Développement d'une base de connaissances interactive dédiée aux barrages

M. Zaïri

Rapport de stage

2ème année ISIMA

présenté par

Mariem ZAIRI

Sujet : Développement d'une base de connaissances interactive dédiée aux barrages

Responsable entreprise : **Stéphan BERNARD**

Responsable ISIMA : **Gilles LEBORGNE**

ISIMA
Institut Supérieur d'Information de Modélisation et de leurs Applications

Cemagref
Sciences, eaux & territoires

Durée : 5 mois du 4 avril au 2 septembre 2011

REMERCIEMENTS

Mes remerciements s'adressent principalement à mon maître de stage, Stéphan Bernard, pour m'avoir accordé sa confiance dans ce projet et pour toute l'aide qu'il m'a apporté afin de me permettre de mieux cerner le travail à réaliser tout au long du stage.

De même, je remercie mon responsable à l'ISIMA, Gilles Leborgne, pour son suivi et son soutien au cours de ce stage.

Mes remerciements vont également à Madame Murielle Mouzat, ma professeure de communication pour m'avoir appris la bonne démarche pour la réalisation d'un rapport et d'une présentation orale.

Enfin, merci à toute l'équipe TSCF, pour son accueil, sa disponibilité et sa bonne humeur permanente.

RÉSUMÉ ET ABSTRACT

RÉSUMÉ

Depuis plusieurs années, des recherches visant à capitaliser la connaissance sur le comportement des barrages sont réalisées au CEMAGREF d'Aix-en-Provence. Ces travaux ont pour but de développer des méthodes d'aide à l'expertise en complément aux méthodes mécaniques pour évaluer la sûreté de fonctionnement des **barrages** en service.

Dans la lignée des précédents travaux réalisés, qui ont permis d'obtenir les différents scénarios de **vieillissement** des barrages, mon travail a consisté à développer une base de connaissances interactive sur les pathologies des barrages.

Cette base de connaissances a pour but de structurer le maximum d'informations sur les barrages, permettant ainsi à ses utilisateurs de mieux en comprendre les différents mécanismes de rupture et de **dégradation**.

Ce projet propose une solution pour stocker les informations de chaque barrage dans une **base de données**, une visualisation par le biais d'une **interface Web** et une gestion des droits d'accès par un système d'authentification.

Mots clés : Barrage, vieillissement, dégradation, interface Web, base de données

ABSTRACT

Research about knowledge capitalization for embankment dams behavior have been conducted for many years in Cemagref, a public research institute, at its center based at Aix-en-Provence. The goals of these works are on one hand to develop expertise assistance methods, in addition to reliability mechanics methods, and on the other hand to assess embankment **dams** operational safety methods.

To complete anterior researches, from which different embankment dam's **aging** scenarios have been obtained, my work consisted in the development of an interactive knowledge database for embankment dams pathology.

This knowledge base is to structure the maximum amount of information on dams and allow its users to have better understanding of the different failure mechanisms and **degradation** of dams.

This project suggests a solution to store information for each dam in a **database**, visualizing them through a **web interface** with the management of access rights thru an authentication system.

Keywords : Dam, aging, degradation, web interface, database

TABLE DES FIGURES ET ILLUSTRATIONS

TABLE DES MATIÈRES

INTRODUCTION

Des accidents liés aux dégradations des barrages, due au vieillissement de ces derniers, à une mauvaise conception ou encore à l'action de l'homme, peuvent se produire et entraîner la rupture de ces ouvrages. Par ailleurs, cette rupture peut avoir des conséquences humaines, économiques ou encore environnementales non négligeables.

Ainsi, il est donc nécessaire de produire des méthodes et des outils permettant de maîtriser la sécurité des ouvrages hydrauliques, afin de pouvoir déterminer la fiabilité de ces derniers.

Parmi différentes méthodes possibles, certains travaux reposent sur l'expertise de ces ouvrages, et plus précisément sur la formalisation des dires d'experts dans le but d'évaluer la performance globale du barrage et sa sûreté de fonctionnement. Ces travaux sont réalisés pour permettre de vérifier que les barrages sont dans un état de sécurité satisfaisant, et si ce n'est pas le cas, d'effectuer des actions correctives pour réhabiliter ces ouvrages.

La thématique de mon stage que j'ai effectué au sein du Cemagref de Clermont-Ferrand, s'inscrit dans ce contexte de formalisation des dires d'experts afin de pouvoir, bénéficier d'un outil permettant d'évaluer la performance d'un barrage vis-à-vis de différents modes de rupture déjà constatés. Le développement d'une base de connaissances interactive des barrages permet entre autres de rassembler en un seul outil un certain nombre de connaissances, de pouvoir faire une interrogation guidée des informations stockées et de fixer une terminologie commune concernant les phénomènes de dégradations des fonctions de ces ouvrages.

Pour commencer, ce rapport présente une synthèse du contexte professionnel.

Elle est suivie d'une partie plus technique. Je commencerai par présenter la partie analyse du projet, puis les choix technologiques, et enfin le développement de l'application.

Ce rapport se terminera par une conclusion de ce stage, comment je l'ai ressenti, une explication des différents problèmes rencontrés et un bilan personnel.

I. PRÉSENTATION DU CONTEXTE DE STAGE

1. Présentation de l'organisme d'accueil : le Cemagref de Clermont-Ferrand

Établissement public à caractère scientifique et technologique, crée en 1981, le Centre national d'Études sur le Machinisme Agricole, le Génie Rural, les Eaux et Forêts (CEMAGREF) se concentre sur l'ingénierie de l'agriculture et de l'environnement. Il dépend du Ministère de l'agriculture et du Ministère de la recherche.

Les activités du Cemagref sont regroupées au sein des 3 départements de recherche qui portent sur l'eau, les écotechnologies et les territoires ; son organisation comprend un pôle recherche et innovation ainsi qu'un pôle soutien à la recherche.

Le CEMAGREF de Clermont-Ferrand accueille 3 Unités de Recherche et d'Expertise (URE) :

- **DFCF** : Dynamiques et Fonctions des espaces ruraux
- **TSCF** : Technologies, Systèmes d'information et procédés pour l'agriculture et l'agro-alimentaire
- **LISC** : Laboratoire d'Ingénierie des Systèmes Complexes

Pour ma part, j'ai été accueillie au sein de l'équipe COPIN de TSCF qui est composé d'une soixantaine d'agents du Cemagref pour réaliser une application proposée par le CEMAGREF d'Aix en Provence.

Le stage a été effectué au Cemagref de Clermont-Ferrand. En effet, l'équipe TSCF travaille sur le système d'information tandis que le Cemagref d'Aix sont spécialisés dans les barrages.

2. Présentation du l'organisme client : le Cemagref d'Aix en Provence

Le Centre d'Aix-en-Provence est l'un des dix groupements du Cemagref, établissement public à caractère scientifique et technologique (EPST). Les activités de recherche et d'expertise sont conduites par les Unités de Recherche (UR) à compétence nationale. Les quatre Unités de Recherche d'Aix-en-Provence travaillent principalement dans deux domaines : l'eau et les espaces ruraux.

- **Eau**
  - **Ur Ouvrages hydrauliques et hydrologie** : Sécurité des ouvrages hydrauliques et hydrologie
  - **Ur Hydrobiologie** : Structuration et fonctionnement biologiques des cours d'eau
  - **Équipe LERM :** Performances et impact environnemental de l'irrigation

- **Espaces ruraux**
  - **Ur Écosystèmes méditerranéens et risques :** Dynamique des espaces agricoles et forestiers, notamment en zone péri-urbaine, risques liés aux feux de forêt

II. PROJET

1. Besoin des experts

1.1. Cahier des charges

Actuellement, les connaissances explicites sur la conception-réalisation et les modes de rupture et dégradation des barrages sont nombreuses mais dispersées. Elles présentent en outre des formats différents tels que des textes, des photographies ou des vidéos. Il est donc intéressant de rassembler, capitaliser et expliciter les connaissances actuelles et futures. Les experts souhaitent donc avoir une base de connaissances dédiée aux barrages et axée plus particulièrement sur leur conception et leur réalisation ainsi que leurs modes de rupture et de dégradation. L'existence d'une telle base de connaissances a plusieurs intérêts et permet de :

- Capitaliser les connaissances, les rassembler en un seul outil et enrichir cet outil au fur et à mesure de la création de nouvelles connaissances.
- Constituer un outil de formation : elle permet par exemple à un utilisateur de trouver des définitions, elle aide à la formation d'un jeune expert.
- Produire un outil de documentation : elle permet alors à un utilisateur de se référer à un barrage qui aurait subi un même désordre ou d'accéder à une préconisation d'action corrective.
- Faciliter la communication entre les utilisateurs notamment par la définition d'un vocabulaire commun. Cet outil de communication peut être à usage interne d'un panel d'experts dialoguant sur un même problème mais aussi à usage externe entre un animateur non expert et un groupe d'experts.

1.2. Étude du cahier des charges

Le cahier des charges étant très permissif, il fallait plutôt se fixer des limites, ou tout du moins, avoir une idée plus précise de ce que serait l'application.

C'est pour cela qu'à partir de ce cahier des charges, j'ai réalisé des croquis pour présenter mes idées à mon tuteur d'une manière plus visuelle. C'est ainsi que certains points ont été améliorés. Comme par exemple la page d'accueil ou l'édition des incidents.

2. Choix technologique

La première question qui s'est posée pour la réalisation de la partie technique du projet à été de choisir une architecture pour l'application.

2.1. La Méthode Modéle-Vue-Contrôleur (MVC)

Le MVC est une architecture et une méthode de conception qui organise l'Interface Homme-Machine d'une application logicielle (dans notre cas un site Web) en trois parties :

- Les données (Modèle).
- L'interface home-machine (Vue).
- La logique de contrôle (Contrôleur).

Grossièrement, cela permet une séparation entre les traitements de données et la présentation.

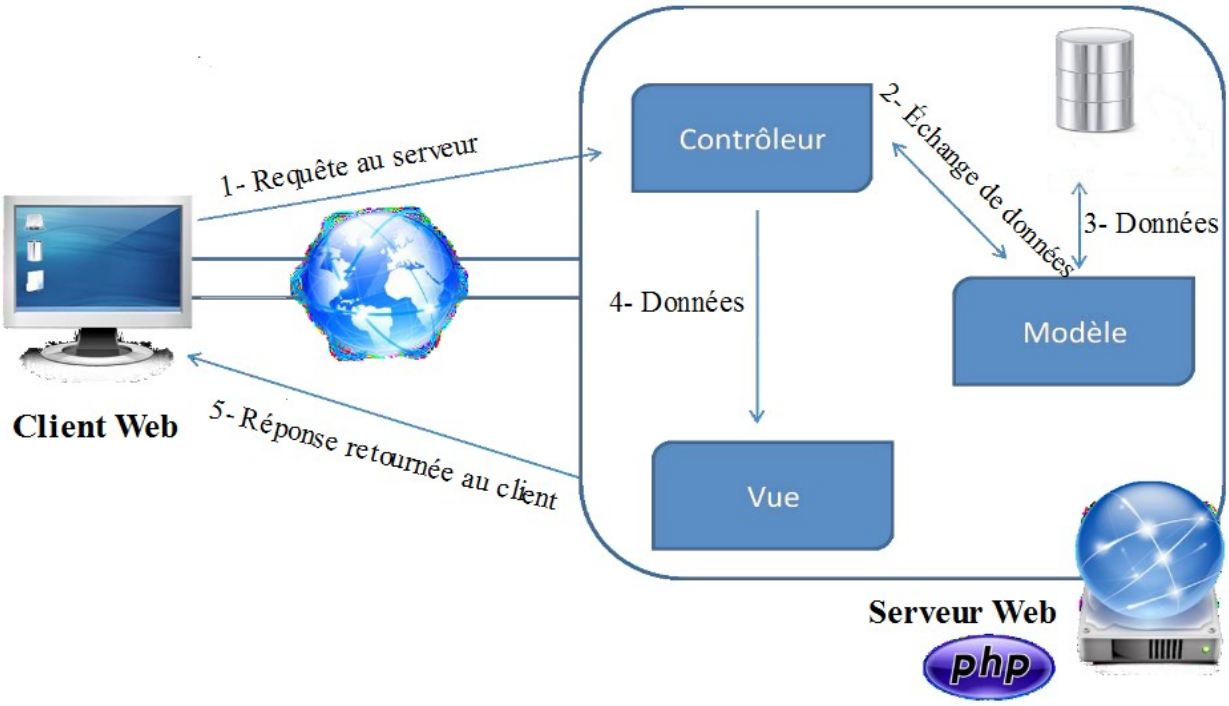

Illustration 1: Interactions entre le modèle, la vue et le contrôleur

2.1.1. Le modèle

Le modèle représente les structures de données. Typiquement, les classes modèles contiennent des fonctions qui aident à récupérer, insérer et mettre à jour des informations de la base de données.

2.1.2. La vue

La vue correspond à l'interface avec laquelle l'utilisateur interagit. Elle se présente sous la forme d'un *template* représentant l'interface, mais sans les données.

2.1.3. Le contrôleur

Il gère l'interface entre le modèle et le client. Il va interpréter la requête de ce dernier pour lui envoyer la vue correspondante. Il effectue la synchronisation entre le modèle et les vues.

2.2. From Scratch, Framework ou CMS ?

Les trois grandes possibilités existantes pour réaliser ce type de site Web sont : le codage « from scratch », c'est à dire en partant de zéro, l'utilisation d'un CMS ou l'utilisation d'un framework.

2.2.1. Codage From Scratch

Le codage « from scratch » consiste à programmer une application du début jusqu'à la fin. Cette solution à été immédiatement abandonnée, car elle est trop lourde et longue à développer, et rendue totalement inutile grâce à la quantité de framework et CMS open source mise à disposition par l'énorme communauté de développeurs Web sur internet.

2.2.2. Utilisation d'un CMS

Un CMS est un système de gestion de contenu. C'est un site Web déjà réalisé qu'il faut ensuite adapter à ses propres besoins. Les fonctions incontournables d'un CMS sont :

- Séparation entre contenu et présentation (système de *templates*).
- Édition de page simplifiée par éditeur *WYSIWYG*.
- Gestion des droits.
- Utilisation d'interface Web d'administration (back-end) pour la gestion du contenu.
- Système de plug-ins ou greffons.

Le CMS semble donc offrir beaucoup d'avantages en fournissant une plate-forme prête à l'emploi et modulaire. La plupart des CMS disposent d'énormément de fonctionnalités, comme un forum, un système de commentaires, et beaucoup de fonctionnalités dites « Web 2.0 », c'est-à-dire communautaires.

2.2.3. Utilisation d'un Framework

Un framework ou kit de développement est un espace de travail modulaire, c'est à dire une suite d'outils et de bibliothèques qui facilitent et accélèrent le développement d'un logiciel. Il contient toutes les fonctions de base utiles au développement d'un type de programme, et permet donc de ne pas avoir besoin de ré-écrire les mêmes fonctions à chaque programme créé. Il en existe dans tous les langages de programmation.

Cette solution est à mi chemin entre le codage « from scratch » et l'utilisation d'un CMS. En effet, l'utilisation d'un framework n'interdit pas le codage, bien que certains comme *Ruby on Rails* ont des fonctionnalités de génération automatique de code.

Parmi les frameworks Web, je me suis plus particulièrement intéressée aux frameworks Web PHP. Dans leur grande majorité, ils sont conçus sur le modèle MVC, ce qui permet de structurer les données. Ils imposent un cadre et des normes de développement qui permettant une programmation propre et modulaire.

De plus, depuis la version 5 de PHP, qui introduit la Programmation Orientée Objet (POO), il est beaucoup plus facile et intuitif de programmer des systèmes modulaires.

2.3. Solution retenue

La solution retenue a été de développer la base de données interactive à l'aide d'un framework, et ceci pour plusieurs raisons :

- Un CMS n'est pas adapté pour le genre de base de données interactive que souhaite les experts du Cemagref.

En effet, l'application a réaliser n'est pas un site public ouvert à tout le monde, c'est à dire que la plupart des fonctionnalités d'un CMS ne seraient que fioriture. Et dans l'hypothèse où l'on utiliserait un CMS, il faudrait désactiver ces fonctionnalités, et donc modifier directement le code source.

- La lenteur d'accès aux bases de données qui est visible surtout à l'affichage des pages est un grand inconvénient d'un CMS.

- L'application représentera les barrages et des données confidentielles, il est alors exclu d'utiliser des éléments « pré-construits » que l'on peut retrouver sur d'autre sites. L'utilisation d'un CMS semble donc difficile, car il faudrait tout transformer.

- En revanche, l'utilisation d'un framework implique le développement sur-mesure de tous les éléments du site à l'aide de fonctions relativement simples. L'apprentissage au développement avec un framework apparaît plus simple qu'avec un CMS.

C'est pourquoi, j'ai décidé d'utiliser un framework pour le développement de l'application.

2.3.1. Comparatif des principaux framework PHP

Il existe de nombreux framework PHP open-source, c'est à dire gratuits, librement modifiables et distribuables. J'ai testé et étudié les plus connus pour trouver le plus adapté.

Zend Framework

Zend Framework est un cadre de développement pour PHP5, orienté objet, créé par Zend. Zend étant le créateur du langage PHP, cela lui confère une grande crédibilité. Ses principales fonctionnalités sont :

- Sécurité : système de protection contre les attaques par injections SQL et des attaques de types cross-site-scripting (XSS).
- Séparation du code en trois couches MVC.
- URL simples et claires.
- Architecture du cœur même de framework totalement modulaire, c'est à dire que l'on peut inclure uniquement les fonctionnalités dont on a besoin.

Grâce à cela et à la grande communauté de développeurs, le Zend Framework bénéficie de beaucoup de fonctionnalités comme un système de template, un système de cache, plusieurs implémentations d'*AJAX*, accès à des sources de données diverses et beaucoup d'autres.

Ce framework est surtout destiné aux sites à forte charge et aux développeurs web de métier qui peuvent se permettre de passer des semaines à le prendre en main.

Symfony

Symfony est un Framework MVC open-source écrit en PHP 5, donc orienté objet. Ses principales fonctionnalités sont :

- Une séparation du code en trois couches, selon le modèle MVC.
- Un système de *template* évolué.
- Des performances optimisées et un système de cache pour garantir des temps de réponse optimums.
- Une gestion des url parlantes, qui permet de formater l'url d'une page indépendamment de sa position dans l'arborescence fonctionnelle.
- Un système de configuration en cascade qui utilise de façon extensive le langage YAML8.

- Un générateur de back-office.
- Un système de gestion de langue (pour internationaliser un site).
- Une couche de mapping objet-relationnel (ORM) et une couche d'abstraction de données.
- Le support de l'*Ajax*.
- Une architecture extensible, permettant la création et l'utilisation de plugins.

Symfony semble le framework le plus évolué. Son générateur de back-office et son système de configuration par langage YAML, à l'instar de Ruby on Rails est particulièrement intéressant.

Il semble une très bonne alternative à Zend, disposant des fonctions équivalentes à celui-ci mais plus simple à maîtriser de par sa conception. Sans oublier la génération automatique de code-source par fichier de configuration YAML. De plus, malgré son jeune âge, il est fiable et éprouvé.

Copix

Copix est un framework pour le langage PHP développé par une communauté de développeurs français. Il est construit en cinq couches : coordination, services, domaines, persistance et présentation. Ses principales fonctionnalités sont :

- Un gestionnaire de langue (internationalisation).
- Un système simple et complet de rewriting d'URL.
- Des DAO automatiques permettant l'accès aux données par de multiples moyens.
- Un système de modules pour découper une application en plusieurs sous-systèmes facilement exportables.
- Un système de cache.
- Un système de *template*.

Le modèle en cinq couches de Copix semble plus complexe et moins naturel que le modèle MVC. De plus, par certains aspects, Copix (notamment son back-end) se rapprochent plus d'un CMS que d'un réel Framework.

2.3.2. Choix final

Après une très forte hésitation entre le Zend Framework et Symfony, j'ai décidé d'utiliser Symfony, étant séduite par ses philosophies : rendre le PHP beaucoup plus confortable, l'*AJAX* plus abordable, l'optimisation du référencement (url rewriting) plus simple, l'internationalisation (i18n) triviale.

Fonctionnement de symfony

L'arborescence d'une application Symfony sur le serveur Web se présente de la manière suivante :

Mon_Projet

- **apps** : *Ensemble des applications Symfony*
  - **mon_appli** : *Application Symfony*
    - **i18n** : *Dossier de traduction*
    - **modules** : *Liste des modules*
      - **mon_module**
        - **actions** : *Actions du module (Contrôleur)*
        - **templates** : *Templates du module (Vue)*
- **cache** : *Fichiers mis en cache*
- **config** : *Fichiers de configuration*
  - **doctrine** : *Type d'ORM (Information dans la page suivante)*
    - **schema.yml** : *Fichier de configuration de la base de données*
- **data** : *Données insérées dans la base*
  - **fixtures** : *Insertion des données par défaut*
  - **sql** : *Scripts de création de BD*
- **lib** : *Librairie de l'application*
  - **filters** : *Configuration des filtres de recherche*
  - **forms** : *Configuration des formulaires*
  - **model** : *Classes PHP correspondantes à la BD*
- **log** : *Fichiers des logs de l'application*
- **test** : *Fichiers des tests unitaires et fonctionnels*
- **web** : *Fichiers utiles pour le rendu*
  - **css** : *Feuilles de style*
  - **images** : *images de l'application*
  - **js** : *Scripts javascript*
  - **uploads** : *Dossier de sauvegarde des documents uploadés*
  - **index.php** : *Fichier chargé par le serveur web*

Le développement de l'application s'est fait en deux temps : l'apprentissage de Symfony et le développement de l'application.

L'apprentissage s'est fait grâce au tutoriel présent sur le site du framework. Ce tutoriel présente une vue globale des fonctionnalités de Symfony.

Le premier choix à faire pour le développement a été celui du mapping des objets relationnels (ORM). Ce mapping permet de faire une correspondance entre la base de données relationnelle et les objets du langage.

2.4. Comparatif des deux ORM

ORM (Object Relational Mapping) est une technique de programmation informatique qui crée l'illusion d'une base de données orientée objet à partir d'une base de données relationnelle.

Symfony se base sur l'un ou l'autre des deux principaux ORM du langage PHP : Doctrine et Propel. Ils ont des fonctionnalités quasi identiques. Ils possèdent tous les deux des opérations CRUD (Opérations de base : Create, Read, Update, Delete). Les deux peuvent générer le code des classes PHP. Cependant Propel est basé sur XML alors que Doctrine est basé sur YAML. Pour la rédaction des requêtes, Propel utilise l'approche Criteria/peer alors que Doctrine utilise un langage dérivé de SQL (le DQL).

Exemple de requête (Récupération du Barrage N° 5) dans les deux ORM :

Sous Propel :

```
$b = new Criteria();
$b->add(BarragePeer::ID, 5);
$items = BarragePeer::doSelect($b);
```

Sous Doctrine :

```
$items = Doctrine_Query::create()
        ->from('Barrage b')
        ->where('b.id = ?', 5)
        ->execute();
```

La solution proposée par Doctrine est donc plus facile à interpréter que celle de Propel. L'affectation de valeurs aux attributs est différente. Propel génère tous les accesseurs alors que Doctrine utilise des propriétés « magiques ». Donc la solution de Propel présente l'avantage d'être compatible avec la majorité des outils de développement comme Eclipse.

Malgré cet inconvénient, Doctrine possède une bonne documentation et l'écriture du modèle est plus facile avec le format YAML que le format XML sans outil supplémentaire. C'est pourquoi j'ai choisi Doctrine pour mon application.

Une fois le modèle d'ORM choisi, il faut implémenter la base de données, donc rédiger le fichier YAML. La structure de schéma est très importante pour l'affichage de l'application car certaines pages sont automatiquement générées avec le schéma relationnel.

Quand cette étape est terminée, le codage de l'application peut commencer.

3. Réalisation de l'application

3.1. Schéma relationnel

Un barrage a des caractéristiques physiques, peut subir des événements tels que des incidents ou des actions correctives. Chacun de ces éléments, décrit dans la base de données, peut être enrichi à l'aide de fichiers texte ou de fichiers multimédias.

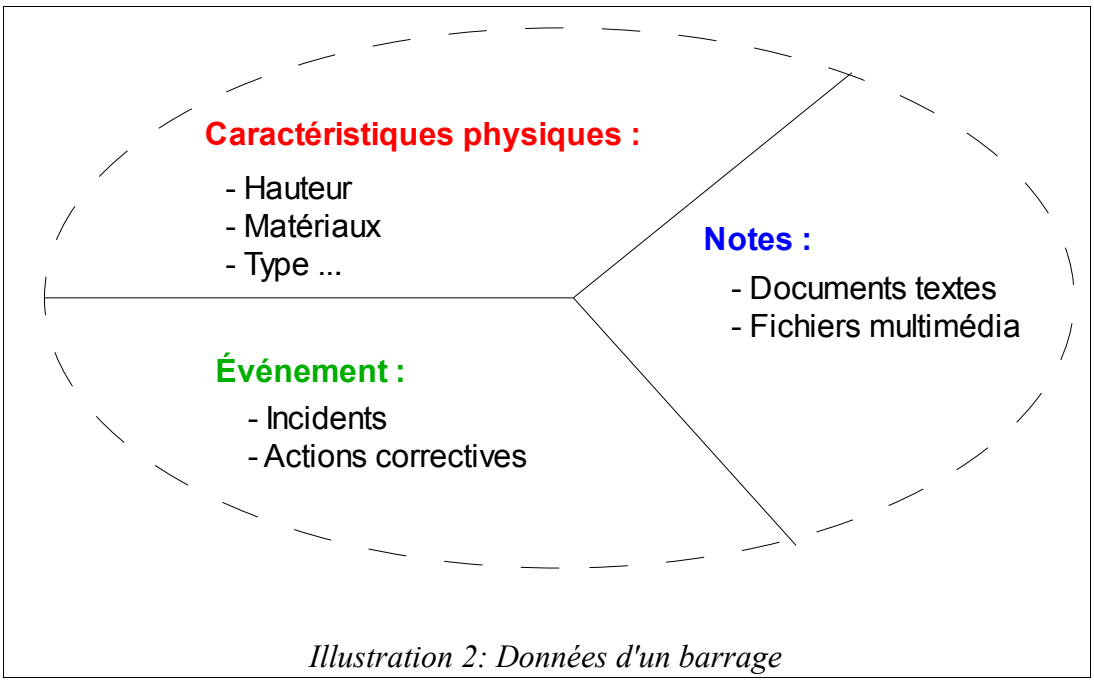

Illustration 2: Données d'un barrage

3.1.1. Caractéristiques physiques

Un barrage possède des caractéristiques physiques simples et d'autres multiples :

Caractéristiques physiques simples

Les caractéristiques physiques simple d'un barrage sont les types de données uniques. Par exemple :

- Le pays (un barrage ne peut appartenir qu'à un seul pays).
- Le type (un barrage ne peut être que d'un seul type à la fois).
- Hauteur maximale par rapport au terrain naturel.
- La capacité.
- Le superficie.

- La longueur.

Ainsi que d'autre données importantes pour son dimensionnement. J'ai choisi l'enregistrement de tout ces derniers dans une seule table « persa_barrage ».

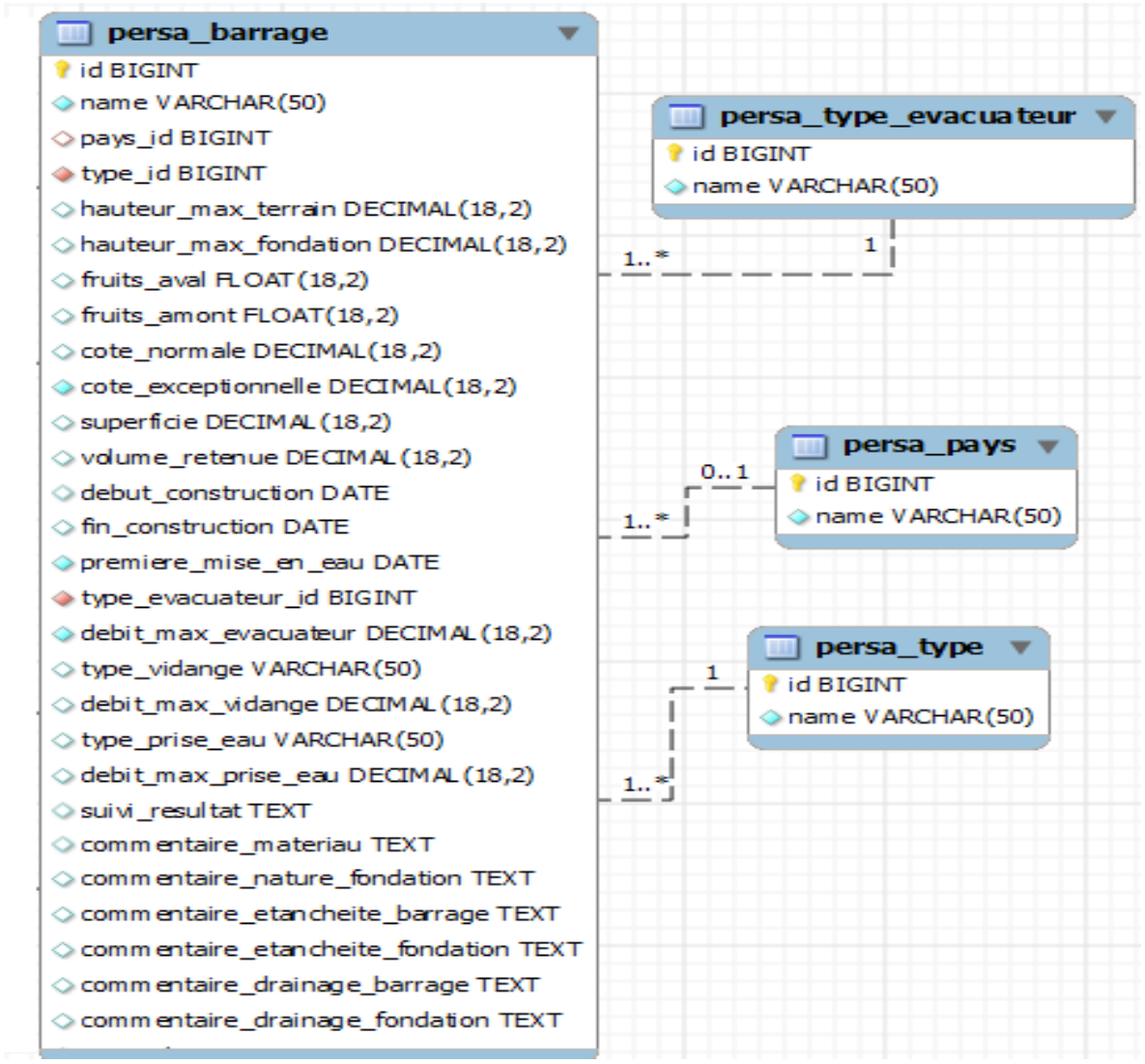

Illustration 3: Schéma relationnel des Caractéristiques physiques simples d'un barrage

« persa_type_evacuateur », « persa_pays » et « persa_type » sont des tables en relation *OneToMany* avec la table « persa_barrage ». Cela permet que les attributs correspondants dans la table « persa_barrage » aient des valeurs appartenant à une liste prédéfinie. Cette liste peut en outre être facilement enrichie au cours du temps.

Caractéristiques physiques multiples

Un barrage a aussi des caractéristiques physiques multiples. En effet, il peut être construit avec plusieurs matériaux, posséder plusieurs dispositifs de mesure, différentes natures de fondations et beaucoup d'autre caractéristiques physiques.

Toutes ces caractéristique sont en relation *ManyToMany* avec la table « persa_barrage ». Cela permet aux utilisateurs de l'application d'avoir des listes déroulantes à choix multiple.

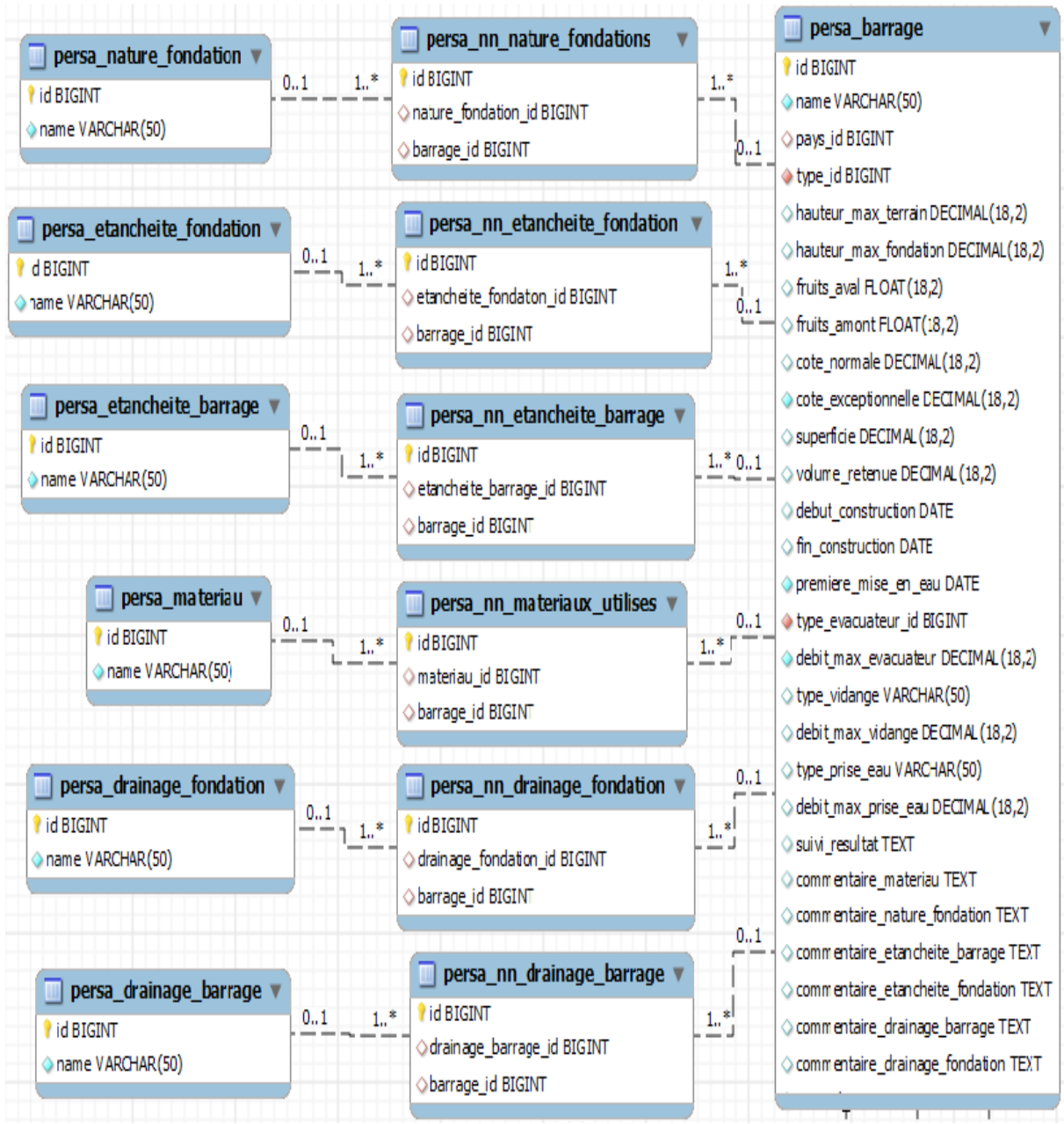

Illustration 4: Schéma relationnel des Caractéristiques physiques multiples d'un barrage

3.1.2. Les événements

Les événements que peut subir un barrage sont des incidents liés à des dégradations ou des phénomènes naturels. Ces incidents peuvent être traités par un ensemble d'actions correctives.

a. Les incidents

Un barrage peut avoir subi un ou plusieurs incidents. Chaque incident peut être lié à une ou plusieurs sources qui désignent les origines de l'incident, et avoir une ou plusieurs conséquences qui participent à la dégradation et deviennent elles-mêmes des incidents. De plus, il y a deux types de modes de rupture d'un barrage. En effet, l'incident peut être lié à des phénomènes naturels ou à la conception-réalisation du barrage.

Cas 1 : Incident lié à une source

Si l'utilisateur de l'application souhaite ajouter un incident de type source, il doit avoir le choix du type de mode de dégradation : Phénomène ou Conception-Réalisation

Cas 2 : Incident lié à une conséquence

Si l'utilisateur choisit d'ajouter un incident de type conséquence, le mode de dégradation est forcément de type Phénomène.

Pour réaliser cela, j'ai d'abord étudié une solution de type *ManyToMany* mais cette solution a été vite abandonnée, car en l'utilisant, le type de mode de dégradation était dupliqué pour chaque catégorie.

- Catégorie A → Phénomène → Type 1
- Catégorie A → Conception-Réalisation → Type 2
- Catégorie B → Phénomène → Type 3
- Catégorie B → Conception-Réalisation → Type 4

On voit bien que pour chaque catégorie il y a deux sous-catégories Phénomène et Conception-Réalisation. Il y aura une redondance des données.

L'héritage est donc la meilleure solution pour gérer cette situation.

b. L'héritage

Schéma Uml des incidents

Pour éviter la redondance des informations dans la base de données on utilise l'héritage :

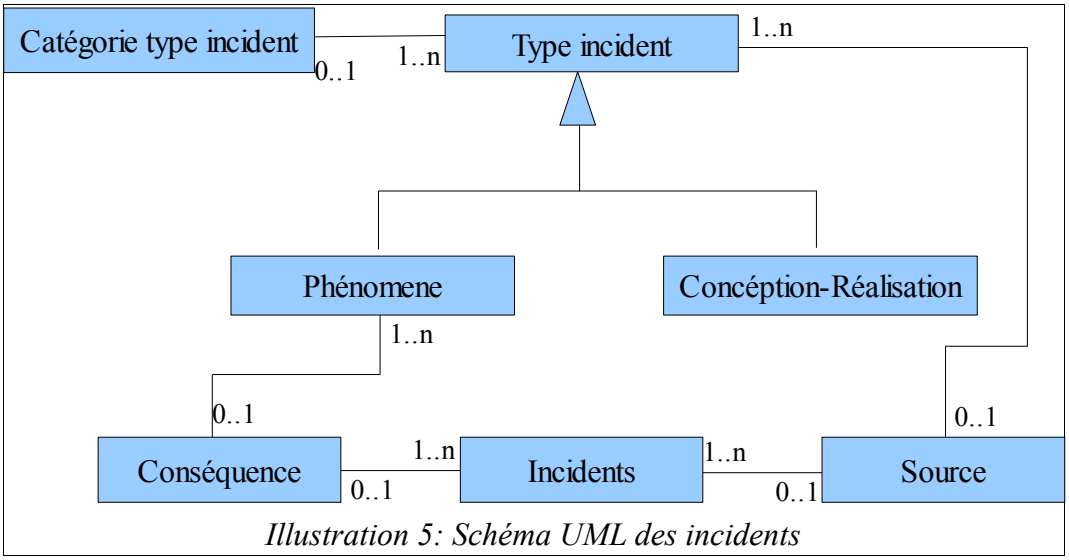

Illustration 5: Schéma UML des incidents

« Phénomène » et « Conception-Réalisation » sont deux tables filles qui héritent de la classe mère « Type incident ».

La table « Conséquence » est une table intermédiaire entre « Phénomène » et « Incidents ». Comme tous les incidents de type conséquence ont un mode de rupture de type Phénomène, il n'y a aucune utilité de les relier avec la table « Type incident ».

La table « Source » est aussi une table intermédiaire mais elle est reliée directement à la classe mère. Ici, l'utilisateur doit pouvoir choisir entre les deux tables filles.

La table « Catégorie type incident » est en relation *OneToMany* avec la table « type incident ». Elle permet de stocker toutes les catégories d'un incident.

Schéma relationnel des incidents

Symfony intègre trois stratégies différentes pour gérer les héritages de table selon les besoins de l'application en termes de performance, d'atomicité, d'efficacité ou bien encore de simplicité. Ces trois stratégies natives sont l'héritage simple, l'héritage par agrégation de colonnes et l'héritage concret.

Pour mon application, j'ai choisi l'héritage par agrégation de colonne. Cet héritage stocke toutes les colonnes, y compris celles des tables filles dans la table mère (ici « type_incidents ») ainsi qu'une colonne type pour identifier les différents types d'enregistrement. Par conséquent, lorsqu'un enregistrement persiste en base de données, une valeur est affectée à cette colonne afin de déterminer à quelle classe il appartient.

Ici les deux tables filles sont représentées par la variable « type » de table « type_incident ».

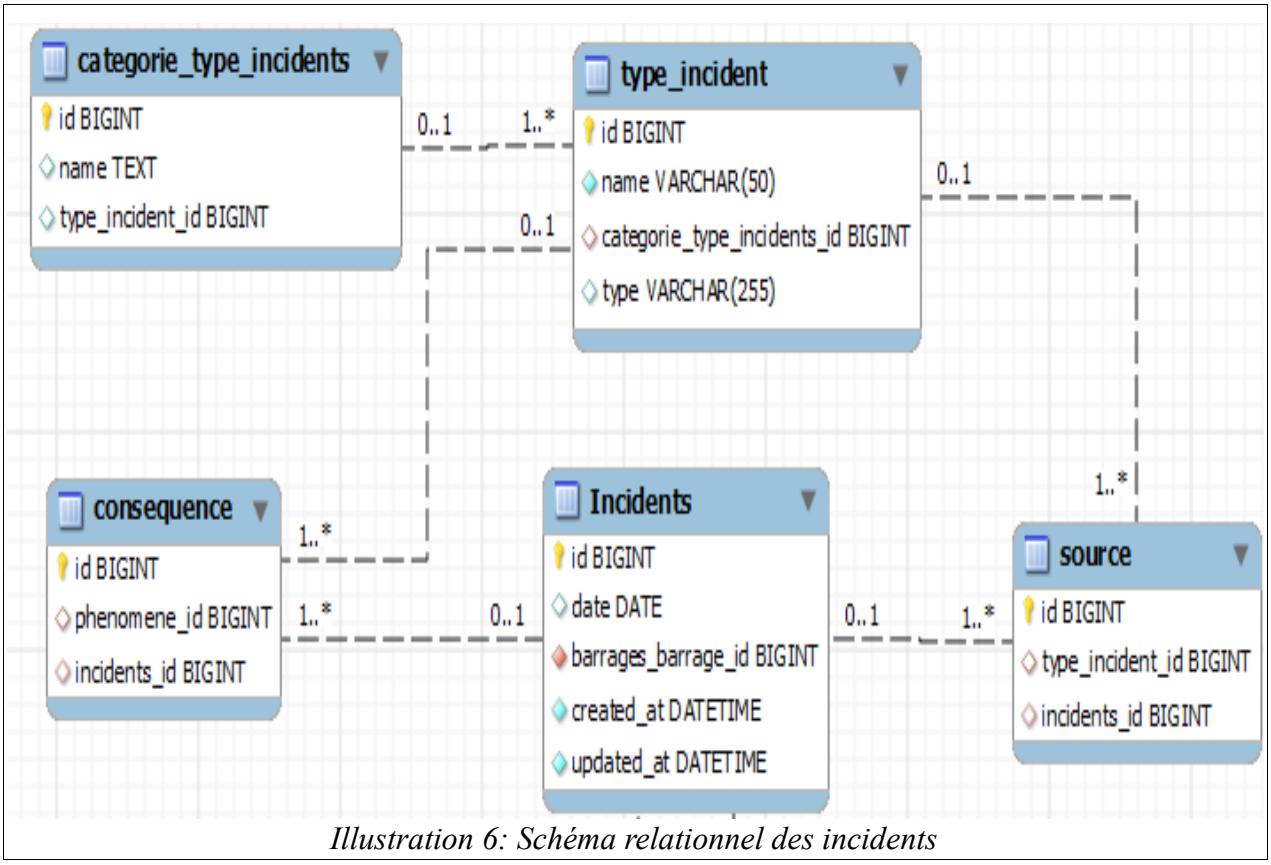

Illustration 6: Schéma relationnel des incidents

c. Les actions correctives

Il peut y avoir plusieurs actions correctives pour un incident. Les actions correctives doivent être associées à l'incident concerné :

Exemple :

- Incident pathologie n°1
  - Action corrective associée n°1
  - Action corrective associée n°2
- Incident pathologie n°2
  - Action corrective associée n°1...

De plus, on a une structure arborescente à trois niveaux : catégories, sous-catégories et type d'action corrective. Pour gérer cette arborescence j'ai utilisé deux relations de type *OneToMany.*

Cependant certaines actions s'arrêtent au niveau 2 (pas de type d'action corrective) il fallait donc gérer ceci avec un booléen.

L'illustration suivante représente l'arborescence des actions correctives. En effet chaque élément de catégorie peut avoir un ou plusieurs éléments de sous catégorie et de même pour ce dernier qui peuvent avoir aucun ou plusieurs type d'action corrective.

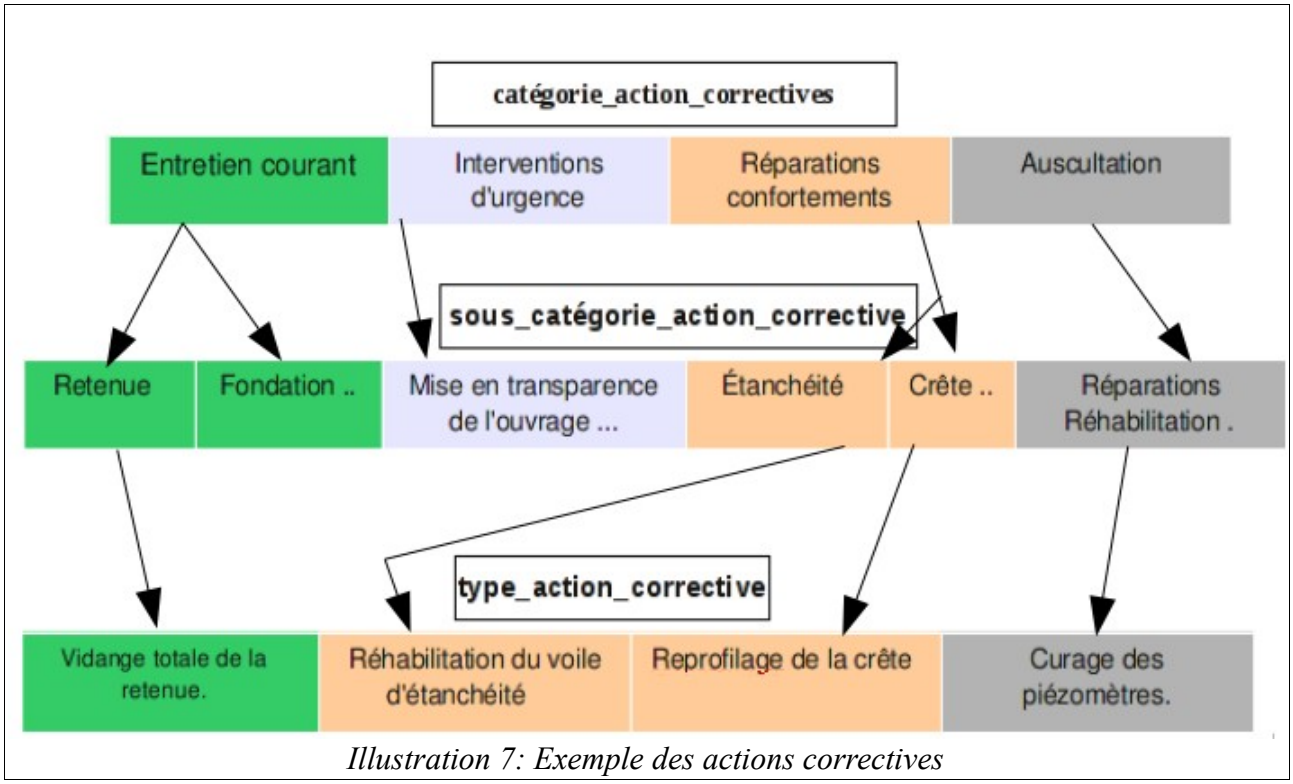

Illustration 7: Exemple des actions correctives

Schéma relationnel des actions correctives

La table « categorie_action_corrective » contient toute les catégories d'action corrective. Chaque élément de cette table est relié à au moins un élément de la table « sous_categorie_action_corrective ». De même pour les éléments de la table « sous_categorie_action_corrective » qui ont une relation *OneToMany* avec les éléments de la table « type_action corrective ».

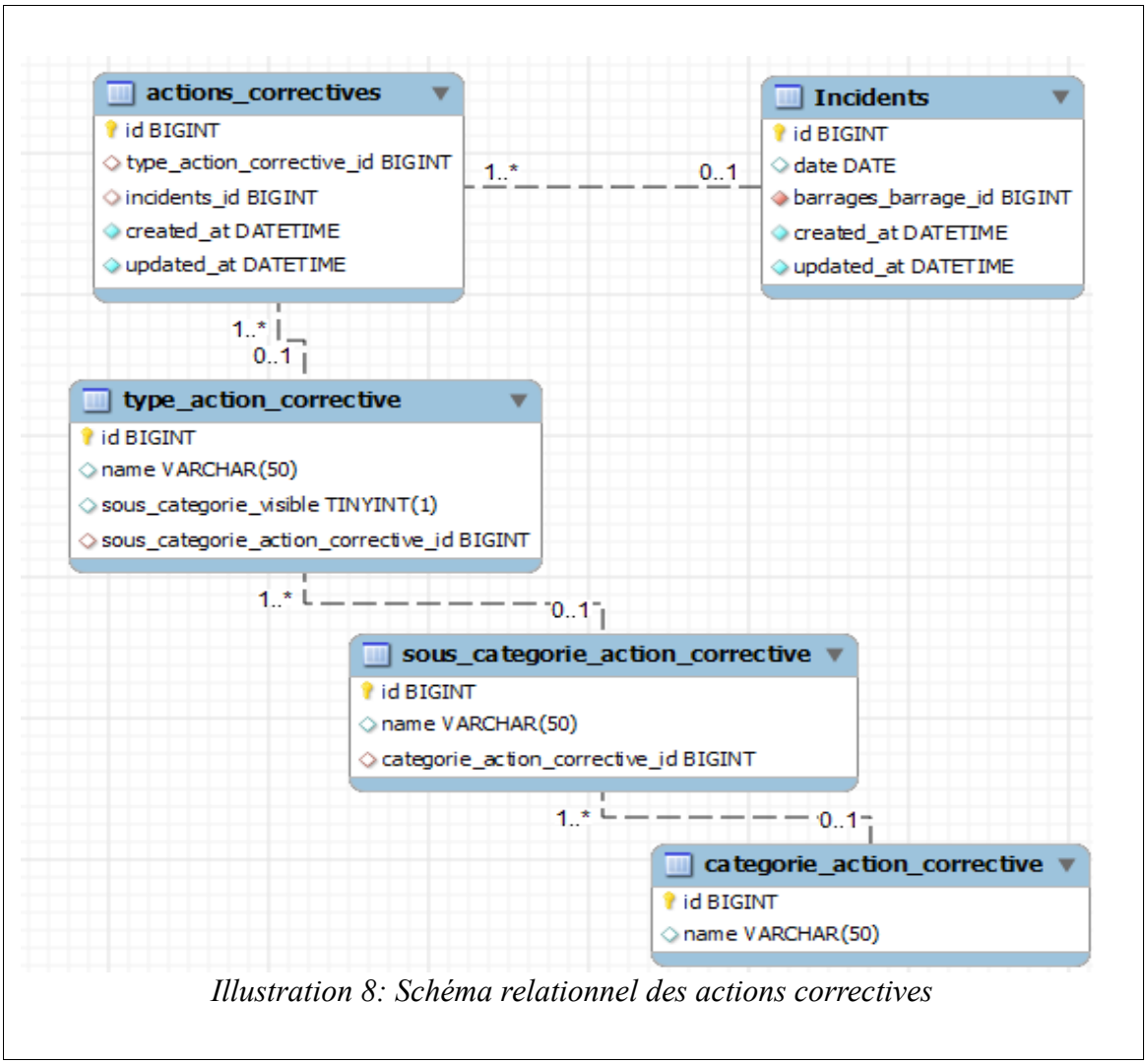

Illustration 8: Schéma relationnel des actions correctives

Le schéma relationnel précédent permet de stocker une arborescence à trois niveaux comme le montre le schéma suivant.

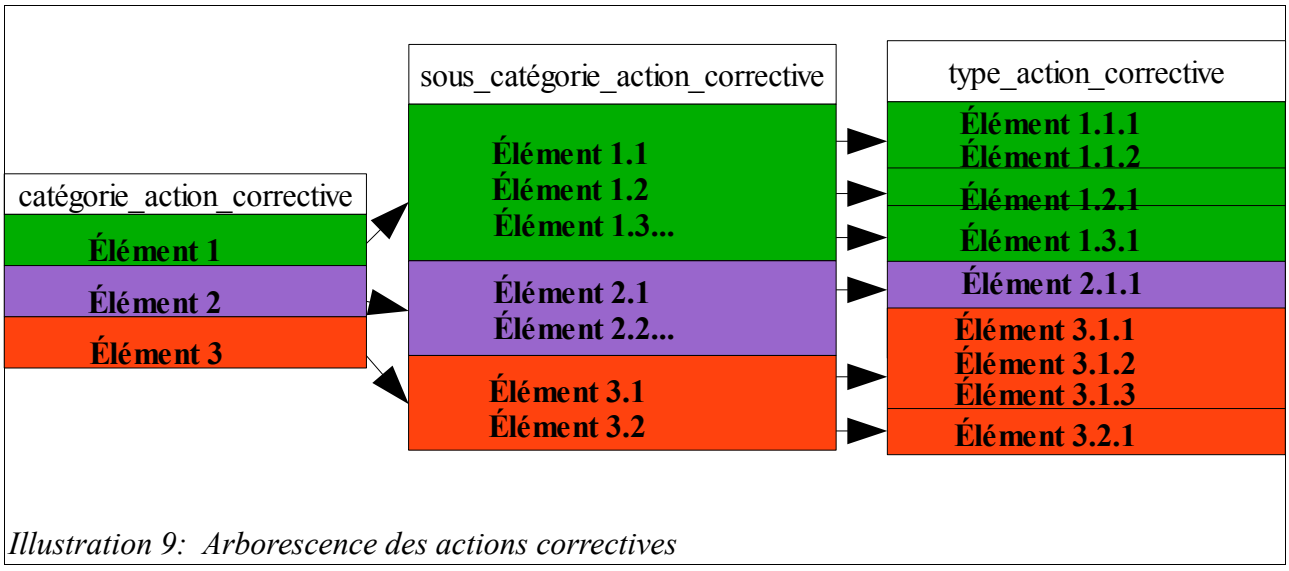

Illustration 9: Arborescence des actions correctives

Pour des raisons de fonctionnalités, l'affichage des relations *ManyToMany* proposé par Symfony via des widgets ne convenait pas à notre application. Symfony affiche les relations *ManyToMany* avec des listes déroulantes à choix multiple qui contient tout les types. Cependant nous voulions une liste déroulante à trois niveaux. Il a donc fallu créer des widgets personnalisés.

d. Les Widgets

Le framework de formulaire de symfony est livré avec un lot de widgets utiles qui fournissent les fonctionnalités de base. Cependant, pour afficher les champs d'un formulaire particulier, il est nécessaire développer des widgets personnalisés.

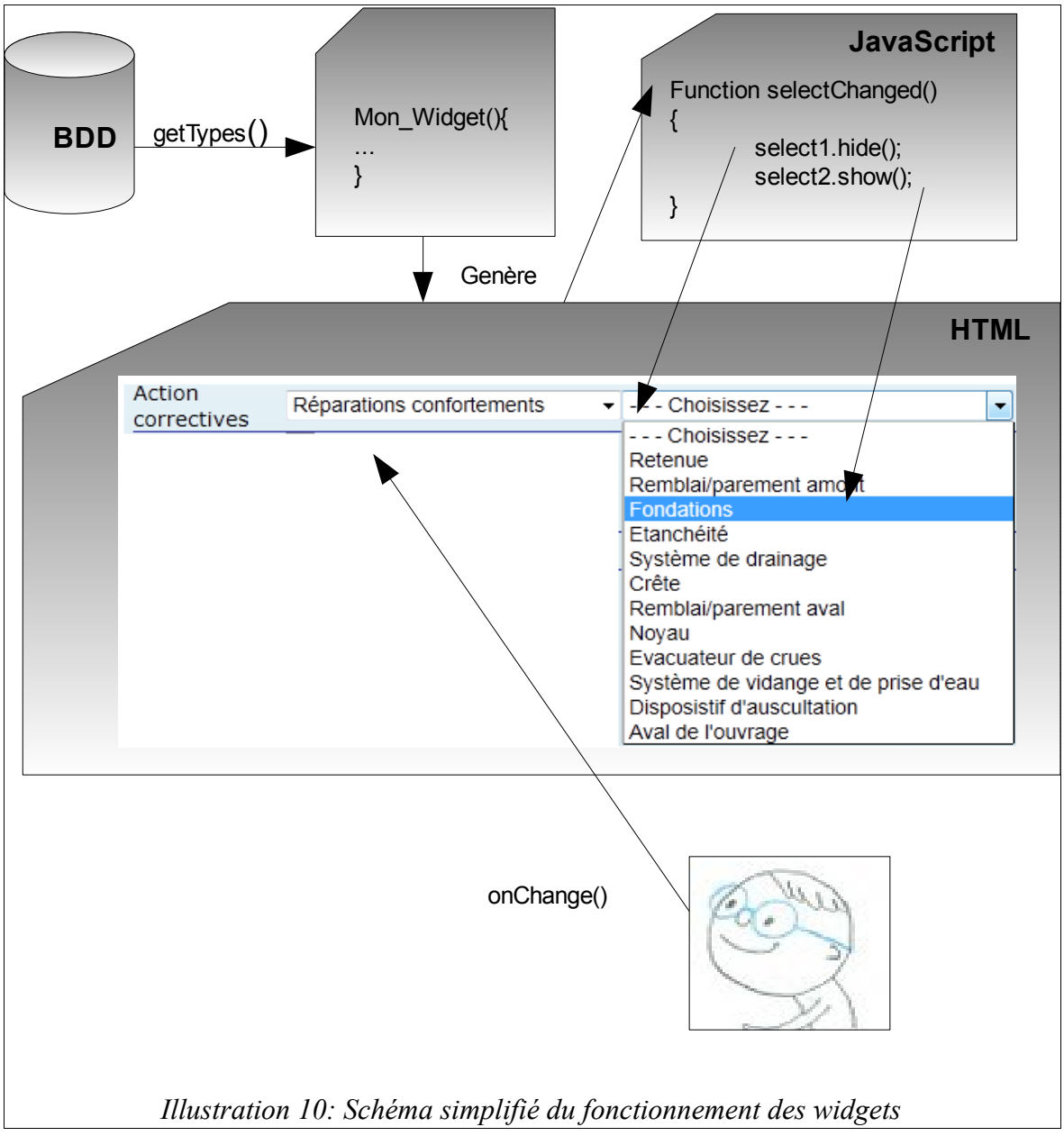

Illustration 10: Schéma simplifié du fonctionnement des widgets

Le code PHP génère le code HTML des différentes listes à afficher à partir de la base de données. Le JavaScript affiche ou cache les listes déroulantes en fonction de la sélection de l'utilisateur.

3.1.3. Notes

En plus des différentes caractéristiques physiques et des divers événements qui peuvent concerner un barrage, son étude peut être enrichie par un ensemble de documents jugés utiles par les experts, tels que des fichiers texte ou multimédia.

Chaque barrage et chaque incident peut être relié à un ensemble de photos, à des documents PDF voire même à des vidéos.

Les utilisateurs de l'application auront la possibilité d'associer des fichiers à leurs barrages à leurs incidents, mais aussi d'attacher des fichiers multimédias à la page d'accueil.

La table multimédia « *persa_media* » est en relation *ManyToMany* avec la table des barrages « *persa_barrage* » : un barrage peut avoir un ou plusieurs fichiers média et un fichier média peut être relié à un ou plusieurs barrages en même temps. Il en est de même pour les incidents.

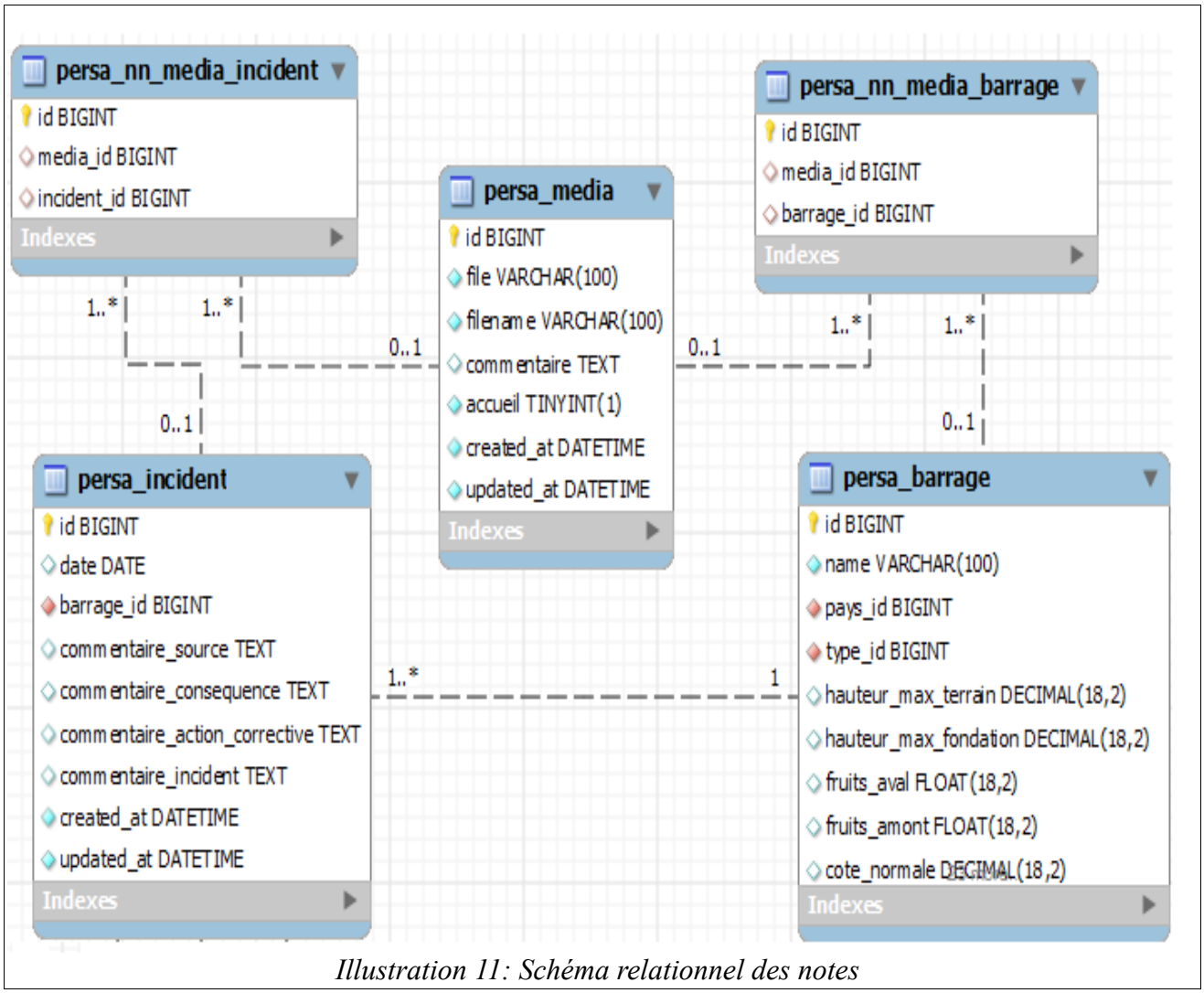

Illustration 11: Schéma relationnel des notes

4. Gestion des utilisateurs

Pour définir les utilisateurs, j'ai utilisé « sfDoctrineGuardPlugin » qui est le plugin à utiliser pour la gestion des utilisateurs de Symfony.

Il comporte des formulaires d'identification et d'inscription très basiques, mais son principal point fort est la gestion des droits associés aux utilisateurs et la possibilité de gérer des groupes.

Le modèle de données du plugin comporte la table des Utilisateurs, des Groupes et des Permissions ainsi que les tables d'associations entre les Utilisateurs et les Permissions, les Groupes et les Permissions et les Utilisateurs et les Groupes.

Pour mon application, on a 6 types de permissions pour 3 groupes d'utilisateurs

Nom	Description
Permission_admin	Permission administrateur
Permission_create	Permission de création de fiche
Permission_update	Permission de mise à jour de ses propres fiches
Permission_update_all	Permission de mise à jour des fiches de tout le monde
Permission_validate	Permission de valider ses propres fiches
Permission_validate_all	Permission de valider les fiches de tout le monde

Illustration 12: Table des permissions

Nom du groupe	Description	Permission du groupe
Group_admin	Groupe administrateur	Permission_admin, Permission_create, Permission_update, Permission_update_all, Permission_validate, Permission_validate_all
Group_expert	Groupe expert	Permission_create, Permission_update, Permission_update_all, Permission_validate
Group_prof	Groupe professeur	Permission_create, Permission_update

Illustration 13: Table des groupes

5. Estimation de la volumétrie de la base de donnée

Les experts barrages ont expliqué que l'application n'enregistra pas plus de 50 barrages. Pour tester la montée en charge de la base de données, j'ai enregistré 1000 barrages. L'enregistrement de ces dernier n'a occupé que 0,4063 Méga-Octet de plus.

On peut en conclure qu'il n'y aura aucun problème de taille dans le futur.

6. Présentation de l'application

6.1. Fonctionnement général

Voir Illustration 14 : Architecture simplifiée de l'application, page 26.

Fichier HTML

Le format HTML (Hypertext Markup Language soit langage de balisage d'hypertexte en français) contient le contenu et la structure de la page Web. C'est le fichier qui sera récupéré par le navigateur, puis lu. Après cette lecture, le navigateur charge les fichiers annexes comme les fichiers CSS, JS, les images, etc. Le format XHTML est basé sur l'XML.

Fichier CSS

Le format CSS (Cascading Style Sheets ou feuilles de style en cascade en français) sert à décrire la présentation des documents HTML et XML. C'est ce fichier qui gère le positionnement du contenu, les couleurs, polices, etc.

Fichier JS

Les fichiers JS contiennent du code Java Script qui est exécuté sur le navigateur du client. Le Java Script permet de modifier le contenu des fichier HTML, de modifier les règles CSS (ce qui permet de créer des animations), mais aussi de récupérer des données via des requêtes *AJAX*.

Fichier TPL

Les fichiers de *template* correspondent aux vues du modèle MVC, ils contiennent des morceaux de code XHTML avec des instructions spécifiques comprises par le moteur de template chargé de fusionner la vue aux modèles (données). Ses vues sont ensuite associées par le contrôleur pour obtenir le fichier HTML final de la page.

Fichier PHP

Ces fichiers contiennent du code PHP. PHP (acronyme récursif pour PHP : Hypertext Preprocessor), est un langage de scripts libre principalement utilisé pour la génération de fichiers HTML dynamiques. Le serveur web utilise l'interpréteur PHP pour générer les pages HTML via PHP qu'il envoie ensuite au client.

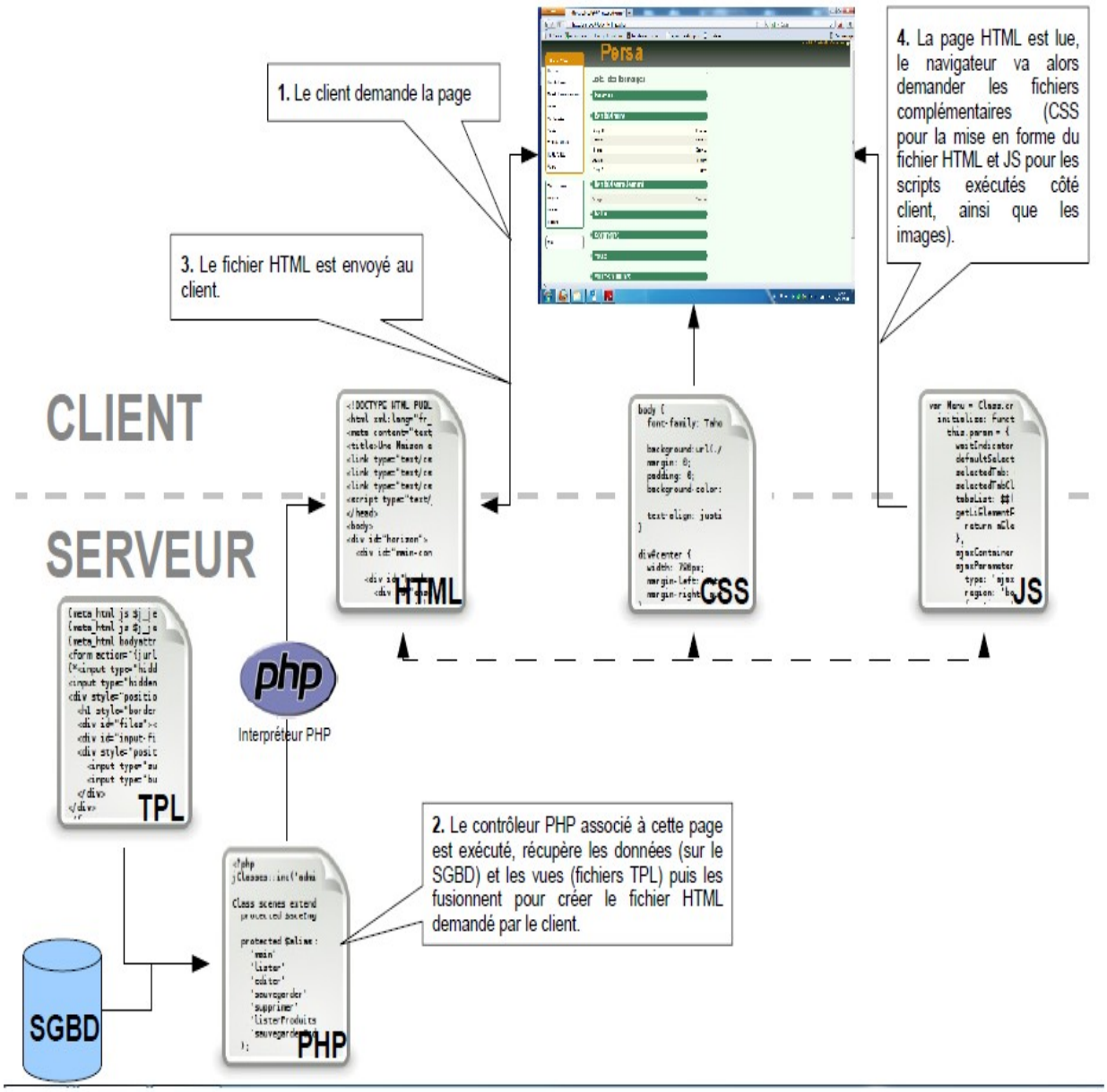

Illustration 14: Architecture simplifiée de l'application

6.2. Front-end

Ici je vais vous détailler les différentes fonctionnalités de l'interface web. Vous pouvez voir ci-dessous la page de connexion de l'application.

Illustration 15: Authentification

Cette page (Illustration 15) permet l'identification d'un utilisateur grâce à un login et un mot de passe. L'utilisateur est obligé de se connecter si il souhaite accéder aux autres pages et fonctionnalités de l'application.

Une fois l'authentification réussie, l'utilisateur est redirigé automatiquement vers la page affichant tous les barrages qui sont répertoriés dans la base de données et triés par type. Cependant, ces droits changent d'un utilisateur à un autre. En effet, certains utilisateurs ont le droit de voir ou de modifier la liste des barrages et d'autres non.

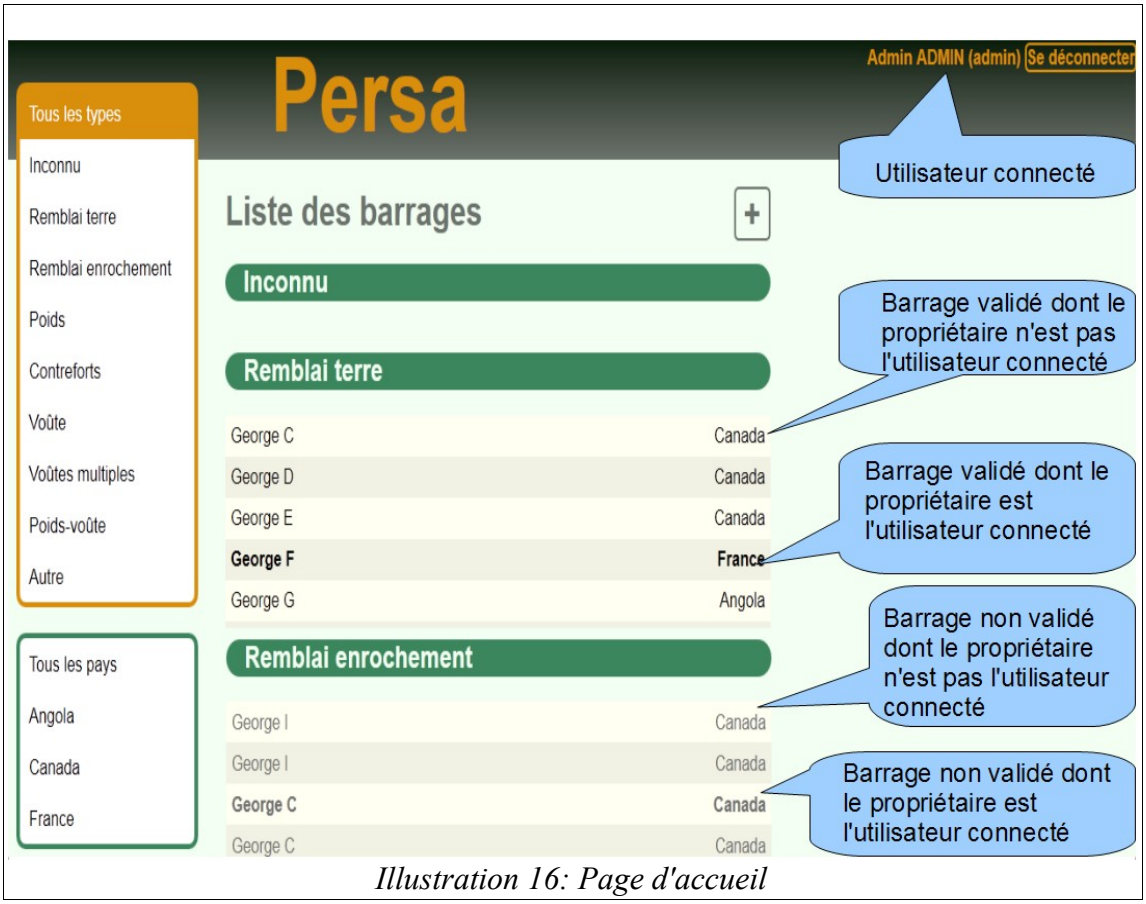

Illustration 16: Page d'accueil

Depuis la page d'accueil l'utilisateur pourra donc afficher les détails d'un barrage et selon ses droits. Il pourra modifier ou supprimer un barrage qui est enregistré dans la base de données.

En effet, il existe 4 types de barrages :

- Les Barrages (en noir et en gras) : ce sont les barrages validés par l'administrateur, il peuvent être affichés par tout le monde et dont le propriétaire est l'utilisateur connecté.

- Les Barrages (en noir) : ce sont les barrages validés par l'administrateur, il peuvent être affichés par tout le monde et dont le propriétaire n'est pas l'utilisateur connecté.

- Les Barrages (en gris et en gras) : ce sont les barrages non validés, leurs données ne sont pas publiques et leur propriétaire est l'utilisateur connecté.

- Les Barrages (en gris) : ce sont les barrages non validés, leurs données ne sont pas publiques et leur propriétaire n'est pas l'utilisateur connecté.

Lorsque le barrage est affiché, l'utilisateur peut également ajouter des nouveaux incidents, mesures, études hydrologiques, ou des médias au barrage. Pour cela il doit cliquer sur les menus « Nouvel incident », « Nouvelle mesure », « Nouvelle hydrologie », ou « Nouveau média » qui sont à gauche de la page.

Un barrage doit avoir obligatoirement un propriétaire. Ce dernier a la possibilité de valider son propre barrage pour le rendre visible à tout le monde.

Illustration 17: Page d'affichage d'un barrage

Si l'utilisateur souhaite ajouter un incident pour un barrage, il sera rédigé vers la page suivante. Pour chaque incident plusieurs sources, conséquences ou actions correctives peuvent être ajoutées.

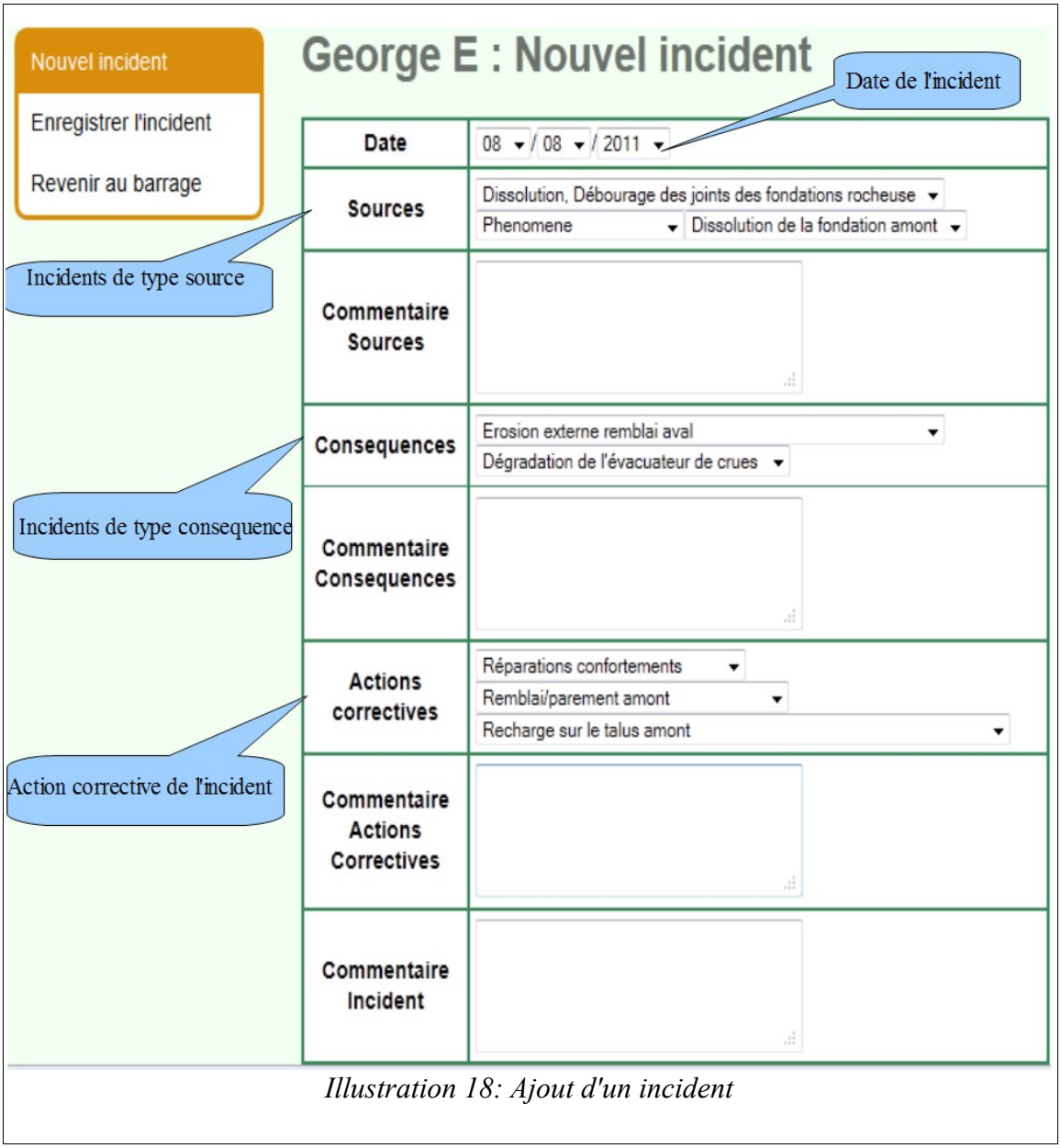

Illustration 18: Ajout d'un incident

De la même façon que les incidents, l'utilisateur de l'application peut ajouter des nouvelles mesures au barrage. Il peut choisir le type de ce dernier à l'aide d'une liste déroulante à 2 niveaux, rédiger un commentaire et mettre le nombre de mesures.

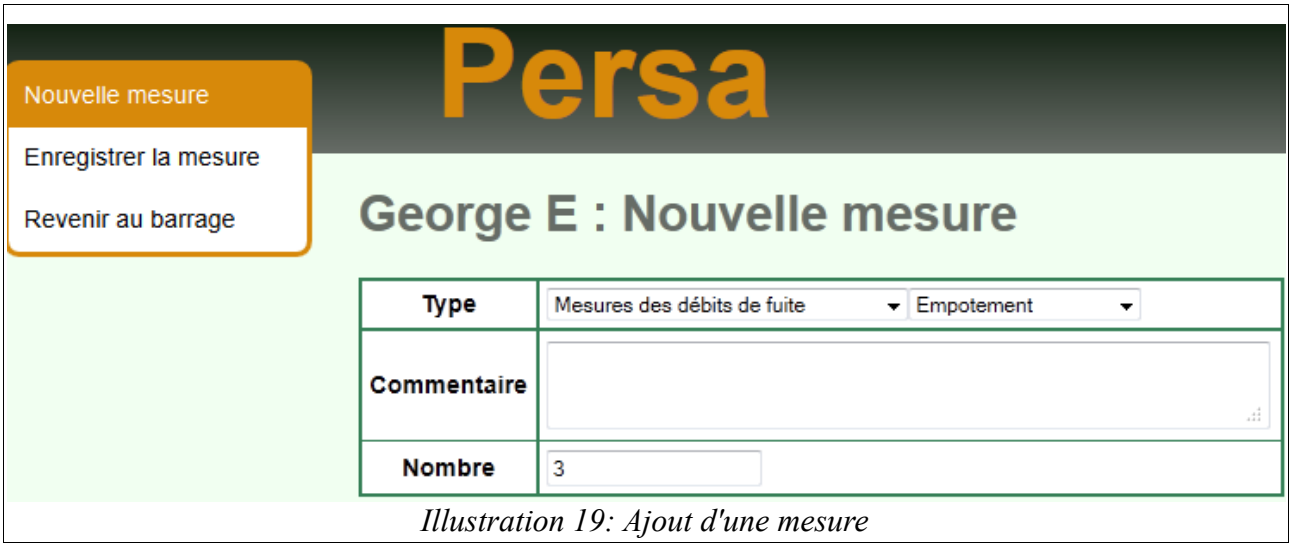

Illustration 19: Ajout d'une mesure

Il est possible d'ajouter pour chaque barrage et/ou incident des fichiers textes, des PDF, des images, des vidéos. L'utilisateur a aussi la possibilité d'attacher des documents uniquement aux barrages qu'il a le droit de modifier.

S'il s'agit d'un document général concernant tout les barrages de la base de donnée il est possible de l'attacher à la page d'accueil.

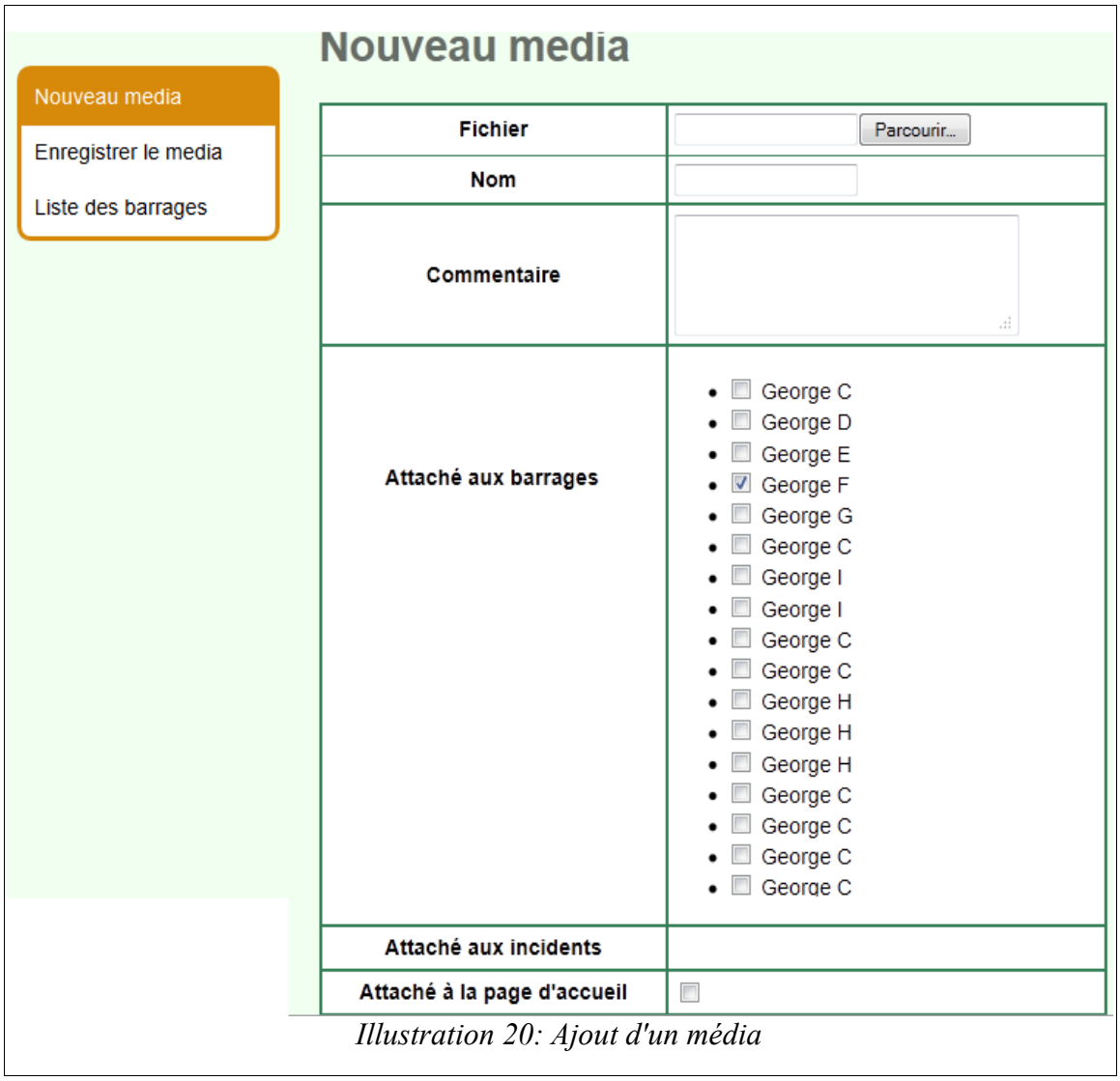

Illustration 20: Ajout d'un média

Une fois l'image enregistrée, l'utilisateur sera redirigé vers une page lui permettant de voir le fichier qu'il a téléchargé au format réduit ou réel. Il aura aussi la possibilité de l'éditer ou de le supprimer.

Illustration 21: Affichage du média

Si il y a eu des études hydrologiques sur un barrage, l'utilisateur peut les ajouter à la base de données à l'aide du menu « Nouvelle hydrologie ».

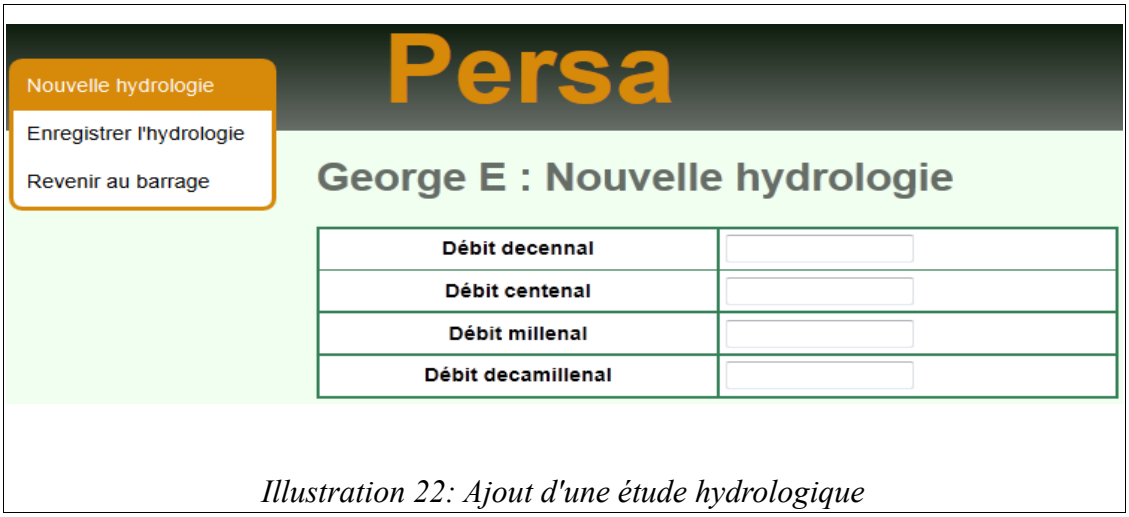

Illustration 22: Ajout d'une étude hydrologique

6.3. Back-end

Le back-end est la partie servant à gérer tout le contenu du site visible par les visiteurs sur le front-end. Par définition, l'accès au back-end doit être protégé par mot de passe puisqu'il permet de modifier le contenu du site. L'interface graphique du back-end devait être claire et pouvoir afficher de grandes quantités d'information, ce qui n'est pas le cas de l'interface du front-end qui est de taille fixe.

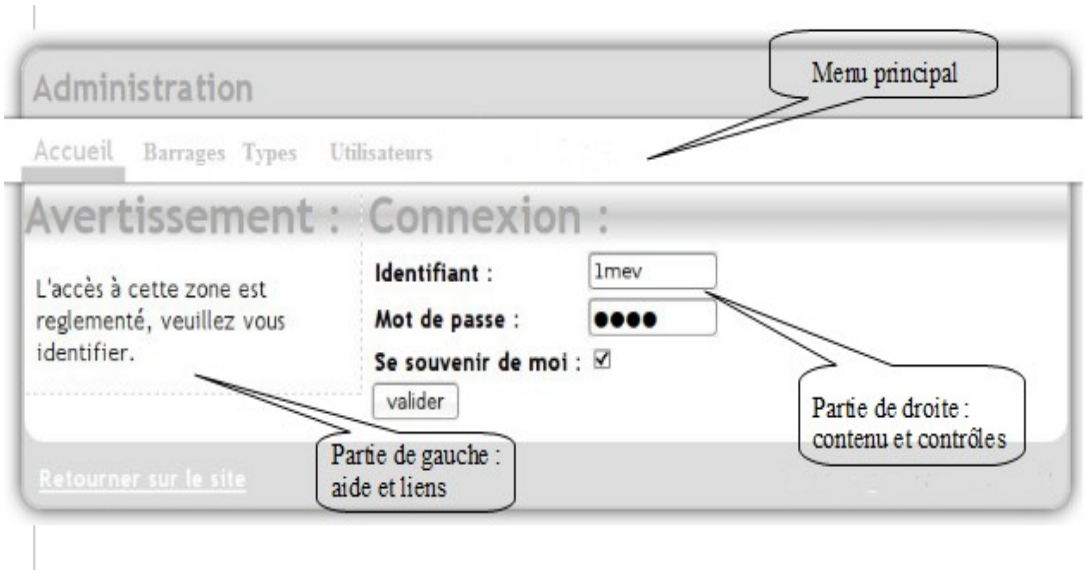

Illustration 23: Page d'identification / Aperçu de la logique de l'interface

L'interface d'administration est découpée en deux parties : la partie de gauche qui sert à afficher de l'aide et des liens vers d'autres fonctions ; et la partie de droite qui contient tout le contenu et les contrôles. Ces deux parties peuvent être chargées indépendamment ou ensemble par *AJAX* sans rechargement du navigateur.

III. RÉSULTATS

1. Résultat obtenu

Actuellement, les fonctions de saisie ont été implémentées. C'est-à-dire que l'utilisateur peut créer, éditer ou supprimer des barrages, des incidents, des actions correctives, et exécuter les sous-fonctions associées comme par exemple : ajouter des documents multimédia, ou des mesures à un barrage donné. De plus, pour toutes les parties principales, l'utilisateur peut afficher des données en fonction de différents critères (Pays, type ..).

La gestion des données multimédia est implémentée. C'est-à-dire que l'utilisateur peut ajouter des images , des vidéos ou des fichier PDF pour enrichir le barrage.

Pour la gestion des utilisateurs et de leur droits, tout est mis en place grâce au plugin « sfDoctrineGuardPlugin ».

Le système de validation fonctionne : l'administrateur peut valider un barrage saisi par un utilisateur, c'est-à-dire publier ses informations pour qu'elles soient visibles à des autres utilisateurs de l'application.

Pour le moment, l'envoie automatique d'un email à l'administrateur en cas de modification des données d'un barrage n'est pas mis en place faute de paramètres d'authentification au serveur SMTP de l'établissement pour l'application.

Coté application backend réservée aux administrateurs, les principaux formulaires de saisie et de contrôles de données ont été générés.

Au niveau des tests automatiques, les principales saisies de données et la navigation ont été testées : la gestion des filtres de recherche pour tous les modules, les suppressions groupées et en cascade des éléments...

L'application respecte les conventions du W3C à l'exception de quelques avertissements générés automatiquement par Symfony. De plus, l'application au niveau du design et de la navigation fonctionne correctement sur la plupart des navigateurs testés (Firefox, Opera, Google Chrome, Safari). Cependant, sur Iceweasel et Konqueror, le menu déroulant ne fonctionne pas correctement.

2. Amélioration future

La partie affichage, création et modification des barrages étant réalisée, il faut maintenant ajouter l'envoi automatique d'un email informatif à l'administrateur en cas de modification de données d'un barrage par un expert. Dans ce cas, l'administrateur pourra valider ou pas ces modifications.

L'autre fonctionnalité est le développement d'un glossaire contenant les termes relatifs au domaine d'étude permettant d'avoir une prise de contact avec la terminologie propre au domaine des barrages.

IV. BILAN

1. Difficultés rencontrées

La principale difficulté a été l'apprentissage de Symfony. En effet, même si le langage de base est le PHP, le framework possède une architecture et des méthodes qui lui sont propres. Ces différences de méthodes sont principalement les accès à la base de données et la création de formulaires.

L'autre problème majeur que j'ai rencontré, était au niveau du schéma relationnel. En effet, celui-ci a été modifié un grand nombre de fois afin d'ajouter les fonctionnalités demandées mais également pour améliorer la cohérence entre les tables. Cela m'a amené à de nombreux dysfonctionnements de l'application qu'il fallait trouver et corriger.

2. Apport pour l'entreprise

Le Cemagref d'Aix en Provence possède maintenant une base de connaissances sur les barrages, les incidents et leurs actions correctives. Maintenant, les experts peuvent consulter l'historique des incidents d'un barrage et en tirer de nombreux enseignements. Cette base de connaissances constitue de plus un moyen efficace pour partager les informations recueillies par chacun.

3. Bilan personnel

Ce projet a permis d'améliorer mes connaissances dans la gestion de projets et les éléments principaux à prendre en compte pour planifier et attribuer des tâches. Ce projet m'a également permis d'apprendre comment fonctionne un système de gestion d'information.

Au niveau du développement, j'ai découvert Symfony et le développement à l'aide d'un framework web. Ce projet était une occasion pour améliorer mes connaissances sur le développement des applications web en prenant en compte les différentes contraintes qui se présentent en plus d'un développement classique.

Ce projet m'a permis également de développer mon sens de l'autonomie, de la réflexion et de la résolution des problèmes et ainsi de bien attribuer les différentes phases de la réalisation au cours du temps.

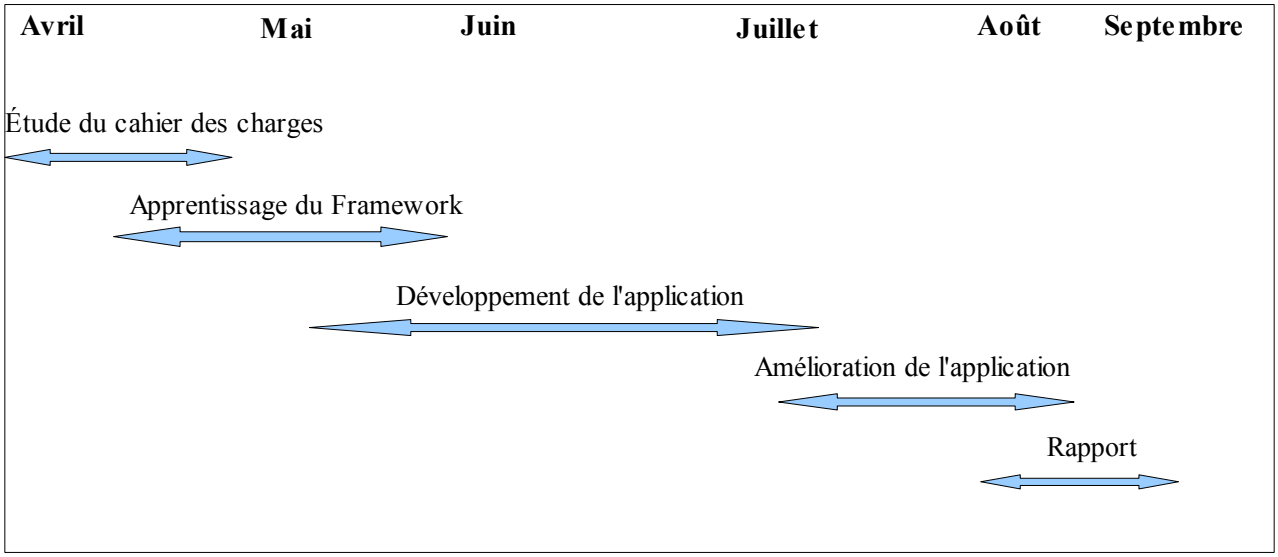

Illustration 24: Chronogramme

CONCLUSION

Ce stage a été sous plusieurs aspects riche d'enseignements.

Le stage consistait à réaliser une base de donnée interactive pour faciliter le travail d'experts. Il m'a permis d'améliorer mes connaissances en création d'application web, et notamment en ce qui concerne le respect strict des standards du Web et l'utilisation poussée de technologies comme l'UML, le SQL, le Java Script, et la POO.

À l'heure actuelle, l'application est prête à être utilisée. On peut donc affirmer que le but qui m'avait été fixé a été atteint.

Le contact avec le monde de la recherche m'a permis de progresser dans de nombreux domaines, notamment sur le thème de l'analyse de données. J'ai aussi découvert le framework Symfony.

En conclusion, mon stage m'a permis de mettre en œuvre des compétences scolaires, professionnelles et humaines pour un sujet intéressant. J'ai de plus acquis de nouvelles compétences dans le domaine du développement web.

GLOSSAIRE

AJAX : Technique utilisant des technologies comme le javascript et le XML pour charger des données dans une page web sans rechargement de la page.

CMS : Un système de gestion de contenu ou SGC ((en) Content Management Systems ou CMS) est une famille de logiciels destinés à la conception et à la mise à jour dynamique de site web ou d'application multimédia.

Éditeur WYSIWYG : What you see is what you get. Champ de saisie de texte similaire à celle d'un traitement de texte, c'est-à-dire avec des options de mises en forme (police, taille du texte, couleur, alignement, etc).

IHM : Interface Homme Machine.

MVC : Le Modèle-Vue-Contrôleur (en abrégé MVC, de l'anglais Model-View-Controller) est une architecture et une méthode de conception qui organise l'interface homme-machine (IHM) d'une application logicielle.

POO : Programmation Orientée Objet. Consiste en la définition et l'assemblage de briques logicielles appelées objet, un objet représente un concept, une idée ou une entité quelconque.

PHP : HyperText Preprocessor est un langage de programmation pour permettre la création de page HTML de manière dynamique.

Template : Un template est un modèle de présentation des données. On parle aussi de « patron » comme en couture ou de gabarit.

YAML : Langage de sérialisation de données qui reprend des concepts d'autres langages comme XML, mais en organisant les données sous forme de listes.

Relation OneToMany : Une relation se produit lorsque chaque enregistrement dans une table A peut avoir plusieurs enregistrements liés à une table B, mais chaque enregistrement dans la table B peut avoir qu'un seul enregistrement correspondant dans la table A.

Relation ManyToMany : Une ligne dans la table A peut avoir plusieurs lignes correspondantes dans la table B, et vice versa.

Fixture : Fichier au format YAML servant à remplir la base de données.

Ruby on Rails : Framework web libre écrit en Ruby (http://www.rubyonrails.org/).

RÉFÉRENCES BIBLIOGRAPHIQUE

- http://www.symfony-project.org/ : Site officiel du Framework de Symfony, on y retrouve des tutoriels, documentation, plug-in...

- http://www.developpez.net/forums/f663/php/bibliotheques-frameworks/symfony/ : Communauté de développeurs en Symfony, très utile lorsque qu'on est bloqué sur quelque chose.

- http://www.php.net/manual/fr/ : Documentation sur le langage PHP.