Guillaume Cothenet  
Yannis Mahiou

# **Description de l’UML**

**Spots** : Spots est une classe contenant un dictionnaire qui a comme clé une Region et comme valeur une liste de Lieu. Elle possède deux méthodes : son constructeur et « AddSpots » qui permet d’ajouter une clé ainsi que sa valeur dans le dictionnaire. Elle utilise les méthodes de la classe XML ou stub afin de charger ou sauvegarder ses données. Nous avons décidé de regrouper nos spots dans un dictionnaire afin de faciliter leur « rangement » (chaque région a une liste de lieu) et on peut donc choisir facilement des lieux à proximité de l’endroit où l’on se trouve puisqu’ils sont « rangé » par région.

**Region** : Region est une classe contenant un nom, une description ainsi qu’un lien vers une image. Elle possède deux méthodes : son constructeur et une redéfinition de la méthode « ToString ». Nous avons choisi cette classe comme clé de notre dictionnaire puisqu’il nous paraissait logique de regroupé les lieux en fonction de leur endroit.

**Lieu** : Lieu est une classe contenant un nom, une note, une description, un lien vers une image et une liste d’avis. Elle possède 5 méthodes : son constructeur, AddAvis (qui permet d’ajouter un avis à sa liste), DeleteAvis (qui permet de supprimer un avis de sa liste), CalculNote (qui permet de faire une moyenne des notes ajoutées par les utilisateurs) et une redéfinition de « ToString ». C’est en quelque sorte le cœur de notre application puisque c’est ce que l’on présente dans notre détail.

**Avis** : Avis est une classe contenant une note, un commentaire, un pseudo et une date. Elle possède 3 méthodes : son constructeur, ChangerAvis (qui permet de changer un avis que l’on a mis) et une redéfinition de « ToString ». Cette classe est venue après la réalisation de la connexion, il nous paraissait logique d’ajouter un avantage aux utilisateur qui avait un compte.

**User** : User est une classe contenant un pseudo, un mot de passe et un mail. Elle possède 2 méthode : son constructeur et une redéfinition de « ToString ». Nous l’avons créé pour instaurer une sorte de hiérarchie entre les utilisateurs et pouvoir ajouter des fonctionnalités disponibles ou non en fonction de notre « grade ».

**AllUser** : AllUser est une classe contenant une liste d’User. Elle possède 5 méthodes : son constructeur, ResearchUser (qui prend en paramètre un string et qui return un User), AddUser (qui permet d’ajouter un utilisateur à sa liste), AddAdmin (qui permet d’ajouter un Admin à sa liste) et DeleteUser (qui permet de supprimer un User). Elle utilise les méthodes de la classe XML ou stub afin de charger ou sauvegarder ses données. Cette classe a été créée afin de stocker tous les utilisateurs de l’application et pouvoir parcourir cette liste à notre guise, pour d’autre méthodes.

**Connexion** : Connexion est une classe contenant un username et un password. Elle possède 2 méthodes : son constructeur et SeConnecter (qui permet de se connecter à l’application). Cette classe nous permet de savoir si un utilisateur est connecté ou non.

**Admin** : Admin est une classe fille d’User, contenant une description. Elle possède 2 méthodes : son constructeur et une redéfinition de « ToString ». On a décidé de la créer afin de distinguer le simple utilisateur et l’admin. Elle nous permet aussi de rajouter certaines fonctionnalités accessibles uniquement aux Admin.

**Article** : Article est une classe contenant un titre, un texte, une date et un Admin. Elle possède 2 méthodes : son constructeur et Modifier (qui permet de modifier un article). On a créé cette classe afin de regrouper tous les textes que l’on voulait afficher dans notre application.

**Topic** : Topic est une classe contenant un nom, un Admin et une liste d’article. Elle possède 2 méthodes : son constructeur et AddArticle (qui permet d’ajouter un article à sa liste). Elle utilise les méthodes de la classe XML ou stub afin de charger ou sauvegarder ses données. Cette classe nous permet d’afficher une liste d’article dans une listebox (par exemple) avec un simple binding sur le Topic en question.

**XML** : XML est une classe qui implémente les interfaces ILoad et ISave. Elle possède 4 méthodes : SaveData (qui permet de serializer toutes les données, LoadUser (qui permet de charger la liste d’utilisateur), LoadSpot (qui permet de charger le dictionnaire de Spots) et LoadTopic (qui permet de charger une liste de Topic. Nous avons créé cette classe afin de sauvegarder dans des fichier les données de l’application.

**Stub** : Stub est une classe qui implémente l’interface ILoad. Elle possède 3 méthodes : LoadUser (qui permet de charger la liste d’utilisateur), LoadSpot (qui permet de charger le dictionnaire de Spots) et LoadTopic (qui permet de charger une liste de Topic. Cette classe à été créé afin de « remplir » l’application de données lors des différents tests effectués.

**ILoad** : ILoad est une interface qui a comme méthode LoadUser, LoadSpot et LoadTopic. Elle permet aux classes XML et Stub de pouvoir deserializer les données utilisées pour le projet.

**ISave** : ISave est une interface qui a comme méthode SaveData(). Elle permet à la classe XML de pouvoir serializer dans des fichiers .xml les données du projet.