RAPPORT DE STAGE

Développement logiciel pour l'équipe SAFE en biométrie au laboratoire GREYC

1 avril 2021 - 3 juin 2021

HARTVICK Colin

L3 Informatique

Enseignant référent – M. ROSENBERGER Christophe

Maître de stage – M. ROSENBERGER Christophe

1 Remerciements

Je tiens à remercier M. Christophe ROSENBERGER, directeur du laboratoire GREYC et maître de stage pour son soutien et son accompagnement tout au long du stage.

Table des matières

2 Introduction

Du 1er avril 2021 au 3 juin 2021, j'ai effectué un stage au sein du laboratoire GREYC situé à Caen. En raison des conditions imposées par la crise sanitaire, j'ai dû le réaliser totalement en télétravail. Au cours de ce stage dans l'équipe SAFE, j'ai pu contribuer au développement logiciel sur la plateforme GREYC Forensics.

Le laboratoire GREYC est un laboratoire de recherche basé à Caen et ayant plusieurs sites en Normandie. Le laboratoire effectue des recherches en sciences du numérique grâce à plusieurs équipes. L'équipe dans laquelle j'ai travaillé est l'équipe SAFE qui mène des recherches en sécurité informatique, architecture, forensique, et biométrie.

Mon stage a constitué en l'élaboration d'une interface utilisateur web pour la plateforme GREYC Forensics, plateforme d'analyse de traces numériques sur des disques, cartes... Cette mission m'a permis de découvrir de nouveaux outils pour la réalisation de sites web et d'approfondir mes connaissances dans ce domaine.

Tout d'abord, je décrirai le cadre du stage, à savoir le laboratoire GREYC et son équipe SAFE. Puis je présenterai ma mission effectuée durant le stage avant de dresser un bilan de celui-ci.

3 Environnement de stage

3.1 Laboratoire GREYC

Le laboratoire GREYC (Groupe de recherche en informatique, image et instrumentation de Caen), fondé en 1995, est un laboratoire de recherche en sciences du numérique associé au CNRS, à l'Université de Caen Normandie et à l'Ecole Nationale Supérieure d'Ingénieurs de Caen. Basé à Caen il contient néanmoins plusieurs autres sites, tous en Normandie : Alençon, Cherbourg, Lisieux, Saint-Lô et Vire. Son directeur est Christophe ROSENBERGER. Les différents membres permanents et non-permanents du GREYC sont divisés en 6 équipes :

- AMMAC : Algorithmes, Modèles de calcul, Aléa, Combinatoire, Complexité.
- CODAG : Contraintes, Data mining, Graphes.
- ÉLÉCTRONIQUE.
- IMAGE.
- MAD : Modèles, Agents, Décision
- SAFE : Sécurité, Architecture, Forensique, biomEtrie.

3.2 Équipe SAFE

L'équipe SAFE, dirigée par Christophe CHARRIER, «mène des activités de recherche en sécurité informatique suivant trois axes avec une continuité des aspects théoriques vers les applications, les trois s'enrichissant mutuellement : Biométrie (définition et évaluation de systèmes biométriques, protection des données biométriques), Architecture et modèles de sécurité (Sécurité réseau (SDN, 5G, 6G), cryptographie appliquée, aléatoire et protection de l'information) et Science de l'investigation (Forensique) (Traitement automatique de langue (TAL), plateforme forensique, protection de la vie privée)» (Site du GREYC). L'équipe possède 11 membres permanents et actuellement 18 non-permanents. SAFE collabore avec des académies nationales (IRISA, XLIM, LITIS), des académies internationales (Université de Toronto, Université d'Oslo) mais aussi des industries (Orange Labs, FIME, TestWe, IN groupe). Parmi les projets qu'elle a réalisés on retrouve OFFPAD (Objet Personnel d'Authentification Hors-ligne appliqué aux hôpitaux et banques en ligne) ou encore IDBLOCKCHAIN (fournisseur des services de sécurité pour la gestion de l'IDentité (ID) numérique dans le respect de la vie privée basé sur les technologies dites blockchain). Actuellement l'équipe SAFE travaille sur 3 projets :

- RiderNet (RIN recherche) : Analyse du comportement visuel d'un motocycliste.
- Réseau Européen P4C : réseau Européen (Norvège-France-Allemagne) en cybersécurité.
- FILTER2 (ANR) : optimisation du passage des fichiers (ou des portions de fichiers) dans les outils d'analyse.

3.3 Environnement de travail

J'ai réalisé le stage à distance, j'ai pu communiquer avec le maître de stage via l'application Discord, que ce soit via des appels vocaux ou chats écrits.

4 Travail réalisé

4.1 Plateforme GREYC Forensics

La plateforme GREYC Forensics est une plateforme de forensique, science qui, selon Wikipédia, «regroupe l'ensemble des différentes méthodes d'analyse fondées sur les sciences afin de servir au travail d'investigation de manière large». La plateforme se concentre sur l'analyse de traces numériques sur un disque dur, une carte, internet, des données réseau... donc sur les traces numériques laissées par un utilisateur lors de l'usage d'un appareil connecté ou d'un logiciel. Le terme trace numérique désigne «les informations qu'un dispositif numérique enregistre sur l'activité ou l'identité de ses utilisateurs au moyen de traceurs tels que les cookies, soit automatiquement, soit par le biais d'un dépôt intentionnel : moteurs de recherche, blogs, sites de réseau social, sites de e-commerce, mais aussi cartes à puce, titres de transport, téléphones mobiles».

Les intérêts de développer une telle plateforme sont multiples : elle pourrait permettre de «réaliser des analyses de haut niveau et automatisable pour des opérationnels de l'investigation (section recherche criminalité numérique de Caen par exemple) et pour des chercheurs (par exemple pour des activités de recherche en protection de la vie privée)». De plus elle serait «un outil intéressant comme support de projets pour des étudiants en informatique pour les faire travailler sur des sujets en lien avec la sécurité, les systèmes d'exploitation, le réseau... Elle pourrait être utilisée à l'ENSICAEN ou dans le cadre du Master E-Secure de l'UNICAEN». Enfin «les compétences du GREYC en traitement automatique de la langue, traitement des images, sécurité... doivent permettre de développer de nouveaux outils de haut niveau pour des investigations numériques».

Ma mission dans ce projet a été de développer une interface utilisateur permettant d'afficher les résultats d'une analyse de disque à travers différents filtres montrant différentes informations.

4.2 Données utilisées

J'ai eu à ma disposition une base de donnée contenant les résultats d'une analyse d'un disque effectuée sur près de 12000 fichiers. Cette base de données se divise en 3 tables différentes :

- **directory**, une table de répertoires avec 3 colonnes :
  - **id** (int) : identifiant du répertoire.
  - **parent** (int) : identifiant du répertoire parent.
  - **name** (string) : nom du répertoire.
- **file**, une table de fichiers avec 7 colonnes :
  - **id** (int) : identifiant du fichier.
  - **parent** (int) : identifiant du répertoire contenant le fichier.
  - **name** (string) : nom du fichier.
  - **extension** (string) : extension affichée du fichier.

— **created_a** (int) : date de création UNIX.
— **modified_at** (int) : date de modification UNIX.
— **size** (int) : taille en bits.

— **file_type**, une autre table de fichiers avec 3 colonnes :
— **file_id** (int) : identifiant du fichier
— **type** (string) : type du fichier
— **extension_correct** (boolean) : extension affichée correspond à la véritable extension du fichier

J'ai ajouté deux tables à cette base de donnée : une table **user**, qui contient les utilisateurs, et une table **dump**, qui contient les informations des analyses. J'ai aussi rajouté une colonne à la table **directory**, nommée **dump**, qui contient l'identifiant de l'analyse à laquelle appartient chaque répertoire pour joindre les tables **dump** et **directory**.

— **user** qui contient 5 colonnes :
— **id** (int) : identifiant de l'utilisateur.
— **email** (string) : mail de l'utilisateur.
— **role** (clob) : rôle de l'utilisateur.
— **password** (string) : mot de passe haché de l'utilisateur.
— **name** (string) : nom de l'utilisateur.

— **dump** qui contient 4 colonnes :
— **id** (int) : identifiant de l'analyse.
— **user** (int) : identifiant de l'utilisateur propriétaire de l'analyse.
— **name** (string) : nom de l'analyse.
— **type** (int) : type de l'analyse. **1** correspond à une analyse de disque et **2** à une analyse de carte.

Ce qui donne au final une base de données qui peut être représentée par le diagramme UML suivant. C'est donc avec cette base de données que j'ai réalisé l'interface.

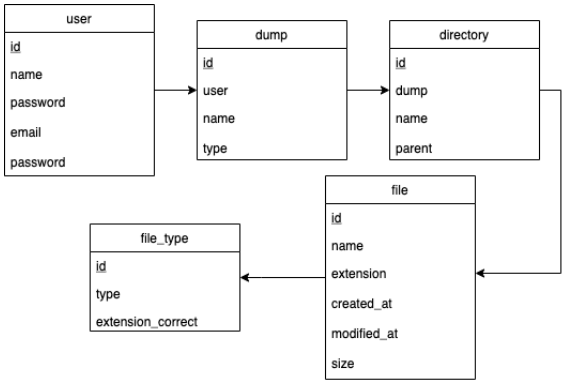


FIGURE 1 – Diagramme UML de la base de données utilisée

4.3 Outils utilisés

Afin de réaliser l'interface utilisateur, j'ai utilisé de nombreux outils facilitant le développement.

4.3.1 Symfony

Symfony est un framework PHP modèle-vue-contrôleur (MVC). Il permet un accès simple à une base de données. Ayant déjà réalisé un projet web modèle-vue-contrôleur en PHP durant l'année de licence, le choix de Symfony s'est fait naturellement. Symfony m'a donc permis de découvrir un nouveau framework mais avec un langage (PHP) et une architecture (MVC) déjà connue. Pour la vue, Symfony permet de developper en twig, un moteur de templates pour PHP. Il permet simplement d'écrire en HTML tout en y incluant des variables PHP.

FIGURE 2 – Logo de Symfony

4.3.2 Bootstrap

Bootstrap est un framework qui simplifie la création du design de site web. Il permet de créer rapidement et facilement des boutons, formulaires, listes... design en utilisant simplement des éléments HTML. Bootstrap m'a été très utile afin d'organiser simplement mes pages et de créer des éléments avec du style rapidement sans devoir utiliser du CSS.

FIGURE 3 – Logo de Bootstrap

4.3.3 Bootstrap Table

Bootstrap table est un outil utilisant Bootstrap qui permet de créer des tableaux interactifs. Il permet d'afficher un grand nombre de données sans ralentissement et de mettre en place des outils utiles comme la pagination ou le tri. Bootstrap Table m'a permis d'afficher un tableau de près de 12000 lignes simplement.

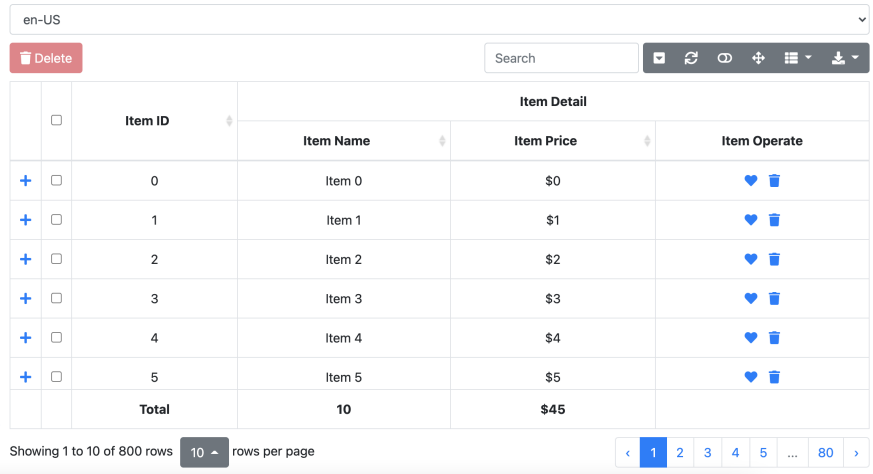


FIGURE 4 – Exemple d'un tableau réalisé avec Bootstrap Table

4.3.4 Google Charts

Google Charts est un outil qui permet l'affichage de graphiques interactifs sur des applications web. C'est un outil simple d'utilisation qui contient de nombreux types de graphiques et permet une grande personnalisation tout en restant relativement simple pour des utilisations basiques.

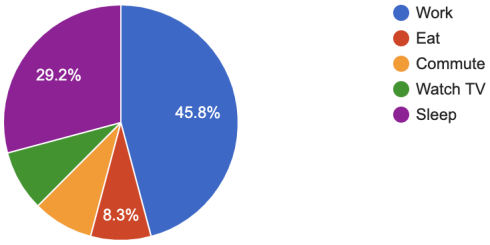


FIGURE 5 – Exemple d'un diagramme circulaire réalisé avec Google Charts

4.3.5 JQuery

JQuery est une bibliothèque JavaScript facilitant l'écriture de script dans le code HTML. Je m'en suis servi dans mes scripts mais j'ai aussi utilisé plusieurs outils issus de JQuery. J'ai donc utilisé select2 qui permet de créer des sélecteurs avec de nombreuses options, mais aussi la fonction slider de JQuery qui permet de réaliser un double curseur.

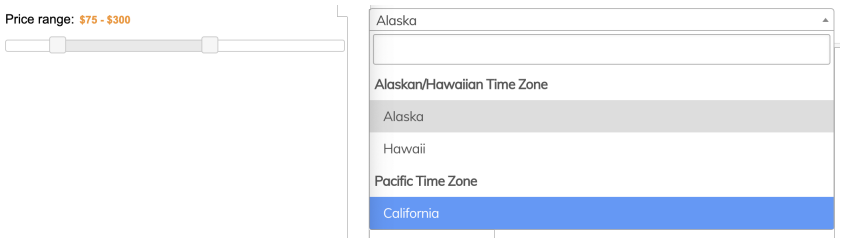


FIGURE 6 – Exemple d'un double curseur et de d'un select2 avec JQuery

4.3.6 D3.js

D3.js est une bibliothèque JavaScript permettant l'affichage de données à l'aide de HTML, SVG, et CSS. J'ai utilisé cette bibliothèque afin de construire une carte de chaleur.

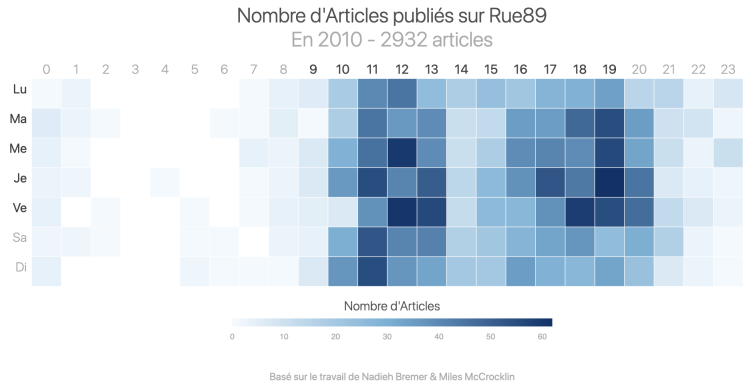


FIGURE 7 – Exemple de carte de chaleur réalisée avec D3.js

4.4 Routage

Le routage du site est assez simple. Il y a 3 types de routes pour l'instant, car seuls les filtres pour une analyse de disque ont été développés :

- / : accueil du site, redirige vers la page de connexion si aucun utilisateur n'est connecté. Si un utilisateur est connecté, affiche la page d'accueil avec la liste des analyses disponibles.
- **/dump/disk/*nom-analyse*/*nom-filtre*** : affiche le filtre pour l'analyse nommée
- **/dump/disk/*nom-analyse*/*nom-filtre*/data** : affiche en JSON les données du filtre pour l'analyse nommée. Utilisée par tous les filtres pour obtenir les données à afficher en AJAX.

On peut facilement imaginer que si l'on développe des filtres pour l'analyse de carte, la route soit **/dump/card/*nom-analyse*/*nom-filtre***. On peut aussi

ajouter les 2 routes /**login** et /**lougout** qui servent respectivement à afficher la page de connexion et à déconnecter l'utilisateur et qui ont été générées automatiquement par Symfony.

Afin d'appliquer les filtres sur une sélection précise de fichier, il y a un formulaire de sélection que l'on détaillera ensuite. On passe les paramètres de la sélection via une query string dans l'url.

4.5 Accès à la base de données

Avec le framework Symfony, on accède à la base de données grâce aux classes **Repository**. Pour faciliter les requêtes pour accéder aux données des analyses, j'ai créé la fonction **request()** dans la classe **FileRepository**. Elle permet d'accéder à des données des tables **file**, **directory** et **file_type** de la base de données.

request() prend plusieurs paramètres :
- **$dump** (int) : identifiant de l'analyse.
- **$select** (string) : select de la requête en SQL.
- **$return** (string) : **"int"** si le résultat attendu est un entier, **"array"** si c'est une liste.
- **$group_by** (string) : colonne à grouper. Si **$group_by** est **file . extension**, le résultat regroupe toutes les mêmes extensions en une ligne.
- **$limit** (int) : nombre maximal de résultats.
- **$offset** (int) : numéro de début des résultats.
- **$sort** (string) : colonne à trier.
- **$order** (string) : **asc** pour un tri croissant et **desc** pour un tri décroissant.
- **$parameters** (array) : liste des paramètres à respecter. Un paramètre est une liste **(requête, nom-variable, variable)**. Par exemple si l'on souhaite sélectionner les fichiers inférieurs à 10 bits, on ajoute **('file.size < var', 'var', 10)** à **$parameters**.

Grâce à la fonction **createQueryBuilder()** de **EntityRepository**, on peut effectuer la requête souhaitée. Avec **from()** on peut effectuer aussi des requêtes sur les tables **directory** et **file_type**. On rajoute une condition avec **setParameter()** et **andWhere()** pour que les données sélectionnées appartiennent à la bonne analyse. De plus on ajoute une jointure entre les tables **directory** et **file** et les tables **file** et **file_type**.

```
$qb = $this->createQueryBuilder('file')
    ->from('App\Entity\Directory', 'dir')
    ->andWhere('dir.dump = :dump')
    ->andWhere('dir.id = file.parent')
    ->setParameter('dump', $dump)
    ->from('App\Entity\FileType', 'file_type')
    ->andWhere('file.id = file_type.id');
```

On définit ensuite les colonnes que l'on souhaite rechercher avec **select()**

```php
$qb = $qb->select($select);
```

Puis on définit certains paramètres de la requête : **GROUP BY** (colonne regroupée), **LIMIT** (nombre maximal de résultats), **OFFSET** (numéro du premier résultat) et **ORDER BY** (colonne ordonnée avec ordre croissant ou décroissant) en fonctions des paramètres de **request()**.

```php
if($group_by != null)
    $qb = $qb->groupBy($group_by);

if($limit != null)
    $qb = $qb->setMaxResults($limit);

if($offset != null)
    $qb = $qb->setFirstResult($offset);

if($sort != null && $order != null)
    $qb = $qb->orderBy($sort, $order);
elseif($sort != null && $order == null)
    $qb = $qb->orderBy($sort, 'ASC');
```

On définit après les conditions dans la liste **$parameters** avec **setParameter()** qui définit un paramètre et **andWhere()** qui ajoute une condition.

```php
foreach ($parameters as $parameter){
    $qb = $qb->andWhere($parameter[0])
    ->setParameter($parameter[1], $parameter[2]);
}
```

Enfin on renvoie le résultat brut en liste ou on le convertit en entier en fonction de **$return**.

```php
switch ($return){
    case "int":
        return intval($qb->getQuery()->getSingleScalarResult());
    case 'array':
        return $qb->getQuery()->getResult();
}
```

4.6 Contrôleurs

Les contrôleurs sont les classes qui gèrent les données et font les appels aux vues à afficher. Ils sont donc des éléments essentiels de l'application.

4.6.1 DefaultController

Ce contrôleur gère la partie avant la sélection d'une analyse. Il est appelé lorsque la route est /, avec la fonction **displayHome()**. Si à ce moment aucun utilisateur n'est connecté, la fonction redirige vers la route /**login** pour afficher la page de connexion. Sinon, on affiche la page d'accueil ci-dessous en ayant récupéré au préalable la liste des analyses appartenant à l'utilisateur connecté avec une requête à la base de donnée via la classe **DumpRepository**.

FIGURE 8 – Page d'accueil

4.6.2 SecurityController

Ce contrôleur possède deux fonctions, **login()** et **logout()**. La première est appelée quand la route est /**login** et permet d'afficher la page de connexion. La seconde est appelée par /**logout** et déconnecte l'utilisateur puis redirige vers la route /. SecurityController a été créé automatiquement par Symfony avec un **make :auth**, ce qui a aussi généré le template de la page de connexion.

4.6.3 DumpController

Ce dernier contrôleur est le plus utilisé, on passe par lui dès que l'on est dans une analyse. Il est utilisé pour afficher les filtres et pour récupérer les données des filtres. On va détailler ses deux fonctions les plus importantes, **displayDiskDump()** qui gère l'affichage des filtre et **getFilterData()** qui gère les données des filtres.

Dans le constructeur de DumpController, on définit les filtres dans une liste dans une variable globale **$filters**. De plus on définit des accès aux classes **FileRepository** et **DumpRepository**, qui permettent de faire des requêtes à la base de données, avec deux autres variables globales : **$fileRepository** et **$dumpRepository** .

displayDiskDump() est une fonction qui est appelée par la route **dump/-disk/*dumpSlug*/*filter***. Elle prend 2 arguments qui sont présents dans l'url, **$dumpSlug** qui est le nom de l'analyse préalablement converti en slug pour

qu'il puisse être dans une url, et **$filter** qui est le nom du filtre que l'on souhaite afficher.

```
public function displayDiskDump(string $dumpSlug, string $filter = null):
    ↪ Response
```

Tout d'abord on souhaite identifier l'analyse. On parcourt donc toutes les analyses et on identifie l'analyse en comparant **$dumpSlug** avec les slugs des analyses.

```
foreach ($this->dumpRepository->findAll() as $dump){
          if($dump->getSlug() == $dumpSlug)
              $this->dump = $dump;
      }
```

Ensuite si il n'y a pas d'analyse identifiée, si aucun utilisateur n'est connecté ou si l'analyse n'appartient pas à l'utilisateur connecté, on ne peut afficher l'analyse donc on redirige vers la page d'accueil. De même, si aucun filtre n'est renseigné ou si il n'existe pas, on redirige vers la page du filtre **Aperçu**.

```
if ($this->dump == null || $this->getUser() == null || $this->getUser()->
    ↪ getId() != $this->dump->getUser())
    return $this->redirectToRoute('home');
if($filter == "" || !in_array($filter, array_column($this->filters,'id')))
    return $this->redirectToRoute('dump.disk.filter', array("dumpSlug" =>
        ↪ $dumpSlug, "filter" => "preview"));
```

Puis on récupère les données pour le formulaire de sélection, formulaire qui sert à appliquer le filtre, par exemple, seulement sur des fichiers de certains types, extensions, taille... Pour récupérer les données on utilise la fonction **request()** de la classe **FileRepository**. On obtient donc la liste des types, des répertoires, des extensions et la plus grande taille d'un fichier de l'analyse.

```
$selectionData = array(
          "directories" => $this->fileRepository->request($this->dump->
              ↪ getId(), 'dir.id, dir.name as text ', 'array', 'text',
              ↪ null, null, null, null),
          "fileTypes" => $this->fileRepository->request($this->dump->getId
              ↪ (), 'file.id, file_type.type as text ', 'array', 'text',
              ↪ null, null, null, null),
          "extensions" => $this->fileRepository->request($this->dump->getId
              ↪ (), 'file.id, file.extension as text ', 'array', 'text',
              ↪ null, null, null, null),
          "maxSize" => $this->fileRepository->request($this->dump->getId(),
              ↪  'MAX(file.size)', 'int', null, null, null, null, null)
      );
```

Enfin on effectue un rendu de la page avec les données suivantes : la liste des filtres, le filtre à afficher, l'analyse active et les données pour le formulaire de sélection.

```php
return $this->render('dump/disk.html.twig',[
    'filtersList' => $this->filters,
    'currentFilter' => $filter,
    'currentDump' => $this->dump,
    'selectionData' => $selectionData,
]);
```

Pour la fonction **getFilterData()**, qui est appelée par la route **/dump/-disk/*dumpSlug*/*filter*/data**, on a aussi 2 paramètres, **$dumpSlug** et **$filter** qui sont les mêmes que pour la fonction précédente. Cette fonction retourne un fichier JSON qui est un format qui permet au rendu de la page une analyse simple. Pour tous les filtres, les données sont reçues via des requêtes AJAX.

```php
public function getFilterData(string $dumpSlug, string $filter):
    ↪ JsonResponse
```

Comme pour **displayDiskDump()**, on identifie l'analyse et on effectue plusieurs tests. Si l'analyse n'existe pas, qu'aucun utilisateur n'est connecté, que l'analyse n'appartient pas à l'utilisateur connecté ou qu'aucun filtre n'est renseigné, on renvoie un JSON vide.

```php
foreach ($this->dumpRepository->findAll() as $dump){
          if($dump->getSlug() == $dumpSlug)
              $this->dump = $dump;
      }

if ($this->dump == null || $this->getUser() == null || $this->getUser()->
    ↪ getId() != $this->dump->getUser() || $filter == "")
   return new JsonResponse();
```

Avant de faire appel à la fonction **request** de **FileRepository** pour récupérer les données, il faut d'abord récupérer les paramètres placés dans l'url en query string. On crée donc une liste et on y ajoute les paramètres si ils sont compatibles. Ensuite on crée une autre liste pour les paramètres pour la fonction **request()** et on les ajoute.

```php
$URLparameters = array();
foreach ($_GET as $k => $v){
          if( ($k == "directories") || ($k == "types") || ($k == "
              ↪ extensions") || ($k == "extensionsType" && ($v == "correct
              ↪ " || $v == "incorrect")) || (($k == "minSize" || $k == "
              ↪ maxSize" || $k == "createdAt" || $k == "modifiedAt") &&
              ↪ is_numeric($v)) )
                 $URLparameters[$k] = $v;
      }

```

```php
$parameters = array();
foreach ($URLparameters as $k => $v){
    switch($k){
        case 'directories' :
            array_push($parameters, array('dir.name in (:'.$k.')', $k, $v));
            break;
        case 'types' :
            array_push($parameters, array('file_type.type in (:'.$k.')', $k,
                ↪ $v));
            break;
        case 'minSize':
            array_push($parameters, array('file.size >= :'.$k, $k, $v));
            break;
        [...]
    }
}
```

Ensuite en fonction du filtre indiqué, on effectue la requête pour accéder aux données et les formater de façon à ce qu'elles soient compréhensibles lors du rendu de la page. Par exemple, comme ci-dessous, avec le filtre **Aperçu**, on effectue 3 requêtes : on demande le nombre de fichiers, de répertoires et la somme des tailles des fichiers. La somme des tailles est convertie de bits en format standard, avec des KB, MB, GB... Pour le filtre **Liste de fichiers**, on demande toutes les informations pour chaque fichier (nom, répertoire, extension, taille...) puis on change le format des tailles et des dates de créations et de modification pour qu'elles soient compréhensibles simplement. Si le filtre renseigné n'existe pas on renvoie un JSON vide.

```php
switch ($filter){
    case 'preview':
        return new JsonResponse(array(
            "files" => $this->fileRepository->request($this->dump->getId(), '
                ↪ count(file.id)', 'int', null, null, null, null, null,
                ↪ $parameters),
            "directories" => $this->fileRepository->request($this->dump->
                ↪ getId(), 'count(DISTINCT(file.parent))', 'int', null, null
                ↪ , null, null, null, $parameters),
            "size" => $this->bytesToStandard($this->fileRepository->request(
                ↪ $this->dump->getId(), 'sum(file.size)', 'int', null, null,
                ↪  null, null, null, $parameters))));

    case 'list-files':
        $rows = $this->fileRepository->request($this->dump->getId(), 'file.
            ↪ name as name, file.extension as extension, file_type.
            ↪ extension_correct as extension_correct, dir.name as directory
            ↪ , file.size as size, file.created_at as created_at, file.
            ↪ modified_at as modified_at, file_type.type as type', 'array',
            ↪  null, null, null, null, null, $parameters);
        $rows = array_map(function ($row){
```

```
            $row['size'] = $this->bytesToStandard($row['size']);
            $row['created_at'] =(gmdate("d/m/Y", substr($row['created_at'
                ↪ ],0,10)));
            $row['modified_at'] =(gmdate("d/m/Y", substr($row['modified_at'
                ↪ ],0,10)));
            $row['extension_correct'] = $row['extension_correct'] == "true" ?
                ↪  "&#10003;" : "";
            return $row;
        }, $rows);
        return new JsonResponse($rows);

    [...]

    default :
        return new JsonResponse();
```

4.7 Filtres

4.7.1 Aperçu

Ce premier filtre est un filtre très simple ; il affiche, comme on peut le voir ci-dessous, le nombre de fichiers, le nombre de répertoires dans lesquels sont les fichiers et la somme totale de la taille les fichiers. Pour récupérer les données on utilise la fonction **get()** de JQuery et ensuite on les insère dans les éléments du filtre.

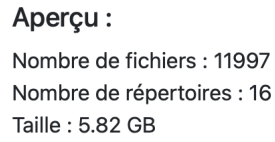


FIGURE 9 – Aperçu

4.7.2 Liste de fichiers

Ce filtre affiche un tableau de 8 colonnes : nom, extension affichée, extension correcte, répertoire, taille, créé le, dernière modification le et type. Il est créé avec Bootstrap Table et ses données sont récupérées avec l'attribut **data-url** de Bootstrap Table. On active la pagination avec l'attribut **data-pagination**. Pour trier les tailles et les dates, on crée 2 fonctions JavaScript **sizeSorter()** et **dateSorter()** car leurs colonnes ne peuvent être triées automatiquement du fait de leur format. On définit quelle fonction utiliser pour trier les colonnes avec l'attribut **data-sorter**.

Nom	Extension affichée	Correcte ?	Repertoire	Taille	Crée le	Dernière modification le	Type
000922	java	✓	test_files	13.82 KB	12/03/2021	08/01/2003	Unknown
077000	txt	✓	077	3.09 MB	12/03/2021	01/05/2006	Unknown
077001	txt	✓	077	158.25 KB	12/03/2021	07/02/2009	Unknown
077002	txt	✓	077	43.92 KB	12/03/2021	30/10/2002	Unknown
077003	txt	✓	077	1.3 MB	12/03/2021	06/02/2009	Unknown
077004	txt	✓	077	884.78 KB	12/03/2021	16/02/2006	Unknown
077005	txt	✓	077	35.68 KB	12/03/2021	30/10/2002	Unknown
077006	txt	✓	077	2.95 KB	12/03/2021	24/05/2249	Unknown
077007	txt	✓	077	75.93 KB	12/03/2021	18/03/2285	Unknown
077008	txt	✓	077	977.98 KB	12/03/2021	16/02/2006	Unknown

Affiche de 1 à 10 sur 11997 lignes 10 lignes par page ‹ 1 2 3 4 5 ... 1200 ›

FIGURE 10 – Liste de fichiers

4.7.3 Vérification d'extensions

Ce filtre affiche un diagramme circulaire de la part d'extension correcte et incorrecte dans les fichiers. Il est réalisé avec Google Charts et sa fonction **PieChart()**, les données sont récupérées avec la fonction **ajax()** de JQuery.

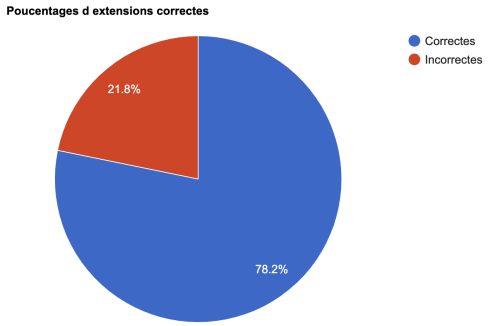
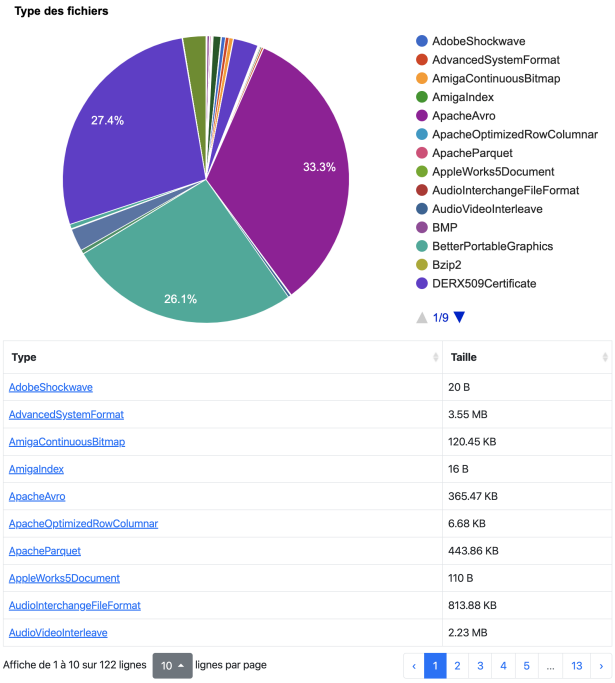


FIGURE 11 – Vérification des extensions

4.7.4 Types de fichiers

Ce filtre affiche aussi un diagramme circulaire de tous les différents types de fichiers. Il affiche la part en taille de fichiers de chaque type. Comme pour le filtre précédent, il est réalisé avec Google Charts. Le filtre affiche en plus un tableau de 3 colonnes : le type, la somme des tailles des fichiers de ce type et la part de la somme des tailles. Ce tableau, réalisé avec Bootstrap Table, permet de mieux visualiser les types avec une petite somme de taille.



Type	Taille
AdobeShockwave	20 B
AdvancedSystemFormat	3.55 MB
AmigaContinuousBitmap	120.45 KB
AmigaIndex	16 B
ApacheAvro	365.47 KB
ApacheOptimizedRowColumnar	6.68 KB
ApacheParquet	443.86 KB
AppleWorks5Document	110 B
AudioInterchangeFileFormat	813.88 KB
AudioVideoInterleave	2.23 MB

Affiche de 1 à 10 sur 122 lignes 10 lignes par page ‹ 1 2 3 4 5 ... 13 ›

FIGURE 12 – Types de fichiers

4.7.5 Création des fichiers et Modification des fichiers

Ces filtres affichent un diagramme linéaire correspondant respectivement au nombre de fichiers créés par jour et au nombre de fichiers modifiés par jour. La date de début du diagramme correspond au premier jour où des fichiers sont créés/modifiés et la date de fin du dernier. Il est réalisé avec Google Charts et sa fonction **LineChart()**. Étant donné que l'intervalle entre la date de début et de fin du diagramme peut être importante, un curseur est rajouté dessous afin de pouvoir sélectionner et voir plus précisément une plus petite intervalle.

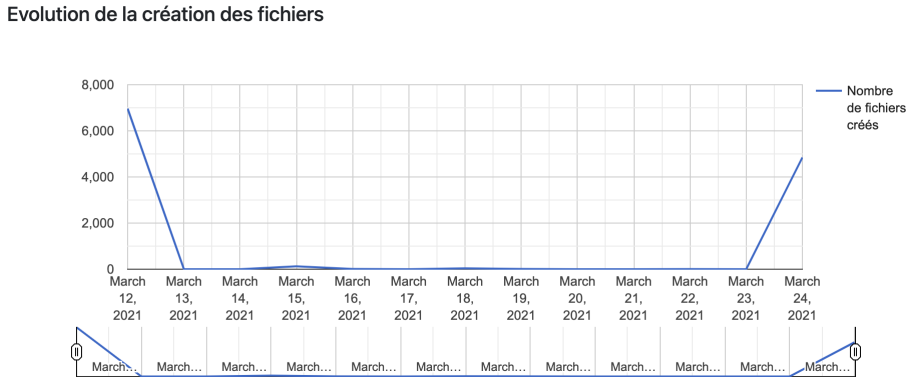


FIGURE 13 – Création des fichiers

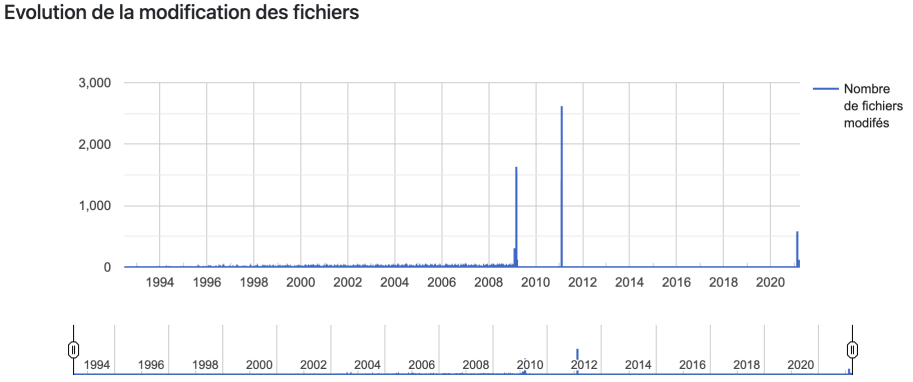


FIGURE 14 – Modification des fichiers

4.7.6 Blocs de données

Ce dernier filtre est réalisé pour une analyse de cartes mémoires. Il permet d'afficher une représentation des données d'une carte. Inspiré de l'outil de défragmentation de Windows, il affiche un tableau de blocs de couleurs, chaque couleur correspondant à un type de donnée. Il est réalisé à l'aide de D3.js. On créé un objet SVG avec D3.js qui sera notre tableau. Afin de le colorer, on crée une liste de couleurs de la taille du nombre de types de données. Ainsi, chaque type de données a une couleur associée dans le tableau. En cliquant sur un bloc de données, son type de données s'affiche à côté du tableau. On pourra aussi afficher aussi la valeur des données à côté du type.

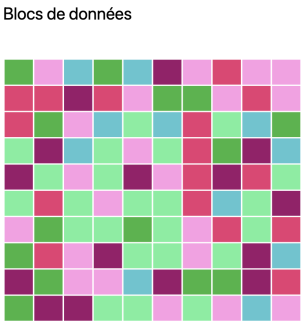


FIGURE 15 – Blocs de données

4.8 Interactivité de la plateforme

4.8.1 Formulaire de sélection

Le formulaire permet d'affiner la sélection de fichier sur laquelle on applique les filtres. Par exemple, il permet d'appliquer un filtre seulement sur les fichiers ayant pour extension .txt. Les paramètres choisis sont ensuite envoyés via l'url en paramètre GET.

Tout d'abord il y a 3 sélecteurs multiples pour la sélection de répertoires, extensions et types créés avec l'outil select2. Ensuite 3 boutons radios pour la vérification d'extension. Puis une glissière pour la taille minimale et maximale avec le fonction **slider()** de JQuery. Enfin deux sélecteurs de dates pour la date de création et de modification. Le bouton **Rechercher** permet de rafraîchir la page avec la sélection du formulaire. Avec un script JavaScript et du JQuery, on récupère les informations du formulaire et on les met en paramètre de la nouvelle url. Le bouton **Vider** permet de remettre à zéro tous les champs du formulaire.

FIGURE 16 – Formulaire de sélection

4.8.2 Interactivité sur les filtres

Pour rajouter de l'interactivité sur les tableaux, on peut cliquer sur l'information d'une case, ce qui la selectionne. Par exemple si on clique sur une case de répertoire **test_files**, ce répertoire est donc sélectionné. Pour rajouter des liens aux tableaux, on utilise l'attribut **data-formatter** de Bootstrap Table. Via la fonction définie par cet attribut, on modifie la valeur des cases souhaitées par

des liens. L'exemple ci-dessous montre, pour le filtre **Liste des fichiers**, que le lien, en cliquant sur l'extension **txt**, renvoie vers ce même filtre mais avec l'extension **txt** en paramètre.

FIGURE 17 – Lien sur la case de l'extension .txt

Pour les diagrammes avec Google Charts, on peut faire le même principe en cliquant sur un élément du graphique, que ce soit un élément du diagramme circulaire ou du linéaire. Pour cela on utilise la fonction JavaScript **event . addListener()** pour la sélection d'un élément. On fait donc appel à une fonction à la sélection, et on y change l'url de la page, en y ajoutant en paramètre l'élément sélectionné. Cela fonctionne pour les filtres **Vérification d'extensions**, **Types des fichiers**, **Création des fichiers** et **Modification des fichiers**

5 Conclusion

Pour conclure, j'ai effectué mon stage de fin d'études de licence au laboratoire GREYC au sein de l'équipe SAFE. Lors de ce stage de 2 mois, j'ai pu mettre en pratique mes connaissances théoriques acquises durant ma formation sur le développement web tout en découvrant de nouveaux outils tels Bootstrap, JQuery ou Symfony.

Ce stage a été enrichissant pour moi, car il m'a permis de découvrir le domaine de la recherche. Cependant je regrette de ne pas avoir pu travailler en présentiel. En effet, j'aurais aimé découvrir le fonctionnement général du laboratoire de recherche au quotidien ainsi que travailler et m'inscrire dans une équipe.

6 Bibliographie

Bootstrap - The most popular HTML, CSS, and JS library in the world.
`https://getbootstrap.com`

Bootstrap Table - An extended table to the integration with some of the most widely used CSS frameworks. (Supports Bootstrap, Semantic UI, Bulma, Material Design, Foundation)
`https://bootstrap-table.com`

D3.js - Data-Driven Documents
`https://d3js.org`

Google Developers - Charts
`https://developers.google.com/chart`

GREYC UMR CNRS 607 - Groupe de Recherche en Informatique, Image, et Instrumentation de Caen
`https://www.greyc.fr/`

Document du GREYC - Vers une plateforme forensique

JQuery `https://jquery.com`

Symfony - High Performance PHP Framework for Web Development
`https://symfony.com`

Twig - The flexible, fast, and secure template engine for PHP
`https://twig.symfony.com`

Wikipedia - Science Forensique
`https://fr.wikipedia.org/wiki/Science_forensique`