***Chasse, Pêche et Tradition***

* **Contexte**

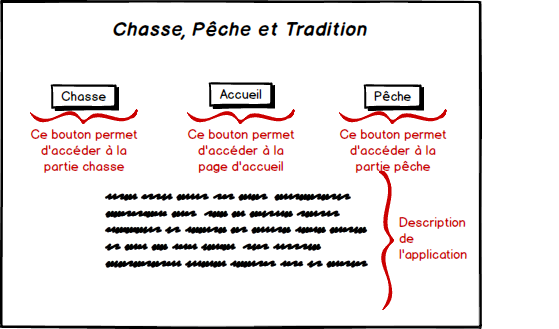
Si comme nous vous êtes mordu de chasse ou de pêche alors cette application est faite pour vous. En effet elle va vous permettre de consulter la liste des animaux (que ce soit pour la chasse comme pour la pêche) en fonction d’une zone.

Prenons un exemple, vous désirez savoir quels gibiers vous pouvez chasser en Auvergne, pas de soucis, vous choisissez l’Auvergne dans les différentes zones sélectionnables puis la liste des animaux chassables sur le territoire Auvergnat apparaît. Même chose si vous cherchez la liste des poissons présents en Auvergne.

*Chasse, Pêche et Tradition* s’adresse essentiellement à un public amateur de chasse et de pêche même si d’autres personnes qui souhaitent tout simplement obtenir des renseignements sur un animal peuvent utiliser notre application.

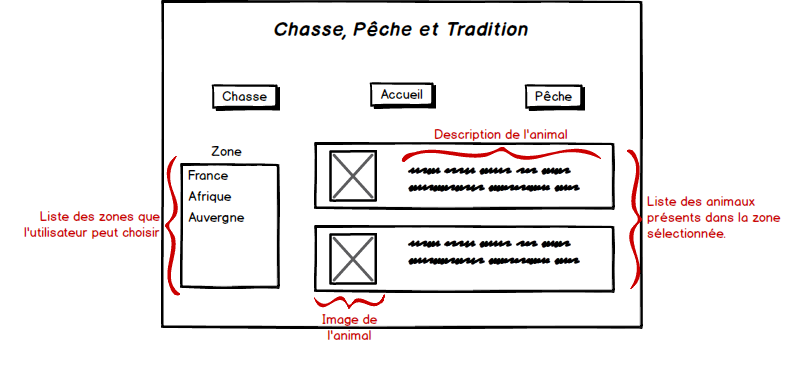
Comme décris auparavant, l’application permet d’accéder à une liste d’animaux situés dans une zone. Elle sera décomposée en deux parties, la partie chasse et la partie pêche. Mais ces deux parties fonctionnent de la même façon. En effet, l’utilisateur aura accès à une liste de zone et lorsqu’il choisit une zone, une liste des animaux présents dans la zone sera à sa disposition. Pour chaque animal, l’utilisateur pourra voir le nom commun, le nom scientifique, une image ainsi qu’un descriptif.

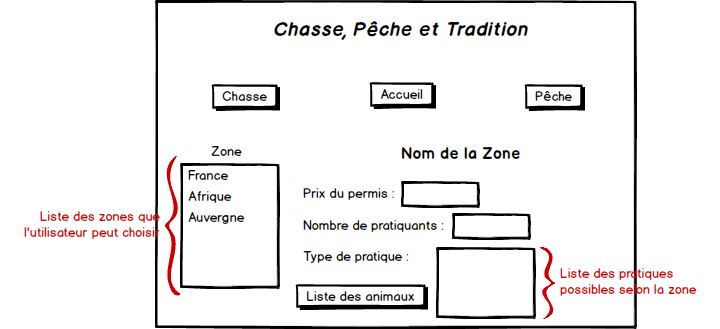
* **Sketch**

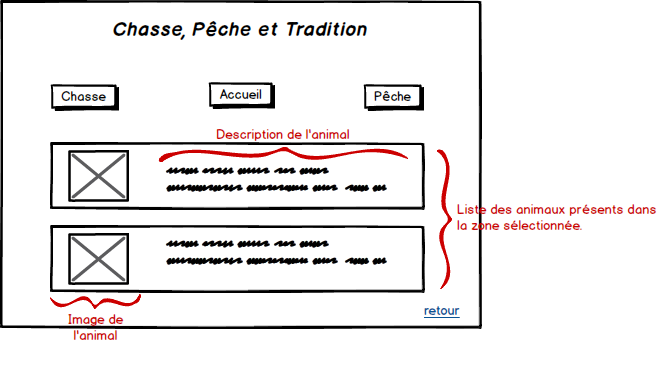


.

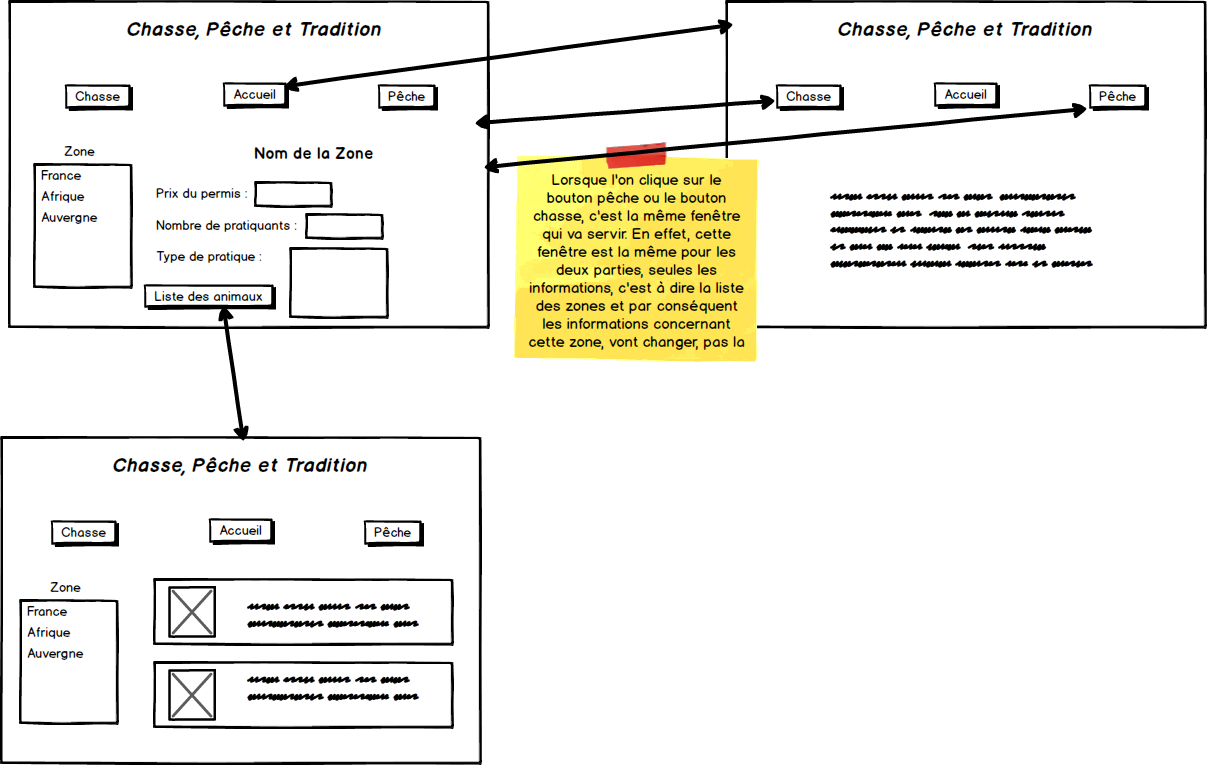
Le sketch ci-dessus représente la page d’accueil de notre application. Cette page est très simple. En effet, on y trouve le nom de l’application en haut et ensuite on trouve trois boutons : Le bouton chasse et le bouton pêche qui permettent d’accéder à la partie correspondante. Ensuite comme son nom l’indique le bouton accueil permet de retourner sur notre page d’accueil c'est-à-dire celle qui est présentée ici. Ce bandeau va rester le même pour toutes les pages seule la partie basse va changer par le biais de contrôle utilisateur. Pour l’accueil, la partie basse de la fenêtre se compose d’un petit texte qui sert de descriptif pour notre application.

Lorsque l’utilisateur va cliquer sur un des boutons Chasse ou Pêche, la page présentée ci-dessous va s’ouvrir.

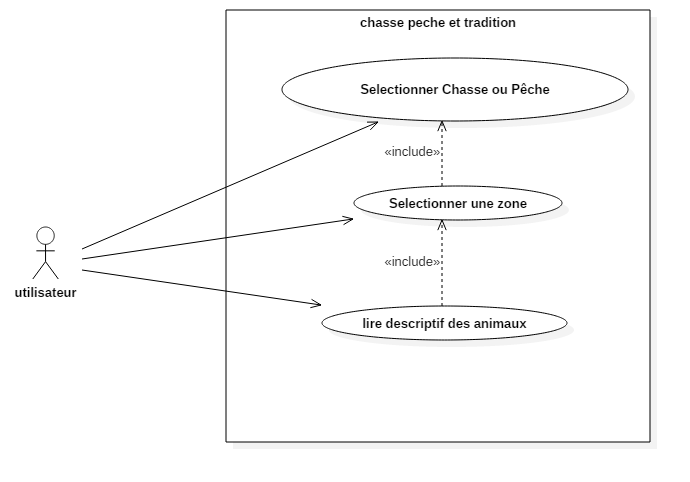


Cette page va être identique selon que l’utilisateur clique sur le bouton Chasse ou sur le bouton pêche. En fonction de la zone choisit dans la liste de gauche, on verra le prix du permis dans cette zone ainsi que le nombre de pratiquants et les types de pratiques. De plus on trouve un bouton qui permet d’accéder à la liste des animaux présents sur la zone. En cliquant sur ce bouton on arrive sur la page suivante.

Cette page est composée d’une liste des animaux présents dans la zone sélectionnée. On trouve les informations pour chaque animal ainsi qu’une photo.

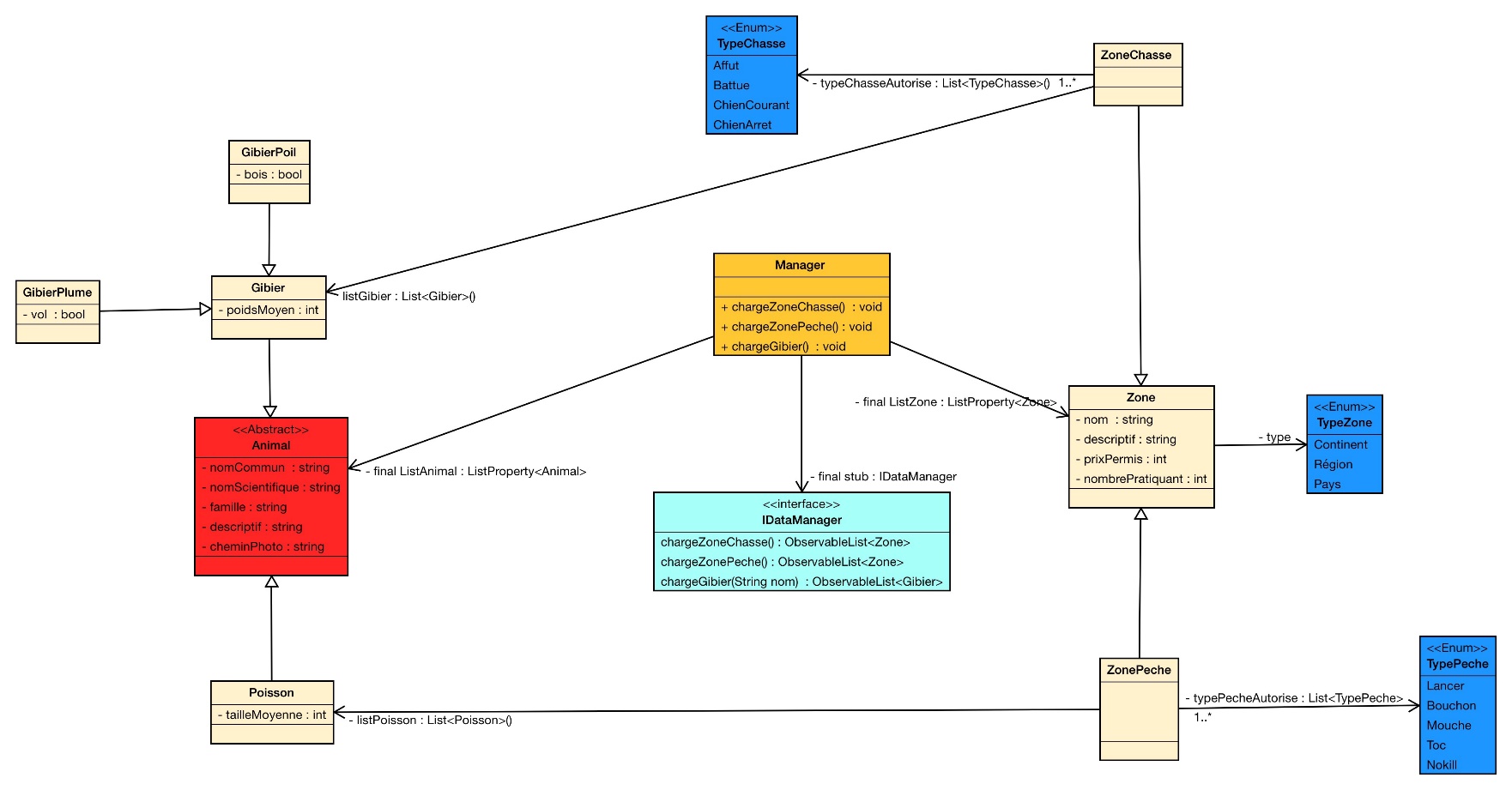


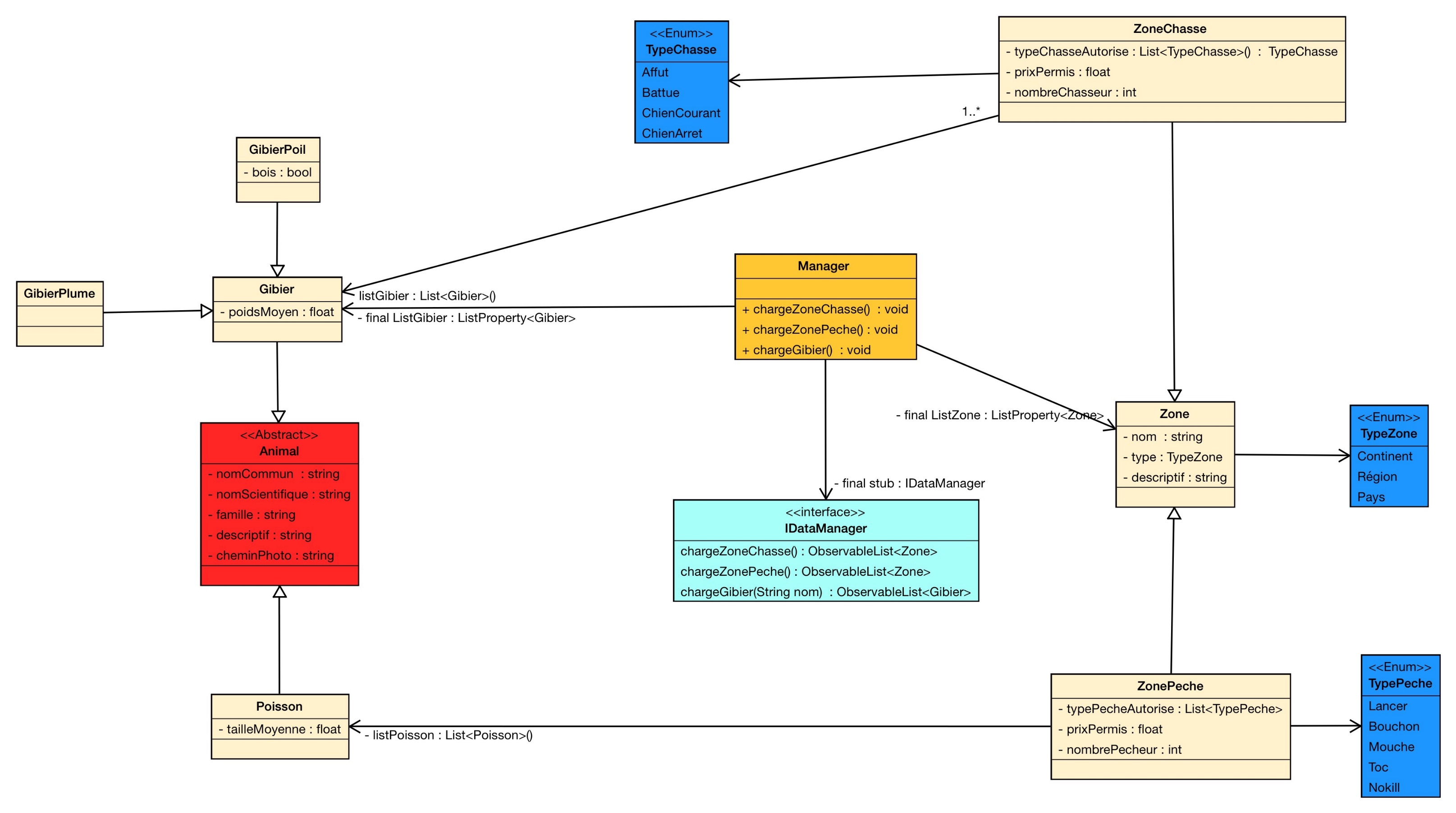
La figure ci-dessus nous montre le story-board de l’application, c'est-à-dire l’interaction entre les vues.

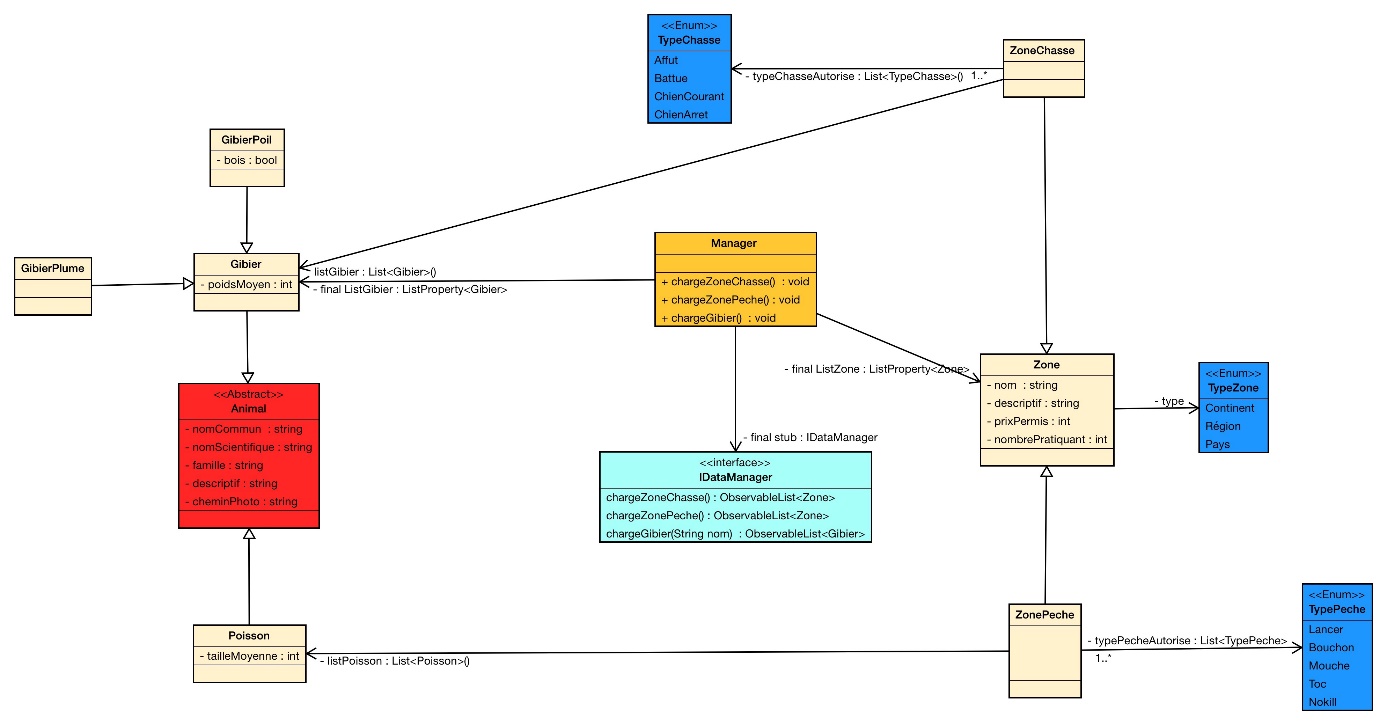
* **Diagramme de cas d’utilisation**

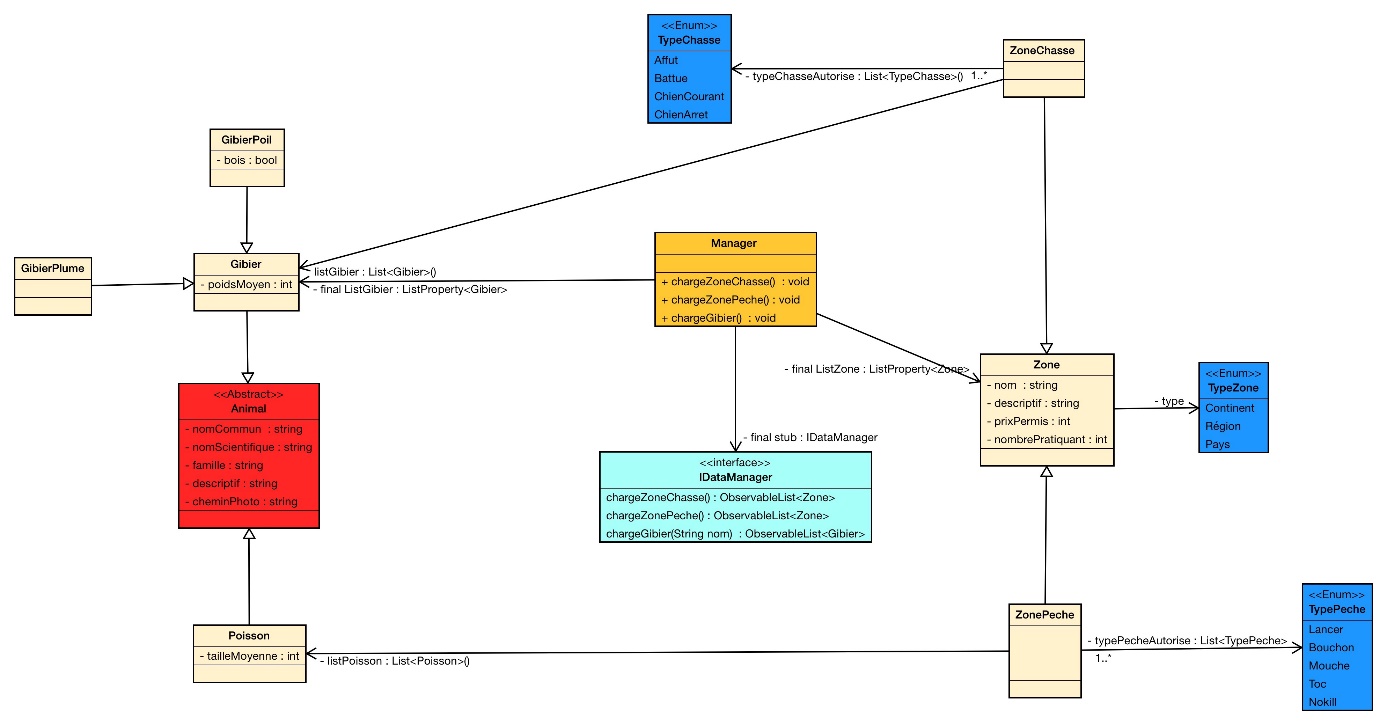
La figure ci-dessus correspond à notre diagramme de cas d’utilisation. On a ici un diagramme très simple car comme nous l’avons précisé précédemment, nous allons commencer par mettre en place une application très simple. Ainsi par la suite ce diagramme pourra être emmené à être modifiée pour ajouter les nouvelles fonctionnalités de l’application.

Ainsi, l’utilisateur peut choisir entre les deux parties de l’application, soit la chasse, soit la pêche. Une fois qu’il a fait cela, il peut sélectionner une zone et ensuite il pourra lire le descriptif des animaux. C’est donc pour cela qu’il y a des relations d’inclusion car on ne peut pas lire un descriptif sans avoir choisit une zone et on ne peut pas sélectionner une zone sans avoir sélectionné une des deux parties.

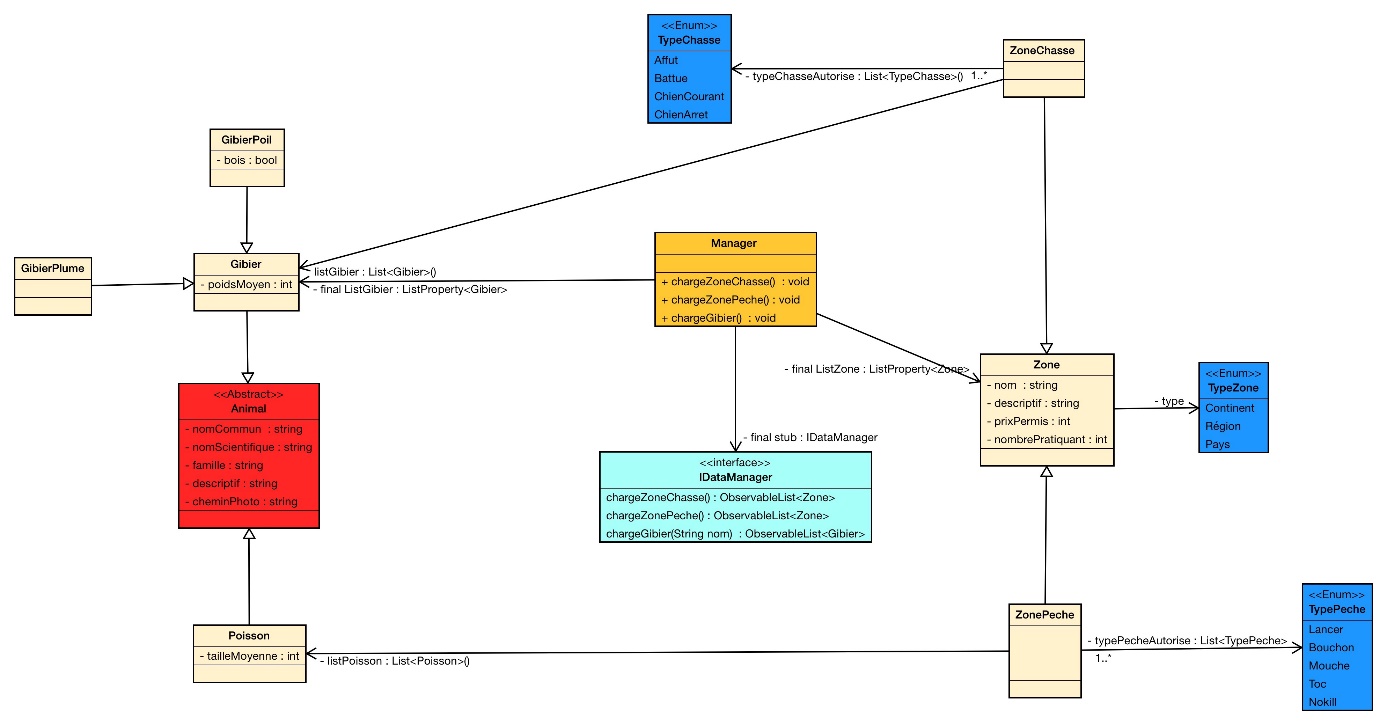
* **Diagramme de classe**

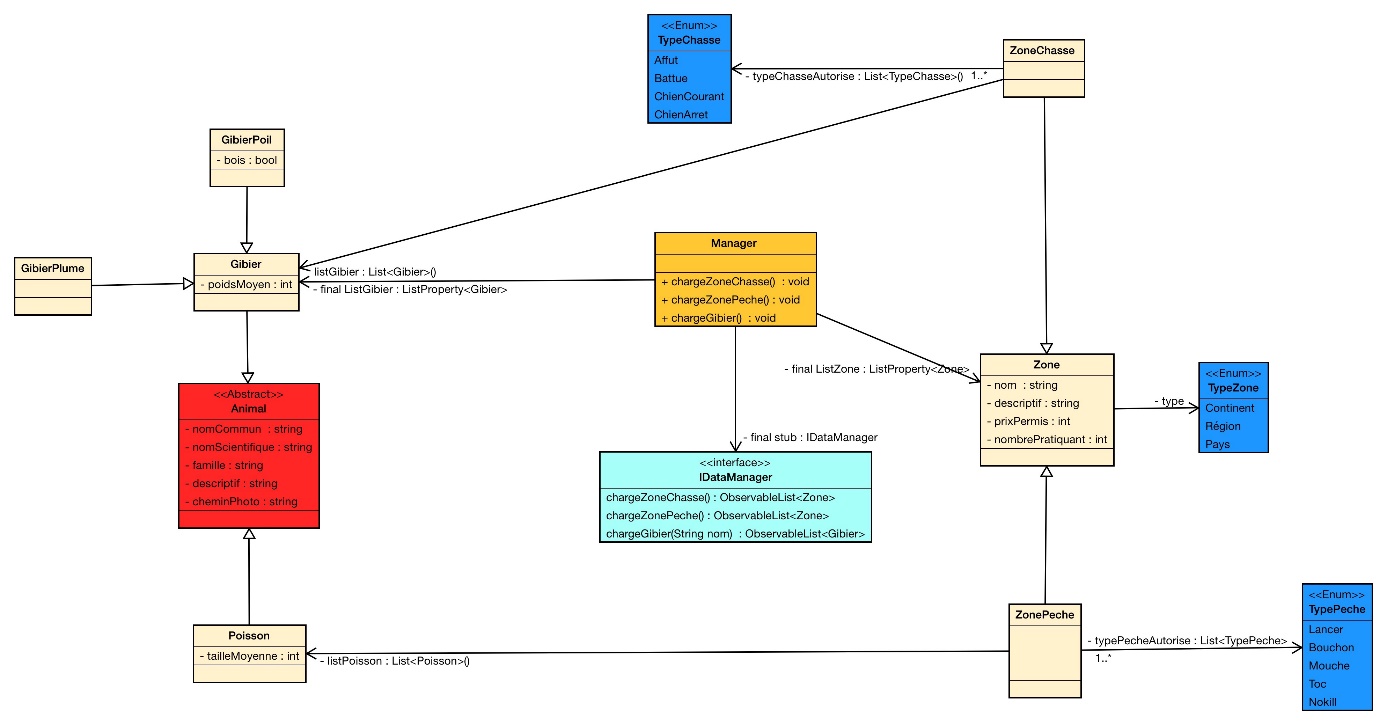
******Ci-dessus, voici le diagramme de classe de notre application. Nous allons donc le détailler et l’expliquer.

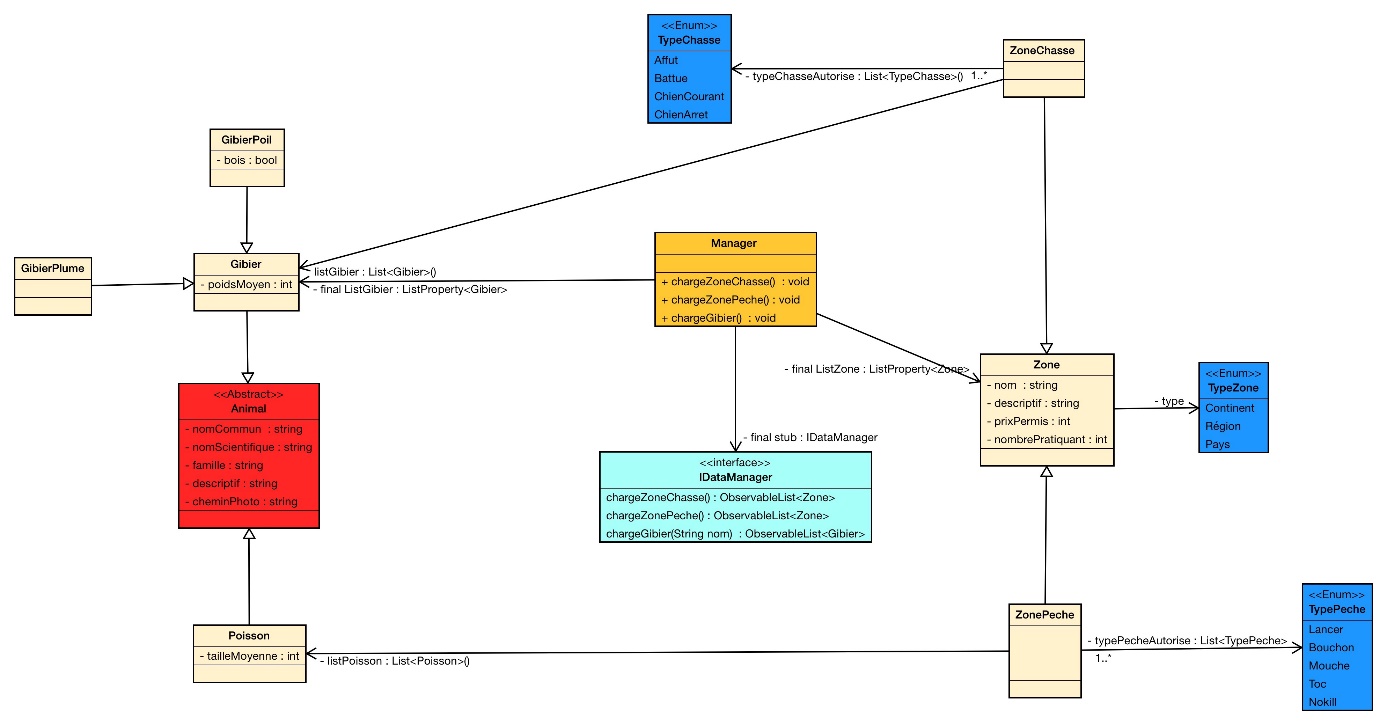
* Pour commencer, voici la classe la plus importante de notre application : la classe Manager. En effet, cette dernière sert de façade (patron de conception) entre le package Métier et le package Vue. Ainsi le manager va faire la relation entre le métier et la vue, c’est-à-dire que les informations du métier qui devront être transmises à la vue passeront par le Manager. On remarque que le Manager possède des relations avec les autres classes par le biais de listes et qu’il possède trois méthodes : ces dernières permettent de charger les listes précisées auparavant. Ainsi la méthode chargeZoneChasse() va charger la liste de zone avec toutes les zones de chasse que l’application possède. Tandis que la méthode chargeGibier() va charger la liste d’animal selon la zone sélectionnée. De même il y a une méthode chargePoisson() qui charge la liste de poisson selon la zone. Le Manager possède aussi un attribut du type IDataManager. IDataManager est une interface qui correspond à la persistance de l’application et qui peut être implémentée par les classes qui correspondent au chargement de donnée. Par exemple dans notre application, on trouve la classe StubDataManager, dans le package Persistance, qui implémente l’interface IDataManager. Cette classe contient les données de l’application en brut. Le Manager est instancié dans le package vue et il est utilisé par chaque vue pour le binding.
* La classe Zone correspond aux différentes zones proposées par l’application. Elle est composé d’un nom, d’un descriptif et d’un type de Zone qui est une énumération des cas que l’on peut trouver (Par exemple une zone peut être un pays.). On trouve aussi le prix du permis correspondant à la zone ainsi que le nombre de pratiquant. Cette classe Zone est la classe mère de deux autres classes : la classe ZoneChasse et la classe ZonePeche que l’on va décrire ci-dessous.



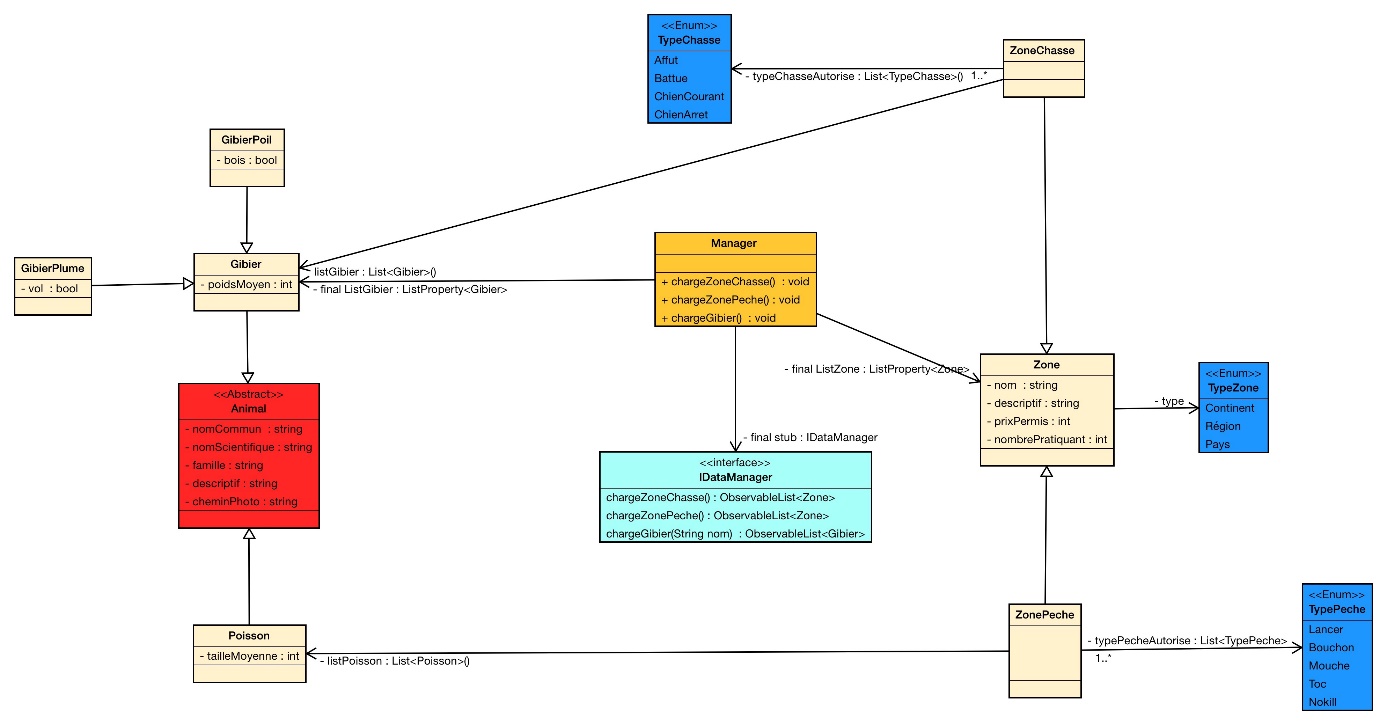
* Comme dit précédemment, la classe ZoneChasse hérite de la classe Zone. Elle est plus spécifique car elle donne plus d’information. En effet, elle a pour attribut, une liste de type de chasse autorisé. Le type de chasse vient d’une énumération reprenant tous les types de chasse que l’application propose. De plus la classe ZoneChasse possède une liste de Gibier.



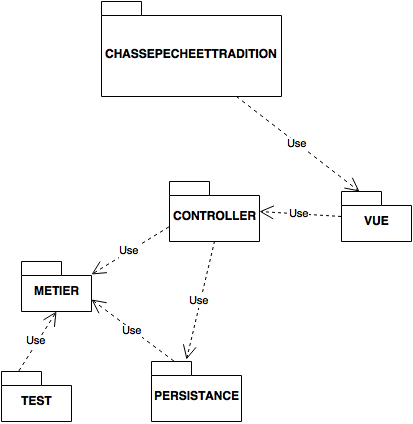
* La classe ZonePeche est similaire à la classe ZoneChasse, seul les informations changent c’est-à-dire que l’on trouve une liste des pratiques de pêche autorisées à la place des types de chasse autorisé. Cette classe possède une liste de Poisson.
* Nous allons maintenant parler de la classe Animal. C’est une classe abstraite qui correspond aux animaux de notre application. Elle est abstraite car on ne peut pas faire d’instanciation. En effet, il faut que l’instanciation d’un animal soit plus précise, c’est-à-dire que l’on fasse la différence entre les poissons et les gibiers. C’est pour cette raison que deux classes vont hériter de celle-ci : La classe Gibier et la classe Poisson. Ces deux classes ont des attributs en commun que l’on retrouve dans la classe Animal : un nom commun, un nom scientifique, une famille, un descriptif et un chemin menant à une photo de l’animal.



* La classe Poisson hérite de la classe Animal et à pour attribut une taille moyenne.

****

* La partie Gibier est plus complexe que la partie Poisson car elle est composée de la classe Gibier qui hérite de la classe Animal mais qui a deux filles : la Classe GibierPlume pour les gibiers à plumes (Bécasse, faisan …) et la classe GibierPoil pour les gibiers à poils (sanglier, cerf …). Un gibier a un poids moyen, un gibier à plumes peut savoir voler alors qu’un gibier à poils peut avoir des bois.
* **Diagramme de Paquetage**

****

Comme on peut le distinguer sur la figure ci-dessus notre projet est constitué de 6 paquetages. Nous allons maintenant décrire quel est de rôle de chaque paquetage.

1. Métier

Le paquetage Métier constitue l’épine dorsale du projet, en effet comme on peut le voir sur le diagramme quasiment tous les autres package l’utilisent. Le métier contient toute les classes qui constituent l’application, c’est à partir de là que sont créés tous ses objets. Il contient aussi toutes les méthodes permettant d’agir sur ces objets (les modifier, leurs ajouter quelque chose...)

1. Vue

Le Paquetage Vue contient lui les différents éléments graphiques de l’application. Ainsi on retrouve dans la vue les différents Contrôles Utilisateurs et page de base. La vue utilise le package Controller car chaque vue a un contrôleur qui lui correspond et qui contient les définitions des méthodes pour les boutons par exemple.

1. Test

Le Paquetage Test permet comme son nom l’indique d’effectuer des tests sur le métier afin de vérifier que tous les éléments du métier (attributs et méthodes) fonctionnent correctement et ne génèrent pas d’exception et que les différentes méthodes effectuent bien la ou les actions que l’on souhaite.

1. Persistance

Le Paquetage Persistance contient les éléments de sauvegarde et de chargement de l’application. En effet ce package permet à chaque utilisation de charger les éléments du métier (obtenir la dernière sauvegarde faite sur les éléments du métier) et après l’utilisation, de sauvegarder ces éléments.

1. Controller

Le Paquetage Controller contient tous les contrôleurs qui permettent le contrôle des vues correspondantes. De plus, ce package utilise le package métier pour avoir accès à la façade c’est-à-dire le Manager et donc pouvoir utiliser les attributs de celui-ci. Ainsi, un Manager est instancié dans le contrôleur de la page principal. Mais il faut aussi instancier une classe de la persistance pour que l’application est accès aux données. C’est pour cette raison que le package Controller utilise le package Persistance.

Le métier possède une classe Manager qui est ce que l’on nomme la façade (patron de conception nommé « Façade »). On utilise ce patron de conception car il permet de donner l’accès au métier par le biais d’une seule classe. Cette classe possède un attribut de type IDataManager. Cette interface impose l’utilisation de méthodes spécifiques permettant la sauvegarde et la lecture des données. Cette interface est implémentée par les classes de la persistance. Ceci est le patron de conception « stratégie ». On utilise ce patron de conception car de cette manière on peut facilement changer le type de Persistance. Par exemple, si on veut que la persistance soit de la persistance binaire alors la classe gérant cette persistance implémente l’interface IDataManager et il suffit de modifier lors de la création du Manager l’attribut de type IDataManager.

* **Perspectives**

Dans un second temps notre application peut être amenée à évoluer. En effet, nous avons pensé à plusieurs autres fonctionnalités ou alors une évolution des fonctionnalités déjà présentes. Pour commencer, on a pensé à l’intégration d’une GoogleMap pour la sélection des zones qui pourrait aussi servir à la création d’une nouvelle zone qui sera délimité par des points sur la GoogleMap. Ensuite, on a pensé à une partie qui permettrait à un utilisateur de se connecter et de saisir ses prélèvements dans une zone. Mais n’ayant pas beaucoup de temps pour la réalisation de ce projet et souhaitant réaliser une application de qualité et qui fonctionne parfaitement, nous allons d’abord laisser de côté ces idées. Mais s’il nous reste du temps nous feront une deuxième version en intégrant les nouveautés.