TDm4

Club d'ornithologie

La présidente d'un club d'ornithologie fait appel à vous pour concevoir et développer un système de gestion d'observation. Elle souhaite avant tout avoir un système robuste et bien documenté qui résistera au temps et qui pourra être adapté dans le futur. En effet, le système informatique précédent, qui avait été développé par le fils d'un ex-membre de l'association, c'est brutalement arrêter de fonctionner et, par manque de documentation adéquate, n'a pas pu être réparé. Ne sachant pas non plus comment et où les données sur les observations étaient stockées, l'association a perdu des années de travail précieux sur le suivi des populations d'oiseaux dans la région.

Le système devra permettre aux adhérents de l'association de réaliser des observations. C'est-à-dire de remplir des fiches d'informations avec le nom de l'espèce observé, le nombre d'oiseaux et l'endroit où ils ont été vus. Les adhérents doivent également avoir accès à une carte géographique de la France sur laquelle ils peuvent visualiser toutes les données recueillies par les membres de l'association. Cette carte devra être paramétrable afin de filtrer les données par type d'oiseau, par année et aussi par observateur (pour voir uniquement ses propres observations).

Grâce à des financements de laboratoires de recherche, obtenus pour leur mission d'étude démographique sur les oiseaux en France, l'association emploie 1 personne à temps plein. Cet employé est principalement chargé de valider les données des observations. En effet, pour que les données puissent être traitées automatiquement, elles doivent respecter un certain format : le nom de l'espèce doit correspondre à un nom répertorié dans la liste officielle, le lieu de l'observation doit contenir un numéro de département valide et, afin d'assurer la traçabilité des données recueillies, l'observateur doit également être identifié, sans ambigüité, comme un membre de l'association. Finalement, cet employé doit également avoir accès à la base de données complète des observations afin de les modifier, supprimer ou ajouter manuellement si besoin est.

L'association utilise déjà une application Web pour la gestion de l'association. La liste des adhérents est stockée dans une table de base de données avec les données suivantes :

Id :int	Nom :String	Prénom :String	CarteAdherentAjour :Boolean

La liste des espèces d'oiseau répertoriée, quant à elle, est stockée au format XML sur le serveur du laboratoire de recherche. Chaque espèce à un identifiant et une dénomination latine que les adhérents ne connaissent pas toujours avec exactitude, d'où la nécessité de faire valider les saisies et de les modifier manuellement si besoin.

Organisation du TDm

Afin de faciliter votre travail de modélisation en trois phases, de plus en plus détaillées, il est fortement conseillé de copier le projet Omondo après chaque grande phase afin de pouvoir récupérer et modifier les diagrammes précédents si besoin est.

Vous pouvez vous aider des diagrammes fournis dans le diaporama du cours pour commencer, mais ils ne sont pas complets et peuvent être grandement améliorés.

À la fin du TDm vous devez rendre, sur l'espace UMTICE, un dossier zippé complet et organisé avec les exportations des différents diagrammes. Libre à vous de nommer les fichiers judicieusement pour qu'on comprenne leurs relations.

Phase 1 – Analyse des besoins

Avant de commencer, lisez attentivement la description du système ci-dessus et soulignez les mots qui correspondent aux fonctionnalités et acteurs importants. Ce sont ses éléments que vous allez utiliser pour faire les diagrammes de cette 1ère phase d'analyse des besoins.

1.1. Modélisez par un **diagramme de cas d'utilisation** le système informatique pour cette association d'ornithologie.

1.2. Pour chaque cas d'utilisation identifié à l'étape 1.1, modélisez, à l'aide de **diagramme de séquence**, les scénarios types entre les acteurs externes et le système. Étant donné que le système interagit avec des acteurs humains, pensez à modélisez certains scénarios qui montrent le comportement du système lors d'une erreur de saisie.

1.3. Au vu de la description du système, dressez un **diagramme de classes** avec les éléments qui sont indispensables.

Phase 2 – Conception niveau abstrait

Lors de cette 2ème phase de conception niveau abstrait, vous allez réfléchir au moyen informatique que vous allez utiliser pour développer un système facile d'utilisation, fiable et robuste.

2.1. Au vu des différentes fonctionnalités proposées par le système, décomposez-le, si possible, en plusieurs composantes indépendantes. Modéliser les fonctionnalités de chacune de ces composantes à l'aide d'un **diagramme de cas d'utilisation**.

2.2. Pour chacun des nouveaux cas d'utilisation identifiés à l'étape 1.2, modélisez, à l'aide de **diagramme de séquences**, les interactions entre les acteurs externes et les différents composants du système. N'hésitez pas à faire cette étape en même temps de l'étape 2.3. C'est en créant le digramme de classes que vous allez pouvoir améliorer et affiner le diagramme de séquence.

2.3. Dressez un **diagramme de classes** avec tous les éléments nécessaires au système. Les composantes du système, identifiées lors de l'étape 2.1, peuvent être utilisées comme paquetage. Une fois que le diagramme de classe est complet, vérifiez sa cohérence avec les diagrammes de séquences de l'étape 2.2. Ajoutez les attributs nécessaires aux classes et les paramètres d'entrée-sortie aux opérations. Vérifier que vous avez représenté le fonctionnement de chaque cas d'utilisation identifiée à l'étape 2.1.

Phase 3 – Conception niveau physique

Lors de cette 3ème phase de conception niveau physique, vous allez choisir le langage de programmation et les systèmes informatiques existants les mieux adaptés pour les fonctionnalités du système, identifiés précédemment. Pour les besoins de l'exercice, nous avons déjà fait ces choix pour vous :

- langage : Java
- gestion de la base de données : SQL Serveur 2012
- affichage de la carte géographique : Api Google Maps

3.1. Reprenez le **diagramme de classes** de l'étape 2.3 et transformez-le en UML pour Java en respectant toutes les contraintes impliquées par ce langage. Modifiez le diagramme afin d'utiliser les classes Java qui implémentent les structures de données appropriées (ArrayList, Pile FIFO, Object…). Vérifier la cohérence et la complétude de votre modèle afin de faciliter la génération automatique de code.

3.2. De la même façon que le diagramme de classe, transformez les **diagrammes de séquence**s en UML pour Java. Proposez également quelques diagrammes de test pour les fonctionnalités les plus importantes ou susceptibles de causer des erreurs. Ces diagrammes pourront ensuite être réutilisés comme bases de tests pour valider le développement.

3.3. Pour finir, reprendre les **diagrammes de cas d'utilisation** et pour spécifier le type des acteurs externes finalement choisit pour le développement. Pensez également à indiquer les codes d'accès à ces acteurs externes afin de faciliter la maintenance du système.