Départ. Informatique
BUT ASPE – S1-10. Qualité de Développement

V. Deslandres

TD1 – Modélisation, Conception

EXERCICE 1 - Couches logicielles

L'architecture *n-tiers* ou *multi-niveaux* est dérivée du modèle client-serveur.

> Le mot *tier* n'a pas la même signification en français et en anglais. En effet, *tier* (et son pluriel *tiers*) se traduit par niveau ou couche, alors que le mot français *tier* désigne une partie d'un tout divisé en trois.

Pour les applications proposées, on soumet une liste de fonctionnalités : pour chacune, préciser si elle relève plutôt de **l'IHM**, du **métier**, ou de la **persistance des données**.

A- Application Assurances

1. Saisir le login/ pwd en tant que Client
2. Vérifier l'authentification d'un utilisateur
3. Choix du type de sinistre par le client
4. Vérification de la couverture d'une personne dans un sinistre
5. Réplication de la BD
6. Saisie des informations administratives Clients
7. Calcul du montant des cotisations d'une personne
8. Déclaration du sinistre : enchaînement des écrans de saisie
9. Enregistrement d'un nouveau sinistre

B- Application « Formation en ligne »

Aujourd'hui de nombreuses formations sont disponibles en ligne (*e-learning, MOOC*). Les fonctionnalités proposées pour ce type d'outil sont les suivantes (tous utilisateurs et à différents niveaux d'abstraction) :

1. Choix de la formation parmi une liste
2. Inscription d'un utilisateur (saisie des informations)
3. Affectation d'un tuteur pour un apprenant
4. Consultation des pré-requis d'un module
5. Saisir les caractéristiques du module (nombre de crédits, vol. horaire, dates, modes d'évaluation)
6. Valider les saisies des caractéristiques
7. Obtenir les modules accessibles par une formation
8. Accéder au contenu d'un module (accès réservé aux inscrits)
9. Ouvrir une session de *chat* entre un tuteur et ses apprenants
10. Annoter les travaux d'un apprenant

C- Jeu « Bataille navale »

Vous connaissez ce jeu : sur un plateau, le joueur peut placer ses bateaux et déclencher ses tirs. Auparavant il a pu se connecter et attendre un autre joueur. Il reçoit l'information quand c'est son tour de jouer et quand la partie est terminée (tous les bateaux d'un joueur ont été coulés). Il voit les tirs de son adversaire et quels sont ses navires touchés ou coulés, les cases du plateau où il a déjà tiré, ainsi que le résultat de ses tirs.

Citez quelques méthodes relatives à chaque couche :
- Présentation
- Applicative
- Persistance

D- Technologie Web : formulaire, architecture MVC

On dispose d'un site où un client s'authentifie pour visualiser ses commandes antérieures. Donner une fonctionnalité pour chaque couche de l'architecture MVC.

EXERCICE 2 – Bug de l'an 2000

Pour beaucoup d'applications des années 80/90, le format de stockage choisi pour représenter une année était un nombre compris entre 0 et 99 : 1966 était stocké comme 66, 1989 comme 89, etc. donc moins d'espace était nécessaire pour stocker seulement deux chiffres.
A l'approche de l'An 2000, il a fallu revoir ce choix de conception.

2.1) Pourquoi ? quel risque posait ce stockage des dates ?

2.2) Certains développeurs ont utilisé des façons différentes pour stocker, et donc lire et manipuler les valeurs stockées dans les variables qui utilisaient le format de date à six chiffres.

> Certains ont stocké les dates comme des **nombres** : 22 janvier 1989 → 890122, 4 décembre 1966 stocké comme 661204, ce qui simplifiait grandement le tri des dates (ordre sur les nombres).
> D'autres ont gardé les **notions de jour, mois, année** (tous des nombres de 2 chiffres max).
> Quelle conséquence cela a eu pour le changement de format ?

EXERCICE 3 – Relation entre Modèle Relationnel et UML

Vous avez en charge une étude pour la réalisation d'une base de données de gestion d'une association. Vous ne disposez que du modèle relationnel (MR) ci-dessous :

```
Compte (#num:entier, categorie:{A|R}, email:chaine, =solde():réel,
        =alerter() :vide)
Recette(#num=>Compte, #date:date, montant:reel, annee:entier)
Dépense(#num=>Compte, #date:date, montant:reel, seuilAlerte:entier,
        annee:entier, =alerter():vide)
```

Et des explications suivantes :

« Les comptes A (actifs) sont les comptes courants de l'association. En cas de solde négatif, une alerte automatique est envoyée au email associé (titulaire du compte). Les comptes R (réserve) sont des comptes sur lesquelles les opérations sont rares et n'interviennent qu'en cas de problème de trésorerie ou d'investissement important. En cas de dépense supérieure à une valeur seuil, une alerte est aussi envoyée au titulaire du compte »

Les méthodes mentionnées (*solde(), alerter()…*) sont ici associées aux relations. Ces méthodes seront placées des procédures stockées par exemple.

3.1 - Réalisez le schéma UML correspondant à ce Modèle Relationnel.

3.2 - Nous proposons **d'analyser la qualité** de cette conception :

a - Serait-il judicieux de faire intervenir une **relation d'héritage** à la place de l'attribut catégorie pour discriminer les comptes A et R ?

b - Donner les **responsabilités** de la classe *Compte* et de la classe *Dépense*. Qu'en pensez-vous ? Il semble y avoir 2 erreurs.

*Indice : dans l'association, il y a 2 types de personnes : des stagiaires présents de 6 à 9 mois (à qui on ouvre des comptes), et des permanents qui sont là depuis plus de 5 ans. Que pensez-vous donc de la responsabilité **relative aux alertes** actuelle ?*

EXERCICE 4 - OTHELLO REVERSI

Soit la classe du Jeu Othello suivante :
Cette classe est-elle **cohésive** ?

Si non, proposer une amélioration.