

Projet Cimetière

Rapport de stage :

Philippe Basuyau

Octobre 2016

« Il n'y a pas de grande tâche
difficile qui ne puisse être
décomposée en petites tâches
faciles »

Shantida (grand maître
Bouddhiste du VII^e siècle)

Remerciement à

A mes 2 Professeurs **Pascal Buguet** et **Sébastien Maloron** qui m'ont été mes mentors sur ce projet. Leurs précieux conseils mon guidé tout au long de la formation et du stage.

Merci aussi à la Mairie d'Estrées Saint Denis, tout particulièrement à :

Mr Pouplin – Mr le Maire

Mr Mussart Jean-noël – Mr le secrétaire général

Elizabeth Morrice – adjointe

Et toute l'équipe de la mairie **Nathalie Tourman, Stéphanie**

Merci à **Richard Chaigneau** de regard neuf pour le bilan de compétence et tous ces conseils.

Un très grand merci à mon épouse **Nathalie**, qui m'a beaucoup porté dans ce projet et toujours soutenu dans mes actions.

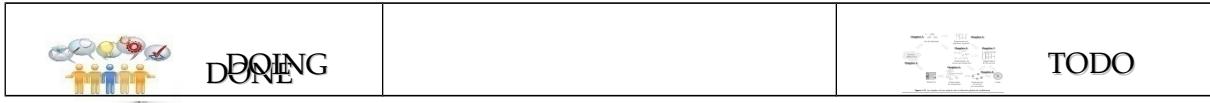
Merci à toutes ces belles rencontres que j'ai pu faire depuis le début de ma reconversion professionnelle.

Table des matières

<u>Cahier des charges fonctionnel</u>	5
I. Présentation générale du problème.....	6
II. Présentation générale du Logiciel.....	6
III. Documentation et terminologie.....	6
IV. Caractère confidentiel.....	8
V. Contexte et motivation de l'action.....	8
VI. Dictionnaire des données.....	9
o L'administrateur.....	9
o L'utilisateur connecté.....	9
o L'utilisateur non connecté.....	9
VII. Description Fonctionnelle.....	12
VIII.....	Contrainte Logiciel
13	
Le Backlog.....	14
o Pour le côté administrateur :.....	14
o Pour le côté agent Territorial :.....	14
Langages et outils utilisés.....	15
I. Le langage.....	15
II. Les logiciels.....	16
Les Diagrammes UML.....	17
I. Le diagramme de cas d'utilisation.....	17
II. Diagrammes d'Activité.....	19
o Diagramme d'activité du secrétaire général.....	19
o Diagramme d'activité d'inscription.....	20
o Diagramme d'activité de l'agent Territorial.....	21
III. Diagramme de navigation.....	22
IV. Diagramme de Séquence.....	23
o Inscription d'un agent territorial.....	24
V. Diagramme de classe.....	25
VI. Modèle physique de données.....	26
VII. Création de la base de données.....	27
VIII.....	Reverse Engineer Database
29	
IX. Diagramme Réseau de la mairie.....	30
X. Diagramme de déploiement.....	31
Les Maquettes.....	32

<u>I.</u>	<u>Vue Administrateur - Secrétaire général.....</u>	33
o	<u>Inscription au droit d'utilisateur.....</u>	33
o	<u>Modification de la fiche utilisateur.....</u>	33
o	<u>Suppression du droit de l'agent territorial.....</u>	34
o	<u>Gestion de la tarification.....</u>	34
o	<u>Gestion topographique du Cimetière.....</u>	35
<u>II.</u>	<u>Vue Utilisateur - Agent Territorial.....</u>	36
o	<u>Création du compte concessionnaire.....</u>	36
o	<u>Modification du compte Concessionnaire.....</u>	36
o	<u>Ayant droit sur la concession.....</u>	37
o	<u>Renouvellement de la concession.....</u>	37
o	<u>Consultation de la concession.....</u>	38
<u>La Partie Développement.....</u>		39
<u>I.</u>	<u>La structure MVC.....</u>	39
<u>II.</u>	<u>Le schéma de fonctionnement de mon MVC.....</u>	40
<u>Le code.....</u>		41
<u>I.</u>	<u>L'inscription.....</u>	41
o	<u>La Sécurité.....</u>	41
o	<u>Le contrôle des champs du formulaire.....</u>	42
<u>II.</u>	<u>Le transfert des données.....</u>	43
o	<u>Le code du DAO.....</u>	43
o	<u>Le code du DTO.....</u>	46
o	<u>Le code du IDAO.....</u>	47
o	<u>Le code des différentes bibliothèques concernées.....</u>	47
o	<u>Le code de configuration des données de connexion.....</u>	49
<u>III.</u>	<u>Le principe de navigation de la vue en inscription.....</u>	50
o	<u>Le schéma de la navigation et de son traitement.....</u>	50
o	<u>Le code de la navigation et de son traitement.....</u>	51
<u>Présentation du Logiciel « Cimetière » à la Mairie.....</u>		53
<u>I.</u>	<u>Présentation avec un diaporama de façon rétrospective sur la mise au pont du logiciel intranet.....</u>	53
<u>II.</u>	<u>Présentation des fonctionnalités du logiciel.....</u>	53
<u>III.</u>	<u>Démonstrations sur rétroprojecteur de son fonctionnement.....</u>	53
<u>Doléance.....</u>		54
<u>I.</u>	<u>Prise en compte des ajustements.....</u>	54
b.	<u>Prise en compte des futures améliorations.....</u>	54
<u>Formation des Agents territoriaux.....</u>		55
<u>I.</u>	<u>Formation du secrétaire générale.....</u>	55
<u>II.</u>	<u>Formation des agents territoriaux.....</u>	55

<u>La Webographie et la bibliographie</u>	56
I. <u>La Webographie</u>	56
II. <u>La Bibliographie</u>	56
<u>L'annexe</u>	57
I. <u>Le script de la création de base</u>	57
II. <u>Les Librairies et Les Modèles</u>	62
III. <u>Les DAO - DTO - IDAO</u>	66
IV. <u>L'inscription côté View et côté Controller</u>	Erreur ! Signet non défini.



Cahier des charges fonctionnel

« Un **cahier des charges fonctionnel** (CDCF) est un document qui doit être respecté lors de la réalisation d'un projet ». Le document est formalisé dans le document NF X 50-151 par L'**AFNOR** (Agence Française de NORmalisation). L'origine du cahier des charges remonte à l'Ancien Régime, avant la Révolution française. Il permet de fixer les règles pour qu'un projet soit mené à bien, sans surprise, et de déterminer les devoirs et les droits de chaque participant.

- o Les besoins essentiels que « le logiciel » doit satisfaire
- o Les conditions d'utilisation prévues
- o Les contraintes imposées par le demandeur

La norme NF X 50-151 s'applique à tous projets de développements industriels, que l'on soit dans le domaine de l'aéronautique, de l'automobile, Par voie de conséquence, elle peut s'appliquer également à l'informatique et aux Systèmes d'Informations.

Voici les avantages de rédiger un cahier des charges logiciel :

- o Structurer le projet
- o Clarifier les idées
- o Identifier les besoins
- o Assurer la bonne compréhension de votre projet par les prestataires

De la mise en place de ce cahier des charges fonctionnel en dérivera le DCU et son Backlog.

I. Présentation générale du problème

A l'heure actuelle, La Mairie d'Estrées Saint Denis souhaiterait informatiser le cimetière. Et que le système puisse alerter sur les renouvellements des concessions échues.

II. Présentation générale du Logiciel

Le logiciel servira à la gestion du cimetière, des concessionnaires et de ses ayants droits. D'indiquer le choix de sépulture (concession, columbarium ou jardin du souvenir), de l'emplacement choisi, de la durée.

Il devra faciliter en premier lieu le renouvellement des différentes sépultures qui arrivent à échéance.

Ce dispositif fera gagné du temps à l'agent territorial qui s'en occupe à l'heure actuelle. Le gain de temps estimer par Monsieur le Maire serait l'équivalent d'un mi-temps, dès la mise en service du logiciel.

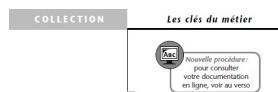
III. Documentation et terminologie

Création d'un dictionnaire métier pour l'emploi des mêmes terminologies et pour une meilleure compréhension globale (Voir Annexe).

Pour commencer lecture du règlement du cimetière, ensuite lecture du livre :

Gérer un cimetière

Guide juridique et pratique de la gestion des cimetières



Gérer un cimetière
Guide juridique et pratique
de la gestion des cimetières

Sous la direction de Philippe DUPUIS
Universitaire et consultant

Éditions L'Atelier
Gérage Juridique et Pratique de la Gestion des Cimetières
Copyright terminé à Paris le 10/10/2012
Avec 120 pages

Le dictionnaire des terminologies

Kalamazoo	Classeur réunissant tous les registres des décès sur la commune.
Concession	La concession funéraire c'est l'emplacement que l'on peut acheter pour une certaine durée dans un cimetière, sans que l'on puisse devenir propriétaire du terrain.
Columbarium	C'est l'endroit où l'on dépose les urnes cinéraires contenant les cendres des morts dans des niches.
Urne	L'urne funéraire aussi appelée « urne cinéraire » est un vase fermé en pierre, en bronze, en marbre, en albâtre, en céramique, en verre ou même en matériaux biodégradables dans lequel les proches d'un défunt conservent ses cendres après sa crémation.
Caveau	C'est une pièce construite en sous-sol des cimetières, on y entre pose les cercueils et des fois les urnes dans le vide sanitaire.
Reprise concession	Lorsqu'une tombe n'est plus entretenue, ni fleuris. La mairie avec l'aide d'une personne assermenté pratique la procédure de reprise de concession. Le processus dure en moyenne 3 ans après le signalement de la mise en place de la procédure, elle peut être stoppé à tout moment pour 2 raisons, soit par un fleurissement ou par un nettoyage de celle-ci. A la fin dès trois années la mairie récupère la sépulture et pratique un « nettoyage », le coût est assez onéreux de 800 à 1600€ environ.
Demande	C'est lorsqu'une personne demande à pouvoir bénéficier d'une concession.
Réduction corps	A partir de 50 ans, le corps se situant dans un cercueil peut en bénéficier, cela se pratique en présence d'un membre de la famille et d'une personne assermenté. Il ouvre le cercueil, récupère les cendres du défunt et les mette en urne funéraire de façon à faire de la place dans la concession.
Concession Restreinte	Préciser les personnes susceptibles d'accéder après leur décès à cette espace. Ces personnes sont appelées des Ayant droits .
Jardin du souvenir	C'est l'endroit où une personne incinérer peut si elle le souhaite faire disperser librement ces cendres dans un endroit réservé dans le cimetière.
Inhumation	Rite funéraire pratiqué dans la majorité des cultures, consistant à l'enfouissement d'un cadavre dans le sol ou dans un caveau. Il est plus communément appelé l'enterrement.
Inhumer	Mettre en terre avec les cérémonies. Aucune inhumation ne peut avoir lieu sans un constat de décès par un médecin ou un officier civil et en principe moins de 24 heures après le décès.

Octroi	Administration chargée de percevoir cette taxe.
Lieu de Dispersion	L'endroit l'on disperse les cendres après crémation
Crémation	C'est l'action de la destruction d'un cadavre par le feu qui se fait par le biais d'un incinérateur.
Exhumation	C'est l'action d'exhumer, soit de retirer un cadavre de terre, d'un caveau, ou d'un tombeau. De sortir de terre ce qui a été enfoui.
Type tombe	Individuelle, collective ou familial
Sépulture	C'est le rituel funéraire qui accompagne l'inhumation.
Espace cinéraire	C'est l'endroit, le lieu où les columbariums sont entreposés.
Famille	Epoux(se), enfants, petit enfants
Collective	Famille éloignée : Tante, Oncle, cousin...

IV. Caractère confidentiel

Les données fournies par la Mairie sur les défunts sont strictement privées.

V. Contexte et motivation de l'action

Gain de temps et d'énergie pour l'agent territorial et gain financier pour la Mairie d'Estrées Saint Denis. Par la suite le choix de l'emplacement directe sur un plan en 2D ou 3D serait un plus.

VI. Dictionnaire des données

o L'administrateur

- Monsieur Jean-Noël MUSSART qui a pour rôle de secrétaire général aura les droits d'administrateur. Il aura le pouvoir de :
- o Création de droit sur le logiciel pour les différents agents territoriaux
 - o Suppression des droits pour ces même personnes
 - o Un accès back office pour les changements :
 - ⊕ Des différents tarifs sur les durées et des autres frais éventuellement

Changement des durées

Création ou modification des sections

⊕ Création ou modification d'allées

⊕ Création ou modification d'emplacement

- o Et les mêmes accès que les agents Territoriaux

o L'utilisateur connecté

- Madame Nathalie TOURMAN qui est un agent territorial aura le rôle de l'utilisateur logiciel, elle aura le pouvoir de :
- o D'inscrire un demandeur de concession, qui devient lui-même concessionnaire
 - o Générer un numéro de dossier
 - o De gérer Le concessionnaire :
 - ⊕ Renseigner le type de sépulture
 - ⊕ Le choix de l'emplacement
 - ⊕ La durée
 - ⊕ La liste des ayants droits
 - ⊕ Le renouvellement
 - o La reprise de concession suite à l'abandon de celle-ci
 - o Gérer le renouvellement des différents types de sépulture

o L'utilisateur non connecté

Cette personne ne pourra pas dépasser la page d'accueil, tant qu'elle ne sera pas authentifiée.

Toutes les informations à traiter par le système (elles sont listées à partir des maquettes).

Code utilisé	Désignation	Type	Longueur	Remarque
idCategorieSepulture	Identifiant cat. sépul.	int	11	PK
categorieSepulture	Catégorie sépulture	char	35	Individuel-familial-collective
idCommune	Identifiant commune	int	11	PK
inseeCommune	N° insee	Char	5	
cpCommune	Code Postal commune	Char	5	
nomCommune	Nom commune	Varchar	50	
IdConcession	Identifiant concession	Int	11	PK
codeConcession	Code concession	Int	5	
dateConcession	Date concession	Date		
dureeConcession	Durée concession	Int	3	Choix de la durée
tarifConcession	Tarif concession	Int	5	Tarif en fonction type et durée
dateFinConcession	Date fin concession	Date		Calculer
idEtatSepulture	Identifi. état sépulture	Int	11	PK
etatSepulture	Etat sépulture	Varchar	25	Libre-Louée-procédures reprise-nettoyage
idJardinDuSouvenir	Identifi. jardin souvenir	Int	11	PK
idPersonne	Identifiant Personne	Int	11	PK
civilitePersonne	Civile	Char	5	Mr-Mme-Mlle
nomPersonne	Nom personne	Varchar	50	
PrenomPersonne	Prénom personne	Varchar	50	
vivante	Vivante	Tinyint	1	boolean
telephonePersonne	Téléphone personne	Int	10	
mobilePersonne	Mobile personne	Int	10	
emailPersonne	Email Personne	Varchar	254	
dateDecesPersonne	Date décès	Datetime		
dateDeNaissance	Date naissance	Datetime		
agePersonne	Age personne	Int	3	Caluler
idSepulture	Identifier Sépulture	Int	11	PK
sectionSepulture	Section sépulture	Char	2	
alleeSepulture	Allée sépulture	Char	2	
numeroDansAllee	Numéro Sépulture	Int	3	
code	Code sépulture	Char	5	
nombreSepulture	Nbre sépulture	Int	5	
nbrDeCorps	Nbr place occupé	Int	1	
placesLibres	Nbre place dispo	Int	1	A l'heure actuelle sur ESD, 3P
reductionCorps	Réduction de corps	Tinyint	1	Seulement a partir de 50 ans
sectionModule	N° section module	Char	2	
rangCase	N° rang case	Int	2	
numeroCase	N° de la case	Int	2	
idTypeDePersonne	Identif. Type personne	int	11	
typePersonne	Type de personne	Varchar	50	Ayant droit

<code>idTypeSepulture</code>	Identif. type sépulture	Int	11	
<code>typeSepulture</code>	Type de sépulture	Varchar	50	Concession-Columbarium
<code>idUser</code>	Identifiant user	Int	11	
<code>Identifiant</code>	Identifiant	Varchar	25	Identifiant d'authentification
<code>userName</code>	Nom du user	Varchar	254	
<code>password</code>	Mot de passe du user	Varchar	254	MDP d'authentification
<code>accessLevel</code>	Niveau d'accès logiciel	Int	3	Donne un accès différent

VII. Description Fonctionnelle

Le logiciel devra permettre l'inscription d'une demande de concession. La fiche de renseignement du concessionnaire sera remplie avec les informations suivantes :

- o Nom Prénom
- o Date de naissance
- o Adresse, code postal, ville, département
- o Numéro de téléphone et numéro de portable
- o Email
- o Non Prénom du conjoint(e), s'il existe
(Téléphone portable, email)
- o Renseignement supplémentaire sur la descendance, si elle existe et si cela est possible

Le logiciel permettra aussi de saisir les choix du concessionnaire sur sa sépulture :

- o Le type de sépulture :
 -  Concession
 -  Columbarium
 -  Jardin du souvenir
- o Le choix de l'emplacement
- o La durée qui est variable selon le choix précédent :
 -  5 ans
 -  10 ans
 -  15 ans
 -  30 ans
 -  50 ans
 -  Perpétuelle, plus Disponible mais existante
- o De définir une liste d'ayants droit si le concessionnaire le souhaite

Le logiciel devra au moment du décès du concessionnaire ou d'un des ayants droits, enregistrer les informations suivantes :

- o La date de décès
- o Le lieu de décès
- o Gestion des emplacements et réduction de corps
- o Date de sortie du caveau
- o Date d'inhumation dans le caveau familial
- o N° concession (Section, allée, numéro)
- o Nom commune en cas de transport de corps
- o Famille (ou lien de parenté), Collective (avec cousin, neveu, tante) ou Individuel
- o Gestion du jardin du souvenir, suivi des dispersions des cendres, dans le cadre de la généalogie
- o Renouvellement des échéances, par une fonction de recherche automatique

VIII. Contrainte Logiciel

Le plus important, pour Monsieur le Maire, c'est le renouvellement de la concession, il faudrait que le logiciel puisse contrôler les durées de chaque type de sépulture et alerter l'agent territorial qui pourra émettre un courrier à la famille pour lui indiquer le terme échu de la concession.

Le Backlog

C'est la liste des fonctionnalités attendues sur le logiciel « E-cimetière », on parle aussi de Users Story (tâches).

o Pour le côté administrateur :

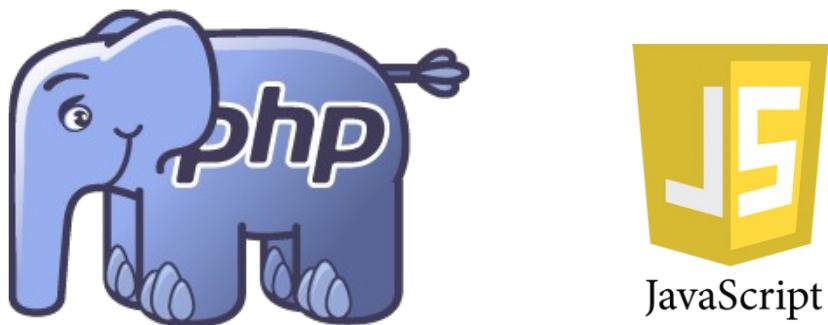
- ❖ Inscription de droit d'agent territorial
- ❖ Gestion des fiches d'agent territoriaux
- ❖ Suppression de droit d'agent territorial
- ❖ Accès Identique que l'agent territorial

o Pour le côté agent Territorial :

- ❖ Inscription d'un concessionnaire (création d'une fiche personnelle)
- ❖ Choix du type de sépulture (Caveau, Columbarium, Jardin du souvenir)
- ❖ Choix du type de concession (Familial, Collective, Individuel)
- ❖ Liste ayant(s) droit(s)
- ❖ Choix emplacement
- ❖ Choix durées
- ❖ Modification de compte du concessionnaire
- ❖ Renouvellement
- ❖ Gestion des abandons
- ❖ Procédure de reprise de la concession

Langages et outils utilisés

I. Le langage



II. Les logiciels



PowerAMC



DONE :
Besoins

DOING :
DCU

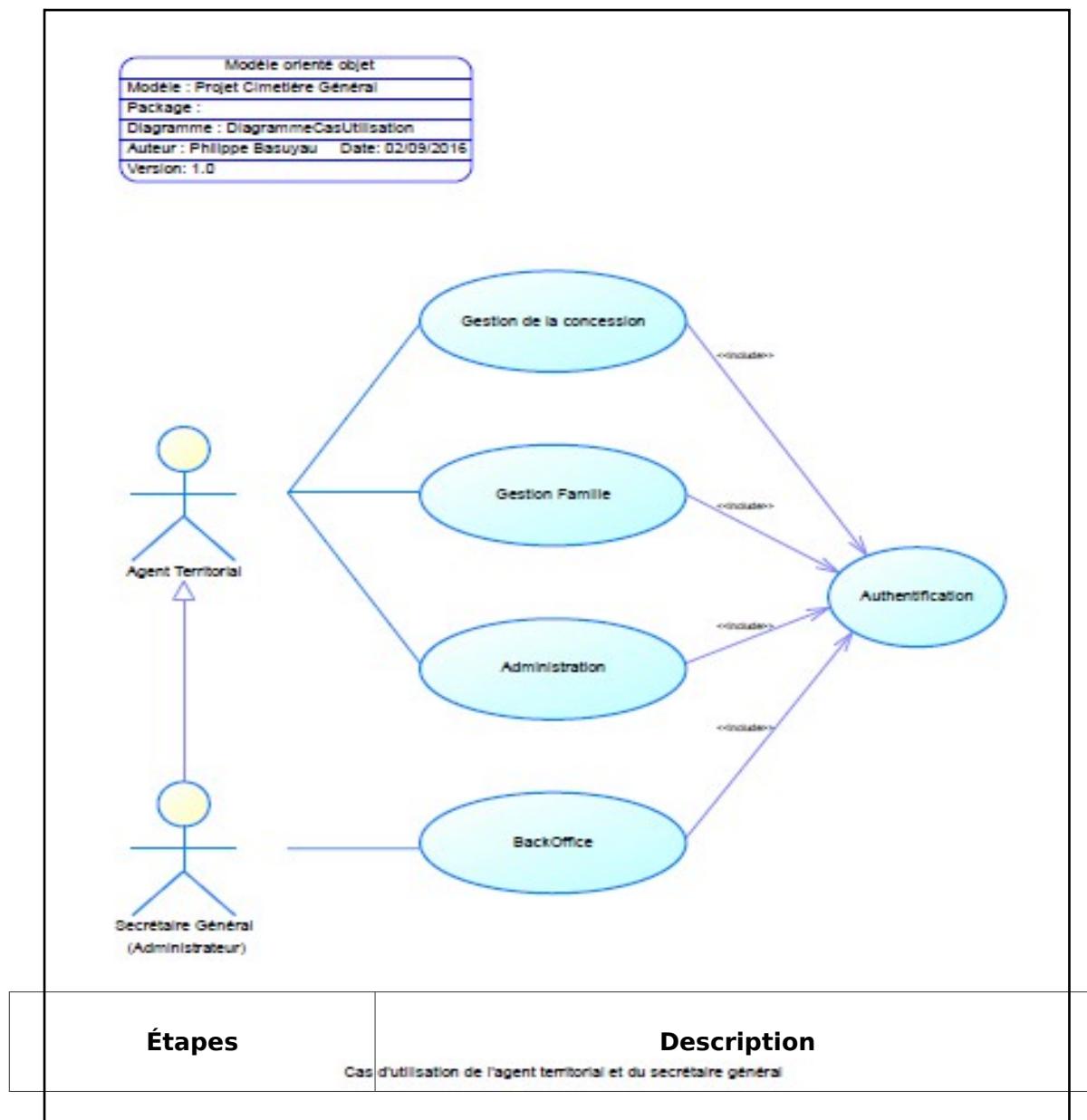
TODO :
DCU

Les Diagrammes UML

Pour ce faire j'ai eu recourt à 2 AGL (Atelier de Génie Logiciel), J'ai utilisé « PowerAMC » pour les diagrammes de cas d'utilisation, d'activité, de navigation, de classes et « StarUML » pour les diagrammes de séquence.

I. Le diagramme de cas d'utilisation

Le diagramme de cas d'utilisation (DCU) est un ensemble de cas d'utilisation (CU). Il formalise graphiquement les besoins des futurs utilisateurs.



Identification du CU	<p>Titre : Projet Cimetière</p> <p>Résumé :</p> <ul style="list-style-type: none"> • Acteur : le secrétaire général de mairie (rôle de l'administrateur) • L'agent territorial de mairie (rôle de l'utilisateur) <p>Date de création : 23/08/2016</p> <p>Date de dernière modification : 31/08/2016</p> <p>Version : 1.0</p> <p>Auteur : Philippe Basuyau</p>
Préconditions	Avoir les droits et être Connecté
Scenario nominal	Création d'un compte concessionnaire
Scenarii d'erreurs du cas nominal	Agent territorial Non Logé - et contrôle si la Personne n'est pas déjà concessionnaire
Scenarii alternatifs	<p>Modification Compte Concessionnaire :</p> <ul style="list-style-type: none"> • Changement adresse • Changement du ou de la conjoint(e) • Changement de la liste des ayants droits • Renouvellement de la concession • Abandon de la concession • Reprise de la concession par la maire
Scenarii d'erreurs des cas alternatifs	Contrôle des modifications et de leurs possibilités de changement
Post-conditions	Avoir les droits
Exigences non fonctionnelles	Rappel De diverses informations sur les concessions
Besoins d'IHM	Les différentes vues en conséquence : Inscription, modification...

Les scenarii sont détaillés avec les DAC (diagrammes d'activité).

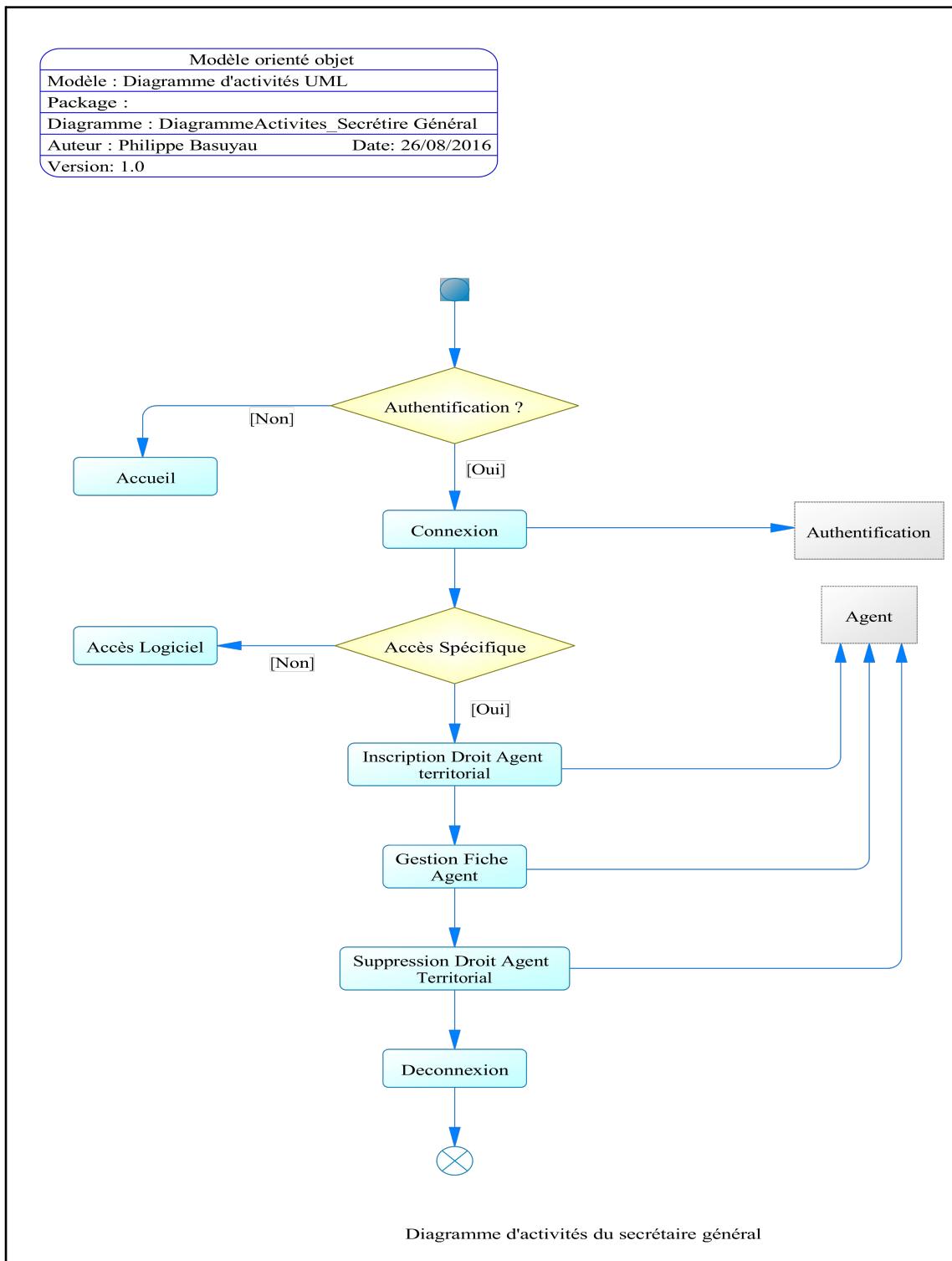
DONE :
DCU

DONE :
DAC

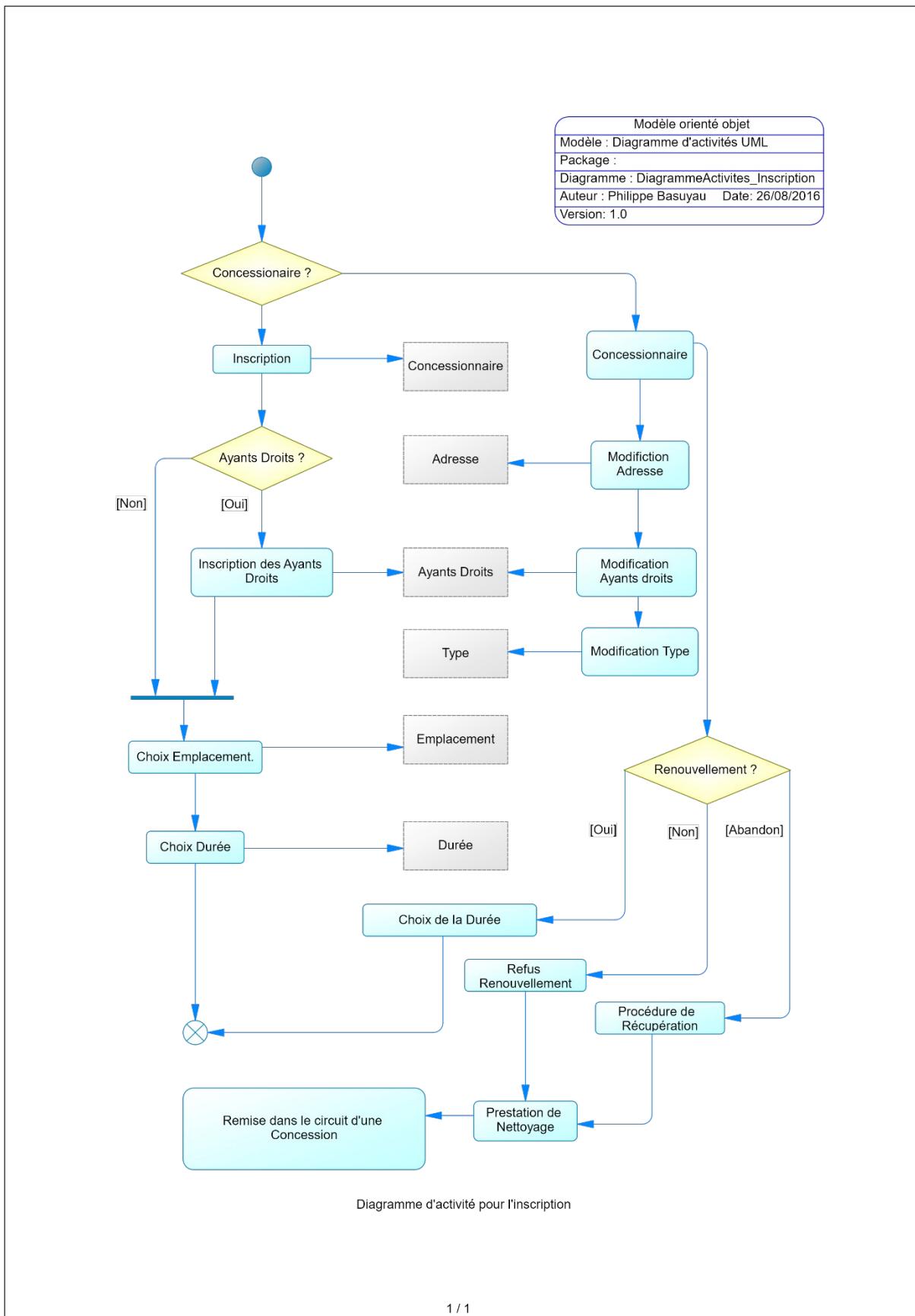
TODO :
DNAV

I. Diagrammes d'Activité

o Diagramme d'activité du secrétaire général

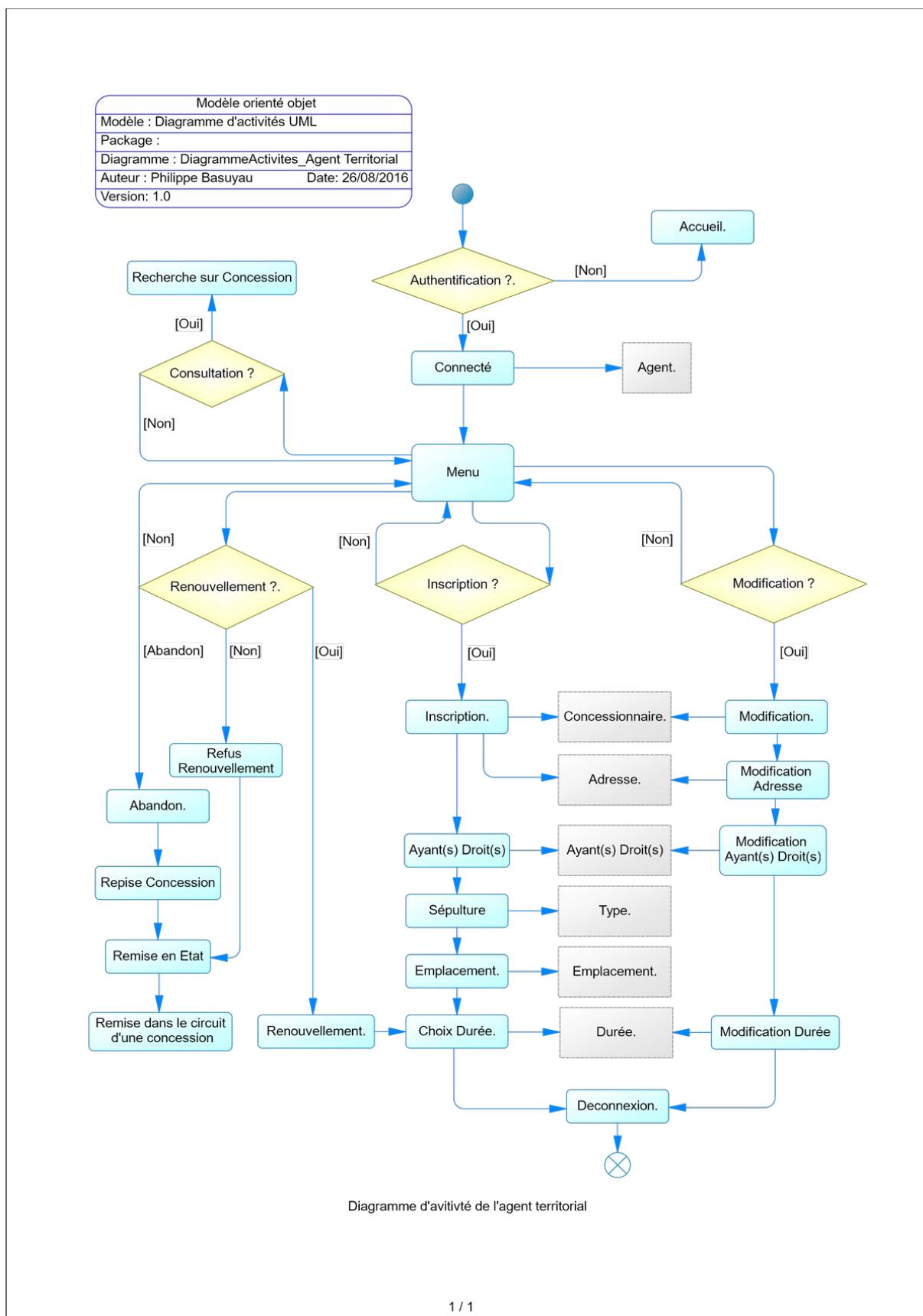


0 Diagramme d'activité d'inscription



o Diagramme d'activité de l'agent Territorial

II.



DONE :
DAC

DOING :
DNAV

TODO :
DSEQ

I. Diagramme de navigation

DONE :
DNAV

DOING :
DSEQ

TODO :
DCL

II. Diagramme de Séquence

III.

Le choix de la programmation de développer en multicouche, pour faciliter la maintenance de celui-ci en premier lieu et en outre pour des raisons de sécurités.

J'utilise le Modèle E-C-B d'Ivar Jacobson qui nous propose 3 stéréotypes pour Entity, Control et boundary (En français B-C-E).

Boundary : Classe de frontière, les objets de ces classes sont visibles et c'est le moyen de communication entre le système et l'extérieur (Interface Homme Machine).

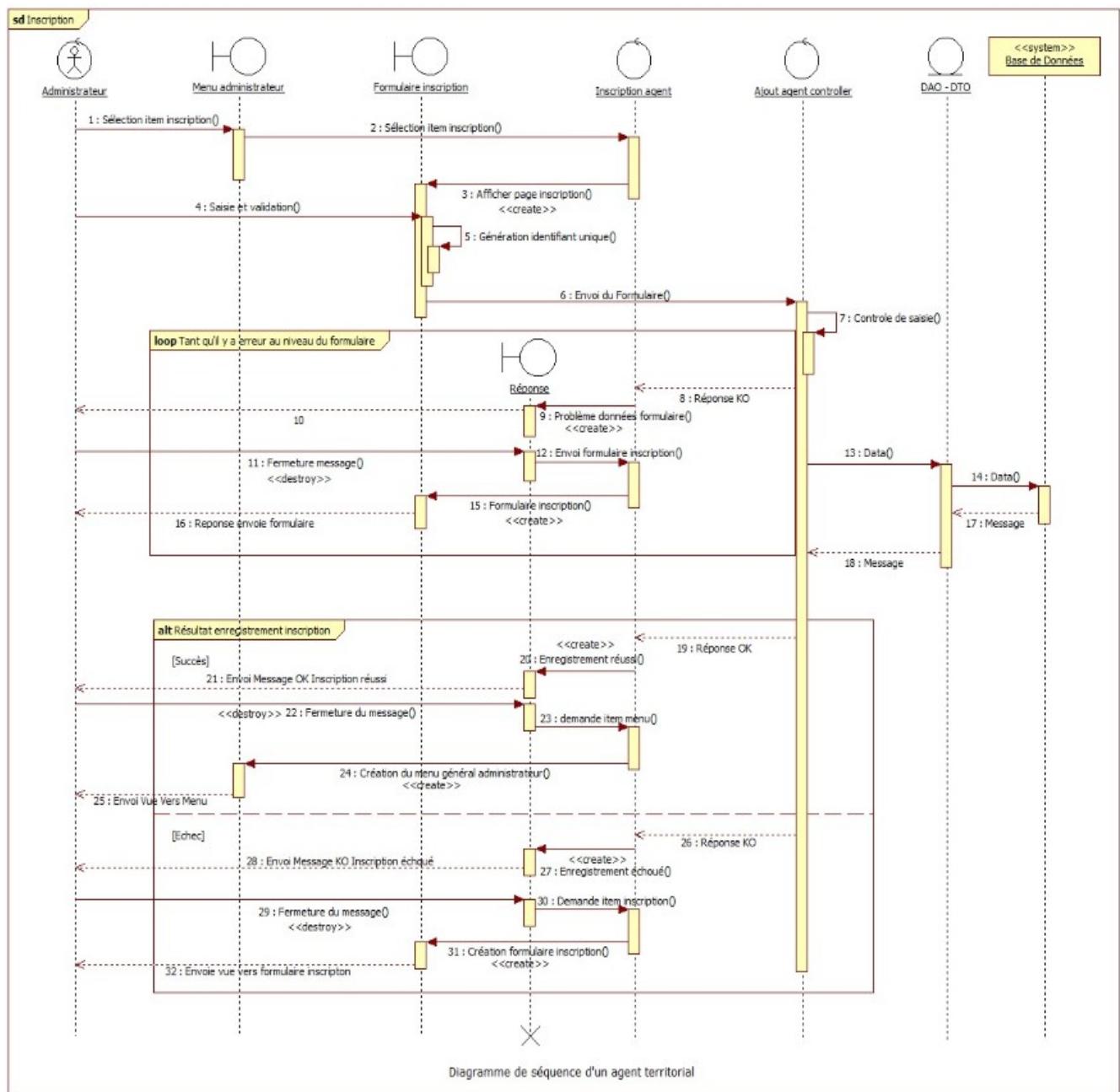
Control : Classe contrôle, les objets de ces classes sont internes au système. Elle contrôle le comportement des cas d'utilisation. Les représentent une activité de contrôle, une règle de gestion, un « chef d'orchestre » pour certain, ...

Entity : Classe entité, les objets de ces classes représentent les données, elles permettent entre autres la persistance des données dans la base de données. Elles sont généralement appelées entités.

Nous avons donc une couche **Boundary** pour tous les écrans, puis une couche **Contrôleur** pour la gestion des vues et le passage des données et enfin une couche Entity où se trouvent entre autres les DAOs pour la persistance des données...

Le diagramme de séquence nous permettra de mieux appréhender le fonctionnement de ce modèle.

O Inscription d'un agent territorial



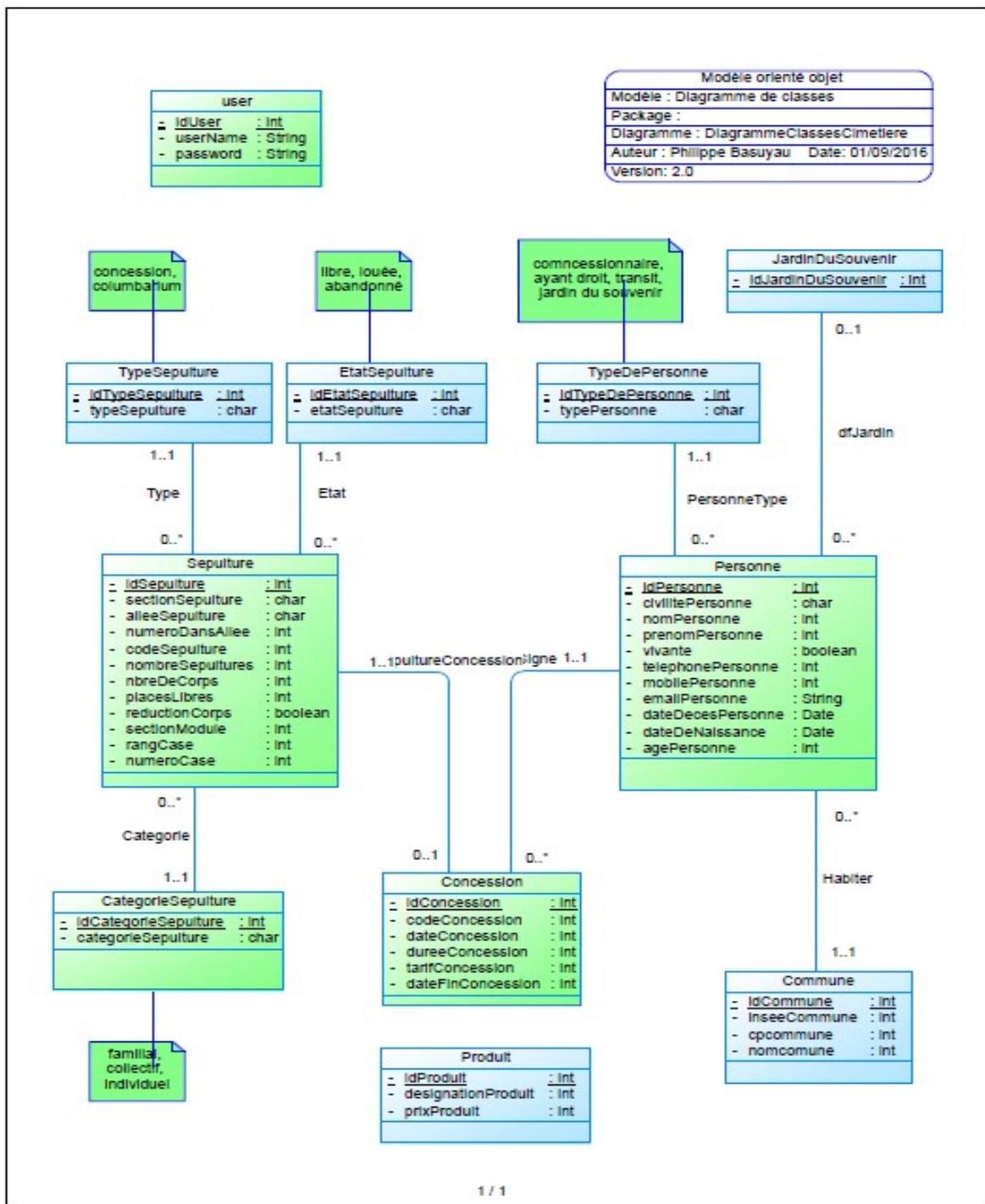
DONE :
DSEQ

DOING :
DCL

TODO :
MDP

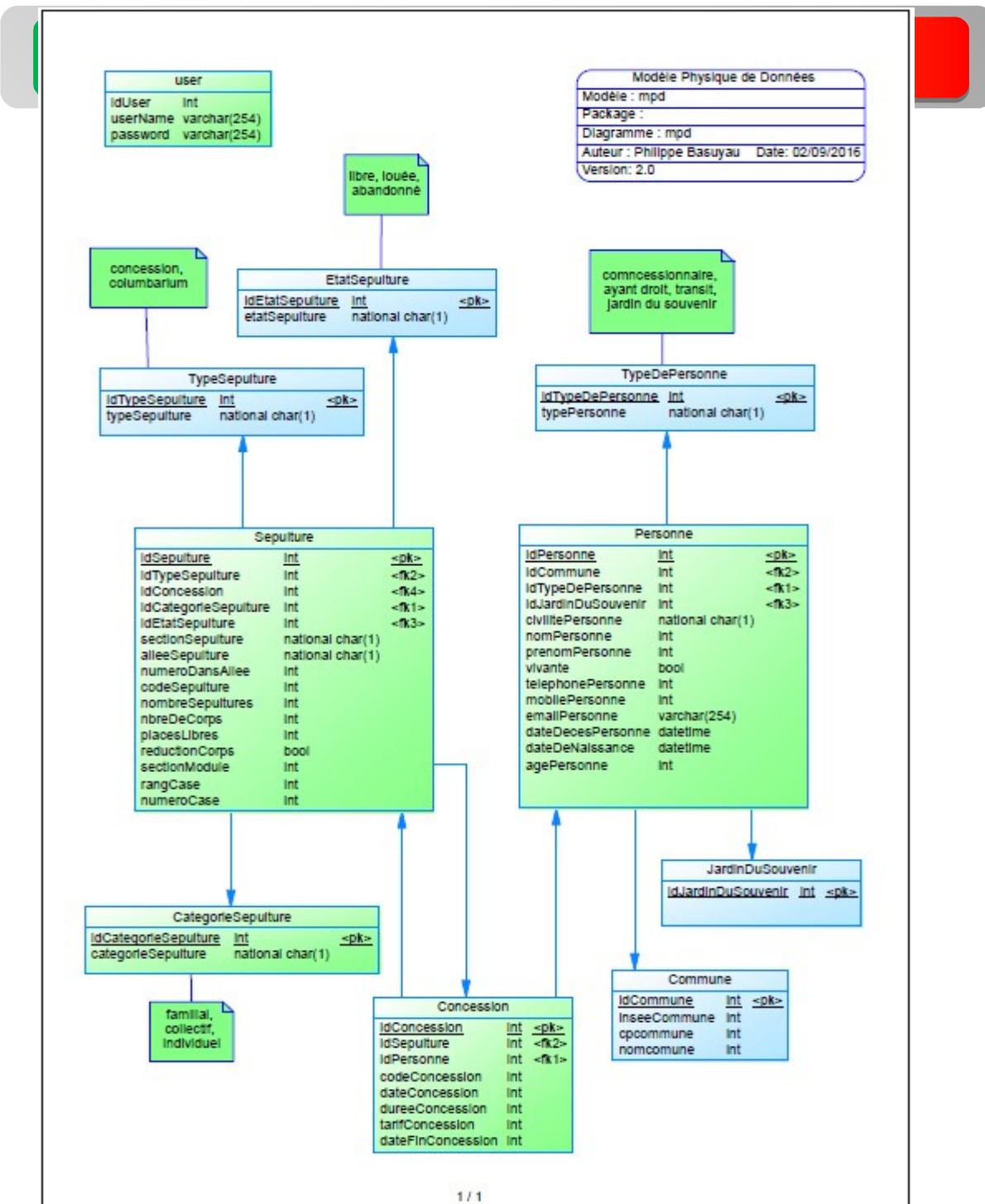
IV. Diagramme de classes

Dans ce diagramme c'est une association inter classes avec multiplicité (exemple : dans une concession, il peut y avoir 0 ou plusieurs personnes et une personne peut être dans 0 ou 1 concession) elle peut être 0-1, 0-* , 1-1, 1-*...



V. Modèle physique de données

Le passage du diagramme de classes au MDP (Modèle Physique de Données) va me permettre d'aborder la formalisation du script de la création de la base de données et donc de la persistance des données.



VI. Création de la base de données

A partir de mon modèle physique de données, je vais pouvoir générer un fichier en .sql pour ma base de données que j'ouvrirais avec MySQL mais plus

précisément « MySQL Workbench » qui est un **SGBD** (Système de Gestion de Base de Données) j'en profiterais pour améliorer et corriger le script afin de mieux ajuster la base de données ainsi que les tables qu'elle comporte. Vous trouverez dans l'annexe le code complet.

Par contre je vous dévoile quelques morceaux, en premier lieu la création de la base avec quelques explications sur sa droite.

```
1  /*=====
2  /* Nom de SGBD : MySQL 5.0
3  /* Date de création : 01/09/2016 12:04:28
4  /*=====
5
6  DROP DATABASE cimetiere;
7  CREATE DATABASE IF NOT EXISTS cimetiere;
8  DEFAULT CHARACTER SET utf8
9  COLLATE utf8_general_ci;
10
11 USE cimetiere;
12
13
14 SET FOREIGN_KEY_CHECKS = 0;
15
16
17 drop table if exists CategorieSepulture;
18
19 drop table if exists Commune;
20
21 drop table if exists Concession;
22
23 drop table if exists EtatSepulture;
24
25 drop table if exists JardinDuSouvenir;
26
27 drop table if exists Personne;
28
29 drop table if exists Sepulture;
30
31 drop table if exists TypeDePersonne;
32
33 drop table if exists TypeSepulture;
34
35 drop table if exists User;
36
```

face la base de données « cimetiere » si elle existe.

réé la base de données « cimetiere » si elle n'existe pas.

u de caractères par défaut, ici je peux utiliser utf8 car il prend en compte les accents et les caractères spéciaux.

pour une question de tri

utilise La base de données « cimetiere »

désactivation des clés étrangères de façon qu'elles laisse se dérouler parfaitement la création des tables. On les réactivera à la fin.

efface la table CategorieSepulture, etc...

Pour info complémentaire : UTF-8 est un jeu de caractères qui permet d'utiliser pratiquement toutes les langues et de nombreux caractères spéciaux utilisés dans le domaine des sciences, des mathématiques et de la technologie

Ici c'est la création de deux tables différentes l'une sans clé étrangère et

```
42 /*=====
43 /* Table : Commune
44 /*=====
45 create table Commune
46 (
47     idCommune          int not null auto_increment,
48     inseeCommune       char(5),
49     cpCommune          char(5),
50     nomCommune         varchar(50),
51     primary key (idCommune)
52 )ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
53
54 /*
55 /* Table : Concession
56 /*=====
57 create table Concession
58 (
59     idConcession        int not null auto_increment,
60     idSepulture         int not null,
61     idPersonne          int not null,
62     codeConcession      int(5),
63     dateConcession      date,
64     dureeConcession     int(3),
65     tarifConcession     int(5),
66     dateFinConcession   date,
67     primary key (idConcession)
68 )ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

Création d'une table sans clé étrangère. On lui indique une clé primaire qui ne peut être nul et qu'elle soit auto-incrémenté. J'ai utilisé « char » car ce sont des variables de longueur fixe, pour celle qui ne le sont pas j'ai utilisé « varchar », même si ici on voit que j'ai mis 50,

ICl on a 1 clé primaire et 2 clés étrangères appelées foreign key (**FK**) qui sont des clés primaires d'autre tables. L'on a la clé primaire sepulture de la table sepulture et idPersonne qui est celle de la table personne. J'utilise aussi **InnoDB**, c'est le seul moteur qui gère les clés étrangères et les transactions.

l'autre avec.

Puis je vais gérer les clés étrangères pour chaque table, si cela est nécessaire et réactivé les clés étrangères à la fin.

```
193 alter table Sepulture add constraint FK_Etat foreign key (idEtatSepulture)
194     references EtatSepulture (idEtatSepulture) on delete restrict on update restrict;
195
196 alter table Sepulture add constraint FK_ConcessionSepulture foreign key (idConcession)
197     references Concession (idConcession) on delete restrict on update restrict;
198
199 alter table Sepulture add constraint FK_Type foreign key (idTypeSepulture)
200     references TypeSepulture (idTypeSepulture) on delete restrict on update restrict;
201
202
203
204
205
206
207
208 SET FOREIGN_KEY_CHECKS = 1;
```

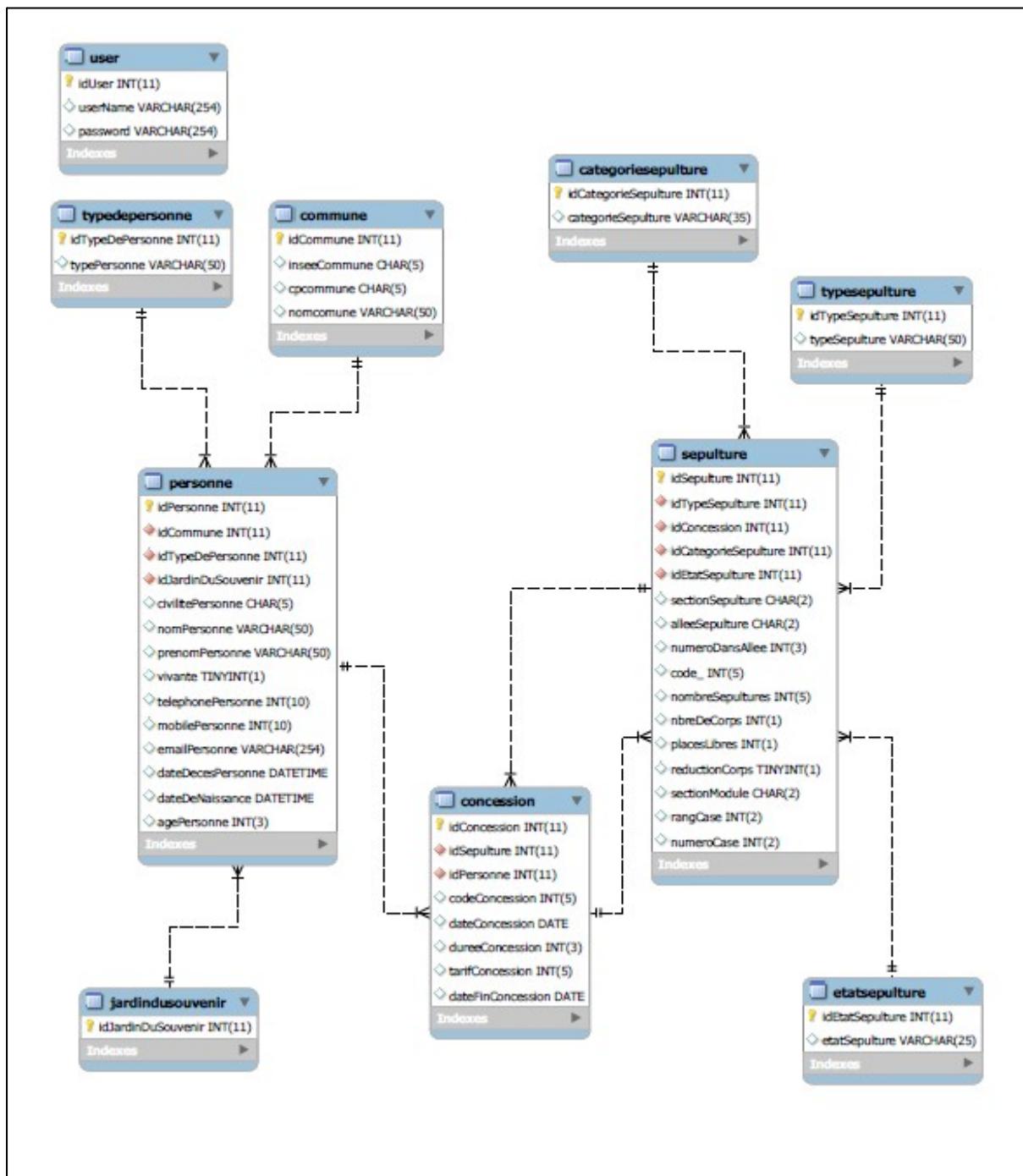
DONE :
Base de
Données

DOING :
Reverse
Engineer

TODO :
Diag. Réseau

VII. Reverse Engineer Database

C'est le diagramme que je produis en faisant un reverse Engineer Database dans le logiciel MySQL Workbench une fois ma base et les tables créées.



DONE :
Reverse
Engineer

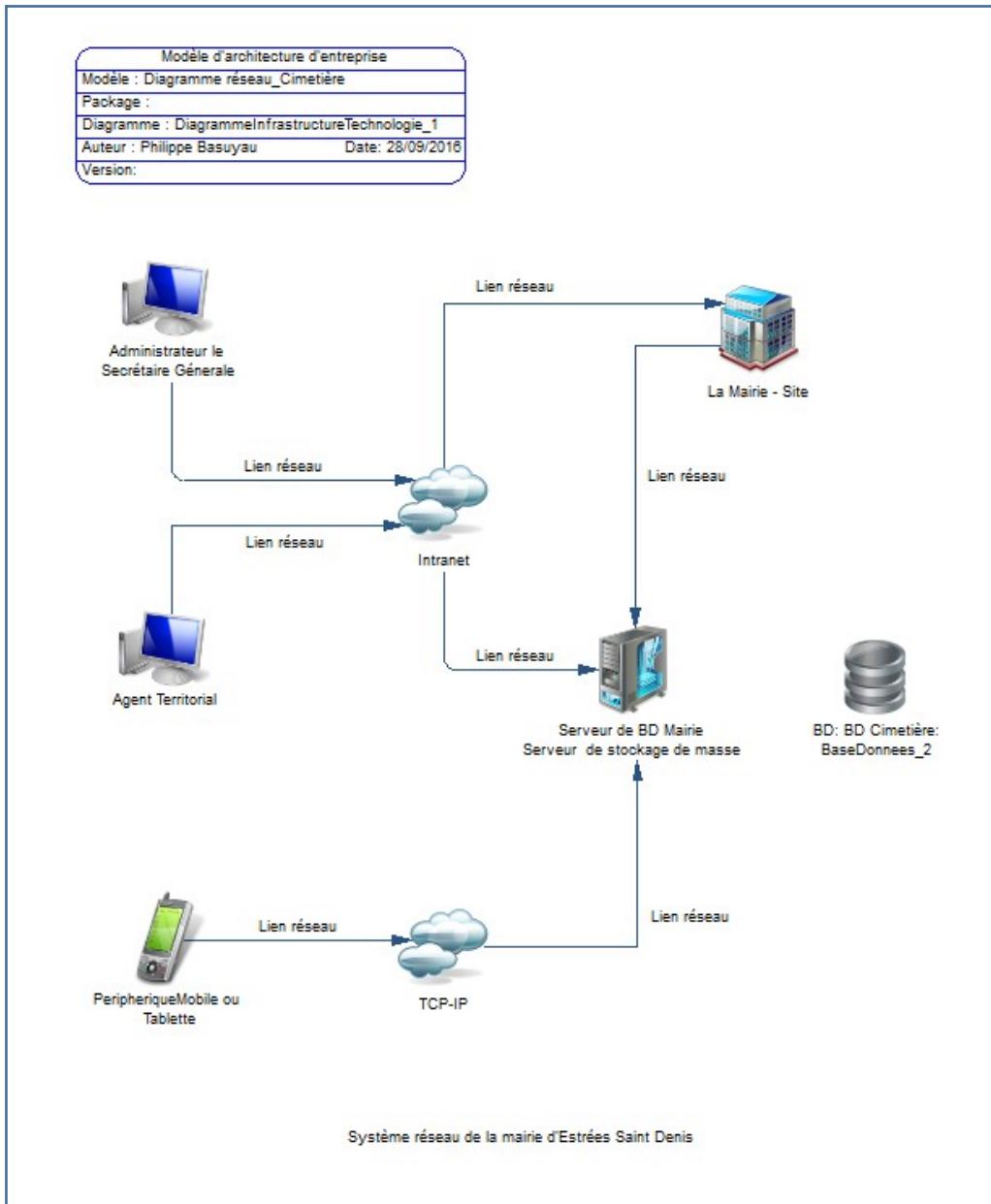
DOING :
Diag. Réseau

TODO :
Diag.
Déploiement

VIII. Diagramme Réseau de la mairie

La structure du réseau de la Mairie, elle possède son propre serveur depuis 1997.

Le serveur est un ProLiant ML350pGen8, avec Microsoft Windows Server 2008 R2 Standard Edition, la License est valable pour 1 serveur et 10 utilisateurs. License et support OEM ROK DVD, verrouillage BIOS



(Hewlett-Packard).

La sauvegarde est stockée sur du HP StorageWorks Ultrium 920, lecteur de bande magnétiques LTO Ultrium (400 Go / 800 Go), Ultrium 3, SAS, interne, 5.25.

DONE :
Diag. Réseau

DOING :
Diag.
Déploiement

TODO :
Les Maquettes

IX. Diagramme de déploiement

C'est le schéma du déploiement de mon logiciel sur l'intranet de la mairie

DONE :

Diag.
Déploiement

DOING :

Les Maquettes

TODO :

Le Code

Les Maquettes

Les maquettes sont des vues des différentes pages du logiciel, elles permettent de montrer aux client le rendu de celle-ci. J'ai utilisé le logiciel « Pencil » il m'a permis de créer et de modéliser les différentes vues, que j'ai présentées à la mairie afin d'avoir avec un échange sur les modifications qui pourraient améliorer le logiciel.

Accueil

[Connexion](#)

Projet-Cimetière



©Philippe Basuyau

[Me Contacter](#)

L'authentification est obligatoire pour l'accès au logiciel. Si on n'est pas authentifié l'on restera sur cette vue.

I. Vue Administrateur - Secrétaire général

O Inscription au droit d'utilisateur

Nous sommes le 29 août 2016

E-Cimetière

Deconnexion

Bonjour, Mr Mussard

Accès Agent

Inscription Agent Territorial ▾

Inscription de droit d'Agent Territorial

Mr ▾

Nom Ici le nom

Mot de passe *****

Prénom Ici le prénom

Confirmer le mot de passe *****

Email Ici le email

Fonction de l'agent Agent Territorial ▾

Autorisation de droit Cocher la case pour l'Autoriser l'accès

Valider

©Philippe Basuyau

[Me Contacter](#)

Tout utilisateur doit être inscrit pour se servir du logiciel.

O Modification de la fiche utilisateur

Nous sommes le 29 août 2016

E-Cimetière

Deconnexion

Bonjour, Mr Mussard

Accès Agent

Gestion Agent territorial ▾

Modification de droit d'Agent Territorial

Choix de l'Agent ▾

Modification de mot de passe

Nom Ici le nom

Nouveau Mot de passe *****

Prénom Ici le prénom

Confirmer le mot de passe *****

Email Ici le email

Fonction de l'agent Agent Territorial ▾

Cocher la case pour l'Autoriser l'accès

Valider

©Philippe Basuyau

[Me Contacter](#)

On pourra corriger le nom et changer le mot de passe si le besoin s'en fait sentir.
© Philippe Basuyau -- Stage Développeur Logiciel -- Projet Cimetière --
Octobre 2016

o

Suppression du droit de l'agent territorial

Suppression de droit d'Agent Territorial

Recherche d'un agent Territorial

Suppression d'un droit

Nom	<input type="text" value="Nom"/>
Prénom	<input type="text" value="Prénom"/>

Cocher la case pour confirmer la suppression de droit

Valider

©Philippe Basuyau

[Me Contacter](#)

L'agent Territorial inscrit peut être amener à évoluer ou à quitter la fonction publique, nous avons donc besoin de pouvoir lui enlever les droits d'utilisation du logiciel.

o

Gestion de la tarification

Les tarifs sont

Nouveau

Création

Nouveau

Ici le nom du produit

Prix en €

Tarification Produit

Changement Tarification

Modification

Recherche produit

Nom du produit

Nouveau tarif

Fonction de l'agent

d'évoluer,

Valider

On doit avoir la possibilité de les

©Philippe Basuyau

[Me Contacter](#)

changer ou de rajouter un nouveau produit avec son prix.

o Gestion topographique du Cimetière



©Philippe Basuyau

[Me Contacter](#)

Le cimetière peut être amené à s'agrandir ou à avoir une refonte complète, on doit donc avoir la possibilité de créer, modifier des sections, des Allées et des numéros.

I. Vue Utilisateur – Agent Territorial

o Création du compte concessionnaire

Nous sommes le 29 août 2016

E-Cimetière

Deconnexion

Bonjour, Mme Nathalie Tourman

Inscription Modification Consultation Renouvellement

Inscription du Concessionnaire

Label

Concessionnaire
Civilité <input type="button" value="Mr"/>
Nom <input type="text" value="Ici le nom"/>
Prénom <input type="text" value="Ici le prénom"/>
Adresse
Adresse <input type="text" value="Ici l'adresse"/>
Ville <input type="text" value="Ici le nom de la ville"/>
Code Postal <input type="text" value="Ici le code postal"/>
Téléphone <input type="text" value="Ici le Numéro de téléphone"/>
Email <input type="text" value="Ici l'email"/>

Naissance

Sexe <input type="radio"/> M <input type="radio"/> F
Date de Naissance <input type="text" value="Ici la date de naissance"/>
Lieu de Naissance <input type="text" value="Ici le lieu de naissance"/>
Département <input type="text" value="Ici le lieu de naissance"/>

Ayant(s) Droit(s)

Etat	Nom	Prénom
Epoux(se) <input type="button"/>	<input type="text" value="Ici le nom"/>	<input type="text" value="Ici le prénom"/>
Enfant <input type="button"/>	<input type="text" value="Ici le nom"/>	<input type="text" value="Ici le prénom"/>
Cousin <input type="button"/>	<input type="text" value="Ici le nom"/>	<input type="text" value="Ici le prénom"/>

Conjoint(e)

Civilité <input type="button" value="Mr"/>
Nom <input type="text" value="Ici le nom"/>
Nom marital <input type="text" value="Ici le nom marital"/>
Prénom <input type="text" value="Ici le prénom"/>

Sépulture

Sépulture <input type="button" value="Choix Type Sépulture"/>
Emplacement <input type="button" value="Section"/> <input type="button" value="Allée"/> <input type="button" value="N°"/>
Durée <input type="button" value="Choix de la durée"/>
Type concession <input type="button" value="Type de concession (familiale)"/>

©Philippe Basuyau Cimetière - Estrées Saint Denis - 60190 Me Contacter

La personne qui vient faire une demande pour une concession devient un concessionnaire et divers choix s'offrent à elle, comme le type de sépulture, l'emplacement, type de concession ...

0 Modification du compte Concessionnaire

Nous sommes le 29 août 2016

E-Cimetière

Deconnexion

Bonjour, Mme Nathalie Tourman

Inscription Modification Consultation Renouvellement

Modification de la concession

Recherche : Label Recherche par Nom Ici le nom Recherche par N° Dossier Ici le numéro de dossier Etat Nom Prénom

Le Concessionnaire

Nom	Ici le nom
Prénom	Ici le nom
Age	Ici le nom

Information

Adresse	Adresse
Ville	Ville
Code postal	Code postal
Téléphone	Téléphone
Email	Email

Conjoint(e)

Civilité	Mr
Nom	Ici le nom
Nom marital	Ici le nom marital
Prénom	Ici le prénom
Téléphone	Téléphone
Email	Email

Ajout ayant droit

Status	Ici le nom	Ici le prénom	Supprimer
Status	Ici le nom	Ici le prénom	Supprimer
Status	Ici le nom	Ici le prénom	Supprimer

Liste des ayant(s) droit(s)

Status	Ici le nom	Ici le prénom	Supprimer
Status	Ici le nom	Ici le prénom	Supprimer
Status	Ici le nom	Ici le prénom	Supprimer

info Sépulture

Capacité Maxi	3
Reduction de corps	0
Occupation	0
Place Restante	3

Valider

cocher la case pour pouvoir valider les modifications

©Philippe Basuyau [Me Contacter](#)

Si le concessionnaire a envie de modifier la durée ou modifier sa liste d'ayant droit, il doit en avoir la possibilité. La réduction de corps amène forcement à une modification de la concession.

0 Ayant droit sur la concession

Le concessionnaire peut s'il le désir changer sa liste d'ayants droit.

E-Cimetière

Deconnexion

Bonjour, Mme Nat**** T*****

Inscription Ayant Droit Consultation Renouvellement

Ayant Droit sur la Concession de :

Recherche Concession : Ici le nom Ici le numéro de dossier

Ayant Droit sur la Concession de :

Nom & prénom

Liste Ayant Droit

Status	Ici le nom	Ici le prénom	Age	Vivant	Modifier	Supprimer
Status	Ici le nom	Ici le prénom	Age	décédé	Modifier	Supprimer
Status	Ici le nom	Ici le prénom	Age	Vivant	Modifier	Supprimer

Ajout Ayant Droit

Civilité	Mr
Nom	Ici le nom
Prénom	Ici le prénom
Date de Naissance	Ici la date de naissance
lien de Parenté	Choix du lien de familial
Adresse	Ici l'adresse
Ville	Ici la ville
Code postal	Ici le code postal
Téléphone	Ici le téléphone
Email	Ici l'email

Ajouter

©Philippe Basuyau [Me Contacter](#)

O Renouvellement de la concession

Nous sommes le 29 août 2016

E-Cimetière

Deconnexion

Bonjour, Mme Nat**** T*****

Inscription Ayant Droit Recapitulatif Renouvellement

Recherche Concession :

Renouvellement Concession

Nom Concessionnaire Prénom

Recherche Concessionnaire

©Philippe Basuyau

[Me Contacter](#)

La concession peut être renouvelée ou la durée peut être modifiée.

O Consultation de la concession

E-Cimetière

Deconnexion

Bonjour, Mme Nat**** T*****

Inscription Ayant Droit Recapitulatif Renouvellement

Recherche Concession :

Prénom

Emplacement occupé

Réduction corps Début de concession

Nbr corps maxi Fin de concession

Emplacement libre

©Philippe Basuyau

[Me Contacter](#)

Vue récapitulative d'une concession avec les informations concernant le concessionnaire.

DONE :
Les Maquettes

DOING :
Le Code

TODO :
L'annexe

La Partie Développement

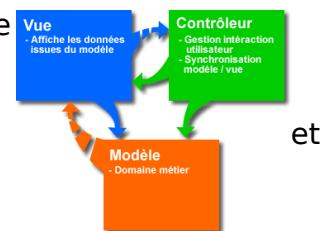
I.

I. La structure MVC

C'est une structure qui a comme type de langage, un langage procédural, la partie orienté objet se situe au niveau des Entités avec les DAO-DTO. Dans mon système mes données entrées sont des **POPO** (plain old PHP object) (c'est un peu l'équivalent du POJO en java (plain old Java object)) et mes sorties sont un tableau ordinal associatif

Le pattern* **MVC** (Model – View – Controller) implique une gestion événementielle et la mise en place du pattern Observer du GOF (Gang Of Four – ceci désigne les quatre informaticiens Erich Gamma, Richard Helm, Ralph Johnson et John Vlissides)

*Pattern : c'est comme un modèle, un patron

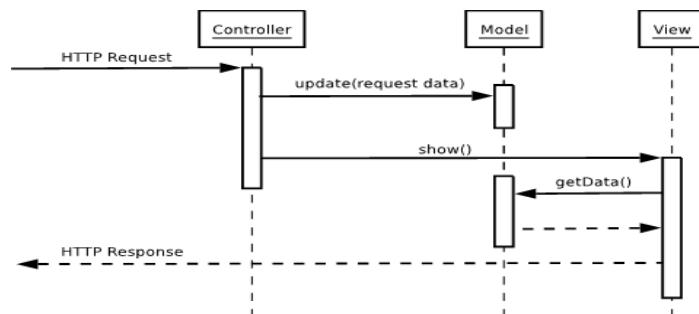


Le principe de MVC est le suivant : une requête sollicite l'action du contrôleur.

Le contrôleur appelle le service métier qui réalise les traitements associés à l'action. Ce service métier pourra appeler N autres services selon ses besoins.

Un service métier peut appeler (ou non) la couche modèle si des interactions doivent être effectuées avec la base de données (récupération de données, mise à jour). Il peut également appeler des services techniques (ex : service d'envoi de mails).

La couche modèle interagit avec la base de données par l'intermédiaire d'une couche d'abstraction plus ou moins grande (ex : PDO pour être près de la base ;



Doctrine ORM pour être plus abstrait).

© Philippe Basuyau -- Stage Développeur Logiciel -- Projet Cimetière -- Octobre 2016

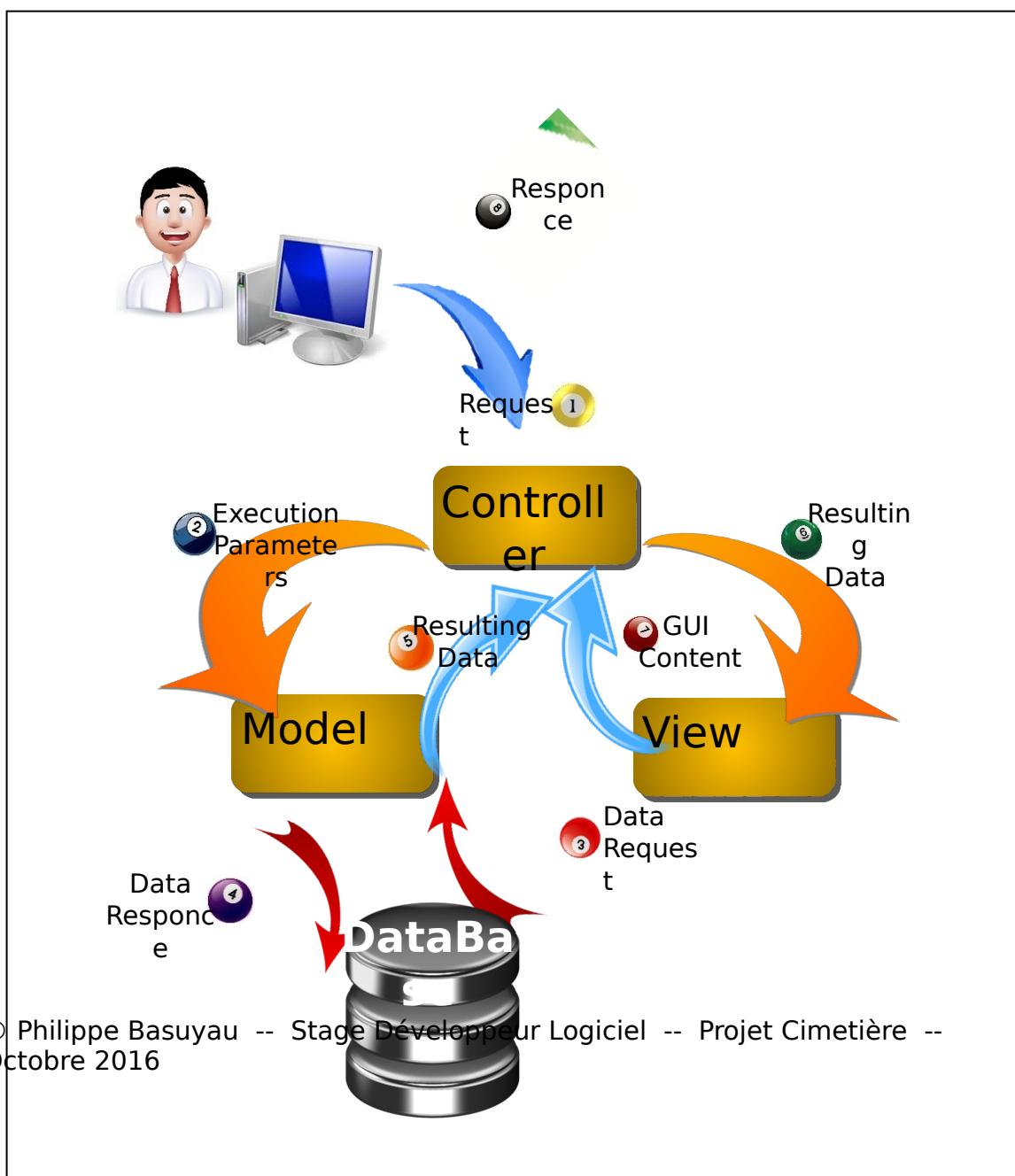
L'application gère les données issues de la base de données via les notions d'objets (appelée entités en Symfony / Zend) ou de tableaux (par défaut en CakePHP).

Une fois remonté jusqu'au contrôleur, celui-ci appelle une vue pour l'affichage. La vue peut utiliser les données issues de la base, toujours par l'intermédiaire des entités (ou des tableaux en CakePHP). La vue peut être au format HTML, JSON, XML, etc... La réponse est envoyée au client.

I. Le schéma de fonctionnement de mon MVC

Le modèle MVC préconise de séparer dans des classes les différentes problématiques :

- Des "vues" (charte graphique, ergonomie)
- Du "modèle" (cœur du métier)
- Des "contrôleurs" (tout le reste : l'enchaînement des vues, les autorisations d'accès, ...)



Le code

I. L'inscription

Pour l'inscription je suis déjà Logué en tant qu'agent territorial ou administrateur qui pour la mairie sera le secrétaire général.

Le code complet de l'inscription de l'agent de la vue et des 2 contrôleurs, du DAO et DTO seront joint en annexe.

o La Sécurité

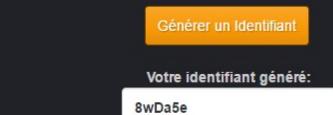
J'ai mis en place un code en JavaScript pour générer un identifiant, il est placé dans le « HEAD » de la vue *inscription*. Pour éviter la découverte par erreur du mot de passe d'un collègue, le login se fait par l'identifiant généré et son mot de passe. Et seul le mot de passe ou le nom de l'agent pourront être modifiés.

```
<HEAD>
    <!-- //le JavaScript dans le head-->
    <SCRIPT LANGUAGE="JavaScript">

        function getIdentifiant() {
            // Définir tout les caractères possibles dans le mot de passe
            var cars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz";
            var long = cars.length;
            wpas = "";
            taille = 6;//je veux un identifiant de 6 caractères

            // je vais utiliser la fonction de Math.random pour générer un identifiant
            for (i = 0; i < taille; i++) {
                wpos = Math.round(Math.random() * long);
                wpas += cars.substring(wpos, wpos + 1);
            }
            // fin du for
            return wpas; // affiche moi le résultat créé
        //fin de la fonction getIdentifiant
    </script>
</HEAD>
```

L'Inscription d'un agent territorial



Dans le *contrôleur ajoutInscriptionController*, je récupère les champs du formulaire au début et pour éviter les injections de code comme les XSS, je mets en place une protection du type FILTER_SANITIZE, qui supprime les balises, et supprime ou encode les caractères spéciaux.

```
//Récupération des données
//pour éviter les injections XSS je place "FILTER_SANITIZE_STRING"
$idUser = filter_input(INPUT_POST, 'idUser', FILTER_VALIDATE_INT);
$identifiant = filter_input(INPUT_POST, 'identifiant', FILTER_SANITIZE_STRING);
$userName = filter_input(INPUT_POST, 'userName', FILTER_SANITIZE_STRING);
$userNameConfirm = filter_input(INPUT_POST, 'userNameConfirm', FILTER_SANITIZE_STRING);
$password = filter_input(INPUT_POST, 'password', FILTER_SANITIZE_STRING);
$passwordConfirm = filter_input(INPUT_POST, 'passwordConfirm', FILTER_SANITIZE_STRING);
$submitAjoutInscription = filter_input(INPUT_POST, 'submitAjoutInscription', FILTER_SANITIZE_STRING);

<?php
// A la place require je sécurise avec require_once
require_once 'conf/config.php';
require_once 'lib/lib-html.php';
require_once 'lib/lib-mvc.php';
require_once 'lib/lib-pdo.php';
require_once 'models/userModel.php';
require_once 'autoload.php';
```

Dans l'index, qui est le point d'entrée du logiciel, je sécurise en remplaçant les « require » pour le chargement de mes bibliothèques dont j'ai besoin pour éviter les doublons de code avec « require_once ».

0 Le contrôle des champs du formulaire

Pour être sûr qu'il n'y a pas de faute sur le nom de l'agent territorial et de son password.

Je le fais saisir 2 fois et je les compare pour vérifier qu'ils sont bien exacts si ce n'est pas le cas il enverra différents messages en fonction de l'erreur commise.

Le nom n'est pas conforme à sa confirmation

Le mot de passe n'est pas conforme à sa confirmation

Si tout se passe normalement le message suivant apparaît :

L'agent territorial Basuyau a été correctement enregistré dans la base de données

```
//Validation des données

// a faire contrôler il fait pas le if suivant
if ($userName != $userNameConfirm) {
    $message = "Le nom n'est pas conforme à sa confirmation";
    $_SESSION['flash'] = $message;
    header('location:index.php?controller=inscriptionController');
} elseif ($password == null || mb_strlen($password) < 5) {
    $message = "Votre mot de passe doit comporter au moins 5 caractères";
    $_SESSION['flash'] = $message;
    header('location:index.php?controller=inscriptionController');
} elseif ($password != $passwordConfirm) {
    $message = "Le mot de passe n'est pas conforme à sa confirmation";
    $_SESSION['flash'] = $message;
    header('location:index.php?controller=inscriptionController');
} elseif ($identifiant == "") {
    $message = "L'identifiant est vide, veuillez remplir le champs";
    $_SESSION['flash'] = $message;
    header('location:index.php?controller=inscriptionController');
} elseif (isset($submitAjoutInscription) && isset($identifiant) && isset($userName) && isset($password))
```

Voici le code correspondant à cette action :

II. Le transfert des données

Je prépare l'initialisation de mon PDO*, du DAO* et tu DTO*

PDO : c'est une extension définissant l'interface pour accéder à une base de données depuis PHP.

DAO : **Data Access Object**, un objet d'accès aux données (issues par exemple de bases de données relationnelles dans une architecture logicielle).

DTO : **Data Transfer Object**, c'est un objet de transfert de données. Il a pour but de simplifier les transferts de données entre les sous-systèmes d'une application logicielle.

0 Le code du DAO

- Les fonctions findAll, findOneById, find, delete, save sont le **Read** : lire (de l'acronyme **CRUD**)

```
<?php

class UserDAO implements IUserDAO {

    /**
     * @var \PDO
     */
    private $pdo;

    /**
     * DAOClient constructor.
     * @param PDO $pdo
     */
    public function __construct(PDO $pdo) {
        $this->pdo = $pdo;
    }

    public function findAll() {
        $sql = "SELECT * FROM user"; // la requete sql
        $rs = $this->pdo->query($sql)->fetchAll();
        return $rs;
    }

    public function findOneById(array $pk) {
//requete sql
        $sql = "SELECT * FROM user WHERE idUser=? ";
//je stocke ma requete sql
        $statement = $this->pdo->prepare($sql);
//je l'execute
        $statement->execute($pk);
//fetch est associé à l'objet PDOStatement il sert à récupérer une ligne
depuis un jeu de résultats
//je mets le résultat dans la variable rs
        $rs = $statement->fetch();
// je renvoie le résultat
        return $rs;
    }
}
```

```

public function findById($pkValue) {
    $sql = "SELECT * FROM user WHERE idUser=?";
    $statement = $this->pdo->prepare($sql);
    $statement->execute(array($pkValue));
    $rs = $statement->fetch();
    return $rs;
}

public function find(array $search) {
    $sql = "SELECT * FROM user ";

    if (count($search) > 0) {
        $sql .= " WHERE ";
        $cols = array_map(
            function($item) {
                return "$item=:{$item}";
            }, array_keys($search)
        );
        //implode : rassemble les éléments d'un tableau en une chaîne de caractère
        $sql .= implode(" AND ", $cols);
    }

    $statement = $this->pdo->prepare($sql);
    $statement->execute($search);

    return $statement->fetchAll(PDO::FETCH_ASSOC);
}

```

Pour info Un tableau récapitulatif sur le CRUD et de sa correspondance en SQL et en http.

Opération	SQL	HTTP
Create	INSERT	POST (en)
Read (Retrieve)	SELECT	GET (en)
Update (Modifier)	UPDATE (en)	PUT (en) et PATCH (en)
Delete (Destroy)	DELETE (en)	DELETE (en)

Celui-ci est un peu à part car il vérifie si c'est un Insert ou un Update.

```
public function save(UserDTO $user) {
// dans cette partie l'on vérifie si l'idUser de l'agent existe
// si getIdUser est null alors il l'ajoute
// sinon il le met à jour
    if ($user->getIdUser() == null) {
        return $this->insert($user);
    } else {
        return $this->update($user);
    }
}
```

Les fonctions insert, update, delete sont le **CUD** (**C**reate **U**pdate **D**elete) de l'acronyme **CRUD**.

```
private function insert(UserDTO $user) {
    $sql = "INSERT INTO user (identifiant, userName, password) VALUES
( ?, ?, ? )";
    $statement = $this->pdo->prepare($sql);
    $statement->execute([
        $user->getIdentifiant(), $user->getUserName(), $user->getPassword()
    ]);
}

private function update(UserDTO $user) {
    $sql = "UPDATE user SET identifiant=?, userName=?, password=? WHERE
idUser=? ";
    $data = array(
        $user->getIdentifiant(), $user->getUserName(), $user->getPassword(),
        $user->getIdUser()
    );
    $statement = $this->pdo->prepare($sql);
    return $statement->execute($data);
}

public function delete(UserDTO $user) {
    if ($user->getIdUser() != null) {
        $sql = "DELETE FROM user WHERE idUser=? ";
        $statement = $this->pdo->prepare($sql);
        return $statement->execute([$user->getIdUser()]);
    }else{
        return "Problème d'argument";
    }
}
```

o Le code du DTO

```
1  <?php
2
3  class UserDTO
4  {
5
6      private $idUser;
7      private $identifiant;
8      private $userName;
9      private $password;
10
11     public function __construct($idUser="", $identifiant="", $userName="", $password="")
12     {
13         $this->idUser = $idUser;
14         $this->identifiant = $identifiant;
15         $this->userName = $userName;
16         $this->password = $password;
17     }
18
19     public function setIdUser($idUser)
20     {
21         $this->idUser = $idUser;
22     }
23
24     public function getIdUser()
25     {
26         return $this->idUser;
27     }
28
29     public function setIdentifiant($identifiant)
30     {
31         $this->identifiant = $identifiant;
32     }
33
34     public function getIdentifiant()
35     {
36         return $this->identifiant;
37     }
}
```

Je déclare en mode privé les variables \$idUser, \$identifiant, \$userName et \$password.

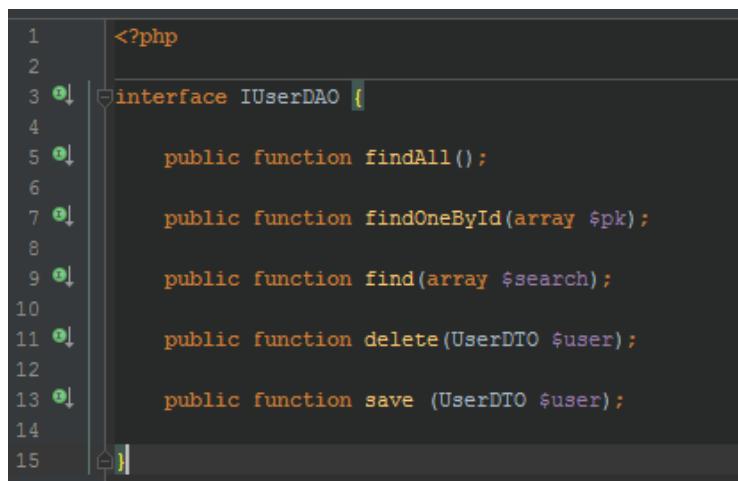
J'affecte mes variables avec les valeurs \$idUser, \$identifiant, \$userName et \$password.

Pour chaque toutes les variables, je fais les setters et les getters.

Bien entendu pour chaque DAO et DTO créé je fais un *DAOTest* et un *DTOTest* afin de m'assurer qu'il fonctionne correctement

0 Le code du IDAO

L'IDAO (l'Interface DAO) me sert de liaison avec le DAO.



```
<?php
interface IUserDAO {
    public function findAll();
    public function findOneById(array $pk);
    public function find(array $search);
    public function delete(UserDTO $user);
    public function save (UserDTO $user);
}
```

fonction « findAll » (trouver tous)

fonction findOneById (trouver un)

fonction « find » (rechercher)

fonction « delete »

fonction « save »

registre)

0 Le code des différentes bibliothèques concernées

Ici j'utilise la bibliothèque qui contient mon PDO qui me sert à la connexion : lib-PDO.php

```
<?php
/*
 * Created by PhpStorm.
 * User: Philippe Basuyau
 * Date: 06/09/2016
 * Time: 11:01
 */

// les bibliotheques dont je vais avoir besoin
require_once 'conf/config.php';

// fonction qui va utiliser le rappeler le code de connexion
function getPDO() {
    $options = [
        PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION
    ];

    return new PDO(DSN, DB_USER, DB_PASS, $options);
}
```

Et de la bibliothèque que me sert à afficher les différentes vues : lib-mvc.php

```
<?php
/**
 * Created by PhpStorm.
 * User: Philippe Basuyau
 * Date: 06/09/2016
 * Time: 11:15
 */
function getViewContent($view, array $params = []){
    //extrait le nom de ma variable: la cle ; la valeur : la valeur
    extract($params);
    ob_start();
    require ROOT_FOLDER."/views/$view.php";
    $content = ob_get_clean();

    return $content;
}

/**
 * @param $view
 * @param array $params
 * @param string $layout
 * @return string
 */
function getResponse($view, array $params = [], $layout = 'default-layout')
{
    //Récupération du contenu de la vue
    $viewContent = getViewContent($view, $params);

    $params['viewContent'] = $viewContent;

    //Application de la vue sur le gabarit
    return getViewContent($layout, $params);
}

/**
 * @param $controller
 * @param $content
 * @param array $attributes
 * @return string
 */
function linkToController($controller, $content, array $attributes =[]){
    $href = "index.php?controller=$controller";
    return htmlLink($href, $content, $attributes);
}
```

o Le code de configuration des données de connexion

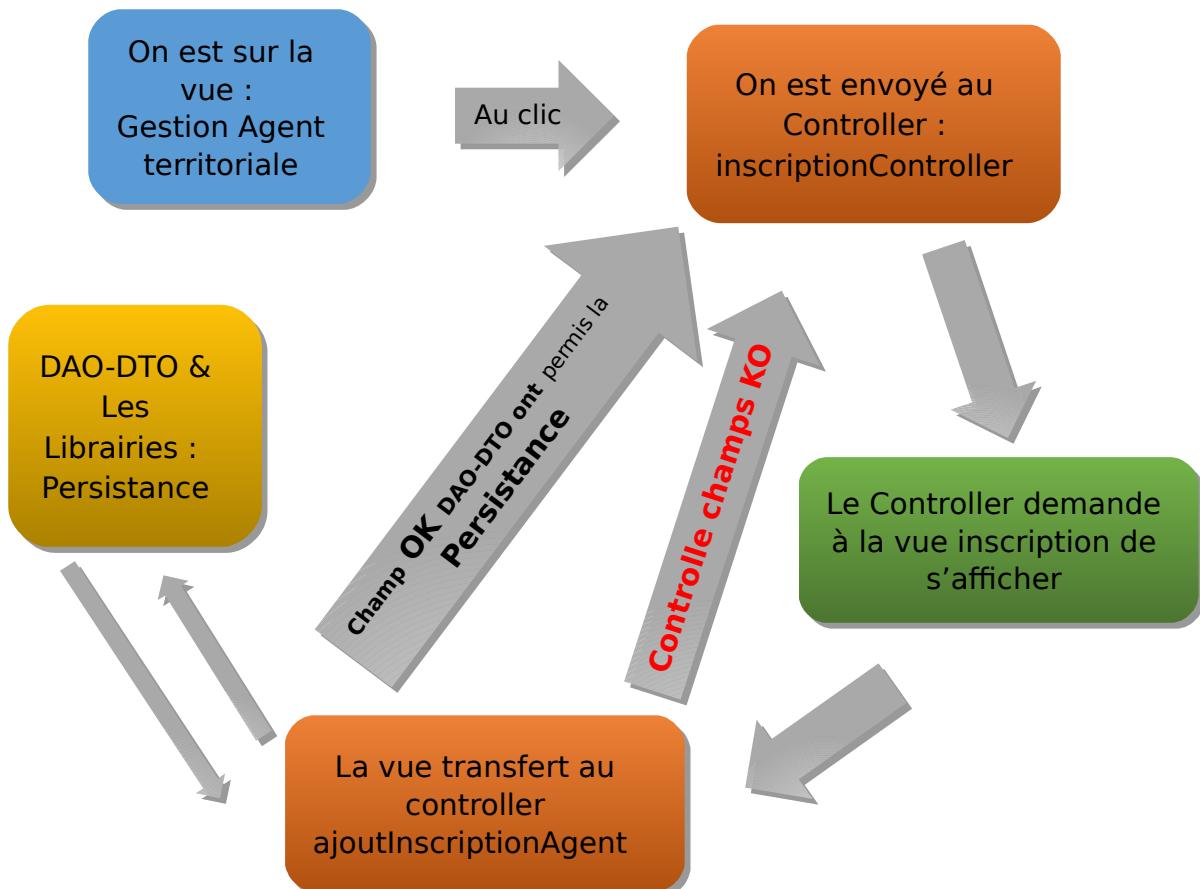
```
/*
 * Configuration pour la connexion BD
 */

define('SERVER', '127.0.0.1');
define('PORT_MYSQL', '3306');
define('DB', 'cimetiere');
define('DSN', 'mysql:host=' . SERVER . ';' . PORT_MYSQL . ':3306' . ';
dbname=' . DB . ';charset=utf8'); // ;dbname=cimetiere;charset=utf8');
define('DB_USER', 'root');
define('DB_PASS', '');
```

III. Le principe de navigation de la vue en inscription

Le principe de fonctionnement de la vue inscription d'un agent territorial en schéma, ci-dessous et à la suite de cela le même chose en version code.

o Le schéma de la navigation et de son traitement



0 Le code de la navigation et de son traitement

La navigation et les contrôles sur les vues

1- Sur la vue gestionAgent :

```
<h1>Gestion Agent Territorial</h1>
<div class="container">

    <div class="row">
        <aside class="col-md-3">
            <ul class="nav nav-pills nav-stacked">
                <li role="presentation" class="active"><a href="index.php?controller=gestionAgentController">Gestion d'Agent</a></li>
                <li role="presentation"><a href="index.php?controller=inscriptionController">Inscription Agent Territorial</a></li>
                <li role="presentation"><a href="index.php?controller=modificationAgentController">Modification Agent Territorial</a></li>
                <li role="presentation"><a href="index.php?controller=suppressionAgentController">Suppression Agent Territorial</a></li>
                <li role="presentation" class="active"><a href="index.php?controller=gestionConcessionController">Gestion Concession</a></li>
                <li role="presentation" class="active"><a href="index.php?controller=gestionCimetiereController">Gestion Cimetière</a></li>
            </ul>
        </aside>
```

Quand je clic sur **Inscription Agent Territorial** de mon menu placé en aside

Cela permet de rediriger vers le contrôleur :

```
<a href="index.php?controller=inscriptionController">
```

2- Sur le controller InscriptionAgent :

Le controller permet d'afficher la vue **inscription**

```
//affichage de la vue
echo getResponse('inscription',
[
    'titre' => $titre,
    'message' => $message,
    'flash' => $flash
]);

```

3- Sur la vue inscriptionAgent :

J'indique que je veux que les données soient en Post pour des raisons de sécurité et je lui dis de passer les informations du formulaire au contrôleur **ajoutInscriptionController** pour qu'il les traite

```
<form method="POST" action="index.php?
controller=ajoutInscriptionController">
```

4- Sur le controller ajoutAgentController :

Si les champs du formulaire correspondent à notre attente alors le controller va les traiter via un DAO et un DTO pour les enregistrer dans la base de données à ce moment-là il y a la persistance des données et on pourra les rappeler à tout moment.

```
// initialisation
$pdo = getPDO();
$dao = new UserDAO($pdo);
$dto = new Utero();

        // Hydratation

        $dto->setIdentifiant($identifiant);
        $dto->setUserName($userName);
        $dto->setPassword($password);

        // Enregistrement
        $dao->save($dto);
```

Je demande aussi au controller d'avoir l'obligeance de m'indiquer que l'enregistrement c'est bien déroulé.

```
// Enregistrement du message en variable de session
$message = "L'agent territorial " . $userName . ' a été correctement
enregistré dans la base de données' ;
$_SESSION['flash'] = $message ;
```

Et je lui demande de rediriger vers la vue inscription via son controller.

```
// Redirection
header('location:index.php?controller=inscriptionController');
```

Présentation du Logiciel « Cimetière » à la Mairie

I. Présentation avec un diaporama de façon rétrospective sur la mise au pont du logiciel intranet

La présentation faite le 05 octobre 2016 pour quelques personnes de la mairie, sur ce qui a été déjà produit et le rendu du logiciel a été très apprécié, je dois faire une présentation vendredi 07 octobre 2016 au secrétaire général et à ses agents.

Je pense avoir des modifications que je gérerais par la suite en tant que mise à jour à moins que cela leur pose de réel problème de fonctionnement.

I. Présentation des fonctionnalités du logiciel

I. Démonstrations sur rétroprojecteur de son fonctionnement

Doléance

I. Prise en compte des ajustements

- ➡ Pour le concessionnaire : avoir un encart sur la vue pour le conjoint, ferait gagner du temps au conjoint, qui a autre chose à faire que de patienter dans une marie dans un moment si éprouvant.
- ➡ Connaitre l'état de concessionnaire :
 - Est-il encore de ce monde ?
 - Quel âge a le concessionnaire ?
 - Bloqué l'âge à la date de son décès

II. Prise en compte des futures améliorations

- ➡ La Police municipale est venue me trouver pour me soumettre une amélioration future concernant la procédure de reprise de concession, avec un système de prise de photo et d'enregistrement avec le code de l'emplacement de la sépulture. Cela leur dégagerait un temps précieux, avec un rappel tous les ans jusqu'à la procédure finale de reprise.
- ➡ Cartographier la carte du cimetière et indiqué par couleur les emplacements libres, occupés, en cours de procédure, en cours de nettoyage. Voir même la colorisation pour les termes échus des concessions.
- ➡ Pouvoir choisir l'emplacement de la concession par le futur concessionnaire directement sur tablette.

Formation des Agents territoriaux

I. Formation du secrétaire général

Création d'une notice en version PDF expliquant la navigation dans le menu et les diverses interactions de la section du secrétaire général avec en plus tous les explications de la section des agents territoriaux.

La deuxième formation sera faite par des inscriptions, modifications et suppressions dans les différentes catégories existantes, pour ce faire je mettrais en place une autre base de données similaire pour la formation de tout le personnel.

I. Formation des agents territoriaux

Création d'une notice en version PDF expliquant la navigation dans le menu et les diverses interactions de la section de l'agent territorial.

La deuxième formation sera faite par des inscriptions, modifications et suppressions de futurs factifs concessionnaires, pour ce faire la aussi je mettrais en place une base de données similaire pour la formation de tout le personnel.

La Webographie et la bibliographie

I. La Webographie

- <http://www.php.net/manual/fr/>
- <https://www.w3.org/>
- <https://validator.w3.org/>
- <http://dev.mysql.com/doc/mysql/fr/>
- <https://openclassrooms.com/>
- <http://www.developpez.com/>
- <https://www.alwaysdata.com/fr/>
- <http://www.w3schools.com/>
- <http://getbootstrap.com/>
-

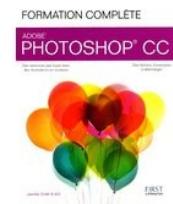
I. La Bibliographie

- Premiers pas en CSS3 et HTML5
De Francis Draillard
- Tout sur le code : Pour concevoir du logiciel de qualité, dans tous les langages
De Steve McConnell
- Formation complète Photoshop CC
De Jennifer SMITH
- Développez votre site web avec le framework Symfony2
D'Alexandre Bacco
- « UML 2 en action : De l'analyse des besoins à la conception »

De Pascal Roques et Franck Vallée

✚ Programmez-en orienté objet en PHP

De Victor Thuillier



II.

III.

L'annexe

I. Le script de la création de base

o La base de données

```
/  
*=====*/  
/* Nom de SGBD : MySQL 5.0 */  
/* Date de création : 01/09/2016 12:04:28 */  
/  
*=====*/
```

```
DROP DATABASE cimetiere  
CREATE DATABASE IF NOT EXISTS cimetiere  
DEFAULT CHARACTER SET utf8  
COLLATE utf8_general_ci;
```

```
USE cimetiere;  
SET FOREIGN_KEY_CHECKS = 0;  
  
drop table if exists CategorieSepulture;  
drop table if exists Commune;  
drop table if exists Concession;  
drop table if exists EtatSepulture;  
drop table if exists JardinDuSouvenir;  
drop table if exists Personne;  
drop table if exists Sepulture;  
drop table if exists TypeDePersonne;  
drop table if exists TypeSepulture;  
drop table if exists User;
```

o Les Tables de la base de données

```
/  
*=====**/  
/* Table : CategorieSepulture */  
/  
*=====**/  
create table CategorieSepulture  
(  
    idCategorieSepulture int not null auto_increment,  
    categorieSepulture varchar(35),  
    primary key (idCategorieSepulture)  
)ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;  
  
/  
*=====**/  
/* Table : Commune */  
/  
*=====**/  
create table Commune  
(  
    idCommune      int not null auto_increment,  
    inseeCommune   char(5),  
    cpCommune      char(5),  
    nomCommune     varchar(50),  
    primary key (idCommune)  
)ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;  
  
/  
*=====**/  
/* Table : Concession */  
/  
*=====**/  
create table Concession  
(  
    idConcession    int not null auto_increment,  
    idSepulture     int not null,  
    idPersonne      int not null,  
    codeConcession   int(5),  
    dateConcession   date,  
    dureeConcession  int(3),  
    tarifConcession  int(5),  
    dateFinConcession date,  
    primary key (idConcession)  
)ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

```

/
*=====
=====
/* Table : EtatSepulture */ 
/
*=====
=====
create table EtatSepulture
(
    idEtatSepulture    int not null auto_increment,
    etatSepulture      varchar(25),
    primary key (idEtatSepulture)
)ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

/
*=====
=====
/* Table : JardinDuSouvenir */ 
/
*=====
=====
create table JardinDuSouvenir
(
    idJardinDuSouvenir  int not null auto_increment,
    primary key (idJardinDuSouvenir)
)ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

/
*=====
=====
/* Table : Personne */ 
/
*=====
=====
create table Personne
(
    idPersonne        int not null auto_increment,
    idCommune         int not null,
    idTypeDePersonne int not null,
    idJardinDuSouvenir int not null,
    civilitePersonne  char(5),
    nomPersonne       varchar(50),
    prenomPersonne   varchar(50),
    vivante          bool,
    telephonePersonne int(10),
    mobilePersonne    int(10),
    emailPersonne     varchar(254),
    dateDecesPersonne datetime,
    dateDeNaissance   datetime,
    agePersonne       int(3),
    primary key (idPersonne)
)

```

)ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

```

/
*=====
=====
/* Table : Sepulture */=====
/
*=====
=====
create table Sepulture
(
    idSepulture      int not null auto_increment,
    idTypeSepulture int not null,
    idConcession     int not null,
    idCategorieSepulture int not null,
    idEtatSepulture int not null,
    sectionSepulture char(2),
    alleeSepulture   char(2),
    numeroDansAllee  int(3),
    code_             int(5),
    nombreSepultures int(5),
    nbreDeCorps       int(1),
    placesLibres      int(1),
    reductionCorps    bool,
    sectionModule     char(2),
    rangCase          int(2),
    numeroCase        int(2),
    primary key (idSepulture)
)ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

/
*=====
=====
/* Table : TypeDePersonne */=====
/
*=====
=====
create table TypeDePersonne
(
    idTypeDePersonne  int not null auto_increment,
    typePersonne       varchar(50),
    primary key (idTypeDePersonne)
)ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

/
*=====
=====
/* Table : TypeSepulture */=====
/
*=====
=====
create table TypeSepulture
(
    idTypeSepulture  int not null auto_increment,
    © Philippe Basuyau -- Stage Développeur Logiciel -- Projet Cimetière --
    Octobre 2016

```

```
typeSepulture      varchar(50),
primary key (idTypeSepulture)
)ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

```

/
=====
=====*/
/* Table : User */
/
=====
=====*/
create table User
(
    idUser      int not null auto_increment,
    identifiant  varchar(25),
    userName     varchar(254),
    password     varchar(254),
    accessLevel   int(3),
    primary key (idUser)
)ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

```

o Les clés étrangères

alter table Concession add constraint FK_ContratSigne foreign key (idPersonne)
 references Personne (idPersonne) on delete restrict on update restrict;

alter table Concession add constraint FK_SepultureConcession foreign key
 (idSepulture)
 references Sepulture (idSepulture) on delete restrict on update restrict;

alter table Personne add constraint FK_Habiter foreign key (idCommune)
 references Commune (idCommune) on delete restrict on update restrict;

alter table Personne add constraint FK_PersonneType foreign key
 (idTypeDePersonne)
 references TypeDePersonne (idTypeDePersonne) on delete restrict on update
 restrict;

alter table Personne add constraint FK_djardin foreign key (idJardinDuSouvenir)
 references JardinDuSouvenir (idJardinDuSouvenir) on delete restrict on update
 restrict;

alter table Sepulture add constraint FK_Categorie foreign key
 (idCategorieSepulture)
 references CategorieSepulture (idCategorieSepulture) on delete restrict on
 update restrict;

alter table Sepulture add constraint FK_Etat foreign key (idEtatSepulture)
 references EtatSepulture (idEtatSepulture) on delete restrict on update
 restrict;

alter table Sepulture add constraint FK_ConcessionSepulture foreign key
 (idConcession)

```
references Concession (idConcession) on delete restrict on update restrict;
alter table Sepulture add constraint FK_Type foreign key (idTypeSepulture)
    references TypeSepulture (idTypeSepulture) on delete restrict on update
restrict;
```

```
SET FOREIGN_KEY_CHECKS = 1;
```

II. Les bibliothèques et Les Modèles

o Lib-controlerModificationAgent.php

```
<?php
/**
 * Created by PhpStorm.
 * User: Philippe_Basuyau
 * Date: 20/09/2016
 * Time: 11:05
 */

/*
 * @param array $data
 */
function tableauUserName( $data ) {

    $x = count($data);

    $html = '';
    $html .='<table class="table table-striped">';

    //--- Les Entêtes
    $html .='<thead>';
    $html .='<tr>';
    $html .='<th>id</th>';
    $html .='<th>Identifiant </th>';
    $html .='<th>Nom Agent</th>';
    $html .='<th>Password</th>';
    $html .='<th></th>';
    $html .='<th></th>';
    $html .='</tr>';
    $html .='</thead>';

    //--- Le Body
    $html .='<tbody>';

    for ($i = 0; $i < $x; $i++) {
        $html .= "<form action='index.php?controller=modifAgentController'
method='POST'>";
        //--- Pour idUsername
        $html .='<tr>';
        $html .='<th>';
        $html .= $data[$i]['idUser'];
        // je cache le champs suivant car je veux pas pouvoir le modifier avec
        :type="hidden"
    }

}
```

```

$html .= '<input type="hidden" value="'.$data[$i]["idUser"].'"'
name="idUser" />';
$html .='</th>';

//--- pour l'identifiant
$html .='<td>';
$html .= $data[$i]['identifiant'];
// je cache le champs suivant car je veux pas pouvoir le modifier avec
:type="hidden"
$html .= '<input type="hidden" value="'.$data[$i]["identifiant"].'"'
name="identifiant" />';
$html .='</td>';

//--- pour les Nom d'Agent
$html .='<td>';
// si je ne veux pas afficher le nom en plus du champs pour la modification
// je commande la ligne suivante
// $html .= $data[$i]['userName'];
$html .= '<input type="text" value="'.$data[$i]["userName"].'"'
name="userName" />';
$html .='</td>';

//--- Pour le password
$html .='<td>';
// si je ne veux pas afficher le nom en plus du champs pour la modification
// je commande la ligne suivante
// $html .= $data[$i]['password'];
$html .= '<input type="text" value="'.$data[$i]["password"].'"'
name="password" />';
$html .='</td>';

//pour le bouton modifier
$html .='<td>';
$html .= "<input type='submit' value='Modifier' name='btModifierAgent'"
id='btModifierAgent' />";
$html .='</td>';

$html .='</tr>'; //fin
$html .= "</form>";//fin

}

$html .='</tbody>';
$html .='</table>';

return $html;
}

© Philippe Basuyau -- Stage Développeur Logiciel -- Projet Cimetière --
Octobre 2016

```

o Lib-controllerSuppressionAgent.php

```
<?php
/**
 * Created by PhpStorm.
 * User: Philippe_Basuyau
 * Date: 20/09/2016
 * Time: 11:05
 */

/*
 * @param array $data
 */
function tableauUserName( $data ) {

    $x = count($data);

    $html = "";
    $html .='<table class="table table-striped">';

    //--- Les Entêtes
    $html .= '<thead>';
    $html .= '<tr>';
    $html .= '<th>id</th>';
    $html .= '<th>Identifiant</th>';
    $html .= '<th>Nom Agent</th>';
    $html .= '<th>Password</th>';
    $html .= '<th></th>';
    $html .= '<th></th>';
    $html .= '</tr>';
    $html .= '</thead>';

    //--- Le Body
    $html .='<tbody>';

    for ($i = 0; $i < $x; $i++) {
        $html .= "<form action='index.php?controller=suppAgentController' method='POST'>";
        //--- Pour idUsername
        $html .= '<tr>';
        $html .= '<th>';
        $html .= $data[$i]['idUser'];
        // je cache le champs suivant car je veux pas pouvoir le modifier avec :type="hidden"
        $html .= '<input type="hidden" value="'. $data[$i]["idUser"] .'" name="idUser" />';
        $html .= '</th>';

        //--- pour l'identifiant
        $html .= '<td>';
        $html .= $data[$i]['identifiant'];

    }

}
```

```

// je cache le champs suivant car je veux pas pouvoir le modifier avec
:type="hidden"
$html .= '<input type="hidden" value="'. $data[$i]["identifiant"].'"'
name="identifiant" />';
$html .='</td>';

//--- pour les Nom d'Agent
$html .='<td>';
// si je ne veux pas afficher le nom en plus du champs pour la modification
// je commande la ligne suivante
$html .= $data[$i]['userName'];
//    $html .= '<input type="text" value="'. $data[$i]["userName"].'"'
name="userName" />';
$html .='</td>';

//--- Pour le password
$html .='<td>';
// si je ne veux pas afficher le nom en plus du champs pour la modification
// je commande la ligne suivante
$html .= $data[$i]['password'];
//    $html .= '<input type="texte" value="'. $data[$i]["password"].'"'
name="password" />';
$html .='</td>';

// pour le bouton Suppression de droit d'agent territorial
$html .='<td>';
$html .= "<input type='submit' value='Supprimer' name='btDeleteAgent' id='btDeleteAgent' />";
$html .='</td>';

$html .='</tr>'; //fin
$html .= "</form>";//fin

}

$html .='</tbody>';
$html .='</table>';

return $html;
}

```

J'aurais pu inclure c'est deux lib-controller dans un seul mais pour le premier je voulais bien décortiquer le principe du code, les prochains seront réunies en un seul et unique car c'est le but ne pas réécrire un morceau de code déjà écrit.

III. Les DAO – DTO – IDAO

o DAO User

```
<?php

class UserDAO implements IUserDAO {

    /**
     * @var \PDO
     */
    private $pdo;

    /**
     * DAOClient constructor.
     * @param PDO $pdo
     */
    public function __construct(PDO $pdo) {
        $this->pdo = $pdo;
    }

    public function findAll() {
        $sql = "SELECT * FROM user";
        $rs = $this->pdo->query($sql)->fetchAll();
        return $rs;
    }

    public function findOneById(array $pk) {
        $sql = "SELECT * FROM user WHERE idUser=? ";
        $statement = $this->pdo->prepare($sql);
        $statement->execute($pk);
        $rs = $statement->fetch();
        return $rs;
    }

    public function findById($pkValue) {
        $sql = "SELECT * FROM user WHERE idUser=?";
        $statement = $this->pdo->prepare($sql);
        $statement->execute(array($pkValue));
        $rs = $statement->fetch();
        return $rs;
    }
}
```

```

public function find(array $search) {
    $sql = "SELECT * FROM user ";

    if (count($search) > 0) {
        $sql .= " WHERE ";
        $cols = array_map(
            function($item) {
                return "$item=:{$item}";
            }, array_keys($search)
        );

        $sql .= implode(" AND ", $cols);
    }

    $statement = $this->pdo->prepare($sql);
    $statement->execute($search);

    return $statement->fetchAll(PDO::FETCH_ASSOC);
}

public function save(UserDTO $user) {
    if ($user->getIdUser() == null) {
        return $this->insert($user);
    } else {
        return $this->update($user);
    }
}

private function insert(UserDTO $user) {
    $sql = "INSERT INTO user (identifiant, userName, password, accessLevel)
VALUES ( ?, ?, ?, ? )";
    $statement = $this->pdo->prepare($sql);
    $statement->execute([
        $user->getIdentifiant(), $user->getUserName(), $user->getPassword(),
        $user->getAccessLevel()
    ]);
}

private function update(UserDTO $user) {
    $sql = "UPDATE user SET identifiant=?, userName=?, password=?,
accessLevel=? WHERE idUser=? ";
    $data = array(
        $user->getIdentifiant(), $user->getUserName(), $user->getPassword(),
        $user->getAccessLevel(),
        $user->getIdUser()
    );
    $statement = $this->pdo->prepare($sql);
    return $statement->execute($data);
}

```

```

public function delete(UserDTO $user) {
    if ($user->getIdUser() != null) {
        $sql = "DELETE FROM user WHERE idUser=? ";
        $statement = $this->pdo->prepare($sql);
        return $statement->execute([$user->getIdUser()]);
    }else{
        return "Problème d'argument";
    }
}

```

o DTO User

<?php

```

class UserDTO
{

    private $idUser;
    private $identifiant;
    private $userName;
    private $password;
    private $accessLevel;

    public function __construct($idUser="", $identifiant="", $userName="",
$password="", $accessLevel="") {
        $this->idUser = $idUser;
        $this->identifiant = $identifiant;
        $this->userName = $userName;
        $this->password = $password;
        $this->accessLevel = $accessLevel;
    }

    public function setIdUser($idUser){
        $this->idUser = $idUser;
    }

    public function getIdUser(){
        return $this->idUser;
    }

    public function setIdentifiant($identifiant){
        $this->identifiant = $identifiant;
    }
}

```

```

public function getIdentifiant(){
    return $this->identifiant;
}

public function setUserName($userName){
    $this->userName = $userName;
}

public function getUserName(){
    return $this->userName;
}

public function setPassword($password){
    $this->password = $password;
}

public function getPassword(){
    return $this->password;
}

public function setAccessLevel($accessLevel){
    $this->accessLevel = $accessLevel;
}

public function getAccessLevel(){
    return $this->accessLevel;
}
}

```

o IDAO user

```

<?php

interface IUserDAO {
    public function findAll();
    public function findOneById(array $pk);
    public function find(array $search);
    public function delete(UserDTO $user);
    public function save (UserDTO $user);
}

```

o Les TEST DAO et DTO

UserDAOTest.php

```
<?php
/**
 * Created by PhpStorm.
 * User: Philippe-Basuyau
 * Date: 16/09/2016
 * Time: 11:08
 */

/*
 * UserDAOTests.php
 */
require_once '../conf/config.php';
require_once '../lib/lib-pdo.php';
require_once '../class/DAO/UserDTO.php';
require_once '../class/DAO/UserDAO.php';

//define("DSN", "mysql:host=localhost;port=3006;dbname=cimetiere");
//define("DB_USER", "root");
//define("DB_PASS", "");

$connexion = getPDO();

$dao = new UserDAO($connexion);

$u2 = new UserDTO("3", "U3", "mdp333");

// $liAffecte = $dao->save($u2);
// echo "<br>";
// echo "<pre>";
// var_dump($liAffecte);
// echo "</pre>";
// echo "<br>";

echo "<br>***** FIND ALL *****";
// OK
$t = $dao->findAll();

echo "<br>";
echo "<pre>";
var_dump($t);
echo "</pre>";
echo "<br>";

echo "<br>***** FIND ONE BY ID*****";
// OK
$to = array("1");
$o = $dao->findOneById($to);

echo "<br>";
echo "<pre>";
var_dump($o);
```

```

echo"</pre>";
echo"<br>";

echo "<br>***** FIND BY ID *****";
// OK
$o = $dao->findById(1);

echo"<br>";
echo"<pre>";
var_dump($o);
echo"</pre>";
echo"<br>";

echo "<br>***** FIND *****";
// OK

$ta = array("idUser" => "3");
$o = $dao->find($ta);

echo"<br>";
echo"<pre>";
var_dump($o);
echo"</pre>";
echo"<br>";

echo "<br>***** DELETE *****";
// OK

$u = new UserDTO(2, "", "");
// $u = new UserDTO();
$o = $dao->delete($u);

echo"<br>";
echo"<pre>";
var_dump($o);
echo"</pre>";
echo"<br>";

echo "<br>***** INSERT *****";
// OK mais renvoie NULL donc a améliorer

$u = new UserDTO();
$u->setUserName("Nouveau");
$u->setPassword("mdp");
$o = $dao->save($u);

echo"<br>";
echo"<pre>";
var_dump($o);
echo"</pre>";
echo"<br>";


echo "<br>***** UPDATE *****";
// OK mais renvoie NULL ou boolean donc a améliorer

```

© Philippe Basuyau -- Stage Développeur Logiciel -- Projet Cimetière -- Octobre 2016

```
$u = new UserDTO(4, "Nouveau", "mdp112");
$o = $dao->save($u);

echo"<br>";
echo"<pre>";
var_dump($o);
echo"</pre>";
?>
```

UserDTO.php

```
<?php
/**
 * Created by PhpStorm.
 * User: Philippe-Basuyau
 * Date: 16/09/2016
 * Time: 11:04
 */

/**
 * UserDTOTest pour savoir si j'envoie bien
 */
require_once '../class/DAO/UserDTO.php';

$u = new UserDTO();

$u->setIdUser(1);
$u->setIdentifiant(fifi);
$u->setUserName("Philippe");
$u->setPassword("mdp123");

echo "<br>", $u->getIdUser();
echo "<br>", $u->getIdentifiant();
echo "<br>", $u->getUserName();
echo "<br>", $u->getPassword();

$u2 = new UserDTO("2", "Muddy water", "Marcy gray", "mdp123");
echo "<br>", $u2->getIdUser();
echo "<br>", $u2->getIdentifiant();
echo "<br>", $u2->getUserName();
echo "<br>", $u2->getPassword();
?>
?>
```