherence
```

- **Installer les dépendances de l'application avec Composer**

Toujours dans l'invite de commande taper :

```
Composer Install
```

- **Création du fichier .env**

- Aller à la racine du projet

- Créer un fichier nommer .env sans extension

- Ouvrir le fichier env.txt

- Copier les données du fichier (Ctrl + C)

- Ensuite coller les données dans le fichier .env que vous avez créer

- **Générer la clé d'application**

Taper dans l'invite de commande :

```
Php artisan key : generate
```

- **Configurer la base de données**

- Ouvrez votre serveur web

- Créer une base de données nommer coherence

- **Migration de la base de données**

Tapez dans l'invite de commande :

- Php artisan migrate
- **Migration des seeders**
Taper dans l'invite de commande :
Php artisan migrate : refresh –seed
 - **Lien Storage**
Taper dans l'invite de commande :
Php artisan Storage : Link
 - **Lancer le serveur de développement**
Taper dans l'invite de commande :
Php artisan serve

Vous pouvez désormais avoir accès à l'application via l'adresse <http://127.0.0.1:8000> ou <http://localhost:8000>

3. Explication des fichiers principaux du projet

Les dossiers de l'application Coherence Risk - CRM suit une structure bien définie pour faciliter l'organisation du code source. Voici une explication des principaux dossiers et de leur objectif dans ladite application :

- **App** : Ce dossier contient le code source de l'application Laravel, y compris les modèles (Models), les contrôleurs (Controllers), les middlewares, et d'autres classes PHP spécifiques au projet.
- **Config** : Contient les fichiers de configuration de l'application, y compris les fichiers de configuration pour la base de données, les services, les sessions
- **Public** : Ce dossier est le point d'entrée de l'application. Il contient les fichiers publics tels que les fichiers CSS, JavaScript, les images, les fichiers de polices
- **Database** : Contient les migrations de base de données, les seeders (alimentateurs de données pour les bases de données), et les factories pour générer des jeux de données de test.
- **Ressources** : Contient les fichiers non exécutables comme les fichiers Blade pour les vues, les fichiers de langues, les fichiers Sass, les fichiers JavaScript non compilés :
 - Resources/View : Contient les fichiers Blade qui représentent les vues de l'application.
 - Resources/Lang : Contient les fichiers de langues pour l'internationalisation
- **Routes** : Contient les fichiers de définition des routes de l'application
- **Storage** : Contient les fichiers générés par l'application, tels que les fichiers de logs, les fichiers de cache, les sessions, etc.
- **Tests** : Contient les fichiers de tests unitaires et d'intégration.
- **Vendor** : Contient les dépendances installées par Composer, le gestionnaire de dépendances de PHP.
- **.env** :

Fichier d'environnement qui stocke les variables d'environnement telles que les informations de base de données, les clés secrètes, etc.

- **Artisan** : Le script artisan, qui fournit une interface en ligne de commande pour diverses tâches de développement.
- **Composer.json et Composer.lock** : Les fichiers de configuration pour Composer, qui gère les dépendances de l'application.

4. Modèles (Models)

4.1. Description des modèles

4.1.1. Poste

Le model Poste représente les différents postes présents dans chaque structure qui utilise l'application. Les principaux attributs sont :

```
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class Poste extends Model
9 {
10     use HasFactory;
11
12     protected $fillable = [
13         'id',
14         'nom',
15         'occupe',
16     ];
17 }
18
```

- Id = l'identifiant du poste
- Nom = le nom du poste
- Occupe = paramètre permettant de savoir si le poste est déjà occupé ou pas

Configuration de la migration Poste :

Chemin : Database/migrations/ 2013_11_09_084133_create_postes_table.php

```
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 class CreatePostesTable extends Migration
8 {
9     /**
10      * Run the migrations.
11      *
12      * @return void
13      */
14     public function up()
15     {
16         Schema::create('postes', function (Blueprint $table) {
17             $table->id();
18             $table->string('nom');
19             $table->string('occupe');
20             $table->timestamps();
21         });
22     }
23
24     /**
25      * Reverse the migrations.
26      *
27      * @return void
28      */
29     public function down()
30     {
31         Schema::dropIfExists('postes');
32     }
33 }
```

4.1.2. User

Le model User représente les utilisateurs de l'application. Les principaux attributs sont :

```
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Contracts\Auth\MustVerifyEmail;
6 use Illuminate\Database\Eloquent\Factories\HasFactory;
7 use Illuminate\Foundation\Auth\User as Authenticatable;
8 use Illuminate\Notifications\Notifiable;
9 use Laravel\Sanctum\HasApiTokens;
10 use Illuminate\Support\Facades\Hash;
11
12 class User extends Authenticatable
13 {
14     use HasApiTokens, HasFactory, Notifiable, HasApiTokens;
15
16     protected $fillable = [
17         'id',
18         'name',
19         'email',
20         'password',
21         'matricule',
22         'poste_id',
23         'suivi_active',
24         'mdp_date',
25         'fa',
26         'tel',
27     ];
28     public function poste() {
29         return $this->belongsTo(Poste::class, 'poste_id');
30     }
31     protected $hidden = [
32         'password',
33         'remember_token',
34     ];
35     protected $casts = [
36         'email_verified_at' => 'datetime',
37     ];
38     public function logoutOtherDevices($password)
39     {
40         if (Hash::check($password, $this->password)) { $this->tokens->each(function ($token, $key) {
41             if ($key != $this->currentAccessToken()->id) { $token->delete(); }});
42         }
43     }
44 }
```

- Id = l'identifiant de l'utilisateur
- Name = le nom et prénom de l'utilisateur
- Email = l'email de l'utilisateur
- Password = le mot de passe de l'utilisateur
- Matricule = le matricule de l'utilisateur
- Poste_id = l'identifiant du poste auquel l'utilisateur est affecté
- Suivi_active = paramètre de suivi d'action de l'utilisateur dans l'application (Save, update, delete et search)
- Mdp_date = date de modification du mot de passe de l'utilisateur
- Fa = paramètre d'identification à double facteur (pas encore employé dans l'application)
- Tel = contact de l'utilisateur

Configuration de la migration User :

Chemin : Database/migrations/ 2014_10_11_000000_create_users_table.php

```
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  class CreateUsersTable extends Migration
8  {
9      /**
10      * Run the migrations.
11      *
12      * @return void
13      */
14     public function up()
15     {
16         Schema::create('users', function (Blueprint $table) {
17             $table->id();
18             $table->string('name');
19             $table->string('email')->unique();
20             $table->timestamp('email_verified_at')->nullable();
21             $table->string('password');
22             $table->unsignedBigInteger('poste_id');
23             $table->foreign('poste_id')->references('id')->on('postes');
24             $table->string('tel');
25             $table->string('matricule')->unique();
26             $table->string('suivi_active');
27             $table->datetime('mdp_date')->nullable();
28             $table->string('fa');
29             $table->rememberToken();
30             $table->timestamps();
31         });
32     }
33
34     /**
35     * Reverse the migrations.
36     *
37     * @return void
38     */
39     public function down()
40     {
41         Schema::dropIfExists('users');
42     }
43 }
44 }
```

4.1.3. Autorisation

Le model Autorisation représente les accès d'un utilisateur aux différentes pages de l'application. Les principaux attributs sont :

```
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class Autorisation extends Model
9  {
10     use HasFactory;
11
12     protected $fillable = [
13         'id',
14         'new_user',
15         'list_user',
16         'new_poste',
17         'list_poste',
18         'historiq',
19         'stat',
20         'new_proces',
21         'list_proces',
22         'eva_proces',
23         'new_risk',
24         'list_risk',
25         'val_risk',
26         'act_n_val',
27         'color_para',
28         'suivi_actp',
29         'list_actp',
30         'suivi_actc',
31         'list_actc_eff',
32         'list_actc',
33         'fiche_am',
34         'list_am',
35         'val_am',
36         'am_n_val',
37         'user_id',
38     ];
39     public function user()
40     {
41         return $this->belongsTo(User::class, 'user_id');
42     }
43 }
```

- Id = l'identifiant de l'autorisation
- New_user = Accès à la page nouveau utilisateur
- List_user = Accès à la page liste des utilisateurs
- New_poste = Accès à la page nouveau poste
- Liste_poste = Accès à la page liste des postes
- Historiq = Accès à la page historique
- Stat = Accès à la page des statistiques
- New_proces = Accès à la page nouveau processus
- List_proces = Accès à la page liste des processus
- Eva_proces = Accès à la page évaluation des processus
- New_risk = Accès à la page nouveau risque
- List_risk = Accès à la page liste des risques
- Val_risk = Accès à la page tableau de validation des risques

- Act_n_val = Accès à la page Risques non validés
- Color_para = Accès à la page paramétrage des couleurs
- Suivi_actp = Accès à la page suivi des actions préventives
- List_actp = Accès à la page liste des actions préventives
- Suivi_actc = Accès à la page suivi des actions correctives
- List_actc_eff = Accès à la page liste des actions correctives effectuées
- List_actc = Accès à la page liste des actions correctives
- Fiche_am = Accès à la page fiche de résolution d'incident
- List_am = Accès à la page liste des incidents
- Val_am = Accès à la page validations des incidents
- Am_n_val = Accès à la page incidents non validés
- User_id = l'identifiant de l'utilisateur liées à ces différents accès

Configuration de la migration Autorisation :

Chemin : Database/migrations/ 2023_12_07_223510_create_autorisations_table.php

```

1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  class CreateAutorisationsTable extends Migration
8  {
9      public function up()
10     {
11         Schema::create('autorisations', function (Blueprint $table) {
12             $table->id();
13             $table->string('new_user');
14             $table->string('list_user');
15             $table->string('new_poste');
16             $table->string('list_poste');
17             $table->string('historiq');
18             $table->string('stat');
19             $table->string('new_proces');
20             $table->string('list_proces');
21             $table->string('eva_proces');
22             $table->string('new_risk');
23             $table->string('list_risk');
24             $table->string('val_risk');
25             $table->string('act_n_val');
26             $table->string('color_para');
27             $table->string('suivi_actp');
28             $table->string('list_actp');
29             $table->string('suivi_actc');
30             $table->string('list_actc_eff');
31             $table->string('list_actc');
32             $table->string('fiche_am');
33             $table->string('list_am');
34             $table->string('val_am');
35             $table->string('am_n_val');
36             $table->unsignedBigInteger('user_id');
37             $table->foreign('user_id')->references('id')->on('users');
38             $table->timestamps();
39         });
40     }
41     public function down()
42     {
43         Schema::dropIfExists('autorisations');
44     }
}

```

4.1.4. Processuse

Le model processuse représente le processus de chaque structure qui utilise l'application. Les principaux attributs sont :

```
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class Processuse extends Model
9 {
10     use HasFactory;
11
12     protected $fillable = [
13         'id',
14         'nom',
15         'description',
16         'finalite',
17     ];
18 }
```

- Id = l'identifiant du processus
- Nom = le nom du processus
- Description = la description du processus
- Finalite = la finalité du processus

Configuration de la migration Processuse :

Chemin : Database/migrations/ 2023_10_10_114949_create_processeses_table.php

```

1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 class CreateProcessusesTable extends Migration
8 {
9     /**
10      * Run the migrations.
11      *
12      * @return void
13      */
14     public function up()
15     {
16         Schema::create('processuses', function (Blueprint $table) {
17             $table->id();
18             $table->string('nom');
19             $table->text('description');
20             $table->text('finalite');
21             $table->timestamps();
22         });
23     }
24
25     /**
26      * Reverse the migrations.
27      *
28      * @return void
29      */
30     public function down()
31     {
32         Schema::dropIfExists('processuses');
33     }
34 }
35

```

4.1.5. Objectif

Le model objectif représente les objectifs de chaque processus défini dans l'application. Les principaux attributs sont :

```

1 <?php
2
3 namespace App\Models;
4
5 use App\Models\Processus;
6
7 use Illuminate\Database\Eloquent\Factories\HasFactory;
8 use Illuminate\Database\Eloquent\Model;
9
10 class Objectif extends Model
11 {
12     use HasFactory;
13
14     protected $fillable = [
15         'id',
16         'nom',
17         'processus_id',
18     ];
19
20     public function processus()
21     {
22         return $this->belongsTo(Processus::class, 'processus_id');
23     }
24 }
25

```

- Id = l'identifiant de l'objectif
- Nom = le nom de l'objectif

- Processus_id = l'identifiant du processus liées a cet objectif

Configuration de la migration objectif :

Chemin : Database/migrations/ 2023_10_10_115038_create_objectifs_table.php

```

1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 class CreateObjectifsTable extends Migration
8 {
9     /**
10      * Run the migrations.
11      *
12      * @return void
13      */
14     public function up()
15     {
16         Schema::create('objectifs', function (Blueprint $table) {
17             $table->id();
18             $table->string('nom');
19             $table->unsignedBigInteger('processus_id');
20             $table->foreign('processus_id')->references('id')->on('processuses');
21             $table->timestamps();
22         });
23     }
24
25     /**
26      * Reverse the migrations.
27      *
28      * @return void
29      */
30     public function down()
31     {
32         Schema::dropIfExists('objectifs');
33     }
34 }
```

4.1.6. Pdf_file_processus

Le model pdf_file_processus représente le fichier PDF ajouter a l'enregistrement d'un processus. Les principaux attributs sont :

```
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class Pdf_file_processus extends Model
9 {
10     use HasFactory;
11
12     protected $fillable = [
13         'id',
14         'pdf_nom',
15         'pdf_chemin',
16         'processus_id',
17     ];
18
19     public function processus()
20     {
21         return $this->belongsTo(Processuse::class, 'processus_id');
22     }
23 }
24
```

- Id = l'identifiant de fichier PDF
- Pdf_nom = le nom du fichier PDF
- Pdf_chemin = le chemin du fichier PDF
- Processus_id = l'identifiant du processus auquel le fichier PDF est associé

Configuration de la migration pdf_file_processus :

Chemin : Database/migrations/ 2023_11_18_212815_create_pdf_file_processeses_table.php

```
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 class CreatePdfFileProcessusesTable extends Migration
8 {
9     /**
10      * Run the migrations.
11      *
12      * @return void
13      */
14     public function up()
15     {
16         Schema::create('pdf_file_processuses', function (Blueprint $table) {
17             $table->id();
18             $table->string('pdf_nom');
19             $table->string('pdf_chemin');
20             $table->unsignedBigInteger('processus_id');
21             $table->foreign('processus_id')->references('id')->on('processuses');
22             $table->timestamps();
23         });
24     }
25
26     /**
27      * Reverse the migrations.
28      *
29      * @return void
30      */
31     public function down()
32     {
33         Schema::dropIfExists('pdf_file_processuses');
34     }
35 }
36
```

4.1.7. Color_para

Le model color_para représente le paramétrage de base des différentes couleurs et intervalles dans l'application. Les principaux attributs sont :

```
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class Color_para extends Model
9 {
10     use HasFactory;
11
12     protected $fillable = [
13         'id',
14         'nbre0',
15         'nbre1',
16         'nbre2',
17         'nbre_color',
18         'operation',
19     ];
20 }
21
```

- Id = l'identifiant du paramètre
- Nbre0 = le nombre sur lequel l'on s'appuie pour apporter des modifications au niveau du paramétrage de base
- Nbre1 = le nombre initial
- Nbre2 = le nombre limite
- Nbre_color = nombre total d'intervalle ou de couleur
- Operation = l'opération qui sera effectuée dans chaque calcul

Configuration de la migration color_para :

Chemin : Database/migrations/ 2024_01_05_130019_create_color_paras_table.php

```

1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 class CreateColorParasTable extends Migration
8 {
9     /**
10      * Run the migrations.
11      *
12      * @return void
13      */
14     public function up()
15     {
16         Schema::create('color_paras', function (Blueprint $table) {
17             $table->id();
18             $table->integer('nbre0');
19             $table->integer('nbre1');
20             $table->integer('nbre2');
21             $table->integer('nbre_color');
22             $table->string('operation');
23             $table->timestamps();
24         });
25     }
26
27     /**
28      * Reverse the migrations.
29      *
30      * @return void
31      */
32     public function down()
33     {
34         Schema::dropIfExists('color_paras');
35     }
36 }
37

```

4.1.8. Color_interval

Le model color_interval représente un intervalle d'évaluation en fonction du paramétrage de base défini. Les principaux attributs sont :

```

1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class Color_interval extends Model
9 {
10     use HasFactory;
11
12     protected $fillable = [
13         'id',
14         'nbre1',
15         'nbre2',
16         'color',
17         'code_color',
18     ];
19 }
20

```

- Id = l'identifiant de l'intervalle
- Nbre1 = le nombre de départ de l'intervalle
- Nbre2 = le nombre d'arriver de l'intervalle

- Color = la couleur de l'intervalle
- Code_color = le code de la couleur de l'intervalle

Configuration de la migration color_interval :

Chemin : Database/migrations/ 2024_01_05_130814_create_color_intervals_table.php

```
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 class CreateColorIntervalsTable extends Migration
8 {
9     /**
10      * Run the migrations.
11      *
12      * @return void
13      */
14     public function up()
15     {
16         Schema::create('color_intervals', function (Blueprint $table) {
17             $table->id();
18             $table->integer('nbre1');
19             $table->integer('nbre2');
20             $table->string('color');
21             $table->string('code_color');
22             $table->timestamps();
23         });
24     }
25
26     /**
27      * Reverse the migrations.
28      *
29      * @return void
30      */
31     public function down()
32     {
33         Schema::dropIfExists('color_intervals');
34     }
35 }
36 }
```

4.1.9. Risques

Le model risque représente le risque de chaque structure qui utilise l'application. Les principaux attributs sont :

```
1 <?php
2
3 namespace App\Models;
4
5 use App\Models\Processus;
6
7 use Illuminate\Database\Eloquent\Factories\HasFactory;
8 use Illuminate\Database\Eloquent\Model;
9
10 class Risque extends Model
11 {
12     use HasFactory;
13
14     protected $fillable = [
15         'id',
16         'nom',
17         'page',
18         'vraisemblance',
19         'gravite',
20         'evaluation',
21         'cout',
22         'vraisemblance_residuel',
23         'gravite_residuel',
24         'evaluation_residuel',
25         'cout_residuel',
26         'date_validation',
27         'processus_id',
28         'statut',
29         'traitement',
30         'poste_id',
31     ];
32
33     public function processus()
34     {
35         return $this->belongsTo(Processus::class, 'processus_id');
36     }
37
38     public function poste()
39     {
40         return $this->belongsTo(Poste::class, 'poste_id');
41     }
42 }
43
```

- Id = l'identifiant du risque
- Nom = le nom du risque
- Page = paramètre permettant de faire la différence entre les risques enregistrer avant et pendant l'incident
- Vraisemblance = indicateur d'évaluation 1
- Gravite = indicateur d'évaluation 2
- Evaluation = le résultat de l'opération (opération est défini dans le paramétrage des couleurs) effectuée entre la vraisemblance et la gravite
- Cout = la somme qui sera dépenser si et seulement si le risque survient
- Vraisemblance_residuel = indicateur d'évaluation résiduel 1
- Gravite_residuel = indicateur d'évaluation résiduel 2
- Evaluation_residuel = le résultat de l'opération (opération est défini dans le paramétrage des couleurs) effectuée entre la vraisemblance résiduelle et la gravite résiduel

- Cout_residuel = la somme qui sera dépenser si le risque survient et après avoir mener à bien les actions préventives
- Date_de_validation = la date a laquelle la ficher risque a été validé
- Processus_id = l'identifiant du processus auquel le risque est lié
- Staut = le niveau de traitement de la fiche risque (valider, en attente de validation, etc...)
- Traitement = l'objectif finale si le risque arrive (réduire le risque, accepter le risque, etc...)
- Poste_id = l'identifiant du poste charger de la validation de la fiche risque

Configuration de la migration risque :

Chemin : Database/migrations/ 2023_10_11_225439_create_risques_table.php

```

1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  class CreateRisquesTable extends Migration
8  {
9      public function up()
10     {
11         Schema::create('risques', function (Blueprint $table) {
12             $table->id();
13             $table->string('nom');
14             $table->string('page');
15             $table->integer('vraisemblance')->nullable();
16             $table->integer('gravite')->nullable();
17             $table->integer('evaluation')->nullable();
18             $table->string('cout')->nullable();
19             $table->string('traitement')->nullable();
20             $table->string('statut')->nullable();
21             $table->integer('vraisemblance_residuel')->nullable();
22             $table->integer('gravite_residuel')->nullable();
23             $table->integer('evaluation_residuel')->nullable();
24             $table->string('cout_residuel')->nullable();
25             $table->datetime('date_validation')->nullable();
26             $table->unsignedBigInteger('processus_id');
27             $table->foreign('processus_id')->references('id')->on('processuses');
28             $table->unsignedBigInteger('poste_id');
29             $table->foreign('poste_id')->references('id')->on('postes');
30             $table->timestamps();
31         });
32     }
33
34     public function down()
35     {
36         Schema::dropIfExists('risques');
37     }
38 }
39

```

4.1.10. Causes

Le model cause représente les causes probables de chaque risque. Les principaux attributs sont :

```
1 <?php
2
3 namespace App\Models;
4
5 use App\Models\Risque;
6 use App\Models\Resva;
7
8 use Illuminate\Database\Eloquent\Factories\HasFactory;
9 use Illuminate\Database\Eloquent\Model;
10
11 class Cause extends Model
12 {
13     use HasFactory;
14
15     protected $fillable = [
16         'id',
17         'nom',
18         'page',
19         'dispositif',
20         'risque_id',
21     ];
22
23     public function risque()
24     {
25         return $this->belongsTo(Risque::class, 'risque_id');
26     }
27 }
28
29 }
```

- Id = l'identifiant de la cause
- Nom = le nom de la cause
- Page = paramètre permettant de faire la différence entre les causes liées à des risques enregistrer avant et pendant l'incident
- Dispositif = le dispositif de contrôle de la cause
- Risque_id = l'identifiant du risque auquel la cause est liée

Configuration de la migration cause :

Chemin : Database/migrations/ 2023_10_11_225524_create_causes_table.php

```

1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 class CreateCausesTable extends Migration
8 {
9     /**
10      * Run the migrations.
11      *
12      * @return void
13      */
14     public function up()
15     {
16         Schema::create('causes', function (Blueprint $table) {
17             $table->id();
18             $table->string('nom');
19             $table->string('dispositif')->nullable();
20             $table->string('page');
21             $table->unsignedBigInteger('risque_id');
22             $table->foreign('risque_id')->references('id')->on('risques');
23             $table->timestamps();
24         });
25     }
26
27     /**
28      * Reverse the migrations.
29      *
30      * @return void
31      */
32     public function down()
33     {
34         Schema::dropIfExists('causes');
35     }
36 }
37

```

4.1.11. Pdf_file

Le model Pdf_file représente le fichier PDF ajouter à l'enregistrement d'un risque. Les principaux attributs sont :

```

1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class Pdf_file extends Model
9 {
10     use HasFactory;
11
12     protected $fillable = [
13         'id',
14         'pdf_nom',
15         'pdf_chemin',
16         'risque_id',
17     ];
18
19     public function risque()
20     {
21         return $this->belongsTo(Risque::class, 'risque_id');
22     }
23 }
24

```

- Id = l'identifiant de fichier PDF
- Pdf_nom = le nom du fichier PDF
- Pdf_chemin = le chemin du fichier PDF
- Risque_id = l'identifiant du risque auquel le fichier PDF est associé

Configuration de la migration Pdf_file :

Chemin : Database/migrations/ 2023_10_23_162437_create_pdf_files_table.php

```

1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 class CreatePdfFilesTable extends Migration
8 {
9     /**
10      * Run the migrations.
11      *
12      * @return void
13      */
14     public function up()
15     {
16         Schema::create('pdf_files', function (Blueprint $table) {
17             $table->id();
18             $table->string('pdf_nom');
19             $table->string('pdf_chemin');
20             $table->unsignedBigInteger('risque_id');
21             $table->foreign('risque_id')->references('id')->on('risques');
22             $table->timestamps();
23         });
24     }
25
26     /**
27      * Reverse the migrations.
28      *
29      * @return void
30      */
31     public function down()
32     {
33         Schema::dropIfExists('pdf_files');
34     }
35 }
36

```

4.1.12. Rejet

Le model rejet représente l'ensemble des fiche risque rejeté après vérification. Les principaux attributs sont :

```
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class Rejet extends Model
9 {
10     use HasFactory;
11
12     protected $fillable = [
13         'id',
14         'motif',
15         'risque_id',
16     ];
17
18     public function risque()
19     {
20         return $this->belongsTo(Risque::class, 'risque_id');
21     }
22 }
23
24 }
```

- Id = l'identifiant du rejet
- Motif = le motif pour lequel le validateur a rejeté le risque
- Risque_id = l'identifiant du risque rejeté

Configuration de la migration Rejet :

Chemin : Database/migrations/ 2023_10_13_150745_create_rejects_table.php

```
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 class CreateRejectsTable extends Migration
8 {
9     /**
10      * Run the migrations.
11      *
12      * @return void
13      */
14     public function up()
15     {
16         Schema::create('rejects', function (Blueprint $table) {
17             $table->id();
18             $table->text('motif');
19             $table->unsignedBigInteger('risque_id');
20             $table->foreign('risque_id')->references('id')->on('risques');
21             $table->timestamps();
22         });
23     }
24
25     /**
26      * Reverse the migrations.
27      *
28      * @return void
29      */
30     public function down()
31     {
32         Schema::dropIfExists('rejects');
33     }
34 }
35 }
```

4.1.13. Action

Le model action représente les différentes actions (préventives et correctives) liées à un risque. Les principaux attributs sont :

```
1 <?php
2
3 namespace App\Models;
4
5 use App\Models\Risque;
6 use App\Models\Resva;
7
8 use Illuminate\Database\Eloquent\Factories\HasFactory;
9 use Illuminate\Database\Eloquent\Model;
10
11 class Action extends Model
12 {
13     use HasFactory;
14
15     protected $fillable = [
16         'id',
17         'action',
18         'type',
19         'page',
20         'date',
21         'poste_id',
22         'risque_id',
23     ];
24
25     public function poste()
26     {
27         return $this->belongsTo(Poste::class, 'poste_id');
28     }
29
30     public function risque()
31     {
32         return $this->belongsTo(Risque::class, 'risque_id');
33     }
34 }
35
```

- Id = l'identifiant de l'action
- Action = le nom de l'action
- Type = le type de l'action (préventive ou corrective)
- Page = le paramètre permettant de savoir où l'action a été enregistrer
- Date = la date limite de traitement de l'action (action préventive uniquement)
- Poste_id = l'identifiant du poste chargé de mener l'action
- Risque_id = l'identifiant du risque auquel l'action appartient

Configuration de la migration Action :

Chemin : Database/migrations/ 2023_10_14_221834_create_actions_table.php

```

1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 class CreateActionsTable extends Migration
8 {
9     /**
10      * Run the migrations.
11      *
12      * @return void
13      */
14     public function up()
15     {
16         Schema::create('actions', function (Blueprint $table) {
17             $table->id();
18             $table->string('action');
19             $table->string('type');
20             $table->string('page');
21             $table->date('date')->nullable();
22             $table->unsignedBigInteger('poste_id');
23             $table->foreign('poste_id')->references('id')->on('postes');
24             $table->unsignedBigInteger('risque_id');
25             $table->foreign('risque_id')->references('id')->on('risques');
26             $table->timestamps();
27         });
28     }
29
30     /**
31      * Reverse the migrations.
32      *
33      * @return void
34      */
35     public function down()
36     {
37         Schema::dropIfExists('actions');
38     }
39 }
40

```

4.1.14. Suivi_action

Le model Suivi_action représente le suivi des actions préventives. Les principaux attributs sont :

```

1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class Suivi_action extends Model
9 {
10     use HasFactory;
11
12     protected $fillable = [
13         'id',
14         'efficacite',
15         'commentaire',
16         'date_action',
17         'date_suivi',
18         'statut',
19         'action_id',
20     ];
21
22     public function action()
23     {
24         return $this->belongsTo(Action::class, 'action_id');
25     }
26 }
27

```

- Id = l'identifiant du suivi
- Efficacité = l'efficacité du suivi
- Commentaire = le commentaire du suivi
- Date_action = la date à laquelle l'action a été mener
- Date_suivi = la date à laquelle le suivi a été mener
- Statut = le statut du suivi (réaliser dans le délai, réaliser hors délai)
- Action_id = l'identifiant de l'action

Configuration de la migration Suivi_action :

Chemin : Database/migrations/ 2023_10_17_093423_create_suivi_actions_table.php

```

1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  class CreateSuiviActionsTable extends Migration
8  {
9      /**
10      * Run the migrations.
11      *
12      * @return void
13      */
14      public function up()
15      {
16          Schema::create('suivi_actions', function (Blueprint $table) {
17              $table->id();
18              $table->string('efficacite')->nullable();
19              $table->text('commentaire')->nullable();
20              $table->date('date_action')->nullable();
21              $table->dateTime('date_suivi')->nullable();
22              $table->string('statut')->nullable();
23              $table->unsignedBigInteger('action_id');
24              $table->foreign('action_id')->references('id')->on('actions');
25              $table->timestamps();
26          });
27      }
28
29      /**
30      * Reverse the migrations.
31      *
32      * @return void
33      */
34      public function down()
35      {
36          Schema::dropIfExists('suivi_actions');
37      }
38  }
39

```

4.1.15. Amelioration

Le model amelioration représente les différents incidents que l'on enregistre dans l'application. Les principaux attributs sont :

```
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class Amelioration extends Model
9 {
10     use HasFactory;
11
12     protected $fillable = [
13         'id',
14         'type',
15         'date_fiche',
16         'date_cloture1',
17         'lieu',
18         'detecteur',
19         'non_conformite',
20         'consequence',
21         'cause',
22         'choix_select',
23         'statut',
24         'date_validation',
25         'date1',
26         'date2',
27         'efficacite',
28         'commentaire_eff',
29         'date_eff',
30         'cause_id',
31         'risque_id',
32     ];
33
34     public function risque()
35     {
36         return $this->belongsTo(Risque::class, 'risque_id');
37     }
38
39     public function cause()
40     {
41         return $this->belongsTo(Cause::class, 'cause_id');
42     }
43 }
44 }
```

- Id = l'identifiant de l'incident
- Type = le type d'incident (non-conformité, contentieux et réclamation)
- Date_fiche = la date à laquelle l'incident a été détecter
- Date_cloture1 = la date à laquelle toute la fiche d'incident a été traiter
- Lieu = le lieu de l'incident
- Detecteur = le détecteur de l'incident
- Non_conformite = le motif de l'incident
- Consequence = les conséquences de l'incident
- Cause = les causes de l'incident
- Choix_select = le résultat des recherches effectué (cause trouver, risque trouver ou cause/risque non trouver)
- Statut = le niveau de traitement de la fiche d'incident
- Date_validation = la date à laquelle la fiche d'incident a été valider
- Date1 = la date de début de vérification de l'efficacité des actions de la fiche d'incident

- Date2 = la date de fin de vérification de l'efficacité des actions de la fiche d'incident
- Efficacite = l'efficacité des différentes actions de la fiche d'incident
- Commentaire_eff = le commentaire de l'efficacité
- Date_eff = la date de vérification de l'efficacité
- Cause_id = l'identifiant de la cause trouvée
- Risque_id = l'identifiant du risque trouvé

Configuration de la migration Amelioration :

Chemin : Database/migrations/ 2023_11_10_231839_create_ameliorations_table.php

```

1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  class CreateAmeliorationsTable extends Migration
8  {
9      public function up()
10     {
11         Schema::create('ameliorations', function (Blueprint $table) {
12             $table->id();
13             $table->string('type');
14             $table->date('date_fiche');
15             $table->date('date_cloture1')->nullable();
16             $table->string('lieu');
17             $table->string('detecteur');
18             $table->string('non_conforme');
19             $table->text('consequence');
20             $table->text('cause');
21             $table->string('choix_select');
22             $table->string('statut');
23             $table->date('date_validation')->nullable();
24             $table->date('date1')->nullable();
25             $table->date('date2')->nullable();
26             $table->date('date_eff')->nullable();
27             $table->string('efficacite')->nullable();
28             $table->text('commentaire_eff')->nullable();
29             $table->unsignedBigInteger('cause_id')->nullable();
30             $table->foreign('cause_id')->references('id')->on('causes');
31             $table->unsignedBigInteger('risque_id')->nullable();
32             $table->foreign('risque_id')->references('id')->on('risques');
33             $table->timestamps();
34         });
35     }
36
37     public function down()
38     {
39         Schema::dropIfExists('ameliorations');
40     }
41 }
42

```

4.1.16. Rejet_am

Le model rejet_am représente l'ensemble des fiches d'incidents rejeté après vérification. Les principaux attributs sont :

```
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class rejet_am extends Model
9 {
10     use HasFactory;
11
12     protected $fillable = [
13         'id',
14         'motif',
15         'amelioration_id',
16     ];
17
18     public function amelioration()
19     {
20         return $this->belongsTo(Amelioration::class, 'amelioration_id');
21     }
22 }
23
```

- Id = l'identifiant du rejet
- Motif = le motif du rejet
- Amelioration_id = l'identifiant de la fiche d'incident

Configuration de la migration Rejet_am :

Chemin : Database/migrations/ 2023_12_10_231543_create_rejet_ams_table.php

```

1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 class CreateRejetAmsTable extends Migration
8 {
9     /**
10      * Run the migrations.
11      *
12      * @return void
13      */
14     public function up()
15     {
16         Schema::create('rejet_ams', function (Blueprint $table) {
17             $table->id();
18             $table->text('motif');
19             $table->unsignedBigInteger('amelioration_id');
20             $table->foreign('amelioration_id')->references('id')->on('ameliorations');
21             $table->timestamps();
22         });
23     }
24
25     /**
26      * Reverse the migrations.
27      *
28      * @return void
29      */
30     public function down()
31     {
32         Schema::dropIfExists('rejet_ams');
33     }
34 }
35

```

4.1.17. Suivi_amelioration

Le model Suivi_amelioration représente le suivi des actions d'une fiche d'incident. Les principaux attributs sont :

```

1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class Suivi_amelioration extends Model
9 {
10     use HasFactory;
11
12     protected $fillable = [
13         'id',
14         'efficacite',
15         'nature',
16         'type',
17         'commentaire',
18         'commentaire_am',
19         'date_action',
20         'date_suivi',
21         'statut',
22         'amelioration_id',
23         'action_id',
24     ];
25
26     public function amelioration()
27     {
28         return $this->belongsTo(Amelioration::class, 'amelioration_id');
29     }
30
31     public function action()
32     {
33         return $this->belongsTo(Action::class, 'action_id');
34     }
35
36 }
37

```

- Id = l'identifiant du suivi
- Efficacite = l'efficacité du suivi
- Nature = la nature du suivi (new, accepté ou non accepté)
- Type =
- Commentaire = le commentaire effectué lors du suivi de l'action
- Commentaire_am = le commentaire effectué lors de la création de la fiche d'incident
- Date_action = la date à laquelle l'action a été menée
- Date_suivi = la date à laquelle le suivi a été effectué
- Statut = le statut du suivi de l'action
- Amelioration_id = l'identifiant de la fiche d'incident
- Action_id = l'identifiant de l'action

Configuration de la migration suivi_amelioration :

Chemin : Database/migrations/ 2023_11_27_012326_create_suivi_ameliorations_table.php

```

1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  class CreateSuiviAmeliorationsTable extends Migration
8  {
9      /**
10      * Run the migrations.
11      *
12      * @return void
13      */
14      public function up()
15      {
16          Schema::create('suivi_ameliorations', function (Blueprint $table) {
17              $table->id();
18              $table->string('efficacite')->nullable();
19              $table->string('nature')->nullable();
20              $table->string('type')->nullable();
21              $table->text('commentaire')->nullable();
22              $table->text('commentaire_am')->nullable();
23              $table->date('date_action')->nullable();
24              $table->dateTime('date_suivi')->nullable();
25              $table->string('statut')->nullable();
26              $table->unsignedBigInteger('amelioration_id');
27              $table->foreign('amelioration_id')->references('id')->on('ameliorations');
28              $table->unsignedBigInteger('action_id');
29              $table->foreign('action_id')->references('id')->on('actions');
30              $table->timestamps();
31          });
32      }
33
34      /**
35      * Reverse the migrations.
36      *
37      * @return void
38      */
39      public function down()
40      {
41          Schema::dropIfExists('suivi_ameliorations');
42      }
43  }
44

```

4.1.18. Historique_action

Le model historique_action représente l'historique de chaque actions (Enregistrement, modification, suppression) mener dans l'application. Les principaux attributs sont :

```
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class Historique_action extends Model
9 {
10     use HasFactory;
11
12     protected $fillable = [
13         'id',
14         'nom_formulaire',
15         'nom_action',
16         'user_id',
17     ];
18
19     public function user()
20     {
21         return $this->belongsTo(User::class, 'user_id');
22     }
23 }
24
```

- Id = l'identifiant de l'action
- Nom_formulaire = le nom de la page ou l'action a été effectuée
- Nom_action = le type d'action effectué
- User_id = l'identifiant de l'utilisateur qui a effectuée l'action

Configuration de la migration historique_action :

Chemin : Database/migrations/ 2023_11_04_103429_create_historique_actions_table.php

```

1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 class CreateHistoriqueActionsTable extends Migration
8 {
9     /**
10      * Run the migrations.
11      *
12      * @return void
13      */
14     public function up()
15     {
16         Schema::create('historique_actions', function (Blueprint $table) {
17             $table->id();
18             $table->string('nom_formulaire');
19             $table->string('nom_action');
20             $table->unsignedBigInteger('user_id');
21             $table->foreign('user_id')->references('id')->on('users');
22             $table->timestamps();
23         });
24     }
25
26     /**
27      * Reverse the migrations.
28      *
29      * @return void
30      */
31     public function down()
32     {
33         Schema::dropIfExists('historique_actions');
34     }
35 }
36

```

4.2. Relations entre les modèles

4.2.1. User & Poste

- Un utilisateur a uniquement droit à un seul poste
- Un poste est lié à un seul utilisateur

4.2.2. Autorisation & User

- Un utilisateur a droit à une seule autorisation
- Une autorisation appartient à un seul utilisateur

4.2.3. Historique_action & User

- Un historique est créé par un et un seul utilisateur

4.2.4. Processuse & Objectif

- Un processus peut avoir plusieurs objectifs
- Un objectif peut appartenir à plusieurs processus

4.2.5. Processuse & Pdf_file_processus

- Un processus a droit à un fichier PDF
- Un fichier PDF appartient à un seul processus

4.2.6. Risque & Poste & Processuse

- Un seul Poste valide un risque
- Un risque appartient à un seul processus
- Un processus peut avoir plusieurs risques

4.2.7. Risque & Pdf_file

- Un risque a droit à un fichier PDF
- Un fichier PDF appartient à un seul risque

4.2.8. Cause & Risque

- Un risque peut avoir plusieurs causes

4.2.9. Action & Risque & Poste

- Un risque peut avoir plusieurs actions
- Une action est validée par un seul poste
- Un poste peut valider plusieurs actions

4.2.10. Risque & Rejet

- Un risque peut être rejeté une seule fois

4.2.11. Action & Poste

- Une action est validée par un seul poste
- Un poste peut valider plusieurs actions

4.2.12. Amelioration & Risque & Cause

- Une Amelioration peut avoir un seul risque ou une seule cause

4.2.13. Amelioration & Rejet_am

- Une amélioration peut être rejeté une seul fois

4.2.14. Suivi_action & Action

- Une action est suivie une seule fois

4.2.15. Suivi_amelioration & Amelioration & Action

- Une amélioration est suivie une seule fois

5. Contrôleurs (Controller)

5.1. Description des contrôleurs

Les contrôleurs sont généralement placés dans le répertoire app/Http/Controllers. Chaque contrôleur est une classe PHP qui hérite de la classe de base Controller fournie par Laravel

5.1.1.AmeliorationController

Le controller AmeliorationController Contient quelques logiques liées à un incident, tels que :

- Logique pour afficher le formulaire
- Logique pour enregistrer un incident
- Logique pour afficher la liste des incidents
- Logique pour enregistrer la date de vérification de l'efficacité et effectuée le suivi de l'efficacité
- Logique pour effectuer les recherches (Risques ou causes) à partir de la fiche d'incident

5.1.2.AuthController

Le controller AuthController Contient quelques logiques liées à l'authentification d'un utilisateur et l'accès à la page d'accueil, ce sont :

- Logique pour déconnecter un utilisateur
- Logique pour la connexion d'un utilisateur

5.1.3.Controller

Ce Controller contient une seule logique qui permet d'afficher la page d'accueil

5.1.4.EtatController

Le controller EtatController contient plusieurs logique liées a l'impression des fichiers, tels que :

- Logique pour imprimer une fiche d'incident
- Logique pour imprimer une fiche risque
- Logique pour imprimer une fiche processus
- Logique pour imprimer les détails d'une action préventive
- Logique pour imprimer les détails d'une action corrective

5.1.5.EvaluationController

Le controller EtatController contient la logique qui permet d'avoir accès à la page d'évaluation de processus

5.1.6.HistoriqueController

Le controller HistoriqueController contient la logique qui permet d'avoir accès à la page d'historique

5.1.7.ListeactionController

Le controller ListeactionController contient plusieurs logiques, tels que :

- Logique pour avoir accès a la page de la liste des actions préventives
- Logique pour avoir accès a la page de la liste des actions correctives effectuées
- Logique pour avoir accès a la page de la liste des actions correctives

5.1.8.ListteamController

Le controller ListteamController contient plusieurs logiques tels que :

- Logique pour la validation d'une fiche d'incident
- Logique pour la mise a jour d'une fiche d'incident
- Logique pour le rejet d'une fiche d'incident

5.1.9.ListprocessusController

Le controller ListprocessusController contient plusieurs logiques, tels que :

- Logique pour afficher la liste des processus
- Logique pour la mise a jour d'un processus

5.1.10. ListerisqueController

Le controller ListerisqueController contient plusieurs logiques, tels que :

- Logique pour avoir accès a la liste des risques
- Logique pour la mise a jour d'un risque

5.1.11. ListeuserController

Le controller ListeuserController contient plusieurs logiques, tels que :

- Logique pour avoir accès a la liste des utilisateurs
- Logique pour la mise à jour d'un utilisateur

5.1.12. ParametfrageController

Le controller ParametfrageController contient plusieurs logiques tels que :

- Logique pour avoir accès a la page de paramétrage des couleurs
- Logique pour la création d'un intervalle de couleur
- Logique pour la suppression d'un intervalle de couleur

5.1.13. PosteController

Le controller PosteController contient plusieurs logiques tels que :

- Logique pour avoir accès a la liste des postes
- Logique pour la mise à jour d'un poste

5.1.14. ProcessusController

Le controller ProcessusController contient deux logiques :

- Logique pour avoir accès a la page de création de nouveau processus
- Logique pour l'enregistrement d'un processus

5.1.15. ProfilController

Le controller ProfilController contient plusieurs logiques tels que :

- Logique pour avoir accès a la page profil d'utilisateur
- Logique pour activer le paramétrage de suivi
- Logique pour la mise a jour du mot du passe
- Logique pour la mise a jour des informations de l'utilisateur

5.1.16. RisqueController

Le controller RisqueController contient plusieurs logiques tels que :

- Logique pour avoir accès a la page nouveau risque
- Logique pour avoir accès a la page validation des risques
- Logique pour enregistrer un risque
- Logique pour effectuer la validation d'un risque
- Logique pour rejeter un risque

5.1.17. StatistiqueController

Le controller StatistiqueController contient plusieurs logiques tels que :

- Logique pour avoir accès a la page des statistiques
- Logique pour effectuer des recherches sur un processus
- Logique pour effectuer des recherches sur un intervalles de date

5.1.18. SuiviactionController

Le controller SuiviactionController contient plusieurs logiques tels que :

- Logique pour avoir accès a la page de suivi des actions préventives
- Logique pour avoir accès a la page de suivi des actions correctives
- Logique pour effectuer le suivi des actions préventives et correctives

5.1.19. UpdateamController

Le controller UpdateamController comporte des logiques permettant d'effectuer la mise a jour d'un incident

5.1.20. UserController

Le controller UserController contient deux (02) logiques :

- Logique pour voir accès a la page de nouvel utilisateur
- Logique pour enregistrer un utilisateur

5.2. Route associée

Chemin : Routes\Web.php

Les routes sont organisées en groupes, ce qui permet de les associer à un middleware (une couche intermédiaire dans une application web qui agit comme un filtre ou une passerelle pour les requêtes HTTP entrantes et sortantes) particulier. Voici une explication détaillée de chaque groupe de routes :

```
Route::get('/Login', [AuthController::class, 'view_login'])->name('login');
```

1

2

3

4

- 1- L'URL
- 2- Le controller au quel la route est liée
- 3- Le nom de la fonction du controller liée à la route
- 4- La méthode name() liées à la route signifie que vous pouvez générer l'URL de cette route en utilisant route('login') ailleurs dans votre application

Routes sans Authentification

Ces routes sont accessibles sans authentification.

- Route de Connexion :

'/Login' : Affiche le formulaire de connexion.

'/auth_user' : Authentifie un utilisateur.

```
/*--Connexion--  
Route::get('/Login', [AuthController::class, 'view_login'])->name('login');  
Route::post('/auth_user', [AuthController::class, 'auth_user']);  
/*--
```

Routes avec Authentification

Ces routes sont regroupées dans un middleware auth, ce qui signifie qu'elles nécessitent une authentification préalable pour y accéder.

```

Route::middleware(['auth'])->group(function () {
    /**
     *--Accueil
     *-----
     */
    /**
     *--Deconnexion
     *-----
     */
    /**
     *--Paramétrage de couleurs
     *-----
     */
    /**
     *--Profil
     *-----
     */
    /**
     *--Utilisateur
     *-----
     */
    /**
     *--Processus
     *-----
     */
    /**
     *--Poste
     *-----
     */
    /**
     *--Risque
     *-----
     */
    /**
     *--Action preventive
     *-----
     */
    /**
     *--Action corrective
     *-----
     */
    /**
     *--Incident
     *-----
     */
    /**
     *--Statistique
     *-----
     */
    /**
     *--Historique
     *-----
     */
    /**
     *--Etat
     *-----
     */
});

```

- Route de l'Accueil :

'/' : Affiche la page d'accueil.

```

    /**
     *--Accueil
     |   Route::get('/', [Controller::class, 'index_accueil'])->name('index_accueil');
     */

```

- Route de Déconnexion :

'/logout' : Déconnecte l'utilisateur.

```

    /**
     *--Deconnexion
     |   Route::get('/logout', [AuthController::class, 'logout'])->name('logout');
     */

```

- Route de Paramétrage des Couleurs :

'/Color parametre' : Affiche la page de paramétrage des couleurs.

'/Color parametre traitement' : Traite les paramètres de couleur.

'/Color interval add traitement' : Traite l'ajout d'intervalles de couleur.

'/Color_interval_delete_traitement/{id}' : Traite la suppression d'un intervalle de couleur.

```
/*--Paramétrage de couleurs-----*/
Route::get('/Color paramettre', [ParametreController::class, 'index_color_risk'])->name('index_color_risk');
Route::post('/Color paramettre traitement', [ParametreController::class, 'color_para_traitement'])->name('color_para_traitement');
Route::post('/Color interval add traitement', [ParametreController::class, 'color_interval_add_traitement'])->name(
    'color_interval_add_traitement');
Route::get('/Color_interval_delete_traitement/{id}', [ParametreController::class, 'color_interval_delete_traitement'])->name(
    'color_interval_delete_traitement');
/*-----*/
```

- Route de Profil :

'/Profil' : Affiche le profil de l'utilisateur.

Diverses routes pour la mise à jour du profil.

```
/*--Profil-----*/
Route::get('/Profil', [ProfilController::class, 'index_profil'])->name('index_profil');
Route::get('/suiviactiveoui', [ProfilController::class, 'suivi_oui']);
Route::get('/suiviactiveton', [ProfilController::class, 'suivi_non']);
Route::get('/mdp_update', [ProfilController::class, 'mdp_update']);
Route::get('/info_update', [ProfilController::class, 'info_update']);
Route::get('/Historique Profil', [ProfilController::class, 'index_historique_profil'])->name('index_historique_profil');
/*-----*/
```

- Route d'Utilisateur :

'/Nouveau utilisateur' : Affiche le formulaire d'ajout d'un nouvel utilisateur.

'/traitement user' : Traite l'ajout d'un nouvel utilisateur.

'/Liste des utilisateurs' : Affiche la liste des utilisateurs.

Diverses routes pour la modification des autorisations des utilisateurs.

```
/*--Utilisateur-----*/
Route::get('/Nouveau utilisateur', [UserController::class, 'index_user'])->name('index_user');
Route::post('/traitement user', [UserController::class, 'add_user'])->name('add_user');
Route::get('/Liste des utilisateurs', [ListeuserController::class, 'index'])->name('index_liste_user');
Route::post('/Modification des autorisations', [ListeuserController::class, 'index_user_modif'])->name('index_user_modif');
Route::post('/traitement modif user', [ListeuserController::class, 'index_modif_traitement'])->name('index_modif_auto');
/*-----*/
```

- Route de Processus :

'/Nouveau Processus' : Affiche le formulaire d'ajout d'un nouveau processus.

'/traitement processus' : Traite l'ajout d'un nouveau processus.

'/Liste Processus' : Affiche la liste des processus.

Diverses routes pour la modification des processus.

```

/*--Processus-----
Route::get('/Nouveau Processus', [ProcessusController::class, 'index_add_processus'])->name('index_add_processus');
Route::post('/traitement processus', [ProcessusController::class, 'add_processus'])->name('add_processus');
Route::get('/Liste Processus', [ListeprocessusController::class, 'index_listeprocessus'])->name('index_listeprocessus');
Route::post('/traitement modif processus', [ListeprocessusController::class, 'processus_modif'])->name('processus_modif');
Route::get('/Evaluation des processus', [EvaluationController::class, 'index_evaluation_processus'])->name('index_evaluation_processus');
*/

```

- Route de Poste :

'/Liste Poste' : Affiche la liste des postes.

Diverses routes pour l'ajout et la modification des postes.

```

/*--Poste-----
Route::get('/Liste Poste', [PosteController::class, 'index_liste_poste'])->name('index_liste_poste');
Route::post('/traitement Poste', [PosteController::class, 'index_add_poste_traitement'])->name('index_add_poste_traitement');
Route::post('/traitement modif Poste', [PosteController::class, 'index_modif_poste_traitement'])->name('index_modif_poste_traitement');
);
Route::get('/get post user', [PosteController::class, '/get_post_user'])->name('get_post_user');
*/

```

- Route de Risque :

'/Nouveau Risque' : Affiche le formulaire d'ajout d'un nouveau risque.

'/traitement risque' : Traite l'ajout d'un nouveau risque.

Diverses routes pour la validation, le rejet et la mise à jour des risques.

```

/*--Risque-----
Route::get('/Nouveau Risque', [RisqueController::class, 'index_risque'])->name('index_risque');
Route::get('/recherche/{processusId}', [RisqueController::class, 'recherche_processus'])->name('recherche_processus');
Route::post('/traitement_risque', [RisqueController::class, 'add_risque'])->name('add_risque');

Route::get('/Validation des risques', [RisqueController::class, 'index_validation_risque'])->name('index_validation_risque');
Route::get('/risque valider/{id}', [RisqueController::class, 'risque_valider'])->name('risque_valider');

Route::post('/traitement rejet', [RisqueController::class, 'risque_rejet'])->name('risque_rejet');

Route::get('/Liste risque', [ListerisqueController::class, 'index_liste_risque'])->name('index_liste_risque');
Route::get('/Mise a jour', [ListerisqueController::class, 'index_risque_actionup'])->name('index_risque_actionup');
Route::post('/Mise a jour risque', [ListerisqueController::class, 'index_risque_actionup2'])->name('index_risque_actionup2');
Route::post('/Mise a jour risque traitement', [ListerisqueController::class, 'index_risque_actionup2_traitement'])->name('index_risque_actionup2_traitement');
*/

```

- Routes des Actions Préventives :

'/Liste Action Préventive' : Affiche la liste des actions préventives.

'/Suivi des actions préventives' : Affiche le suivi des actions préventives.

```

/*--Action preventive-----
Route::get('/Liste Action Preventive', [ListactionController::class, 'index_ap'])->name('index_ap');
Route::get('/Suivi des actions preventives', [SuiviactionController::class, 'index_suiviaktion'])->name('index_suiviaktion');
Route::post('/Suivi_action/{id}', [SuiviactionController::class, 'add_suivi_action'])->name('add_suivi_action');
*/

```

- Routes des Actions Correctives :

'/Liste Action Corrective' : Affiche la liste des actions correctives.

'/Liste Action Corrective effectuée' : Affiche la liste des actions correctives effectuées.

'/Suivi des actions correctives' : Affiche le suivi des actions correctives.

```
/*--Action corrective-----*/
Route::get('/Liste Action Corrective', [ListeactionController::class, 'index_ac'])->name('index_ac');
Route::get('/Liste Action Corrective effectuée', [ListeactionController::class, 'index_ac_eff'])->name('index_ac_eff');
Route::get('/Suivi des actions correctives', [SuiviactionController::class, 'index_suiviactionc'])->name('index_suiviactionc');
Route::post('/Suivi_actionc/{id}', [SuiviactionController::class, 'add_suivi_actionc'])->name('add_suivi_actionc');
/*-
```

- Routes des Incidents :

Diverses routes pour la gestion des incidents et des validations.

```
/*--Incident-----*/
Route::get('/fiche_amelioration', [AmeliorationController::class, 'index'])->name('index_amelioration');
Route::get('/get-cause-info/{id}', [AmeliorationController::class, 'get_cause_info']);
Route::get('/get-risque-info/{id}', [AmeliorationController::class, 'get_risque_info']);
Route::post('/add_amelioration', [AmeliorationController::class, 'index_add'])->name('index_add');
Route::get('/liste_amelioration', [AmeliorationController::class, 'index_liste'])->name('index_amelioration_liste');
Route::get('/validation_amelioration', [ListeamController::class, 'index_validation'])->name('index_validation_amelioration');
Route::get('/am_valider/{id}', [ListeamController::class, 'am_valider'])->name('am_valider');
Route::post('/am_rejet', [ListeamController::class, 'am_rejet'])->name('am_rejet');
Route::get('/amelioration_up', [ListeamController::class, 'index_amup'])->name('index_amup');
Route::post('/amelioration_up2', [ListeamController::class, 'index_amup2'])->name('index_amup2');
Route::post('/amelioration_up_add', [ListeamController::class, 'index_amup_add'])->name('index_amup_add');
Route::post('/amelioration_up_traitement', [UpdateamController::class, 'amup_traitement'])->name('amup_traitement');
Route::post('/amelioration_up2_traitement', [UpdateamController::class, 'amup2_traitement'])->name('amup2_traitement');
Route::post('/amelioration_up2_add_traitement', [UpdateamController::class, 'amup2_add_traitement'])->name('amup2_add_traitement');
Route::post('/traitement_date', [AmeliorationController::class, 'date_recla'])->name('date_recla');
Route::post('/traitement_eff', [AmeliorationController::class, 'eff_recla'])->name('eff_recla');
Route::get('/am_update/{id}', [UpdateamController::class, 'am_update'])->name('am_update');
/*-
```

- Route des Statistiques :

'/Statistique' : Affiche les statistiques.

Diverses routes pour obtenir des données statistiques.

```
/*--Statistique-----*/
Route::get('/Statistique', [StatistiqueController::class, 'index_stat'])->name('index_stat');
Route::get('/get_processus/{id}', [StatistiqueController::class, 'get_processus'])->name('get_processus');
Route::get('/get_date', [StatistiqueController::class, 'get_date'])->name('get_date');
/*-
```

- Route de l'Historique :

'/Historique' : Affiche l'historique.

```
/*--Historique-----*/
Route::get('/Historique', [HistoriqueController::class, 'index_historique'])->name('index_historique');
/*-
```

- Route de l'État :

Diverses routes pour obtenir l'état des incidents, risques, processus, actions préventives et correctives.

```
/*--Estat-----*/  
Route::post('/Etat am', [EtatController::class, 'index_etat_am'])->name('index_etat_am');  
Route::post('/Etat risque', [EtatController::class, 'index_etat_risque'])->name('index_etat_risque');  
Route::post('/Etat processus', [EtatController::class, 'index_etat_processus'])->name('index_etat_processus');  
Route::post('/Etat actionp', [EtatController::class, 'index_etat_actionp'])->name('index_etat_actionp');  
Route::post('/Etat actions', [EtatController::class, 'index_etat_actions'])->name('index_etat_actions');  
/*-----*/
```

6. Vues (View)

Chemin : resources\views

Dans chaque répertoire de vue, vous pouvez avoir différents fichiers représentant les différentes parties de l'interface utilisateur de l'application.

Structure des pages de l'application :

/ressources (Dossier)

-----/views (Dossier)

-----/add (Dossier)

-----/color_risque.blade.php

→ Paramétrage de couleur

-----/ficheamelioration.blade.php

→ Nouvel incident

-----/processus.blade.php

→ Nouveau processus

-----/processuseva.blade.php

→ Nouveau risque

-----/res-va.blade.php

→ Nouvel utilisateur

-----/auth (Dossier)

-----/login.blade.php

→ Page de connexion

-----/etat (Dossier)

-----/actionc.blade.php

→ Aperçue d'une action corrective à imprimer

-----/actionp.blade.php

→ Aperçue d'une action préventive à imprimer

-----/amelioration.blade.php

→ Aperçue d'un incident à imprimer

-----/processus.blade.php

→ Aperçue d'un processus à imprimer

-----/risque.blade.php

→ Aperçue d'un risque à imprimer

-----/historique (Dossier)

-----/historique.blade.php

→ Historique

-----/liste (Dossier)

-----/actioncorrective.blade.php

→ Liste des actions correctives

-----/actioncorrectiveeff.blade.php

→ Liste des actions correctives effectuées

-----/actionpreventive.blade.php

→ Liste des actions préventives

-----/amelioration.blade.php

→ Suivi des incidents

-----/poste.blade.php

→ Liste des postes

-----/processus.blade.php	→Liste des processus
-----/risque.blade.php	→Liste des risques
-----/user.blade.php	→Liste des utilisateurs
-----/user_modif.blade.php	→Modification des autorisations
-----/statistique (Dossier)	
-----/index.blade.php	→Statistique
-----/tableau (Dossier)	
-----/evaproces.blade.php	→Evaluation des processus
-----/valideam.blade.php	→Validation des incidents
-----/validecause.blade.php	→Validation des risques
-----/traitement (Dossier)	
-----/actionup.blade.php	→Risques non validés
-----/actionup2.blade.php	→Modification de risque non validé
-----/amup.blade.php	→Incidents non validés
-----/amup2.blade.php	→Modification des incidents non validés
-----/amup_add.blade.php	→Ajouter une nouvelle action a un incident
-----/suiviaction.blade.php	→Suivi des actions préventives
-----/suiviactionc.blade.php	→Suivi des actions correctives
-----/user (Dossier)	
-----/profil.blade.php	→Profil d'utilisateur
-----/app.blade.php	→Corps des différentes pages
-----/menu.blade.php	→Accueil

7. Evènements (Event)

7.1. Liste des évènement utilisés

Chemin : app\Events

Un évènement est un mécanisme qui permet de déclencher des alertes en temps réel afin de notifier l'utilisateur d'une nouvelle action effectuée dans l'application.

- **NotificationAmcorrective** : Nouvelle action corrective
- **NotificationAmnew** : Nouvel incident
- **NotificationAmrejet** : Incident rejeté
- **NotificationAmvalider** : incident validé
- **NotificationAnon** : Risque rejeté
- **NotificationApreventive** : Nouvel action préventive
- **NotificationAup** : Mise à jour détectée sur un incident
- **NotificationProcessus** : Nouveau processus
- **NotificationRisque** : Nouveau risque
- **NotificationRisqueup** : Mise à jour détectée sur un risque
- **UserInactive** : Session de l'utilisateur a expiré

7.2. Fonctionnement

Pour créer un évènement et le déclencher après une action, il faut :

- Créer un évènement dans le répertoire (app\Events) suivi de l'extension .PHP
- Ensuite entrer le paramètre de l'évènement :

```

<?php

namespace App\Events;

use Illuminate\Broadcasting\Channel;
use Illuminate\Broadcasting\InteractsWithSockets;
use Illuminate\Broadcasting\PresenceChannel;
use Illuminate\Broadcasting\PrivateChannel;
use Illuminate\Contracts\Broadcasting\ShouldBroadcast;
use Illuminate\Foundation\Events\Dispatchable;
use Illuminate\Queue\SerializesModels;

use App\Models\Action;
use App\Models\User;

class NotificationAcorrective implements ShouldBroadcast
{
    use Dispatchable, InteractsWithSockets, SerializesModels;

    public function broadcastOn()
    {
        return ['my-channel-ac'];
    }

    public function broadcastAs()
    {
        return 'my-event-ac';
    }
}

```

ENTRER UN NOM DE CHANNEL

ENTRER LE NOM DE L'EVENEMENT

- Aller dans le fichier nommer .env, ensuite entrer les coordonnées suivantes :

PUSHER_APP_ID=1708507

PUSHER_APP_KEY=9f9514edd43b1637ff61

PUSHER_APP_SECRET=fdbbe1b88552f50c55bc0

PUSHER_APP_CLUSTER=eu

- Ensuite pour déclencher l'évènement, aller le controller de votre choix, après avoir effectuée une requête, entrer : event (new NotificationAcorrective ()) ;
- De plus, importer les parametres de l'évènement (use App\Events\NotificationAcorrective);
- Pour afficher l'évènement, aller dans la page ou vous souhaitez afficher l'évènement, et taper :

```
<script>
    Pusher.logToConsole = true;

    var pusher = new Pusher('9f9514edd43b1637ff61', {
        cluster: 'eu'
    });

    var channel = pusher.subscribe('my-channel-ac');
    channel.bind('my-channel-ac', function(data) {
        Swal.fire({
            title: "Alert!",
            text: "Nouvelle(s) action(s) corrective(s)",
            icon: "info",
            confirmButtonColor: "#00d819",
            confirmButtonText: "OK",
            allowOutsideClick: false,
        }).then((result) => {
            if (result.isConfirmed) {
                location.reload();
            }
        });
    });
</script>
```

LE NOM DE L'EVENEMENT

LE NOM CHANNEL

8. Base de données

