NF17

Projet 6 :
Gestion d'un BDE

Aurélien Dupin
Azzeddine Benabbou
Sylvain Verhille
Heni Bellaaj

Sommaire

I. Note de clarification

1. Objectifs

Ce projet a pour but la création d'une application web qui s'appuie sur une base de données structurée de façon optimale permettant au BDE de gérer :

- Ses adhérents.
- Les associations qu'il fédère.
- Les responsables des associations.
- Les comptes des associations.
- Les inventaires.
- Les rapports d'activités et de subvention des associations.

2. Suppositions

- Les membres ont cotisé au moins une année. En effet, ils n'ont aucune raison d'être dans la base de données du BDE dans le cas contraire.
- Tous les étudiants possèdent un numéro d'étudiant ou INE.
- Il n'y a que trois types de dirigeants : les présidents, trésoriers et secrétaires.

3. Contraintes

- Un étudiant ne peut participer à une association étudiante à une date donnée que s'il est à jour de sa cotisation à cette date.

- Un membre ne peut évidemment pas participer deux fois à la même association étudiante dans un même laps de temps, malgré que la représentation UML le permette.

- Un membre ne peut diriger une association étudiante que s'il participe à cette association étudiante.

- D'après la supposition, il n'y a que trois types de dirigeants. Par conséquent, la classe dirigeant est abstraite.

- Lorsqu'une nouvelle comptabilité commence, par défaut, le solde de début est égal au solde de fin de la comptabilité précédente.

- Une association étudiante ne peut pas avoir deux comptabilités différentes à la même date.

- Une comptabilité ne peut être gérée que par un et un seul trésorier. Ce qui signifie qu'un changement de trésorier entraîne une nouvelle comptabilité. Un trésorier finit sa comptabilité avant d'être remplacé.

- Les dépenses sont soit des services, soit du matériel. La classe dépense est une classe abstraite.

- Une salle ne peut pas être louée deux fois en même temps. Si une salle doit réunir plusieurs associations étudiantes à un même moment, l'une d'entre elles devra faire la réservation à son nom.

- Toutes les locations sont soit des périodes courtes, soit des périodes longues. Autrement dit, la classe location est une classe abstraite.

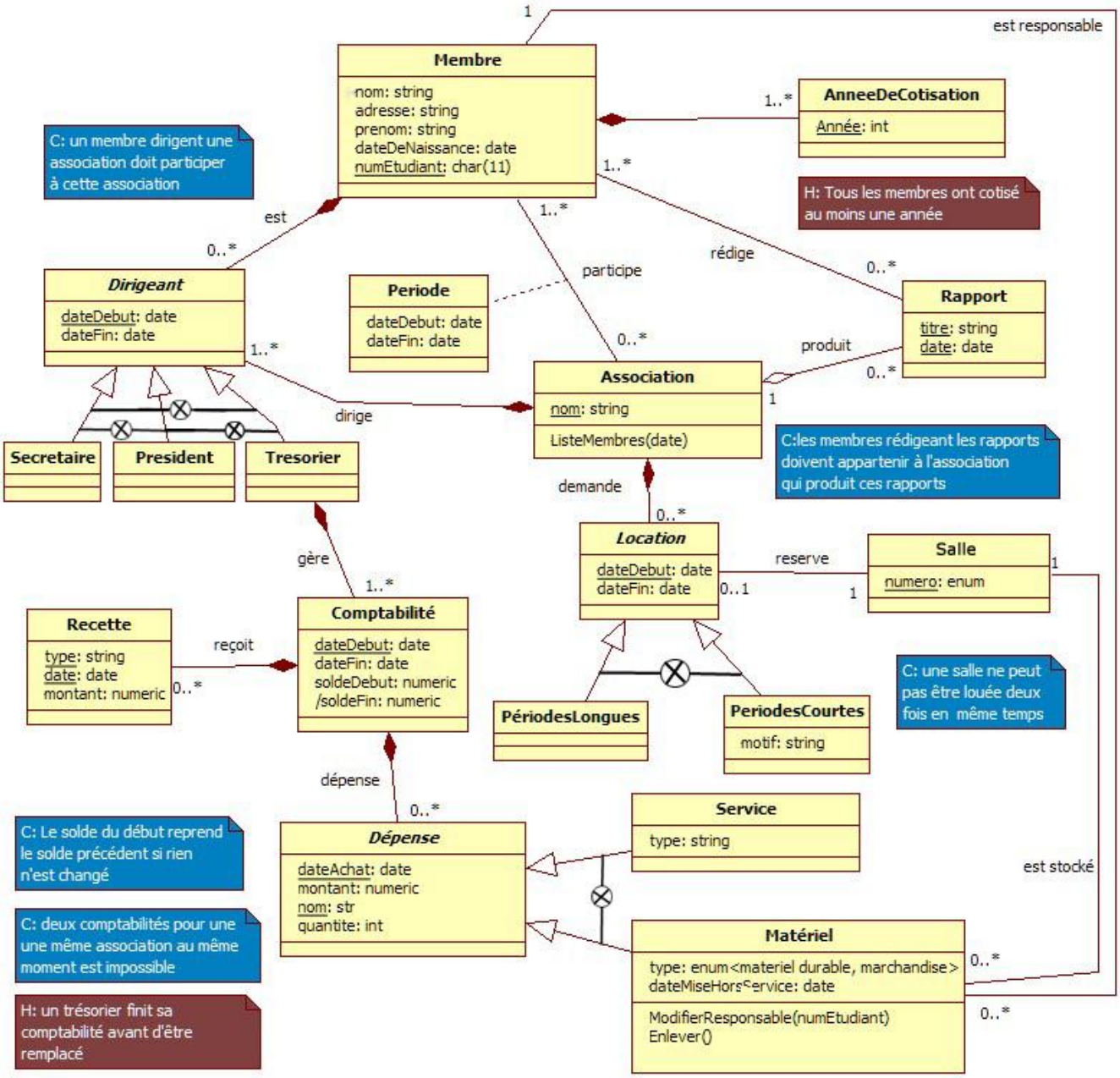
- Deux recettes de même type à la même date ne peuvent pas exister ensemble. Il faudra additionner les montants. De même pour les dépenses, il faudra additionner les montants et les quantités.

- Dans les différentes classes comportant un attribut dateFin, cet attribut peut rester nul tant qu'il est ignoré de l'utilisateur.

4. Méthodes

- La méthode ListeMembre affiche à l'écran la liste des membres à une date passée en paramètre. Pour le cas des dirigeants, leur rôle devra aussi être affiché.

II. Diagramme UML



III. Relationnel

Association (#nom: string)

Membre (#numEtudiant : char (11), nom : string, adresse : string, prenom : string, dateDeNaissance: date)

Periode (#numEtu: char (11)=>Membre (numEtudiant), #nomAsso: string=>Association(nom), #dateDebut : date, dateFin : date)

AnneeDeCotisation (#numEtu : char (11)=>Membre (numEtudiant), #Annee : int)

Rapport (#id: char (5), titre : string, date: date, nomAsso: string=>Association(nom)) avec id clé artificielle, nomAsso NOT NULL

Rediger (#id: char (5) =>Rapport (id), #numEtudiant: char (11) =>Membre (numEtudiant))

PROJ (Rapport,id) C PROJ (Rediger,id)

PROJ(Association, nom) C PROJ(Periode, nomAsso)

Location (dateDebut: date, dateFin: date, #nomAsso: string=>Association (nom), motif: string, type: enum {PeriodesLongues, PeriodesCourtes}, #num: enum=>Salle(numero)) avec dateDebut unique not null

Vue PeriodesLongues = PROJ (REST (Location, type = PeriodeLongues), dateDebut, dateFin, nomAsso,num)

Vue PeriodesCourtes = PROJ (REST (Location, type = PeriodeCourtes), dateDebut, dateFin, nomAsso, motif, num)

Salle (#numero: enum) l'énumération contient tous les numéros de salles de l'UTC, ce type permet de ne pas avoir d'erreur de salles.

Dirigeant (#dateDebut: date, dateFin: date, #numEtu: char(11) =>Membre (numEtudiant), #nomAsso: string=>Association (nom), type: enum {Secretaire, President, Tresorier})

Vue Secretaire = PROJ (REST (Dirigeant, type = Secretaire), dateDebut, dateFin, numEtu, nomAsso)

Vue President = PROJ (REST (Dirigeant, type = President), dateDebut, dateFin, numEtu, nomAsso)

Vue Tresorier = PROJ (REST (Dirigeant, type = Tresorier), dateDebut, dateFin, numEtu, nomAsso)

PROJ (Association, nom) C PROJ (Dirigeant, nomAsso)

Comptabilité (#dateDebut: date, dateFin: date, soldeDebut: numeric, #dateDirigeant: date=>Dirigeant (dateDebut), #numDirigeant=>Dirigeant(numEtu), #nomAsso=>Dirigeant(nomAsso)) soldeFin attribut dérivé de Comptabilité et seul les dirigeants de type trésorier peuvent gérer une comptabilité

PROJ (REST (Dirigeant, type = Tresorier), dateDebut) C PROJ (Comptabilité, dateDirigeant)

Recette (#id: char(5), type: string, date: date, montant: numeric, #dateComptabilité: date=>Comptabilité (dateDebut)) avec id clé artificielle

Dépense (#id: char (5), dateAchat: date, montant: numeric, nom: string, quantite: int, #dateComptabilité: date=>Comptabilité (dateDebut), typeService: string, typeMateriel: enum {materiel durable, marchandise}, dateMiseHorsService: date, num: enum=>Salle(numero), numEtu: char(11)=>Membre (numEtudiant), type: enum {Service, Matériel}) avec id clé artificielle, num NOT NULL, numEtu NOT NULL

Vue Service = PROJ (REST (Dépense, type = Service), id, dateAchat, montant, nom, quantite, dateComptabilité, typeService)

Vue Matériel = PROJ (REST (Dépense, type = Matériel), id, dateAchat, montant, nom, quantite, dateComptabilité, typeMateriel, dateMiseHorsService, num, numEtu)

Les contraintes de la note de clarification doivent toujours être appliquées.

IV- Normalisation

Membre (nom : string, prénom : string, adresse : string, dateDeNaissance : date, numEtudiant : char[11])

numEtudiant->nom
numEtudiant->prénom
numEtudiant->adresse
numEtudiant->numEtudiant
numEtudiant->dateDeNaissance
nom->nom
prénom ->prénom
nom, prénom, dateDeNaissance ->adresse
dateDeNaissance ->dateDeNaissance
nom, prénom, dateDeNaissance->numEtudiant

On en déduit que les clés sont numEtudiant ou nom, prénom, dateDeNaissance. La normalisation 1NF est respectée car tous les attributs sont des atomes, si on considère que la date est atomique. De plus, aucun attribut ne faisant pas partie d'une clé ne dépend que d'une partie de clé. La normalisation 2NF est elle-aussi respectée. Par ailleurs, tous les attributs n'appartenant pas à une clé ne dépendent pas d'une clé. Par conséquent, la normalisation 3NF est respectée. Enfin, aucune clé ou partie de clé ne dépendent d'un attribut ne faisant pas partie d'une clé. La forme normale de Boyce-Codd est respectée.

Periode (#numEtu: char (11)=>Membre (numEtudiant), #nomAsso: string=>Association(nom), #dateDebut : date, dateFin : date)

numEtu->numEtu
nomAsso->nomAsso
dateDebut->dateDebut
dateFin->dateFin
numEtu, nomAsso, dateDebut->dateFin

On en déduit que la clé est dateDebut, dateDirigeant, numDirigeant, nomAsso. La normalisation 1NF est respectée car tous les attributs sont des atomes. De plus, aucun attribut ne faisant pas partie d'une clé ne dépend que d'une partie de clé. La normalisation 2NF est elle-aussi respectée. Par ailleurs, tous les attributs n'appartenant pas à une clé ne dépendent que d'une clé. Par conséquent, la normalisation 3NF est respectée. Enfin, aucune clé ou partie de clé ne dépendent d'un attribut ne faisant pas partie d'une clé. La forme normale de Boyce-Codd est respectée.

Comptabilité (dateDebut: date, dateFin: date, soldeDebut: numeric, dateDirigeant: date=>Dirigeant (dateDebut), numDirigeant=>Dirigeant(numEtu), nomAsso=>Dirigeant(nomAsso)).

Rappelons que dateFin peut être nul, s'il est ignoré de l'utilisateur.

dateDebut->dateDebut
dateDebut, dateDirigeant, numDirigeant, nomAsso ->dateFin
dateDebut, dateDirigeant, numDirigeant, nomAsso ->soldeDebut
dateDebut, dateDirigeant, numDirigeant, nomAsso ->soldeFin
dateFin->dateFin
soldeDebut->soldeDebut
dateDirigeant->dateDirigeant
numDirigeant->numDirigeant
nomAsso->nomAsso

On en déduit que la clé est dateDebut, dateDirigeant, numDirigeant, nomAsso. La normalisation 1NF est respectée car tous les attributs sont des atomes. De plus, aucun attribut ne faisant pas partie d'une clé ne dépend que d'une partie de clé. La normalisation 2NF est elle-aussi respectée. Par ailleurs, tous les attributs n'appartenant pas à une clé ne dépendent que d'une clé. Par conséquent, la normalisation 3NF est respectée. Enfin, aucune clé ou partie de clé ne dépendent d'un attribut ne faisant pas partie d'une clé. La forme normale de Boyce-Codd est respectée.

Dépense (#id: char (5), dateAchat: date, montant: numeric, nom: string, quantite: int, #dateComptabilité: date=>Comptabilité (dateDebut), typeService: string, typeMateriel: enum {materiel durable, marchandise}, dateMiseHorsService: date, num: enum=>Salle(numero), numEtu: char(11)=>Membre (numEtudiant), type: enum {Service, Matériel})

id->id
dateAchat->dateAchat
montant->montant
nom->nom
quantite->quantite
dateComptabilité->dateComptabilité
typeMateriel-> typeMateriel
typeService -> typeService
dateMiseHorsService->dateMiseHorsService
numEtu->numEtu
type->type
id->dateAchat
id->montant
id->nom
id->quantite
id-> typeMateriel
id->typeService
id->dateMiseHorsService
id->numEtu
id->type
dateAchat, nom, numEtu, dateComptabilité ->quantite

dateAchat, nom, numEtu, dateComptabilité ->montant
dateAchat, nom, numEtu, dateComptabilité ->type
dateAchat, nom, numEtu, dateComptabilité ->typeService
dateAchat, nom, numEtu, dateComptabilité ->typeMateriel
dateAchat, nom, numEtu, dateComptabilité ->dateMiseHorsService
dateAchat, nom, numEtu, dateComptabilité->id

On a ainsi deux clés possibles : id,dateComptabilité ou dateAchat, nom, numEtu, dateComptabilité. La normalisation 1NF est respectée car tous les attributs sont des atomes. De plus, aucun attribut ne faisant pas partie d'une clé ne dépend que d'une partie de clé. La normalisation 2NF est elle-aussi respectée. Par ailleurs, tous les attributs n'appartenant pas à une clé ne dépendent que d'une clé. Par conséquent, la normalisation 3NF est respectée. Enfin, aucune clé ou partie de clé ne dépendent d'un attribut ne faisant pas partie d'une clé. La forme normale de Boyce-Codd est respectée.

V- Organigramme

1. Utilisateurs du produit

- **Les visiteurs**
  - Personnes normales, qui n'ont pas de tâches ou de rôles déterminés.
  - C'est aussi les utilisateurs non authentifiés.
- **Les adhérents BDE**
  - Visiteurs authentifiés et qui sont adhérents au BDE.
- **Les présidents des associations**
  - Adhérents BDE qui jouent un rôle spécifique : « Président »
- **Les secrétaires des associations**
  - Adhérents BDE qui jouent un rôle spécifique : «Secrétaire»
- **Les trésoriers des associations**
  - Adhérents BDE qui jouent un rôle spécifique : « Trésorier »
- **Les administrateurs**
  - Super utilisateurs qui ont tous les droits.

2. Fonctions accessibles par chaque utilisateur

Le produit doit permettre aux **visiteurs** de :

- Demander une adhésion au BDE (et attendre sa validation).
- Parcourir la liste des associations.

Les visiteurs ne peuvent pas adhérer directement à une association puisqu'il faut au préalable qu'ils soient adhérents BDE.

Le produit doit permettre aux **adhérents BDE** de :

- Adhérer à une association.
- Parcourir la liste des associations.
- Consulter la liste des membres et des dirigeants de l'association à laquelle ils appartiennent.
- Rédiger des rapports pour l'association à laquelle ils appartiennent.
- Consulter l'archive des rapports de l'association à laquelle ils appartiennent.

Le produit doit permettre aux **dirigeants** ces fonctions communes :

- Sortir des listes des membres et des dirigeants pour une période donnée pour l'association à laquelle ils appartiennent.

Le produit doit permettre aux **présidents des associations** de :

- Réaliser toutes les fonctions «adhérents BDE ».
- Ajouter et supprimer des membres de l'association.

Le produit doit permettre aux **trésoriers des associations** de :

- Réaliser toutes les fonctions «adhérents BDE ».
- Consulter l'archive des comptabilités de l'association.
- Ajouter, supprimer, modifier une recette pour l'association.
- Ajouter, supprimer, modifier une dépense (service ou matériel) pour l'association.
- Dresser le bilan.

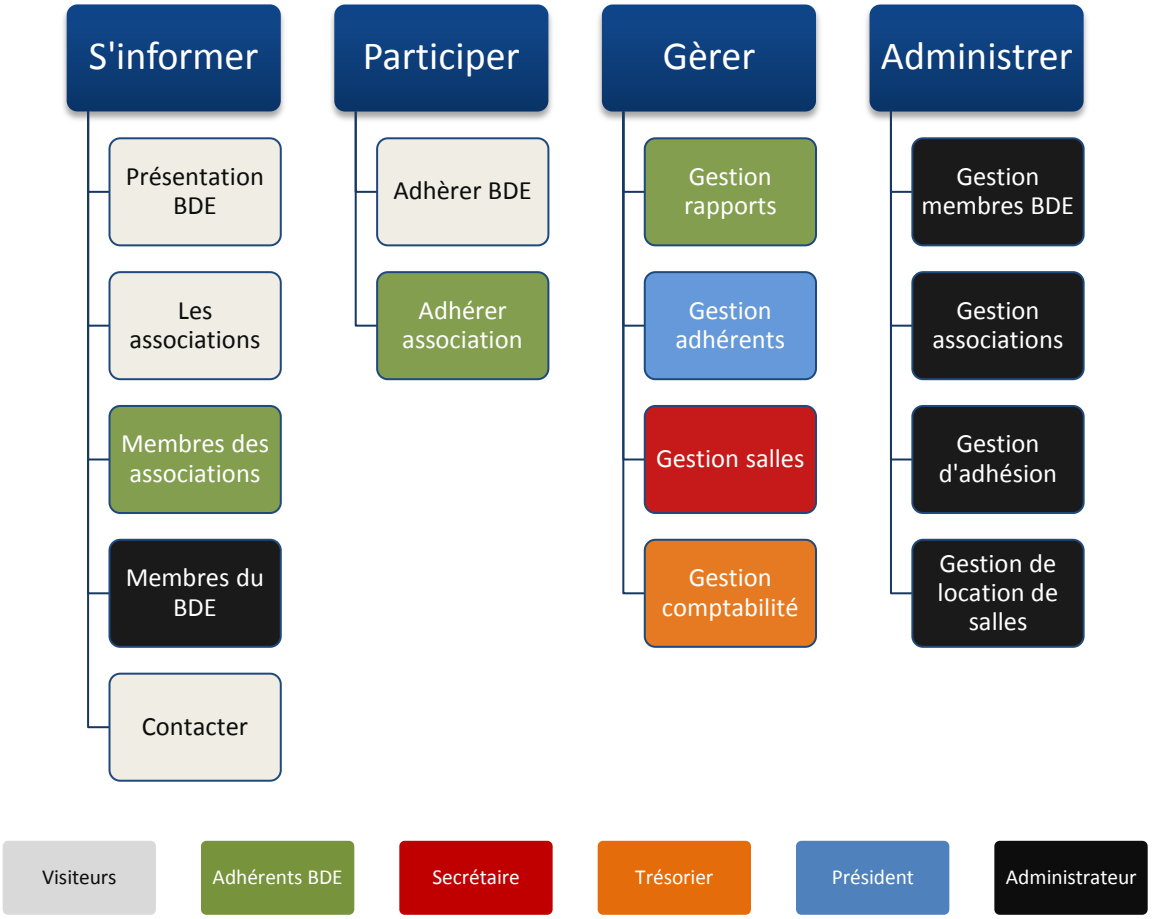
Le produit doit permettre aux **secrétaires des associations** de :

- Réaliser toutes les fonctions « adhérents BDE »
- Consulter la liste des salles disponibles pour la location.
- Demander une location de salle.

Le produit doit permettre aux **administrateurs** de :

- Réaliser toutes les fonctions citées précédemment et sans restriction. (Par exemple : l'administrateur peut gérer la comptabilité de n'importe quelle association sans qu'il y soit adhéré).
- Valider ou non la demande d'adhésion au BDE soumise par un visiteur.
- Consulter les archives des membres BDE.
- Ajouter, supprimer et modifier des membres du BDE.
- Lister les membres non à jour dans leurs cotisations.
- Ajouter, supprimer et modifier des associations.
- Valider ou non une demande de location de salle soumise par une association.
- Louer une salle à une association sans demande préalable.

3. Les pages du produits



Une page accessible par un visiteur est accessible par tous les autres utilisateurs.

Une page accessible par un « adhérents BDE » est accessible par tous les autres utilisateurs sauf « Visiteurs ».

Une page accessible par un dirigeant (« Secrétaire », « Trésorier » ou « Président ») n'est accessible que par ce dirigeant et l'administrateur.

Une page accessible par un Administrateur n'est accessible par aucun autre utilisateur.