

Goubin Régis
Morel Victor
B3121

COMPTE RENDU DASI : PROJET COLLECT'IF

I.Introduction

-> COLLECT'IF, mais qu'est ce donc ?

COLLECT'IF est un service qui permet de faire des activités avec les autres adhérents de l'association. Si un adhérent veut faire une activité un jour précis, il n'a qu'à en faire la demande. S'il y a assez de participants pour jouer, un événement est créé. Ce dernier se verra affecter un lieu par l'administrateur.

Lorsqu'un lieu est affecté à un lieu, un mail est envoyé aux participants afin de les informer qu'ils vont pouvoir s'amuser ce jour là.

-> Description des cas d'usage

Adhérent

La seule demande d'un adhérent est de pouvoir faire les activités qu'il aime. Il doit donc pouvoir s'inscrire et se connecter à son compte. Ensuite, il doit être capable de consulter ses demandes et surtout d'en faire. Il a juste à déterminer une date et une activité et espérer que d'autres personnes souhaitent jouer avec lui.

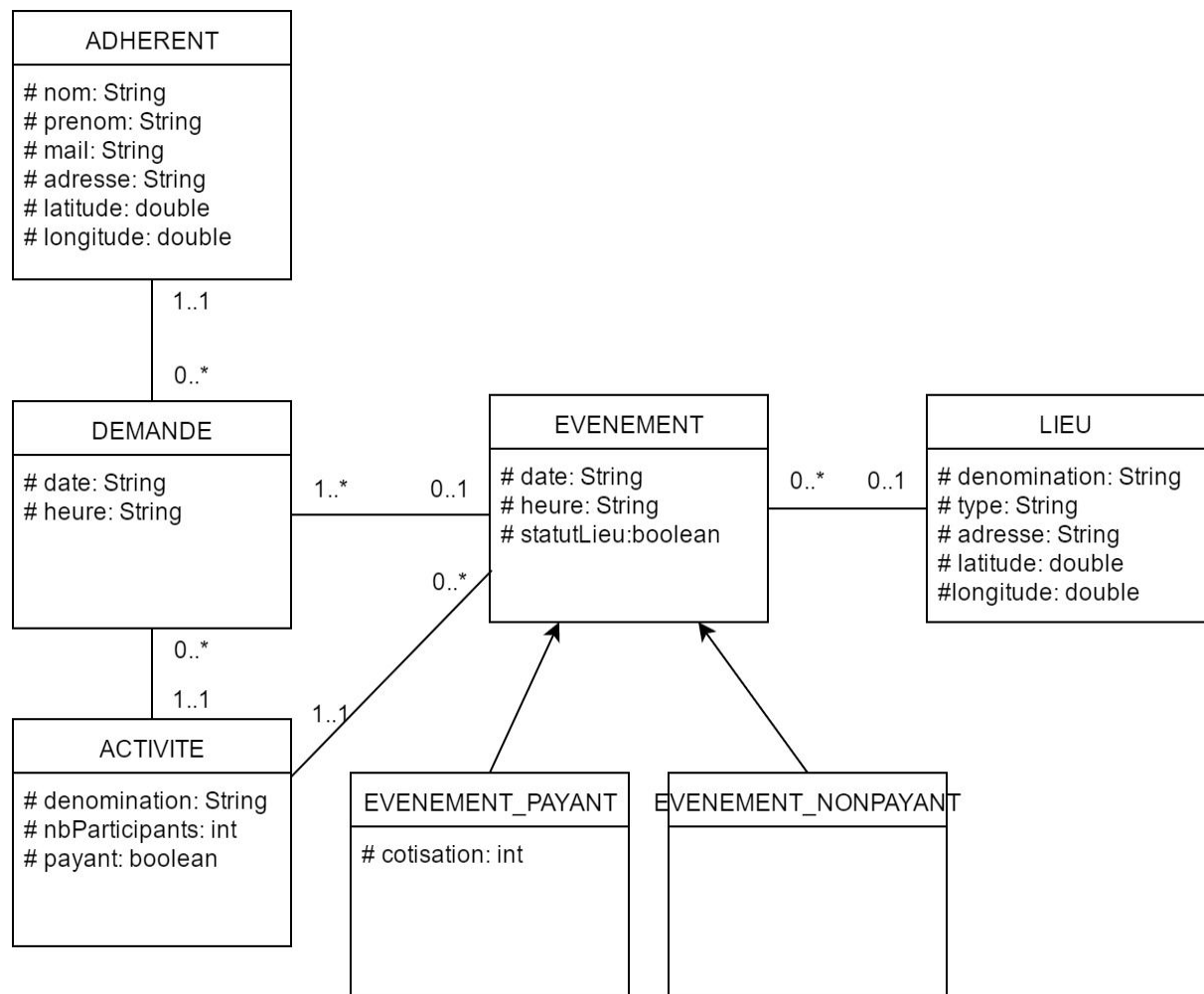
Administrateur

L'administrateur doit être en mesure de faire vivre son association. Il va donc avoir comme possibilité d'affecter des lieux et des cotisations (au besoin) aux événements afin qu'ils puissent se passer. Il aura donc besoin de voir la liste des événements mais aussi une carte sur laquelle il pourra voir où se situe les lieux disponibles (pour accueillir un événement) et les participants d'un événement, pour affecter un lieu cohérent.

II. Analyse

II.1 Modèle du domaine

Le cahier des charges demande à ce qu'un adhérent puisse faire des demandes. Les demandes contiennent une date et une activité. Quand un nombre de demandes suffisant a été fait le même jour pour la même activité, on crée un événement. On distingue deux types d'événements : les événements payants et les événements non payants (cela dépend de l'activité). Aussi, l'événement se verra affecter un lieu par l'administrateur. En prenant en compte toutes ces contraintes, on a conçu le diagramme suivant :



Comme l'indique la structure ci-dessus, nous avons effectués les divers choix suivants :

Utilisation de l'héritage : Les activités pouvant être gratuites ou payantes, on définit des événements payants et non payants. Les événements non payants n'ont pas de nouveaux attributs. Cependant, pour prendre en compte le caractère payant de certains événements, on ajoute à EVENEMENT_PAYANT un attribut cotisation


Enchaînement logique :

Un adhérent fait des demandes, il nous a donc fallu mettre en relation ces deux classes. Etant donné qu'une demande se fait sur une activité, on associe à une demande une activité. Un événement est créé par plusieurs demandes et est associé, comme une demande, à une activité. Ensuite, on associe également à EVENEMENT un lieu (zero lieu peuvent être associé à un événement car le lieu n'est pas connu lors de la création d'un événement).

II.2 Cas d'utilisation et IHM

II.2.1 IHM communes

IHM Connexion



The screenshot shows a web browser window with a login form. The form is titled "IHM Connexion" in the top right corner. It contains an "Email :" label followed by a text input field. To the right of the input field is a "Se connecter" button. At the bottom of the form, there is a link "Pas encore membre ?" followed by an "INSCRIPTION" button.

Email :

Se connecter

Pas encore membre ?

Intention	Contrôle	Action	Réponse
Se connecter	Bouton "Se Connecter"	Clic	si "admin" : afficher IHM administrateur accueil sinon appel de "seConnecter" si null, afficher un message d'erreur si non null, afficher IHM "adhérent accueil"
Inscription	Bouton "Inscription"	Clic	afficher l'IHM "Inscription"

II.2.2 IHM Adhérent

IHM Inscription

Intention	Contrôle	Action	Réponse
S'inscrire	Bouton "Inscription"	Clic	appel "creerAdherent" si null afficher un message d'erreur sinon afficher "adhérent"

IHM Adhérent Accueil

Accueil Faire Demande Déconnexion

Mes demandes :

id Demande	Nom Activité	Date et moment	Lieu Affecté ?

Intention	Contrôle	Action	Réponse
Aller sur l'accueil	Bouton "Accueil"	Clic	aller à IHM "Adhérent Accueil"
Pouvoir faire une demande	Bouton "FaireDemande"	Clic	aller à l'IHM "Adherent Faire Demande"
Se déconnecter	Bouton "Déconnexion"	Clic	aller à l'IHM "Connexion" et appel de "seDeconnecter"

IHM Adhérent Faire Demande

Accueil Faire Demande Déconnexion

Activité :

Date :
 / /

Moment :

Faire une demande

Activités :

Nom Activité	Payant

Intention	Contrôle	Action	Réponse
aller à l'accueil	Bouton "Accueil"	Clic	afficher IHM "Adherent Accueil"
Pouvoir faire une demande	Bouton "Faire Demande"	Clic	afficher IHM "Adherent Faire Demande"
Faire une demande	Bouton "Faire une demande"	Clic	si null afficher un message d'erreur sinon afficher un message confirmant la création de la demande
Se déconnecter	Bouton "Déconnexion"	Clic	afficher IHM "Connexion" et appel de "seDeconnecter"

II.2.3 IHM Administrateur

IHM Administrateur Accueil

Accueil

Carte

Déconnexion

id Evenement :

id Evenement :

id Lieu :

PAF :

id Lieu	Adresse Lieu

id Evenement	Nom Activité	Date et moment	Lieu	PAF

Intention	Contrôle	Action	Réponse
aller à l'accueil	Bouton "Accueil"	Clic	afficher l'IHM "Administrateur Accueil"
voir la carte	Bouton "Carte"	Clic	afficher l'IHM "Administrateur Carte"
affecter un lieu	Bouton "Affecter Lieu"	Clic	appel "affecterLieu" si null afficher message erreur sinon afficher message de confirmation
affecter une cotisation	Bouton "Affecter PAF"	Clic	appel "affecterPAF" si null afficher message erreur sinon afficher message de confirmation
se déconnecter	Bouton "Déconnexion"	Clic	afficher l'IHM "Connexion" et appel de "seDeconnecter"

Accueil

Carte

Déconnexion

Enlever Lieux

Afficher Lieux

id Evenement :

Afficher Participants

Intention	Contrôle	Action	Réponse
Aller à l'accueil	Bouton "Accueil"	Clic	afficher IHM "Administrateur Accueil"
voir la carte	Bouton "Carte"	Clic	afficher IHM "Administrateur Carte"
enlever les points des lieux	Bouton "Enlever Lieux"	Clic	si points des lieux présents, les enlever, ne rien faire sinon
placer les points des lieux	Bouton "Afficher Lieux"	Clic	si points des lieux absent les afficher avec un appel de "listeLieux", ne rien faire sinon
afficher les participants d'un événement sur la carte	Bouton "Afficher Participants"	Clic	afficher les points des participants de l'événement choisi et supprimer ceux qui existaient (s'il en existait)
se déconnecter	Bouton "Déconnexion"	Clic	afficher l'IHM "Connexion" et appel de "seDeconnecter"

II.3 Spécification des Services

Selon l'utilisateur, notre application proposera différents services.

III.3.1 Service Commun

ServiceMetier.seConnecter

static Adherent **seConnecter**(String mail)

seConnecter permet aux adhérents et à l'administrateur de se connecter grâce à leur mail. seConnecter initialise l'entityManager. Si l'utilisateur rentre un mail, le service renvoie l'adhérent de ce mail s'il existe, null sinon (donc si l'on rentre "Admin", l'entityManager est initialisé et renvoie null) .

ServiceMetier.seDeconnecter

static void **seDeconnecter**()

seDeconnecter destroy l'entityManager.

III.3.2 Service Adhérent

ServiceMetier.creerAdherent

static Adherent **creerAdherent**(String nom, String prenom, String mail, String adresse)

createAdherent permet de créer un adhérent à partir d'informations mises en paramètre. Le service demande donc un prenom, un nom, un mail et une adresse. A partir de cette dernière, la localisation de l'adhérent est automatiquement créée. Le service vérifie aussi que le mail n'est pas déjà utilisé par un autre adhérent, si c'est le cas, on renvoie null, sinon, on renvoie l'adhérent créé

ServiceMetier.creerDemande

static Demande **creerDemande**(Adherent A, String activite, Date date, String moment)

faireDemande permet à un adhérent de faire une demande d'activité. Pour créer une demande complète, le service demande donc un adhérent à qui associer la demande, ainsi qu'une date et un moment de la journée. On initialise juste le jour, le mois et l'année de la date. Le moment correspond au moment de la journée, soit : matin, après-midi et soir. On vérifie quand même que l'adhérent n'a pas déjà fait une demande au même moment, dans ce cas on renvoie null. Aussi, dans le cas de la création de la demande, on regarde combien de personne ont fait la même demande que nous, s'il y a suffisamment d'adhérents, on crée un évènement. Dans le cas où la demande a bien été créée, on la renvoie.

ServiceMetier.historiqueDemandes

```
static List<Demande> historiqueDemandes(Adherent adherent)
```

historiqueDemande permet de récupérer les demandes qu'a fait un adhérent. Pour faire cela, le service a simplement besoin de l'adhérent. On retourne la liste de demande de cet adhérent.

ServiceMetier.getActivites

```
static List<Activite> getActivites()
```

listeActivite renvoie la liste des activités que l'association peut organiser.

III.3.3 Service Administrateur

ServiceMetier.getEvenement

```
static List<Evenement> getEvenement()
```

listeEvenement renvoie la liste des événements existants.

ServiceMetier.getLieux

```
static List<Lieu> getLieux()
```

getLieux renvoie la liste des lieux où peuvent se dérouler des activités.

ServiceMetier.consulterParticipants

<code>static List<Adherent> consulterParticipants(Long id)</code>
--

consulterParticipants permet d'avoir la liste des participants à un événement. Pour cela, le service à besoin de connaître l'id de l'événement dont on souhaite avoir l'information (ce service permet d'avoir les participants d'un événement plus facilement qu'avec l'appel du service listeEvenement).

[ServiceMetier.affecterPAF](#)

<code>static Evenement affecterPAF(Long idEvenement, int cotisation)</code>
--

affecterPAF permet d'affecter le prix de la cotisation à un événement payant. Pour cela, elle demande l'id de l'événement à modifier. Si l'événement n'est pas payant ou n'existe pas, le service renvoie un objet null, sinon elle renvoie l'événement sur lequel on a effectué la modification. Aussi, si un prix a déjà été fixé, on n'affecte pas la cotisation demandée et on renvoie null.

[ServiceMetier.affecterLieu](#)

<code>static Evenement affecterLieu(Long idEvenement, Long idLieu)</code>
--

affecterLieu permet d'affecter le lieu à un événement. Pour cela, le service demande l'id de l'événement à modifier et l'id du lieu qu'on souhaite lui affecter. Si l'événement n'existe pas, le service renvoie un objet null, sinon elle renvoie l'événement sur lequel on a effectué la modification. Aussi, si un lieu a déjà été déterminé, on n'affecte pas le lieu demandé et on renvoie null.