CNAM PARIS

UE NSY209

M.POLLET et M. BELLEBIA

‘S.E.L Services’

Projet de développement d’une application mobile distribuée, pour faciliter le fonctionnement d’associations promouvant les échanges et services locaux.

DOSSIER DE CONCEPTION

03 MAI 2018

Auteur :

Vivien SERE

Historique des versions

N° de version	Description	Date
0.0	Version initiale	31 mai 2017
1.0	Version modifiée suite au développement des 1° version de l'application bureau, application mobiles et serveur d'application	03 mai 2018

Lexique des termes utilisés dans le document

Terme	Définition
MLCC	Monnaie Locale Complémentaire Citoyenne
SEL	Système d'Échange Local
Android	Système d'exploitation pour appareil mobile basé sur le noyau Linux et développé actuellement par Google
RPC	Remote Procedure Call : protocole d'appel de procédure à distance
IHM	Interface Homme Machine
EJB	Enterprise JavaBeans : architecture de composants logiciels côté serveur pour Java EE.
DTO	Data Transfer Object : objet de transfert des données
SQL	Structured Query Langage : langage informatique normalisé servant à exploiter les bases de données
SQLite	Base de donnée embarquée dans les téléphones mobiles
DAO	*Data Access Object* : Objet d'accès aux données
JPA	Java Persistence API : interface de programmation permettant d'organiser des données relationnelles
ORM	Mapping Object Relationnel : correspondance entre une base de donnée orientée objet et une base de donnée relationnelle
REST	representational state transfer: architecture pour application distribuée
GWT	(*Google Web Toolkit*) Framework développé par Google permettant de créer et maintenir des applications web dynamiques mettant en œuvre JavaScript, en utilisant le langage et les outils Java aussi bien côté serveur que client
API	*Application Programming Interface* (interface de programmation)
JSON	*JavaScript Object Notation* : format de donnée permettant de représenter l'information et de la faire circuler
ACID	*Atomicité Cohérence Isolation Durabilité* : caractéristiques d'une opération de transaction
NFC	*Near Field Communication* : communication en champ proche – technologie de communication sans fil de courte portée
HTTPS	HyperText Transfer Protocol Secure : combinaison de HTTP avec une couche de chiffrement
XML	*Extensible Markup Language:* méta langage informatique de balisage générique

Table des matières

1 Vue d’ensemble

1.1 Contexte

Le but de ce projet est de fournir une application mobile distribuée à des associations dont le but est de favoriser les échanges locaux.

Le but de ces associations pour leurs membres est de retrouver l’entraide et les liens sociaux entre les personnes qui pouvaient exister autrefois.

Ces associations peuvent être de type ***SEL*** (Système d'Echange Local) qui utilisent de la monnaie virtuelle, ***MLCC***(Monnaie Locale Complémentaire Citoyenne) voire AMAP (Association pour le Maintien d'une Agriculture Paysanne).

Cette application doit pouvoir être utilisée par les membres de l’association mais aussi par le bureau de l’association.

Le bureau de l’association devra pouvoir réaliser avec cette application :

- la gestion des membres de l’association
- la gestion des offres et demandes des adhérents
- envoyer des messages aux adhérents
- la gestion des feuilles de richesses (nom donné au compte en unité de monnaie virtuelle, des adhérents)

Les adhérents de l’association devront pouvoir réaliser avec cette application :

- la gestion de leurs offres et demandes
- la gestion de leur profil
- la consultation des offres et demandes disponibles
- la consultation et le filtrage des messages/notifications à recevoir
- la géolocalisation d'un adhérent
- la géolocalisation des adhérents ayant des critères à respecter
- le paiement d'un autre adhérent pour un service rendu (en monnaie de l'association)

1.2 Environnement de développement

Réalisation	Outil
Applicatif en Java	Eclipse Oxygen
Applicatif sur plate-forme mobiles	Android Studio
Gestion du développement logiciel	Github
Rédaction de documents	LibreOffice
Conception et modélisation	Visual Paradigm
Poste de développement local	UBUNTU 16.04
Mobile de tests	Tablette NEXUS 7 (2013) (Android 6) Samsung Galaxy S5 NEO (Android 6)

Pour l'application utilisée par le bureau, il sera utilisé le framework ***GWT*** pour pouvoir coder en Java aussi bien du côté serveur que client. L'application sera installée sur un ordinateur de l'association.

Pour le serveur, il sera utilisé un ordinateur de l'association. Celui-ci devra toujours être autant que possible, accessible par les mobiles. Pour les associations de plus grandes importances, il devra pouvoir être déployé sur un serveur d'application distant payant.

2 Architecture

2.1 Vue d'ensemble

L'application mobile sera développé sur la plate forme ***Android***. Elle sera utilisée par les adhérents et permettra d'accéder aux différents services. L'application ***GWT*** sera utilisée par le bureau de l'association et permettra d'assurer l'administration de l'association. Un serveur d'application permettre de fournir les services aux applications ***Android*** et à l'application du bureau de l'association. Ce serveur peut être installé sur le même ordinateur que l'application bureau pour les association de taille modeste ou sur un serveur distant pour les associations plus importantes.

Le serveur pourra également envoyer des notifications par SMS ou par Email aux adhérents.

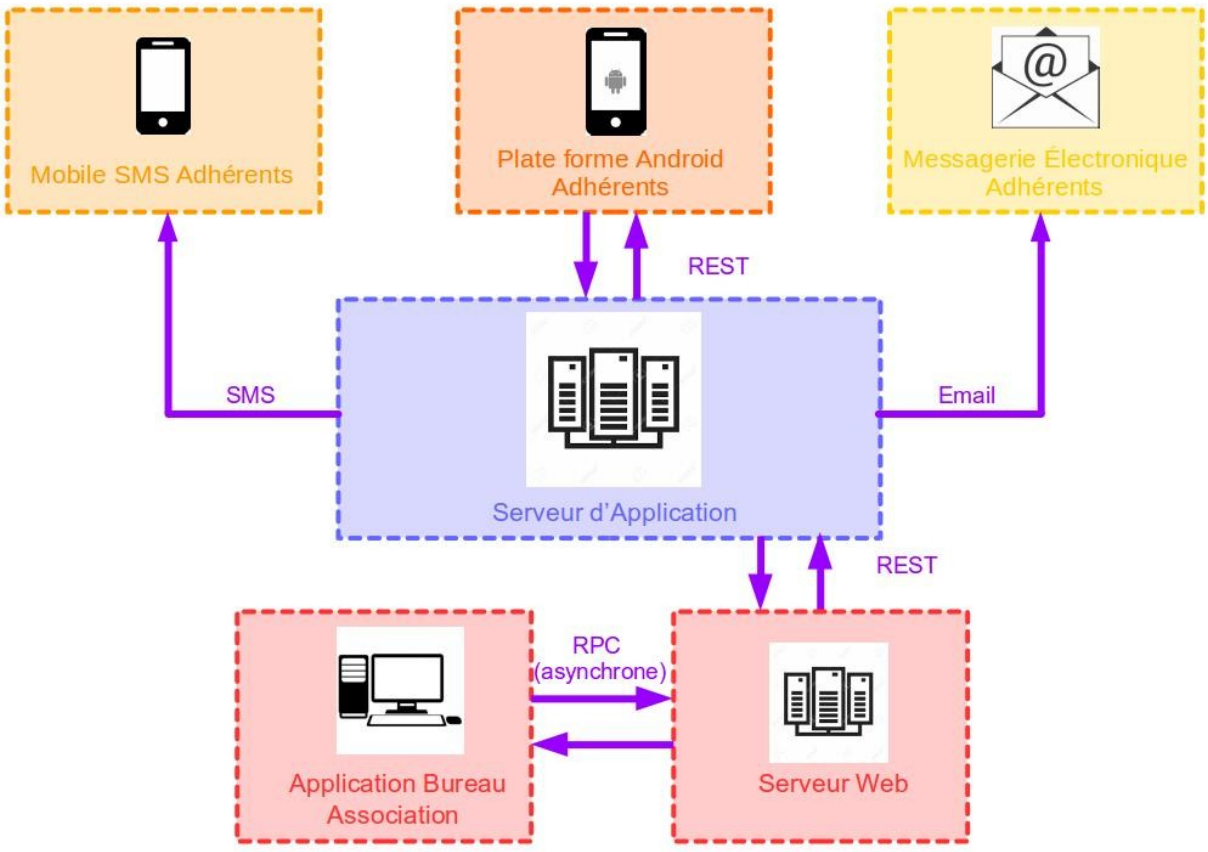


Figure 1 : *Architecture d'ensemble*

La communication entre les plate-formes Android et le serveur d'application se fera via un web service ***REST***.

La communication entre l'application bureau association et le serveur d'application se fera par ***RPC*** (Remote Procedure Call) asynchrone via le Framework ***GWT***, puis par un web service ***REST***.

2.2 Côté clients

2.2.1 Bureau association

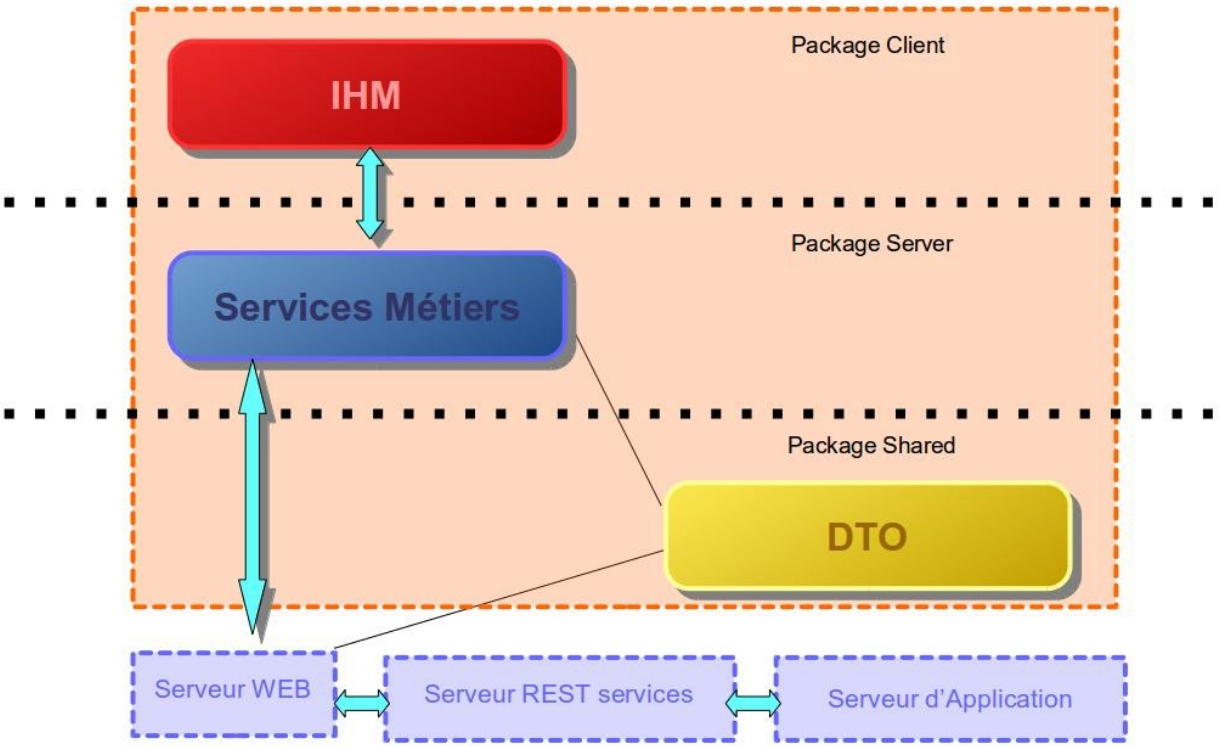


Figure 2 : *Architecture Application Bureau Association*

L'application bureau sera réalisée avec le framework ***GWT*** qui permet de développer en *JAVA* aussi bien côté serveur que côté client. La partie cliente étant transformée automatiquement en code *JAVASCRIPT* optimisé et compatible avec tous les navigateurs

web les plus courants. Cela permet de bénéficier des fonctionnalités du langage *JAVA* et de n'avoir qu'une seule interface de programmation.

Le framework ***GWT*** organise ses applications en 3 packages principaux : *client, server, shared*. Dans le package « *client* », on trouve la couche ***IHM*** (Interface Homme Machine), qui sera transformée en code *JAVASCRIPT*. Dans le package « *server* » on trouve les services déployé sur un serveur. La couche ***IHM*** accède à ses services par ***RPC*** via des interfaces *JAVA*. Pour cette application seules ces interfaces seront présentes, leur implémentation se feront dans le serveur d'application. La communication avec le serveur d'application se fera par un web service ***REST***. Dans le package « *shared* » sont présents les éléments partagés entre le serveur et le client. Dans l'application bureau, cela concerne les ***DTO*** (Data Transfer Object). Ces objets permettront de transférer les données acquises par la couche ***IHM*** vers le serveur.

2.2.2 Utilisateur mobiles

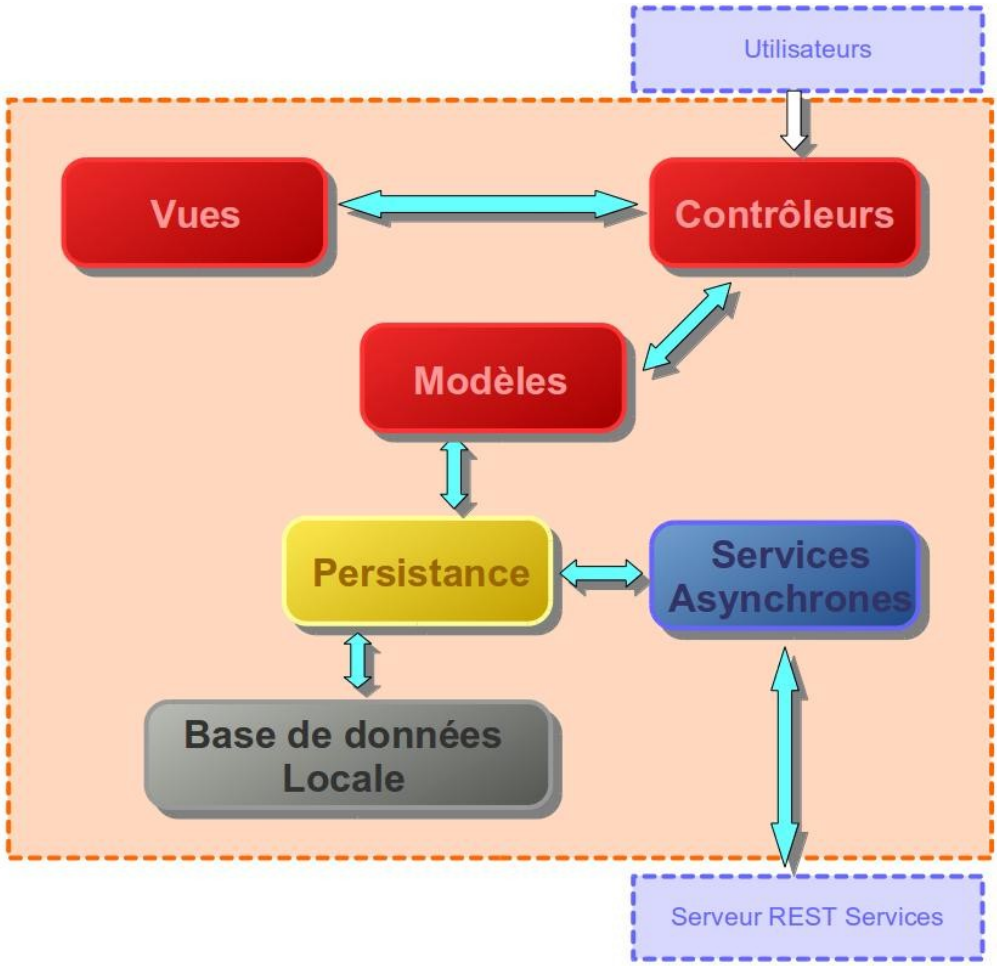


Figure 3 : *Architecture Application Android*

L'application ***Android*** est composée de plusieurs élément:

- ***Vues***: Ce qui est affiché à l'écran. Elles présentent les résultats des traitements. Sous ***Android*** elles seront organisées sous un répertoire '*layout*' en fichier ***xml***.
- ***Contrôleurs*** : C'est la couche qui réagit aux actions de l'utilisateur. Sous ***Android*** ils seront organisés sous forme d'activités et sont étroitement liés aux vues. Dans la présente application, ils seront rassemblés sous le package '*activity*'.
- ***Modèles*** : Ils contiennent les données à afficher et gèrent leur traitement. Dans l'application, ils seront regroupés sous le package '*models*'.

  Ces 3 éléments constituent le modèle architectural ***MVC*** (Modèle vue contrôleur). Le contrôleur réagit aux actions de l'utilisateur en faisant appel aux données du modèle. Et il indique à la vue les valeurs de ces données à afficher.
- ***Persistance*** : c'est la couche qui gère l'accès à la base de données locale. Elle y effectue les opérations demandées la plupart du temps par les ***modèles*** pour obtenir les valeurs de leurs données (opérations traduites en requêtes ***SQL***). Dans l'application, elle sera représentée sous le package '***dao***'.
- ***Services Asynchrones :*** Ce sont les services qui tournent en tâches de fond pour notamment vérifier si le mobile est connecté à internet, récupérer les données du serveur distant, mettre à jour la base locale et/ou la base du serveur distant. Dans l'application, ils seront rassemblés sous le package '*background*'.
- **Base de données Locale** : Bases locales dans lesquelles seront stockées les données. La base locale est de type SQLite (base de données SQL locale gérée par Android). Les données relatives au profil utilisateur seront stockées dans un fichier partagé de type **SharedPreferences** (fichier spécifique à Android). Dans l'application, elle sera également gérée par le package '**dao**'.

2.3 Côté serveur

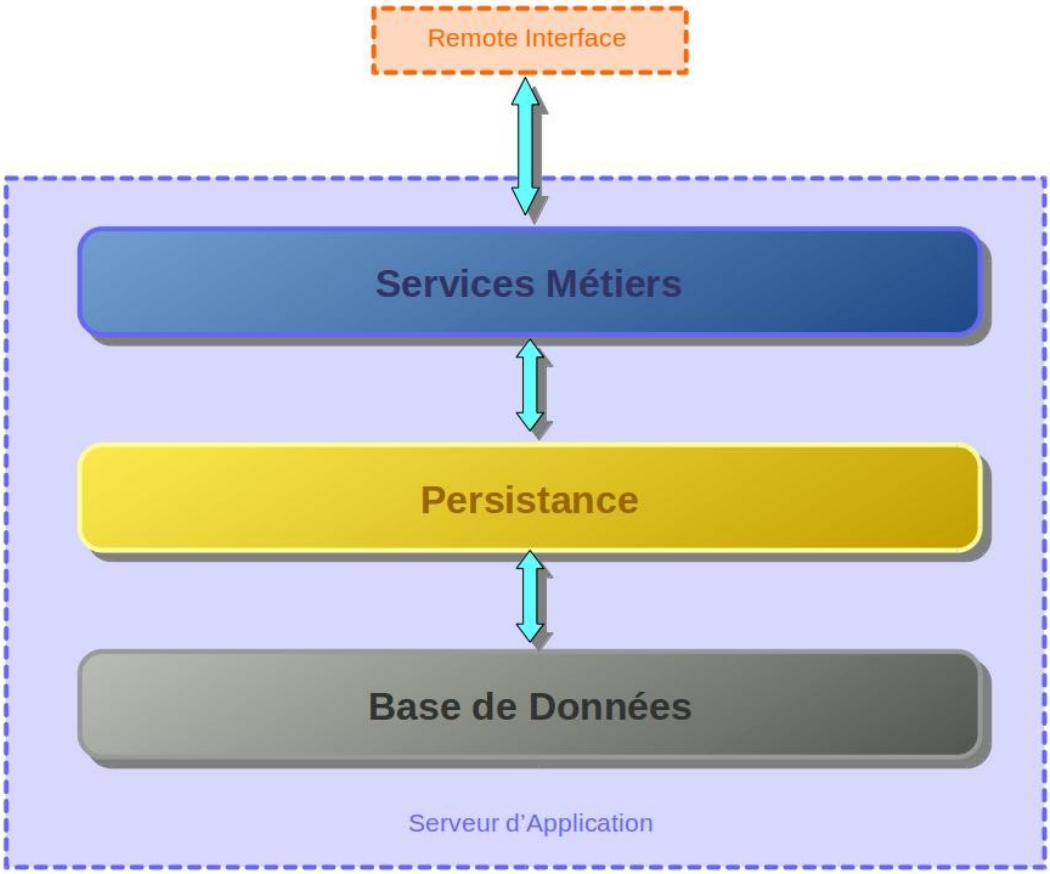


Figure 4: *Architecture Serveur d'Application*

Le serveur d'application utilisé sera **GLASSFISH**.

Le serveur d'application est composé également de plusieurs couches :

- ***Services métiers :*** elle fournit les services distants demandés par les applications ***Android*** ou bureau Association. Elle transmet également les notifications aux adhérents concernés (par SMS, Email ou l'application ***Android***). Elle utilise pour cela les données fournies par la couche persistance.
- ***Couche Persistance :*** elle sera composée de ***DAO*** qui géreront l'accès à la base de donnée. Cet accès se fera par l'intermédiaire d'objet *Entity* qui représenteront les tables en base. Le mappage ***ORM*** entre les entities et les tables se fera suivant une implémentation de la norme ***JPA*** (EclipseLink ou Hibernate).
- ***Base de Données:*** assure le stockage des données. Une solution gratuite sera utilisée : **PostgreSQL**.

Le serveur d'application sera réalisé avec des composants ***EJB*** pour faciliter la gestion des éléments suivants :

- Injection de dépendance
- Les transactions
- L'utilisation d'intercepteurs
- L'accès concurrent aux services
- La sécurité

2.4 Communication clients-serveur

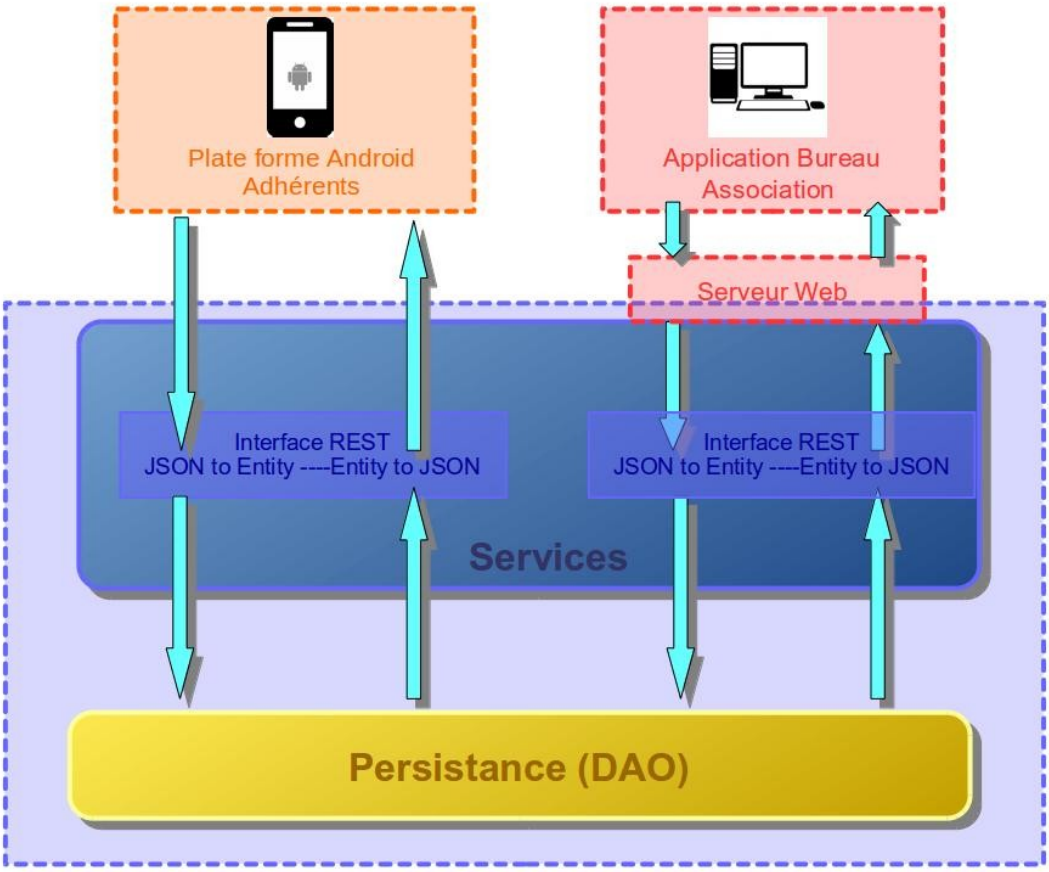


Figure 5 : *communication clients-serveur*

Les applications Android communiqueront avec le serveur via une interface ***REST***. Les données seront échangées au format ***JSON***. Les services du serveur d'application devront donc pouvoir être capable de transformer les données au format ***JSON*** en *Entity* et inversement.

L'application Bureau association communiquera par l'intermédiaire d'un serveur Web (ici **Apache TOMCAT**) . Ce serveur échangera également via une interface REST au format ***JSON***.

Au préalable, pour les 2 applications, les données devront être sérialisées (implements Serializable) au format ***DTO***. Ce sont ces DTOs qui seront transformés en JSON.

3 Use Cases

3.1 Acteurs

3.1.1 Application Web côté bureau

- acteurs principaux :

  - l'association : représente les membres du bureau de l'association, assure le bon fonctionnement et la gestion du système

- acteurs secondaires:

  - service Notification : notifie les adhérents de toutes nouvelles informations jugées nécessaires par l'association

  - Base de Données : assurera la persistance des informations

  - service Transaction : assure que l'opération de transfert en monnaie virtuelle d'une feuille de richesse à une autre feuille de richesse ait les propriétés ***ACID***

3.1.2 Application Mobile

- acteurs principaux :

  - l'adhérent: membre de l'association, utilisera l'application mobile

- acteurs secondaires:

  - service Notification : notifie les adhérents de toutes nouvelles informations jugées nécessaires par l'association

  - Base de Données : assurera la persistance centralisée des informations

  - Base de Données Locale : assurera la persistance localement dans le mobile des informations

- service Transaction : assure que l'opération de transfert de monnaie virtuelle d'une feuille de richesse à une autre feuille de richesse ait les propriétés « ***ACID*** »

- service Géolocalisation : fournit les services permettant de localiser une adresse sur une carte

- service Mise à jour : mets à jours la base de données locale et centrale

3.1.3 SMS et Mails

- acteurs principaux :

  - l'adhérent : membre de l'association, utilisera les SMS et/ou les mails

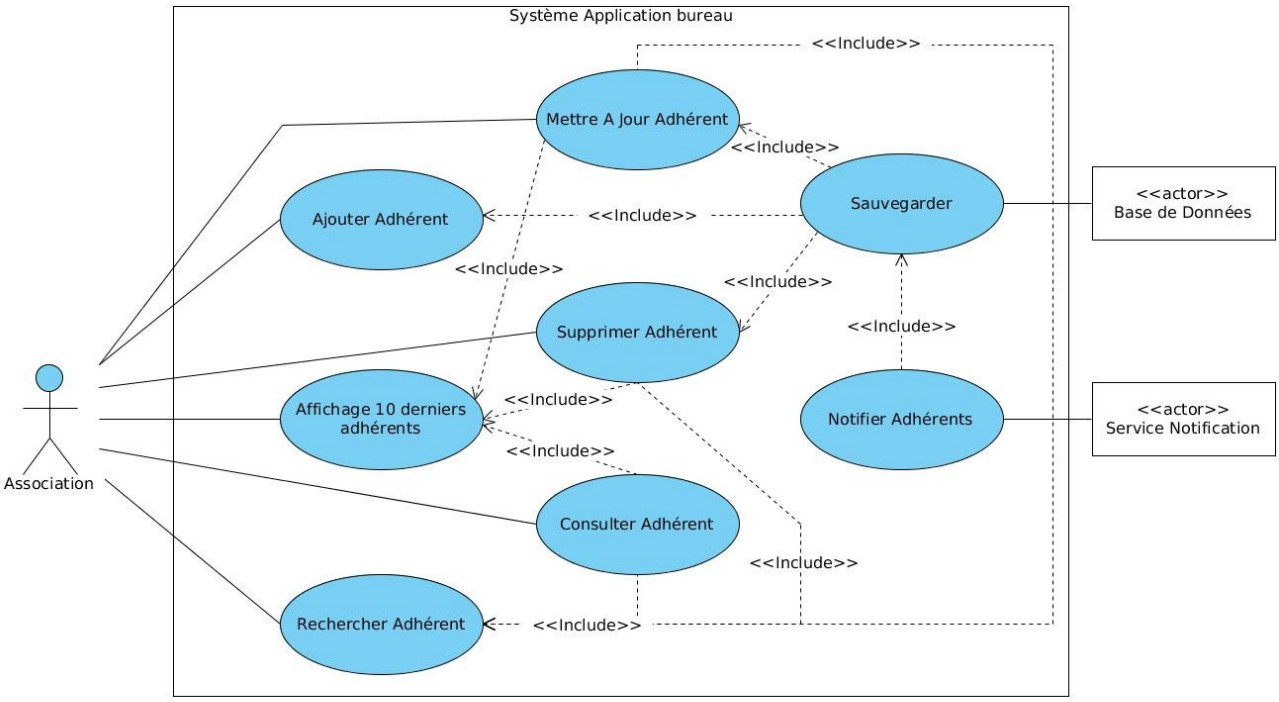
  - l'association : représente les membres du bureau de l'association, assure le bon fonctionnement et la gestion du système

- acteurs secondaires:

  - service Notification : notifie les adhérents de toutes nouvelles informations jugées nécessaires par l'association

3.2 Application Web côté bureau : gérer les adhérents

3.2.1 Diagramme



Cas d'utilisation: Application Web côté bureau - gérer adhérents

3.2.2 Affichage des 10 derniers adhérents modifiés

- *Événement déclencheur* : l'association choisit «Gérer Adhérents» dans le menu de l'accueil.
- *Pré Condition* : accès au serveur, au moins 1 adhérent créé sinon affichage d'un message correspondant,
- *Post Condition* : intégrité des données, données non modifiées, si le nombre d'adhérent est inférieur à 10 l'intégralité des adhérents sera affichée
- Scénario nominal :

  1. l'association sélectionne «Gérer Adhérents» dans le menu
  2. l'application affiche une liste des adhérents
  3. l'association sélectionne « retour »
  4. la fenêtre « accueil » s'affiche

- *Scénarii alternatifs* :

  2a. pas d'adhérent, affichage d'un message informatif

  3a. l'association clique sur un adhérent : → cas d'utilisation « consulter adhérent »

  3b. l'association clique sur « ajouter adhérent » : → cas d'utilisation « Ajouter Adhérent »

  3c. l'association clique sur « supprimer » en face un adhérent : → cas d'utilisation « Supprimer Adhérent »

  3d. l'association clique sur « rechercher » adhérent : → cas d'utilisation « Rechercher Adhérent »

3.2.3 Ajouter Adhérent

- *Événement déclencheur* : l'association choisit «Ajouter Adhérents» dans la fenêtre d'affichage de la liste des adhérents.
- *Pré Condition* : cas d'utilisation « Affichage des 10 derniers adhérents modifiés » ou cas d'utilisation « Rechercher Adhérent » terminés.
- *Post Condition* : intégrité des données, données ajoutées en base et envoie de notifications si ok, données non modifiées sinon, mise à jour des 10 derniers adhérents
- Scénario nominal :

  1. l'association sélectionne «Ajouter Adhérents» dans le menu
  2. l'application affiche un formulaire de saisie adhérent
  3. l'association saisit le formulaire
  4. l'association valide le formulaire
  5. une fenêtre de confirmation apparaît
  6. l'association confirme
  7. sauvegarde en base de l'adhérent : → cas d'utilisation « Sauvegarde »
  8. notification envoyé : → cas d'utilisation « Notifier Adhérents »
  9. retour à la fenêtre des 10 derniers adhérents avec en tête l'adhérent créé

- *Scénarii alternatifs* :

  4a. l'association annule, une fenêtre de confirmation apparaît, si oui retour aux 10 derniers adhérents, si non retour à la saisie du formulaire

  6a. l'association annule la confirmation de création : retour à la saisie du formulaire

  7a. la sauvegarde ne réussit pas

  8a. la notification n'est pas envoyée

  9a. un message d'erreur de création de l'adhérent apparaît, retour aux 10 derniers adhérents

3.2.4 Rechercher Adhérent

- *Événement déclencheur* : l'association choisit «Rechercher Adhérents» dans la fenêtre d'affichage de la liste des adhérents.
- *Pré Condition* : cas d'utilisation « Affichage des 10 derniers adhérents modifiés » ou cas d'utilisation « Rechercher Adhérent » terminés.
- *Post Condition* : liste des adhérents trouvés affichés, message d'erreur sinon
- Scénario nominal :

  1. l'association sélectionne « Rechercher Adhérents» dans le menu

  2. l'application affiche un formulaire de saisie de critère de recherche adhérent

  3. l'association saisit le formulaire

  4. l'association valide le formulaire

  5. une fenêtre de confirmation apparaît

  6. l'association confirme

  7. la liste des adhérents apparaît

  8a. l'association clique sur un adhérent : → cas d'utilisation « consulter adhérent »

  8b. l'association clique sur « supprimer » en face un adhérent : → cas d'utilisation « Supprimer Adhérent »

  8c. l'association appuis sur « retour » : retour à l'écran précédent

- *Scénarii alternatifs* :

  4a. l'association annule, une fenêtre de confirmation apparaît, si oui retour aux 10 dernier adhérents, si non retour à la saisie du formulaire

  6a. l'association annule la confirmation de recherche: retour à la saisie du formulaire

  7a. aucun adhérent trouvé : affichage message correspondant

3.2.5 Consulter Adhérent

- *Événement déclencheur* : l'association choisit «Consulter Adhérents» dans la fenêtre d'affichage de la liste des adhérents.
- *Pré Condition* : cas d'utilisation « Affichage des 10 derniers adhérents modifiés » ou « Rechercher Adhérent » affichent une liste d'adhérent, ces listes ne doivent pas être vide (au moins un adhérent doit être créé)
- *Post Condition* : fiche adhérent affichée, mise à jour des 10 derniers adhérents
- Scénario nominal :

  1. l'association sélectionne un adhérent de la liste affichée

  2. la fiche adhérent apparaît

  3. l'association consulte la fiche

  4. l'association retourne à la liste initiale

- *Scénarii alternatifs* :

  4a. l'association supprime la fiche : → cas d'utilisation « Supprimer Adhérent »

  4b. l'association modifie la fiche : → cas d'utilisation « Mettre à jour Adhérent »

  4c. l'association accède à la feuille de richesse : → cas d'utilisation « gérer feuille de richesse »

3.2.6 Supprimer Adhérent

- *Événement déclencheur* : l'association choisit «Supprimer Adhérents» dans la fenêtre d'affichage de la liste des adhérents.
- *Pré Condition* : cas d'utilisation « Affichage des 10 derniers adhérents modifiés » ou « Rechercher Adhérent » affichent une liste d'adhérent, ces listes ne doivent pas être vide (au moins un adhérent doit être créé), une fiche adhérent est affichée
- *Post Condition* : intégrité des données, données supprimées en base et envoie de notifications si ok, données non modifiées sinon, mise à jour des 10 derniers adhérents
- Scénario nominal :

  1. l'association sélectionne « supprimer »
  2. une fenêtre de confirmation apparaît
  3. l'association confirme la suppression
  4. l'association retourne à l'écran précédent

- *Scénarii alternatifs* :

  3a. l'association annule la suppression : retour à l'écran précédent

3.2.7 Mettre à Jour Adhérent

- *Événement déclencheur* : l'association choisit «Mettre à jour Adhérents» dans la fenêtre d'affichage de la liste des adhérents ou d'une fiche adhérent.
- *Pré Condition* : cas d'utilisation « Consulter adhérent » en cours ou affichage d'une liste d'adhérent (non vide)
- *Post Condition* : intégrité des données, données modifiées en base et envoie de notifications si ok, données non modifiées sinon, mise à jour des 10 derniers adhérents
- Scénario nominal :

  1. l'association sélectionne «Mettre à Jour Adhérents» dans le menu
  2. l'application affiche un formulaire de saisie adhérent pré rempli avec les données de la fiche

3. l’association saisit le formulaire

4. l’association valide le formulaire

5. une fenêtre de confirmation apparaît

6. l’association confirme

7. sauvegarde en base de l’adhérent : → cas d’utilisation « Sauvegarde »

8. notification envoyé : → cas d’utilisation « Notifier Adhérents »

9. retour à l'écran précédent

- *Scénarii alternatifs* :

  4a. l’association annule, retour à la fiche adhérent

  6a. l’association annule la confirmation de création : retour à la fiche adhérent

  7a. la sauvegarde ne réussit pas

  8a. la notification n’est pas envoyée

  9a. un message d’erreur de modification de l’adhérent apparaît, retour à l'écran précédent

3.2.8 Sauvegarder

- *Événement déclencheur* : l’association confirme une modification en base de donnée
- *Pré Condition* : tous les cas d’utilisation ayant pour but de modifier la base de donnée s’étant déroulés suivant le scénario nominal
- *Post Condition* : intégrité des données, données modifiées en base et envoie de notifications si ok, données non modifiées sinon,
- Scénario nominal :

  1. une requête de type « Create » « Update » ou « Delete » est envoyée en base de donnée

  2. l’opération est transactionnelle

  3. l’opération réussit

4. une réponse comme quoi l'opération s'est bien déroulée est envoyé au service de notification : → cas d'utilisation « Notifier Adhérent »

- *Scénarii alternatifs* :

  3a. l'opération ne réussit pas

  4a. une réponse comme quoi l'opération n'a pu être réalisée est envoyée à l'application Web

3.2.9 Notifier Adhérent

- *Événement déclencheur* : la base de donnée envoie un message au service de notification comme quoi une modification de cette dernière a bien été réalisée
- *Pré Condition* : le message envoyé par la base de donnée au service de notification est bien reconnu par ce dernier
- *Post Condition* : un message est envoyé par SMS, par mail et par l'application mobile aux adhérents
- Scénario nominal :

  1. le service de notification reçoit un message

  2. le service de notification lit le message

  3. le service de notification traduit le message

  4. le service de notification récupère les mails, n° de portable et n° adhérent de tous les adhérents en base de donnée (requête de type SELECT)

  5. le service de notification envoi une requête à la base de donnée pour créer une notification pour tous les adhérents concernés par la notification.

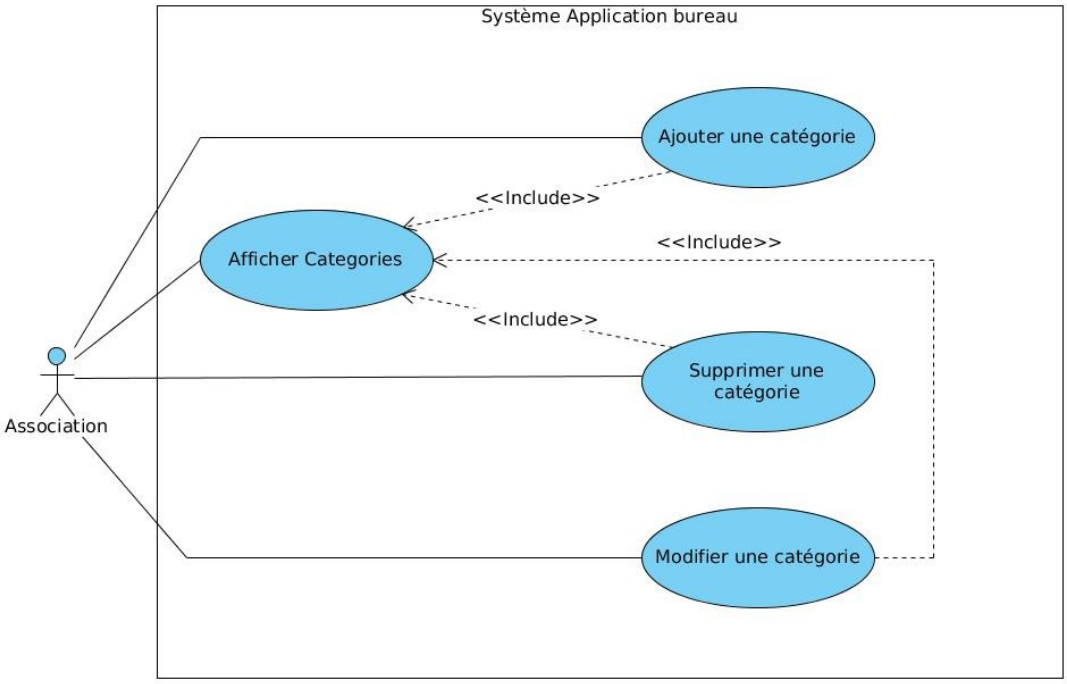
  6. le service de notification envoie le message traduit aux adhérents par mail, SMS et l'application mobile

- *Scénarii alternatifs* :

  Sans Objet

3.3 Application Web côté bureau : gérer les catégories

3.3.1 Diagramme



Cas d'utilisation: Application Web côté Bureau - Gérer les catégories

Powered By Visual Paradigm Community Edition

3.3.2 Afficher catégories

- *Événement déclencheur* : l'association choisit «Gérer les catégories» dans le menu de l'accueil.
- *Pré Condition* : SO
- *Post Condition* : intégrité des données, données non modifiées,
- Scénario nominal :

  1. l'association sélectionne «Gérer les catégories» dans le menu
  2. l'application affiche une liste des catégories
  3. l'association sélectionne « retour »
  4. la fenêtre « accueil » s'affiche

- *Scénarii alternatifs* :

  2a. la liste est vide : un message informe que la liste est vide

  3a. l'association clique sur « supprimer » en face un type: → cas d'utilisation « Supprimer une catégorie»

  3b. l'association clique sur «modifier» : → cas d'utilisation «modifier une catégorie»

3.3.3 Supprimer une catégorie

- *Événement déclencheur* : l'association choisit «Supprimer» dans la liste
- *Pré Condition* : cas d'utilisation « Affichage des catégories» affichent une liste, la liste ne doit pas être vide
- *Post Condition* : intégrité des données, données supprimées en base, mise à jour de la liste

- Scénario nominal :

  1. l'association sélectionne « supprimer »
  2. une fenêtre de confirmation apparaît
  3. l'association confirme la suppression
  4. l'association retourne à la liste des catégories

- *Scénarii alternatifs* :

  3a. l'association annule la suppression : retour à l'écran précédent

3.3.4 Modifier une catégorie

- *Événement déclencheur* : l'association choisit «Modifier» dans la fenêtre d'affichage de la liste des catégories.
- *Pré Condition* : cas d'utilisation « Affichage des catégories» affichent une liste, la liste ne doit pas être vide
- *Post Condition* : intégrité des données, données supprimées en base, mise à jour de la liste
- Scénario nominal :

  1. l'association sélectionne «Modifier » en face un type
  2. l'application affiche un formulaire de saisie de la catégorie pré rempli
  3. l'association saisit le formulaire

4. l’association valide le formulaire

5. une fenêtre de confirmation apparaît

6. l’association confirme

7. sauvegarde en base du type

8. retour à l'écran précédent

- *Scénarii alternatifs* :

  4a. l’association annule, retour à la liste

  6a. l’association annule la confirmation de création : retour à la liste

  7a. la sauvegarde ne réussit pas

3.3.5 Ajouter une catégorie

- *Événement déclencheur* : l’association choisit «Ajouter » dans la fenêtre d'affichage de la liste des catégories.
- *Pré Condition* : cas d’utilisation « Afficher catégories» terminés.
- *Post Condition* : intégrité des données, données ajoutées en base, mise à jour de la liste
- Scénario nominal :

  1. l’association sélectionne «Ajouter» à l’écran

  2. l’application affiche un formulaire de saisie de la catégorie

  3. l’association saisit le formulaire

  4. l’association valide le formulaire

  5. une fenêtre de confirmation apparaît

  6. l’association confirme

  7. sauvegarde en base de la catégorie

  8. retour à la liste modifiée

- *Scénarii alternatifs* :

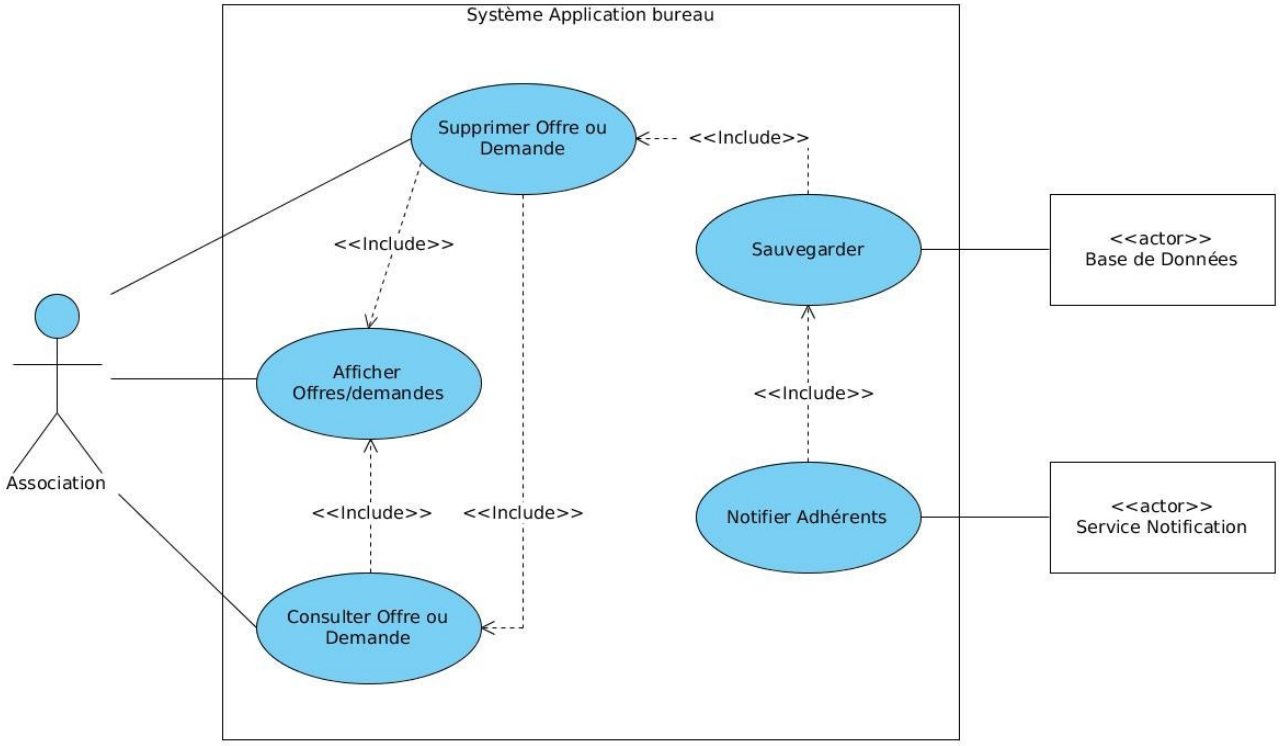
  4a. l'association annule, retour à la liste

  6a. l'association annule la confirmation de création : retour à la saisie du formulaire

  7a. la sauvegarde ne réussit pas

3.4 Application Web côté bureau : gérer les offres et demandes

3.4.1 Diagramme



Cas d'utilisation: Application Web côté bureau - Gérer Offres et Demandes Ponctuelles

3.4.2 Afficher Offres/demandes

- *Événement déclencheur* : l'association choisit «Gérer Offres et Demandes» dans le menu de l'accueil.
- *Pré Condition* : accès au serveur, au moins 1 offre ou demande ponctuelle créée dans le catalogue sinon affichage d'un message correspondant,
- *Post Condition* : intégrité des données, données non modifiées,
- Scénario nominal :

  1. l'association sélectionne «Gérer Offres et Demandes» dans le menu
  2. l'application affiche une liste des offres et demandes
  3. l'association sélectionne « retour »
  4. la fenêtre « accueil » s'affiche

- *Scénarii alternatifs* :

  3a. l'association clique sur « supprimer » en face une offre: → cas d'utilisation « Supprimer Offre ou Demande»

  3b. l'association clique sur «consulter» : → cas d'utilisation «Consulter Offre ou Demande»

3.4.3 Consulter Offre ou Demande

- *Événement déclencheur* : l'association choisit «Consulter 1 Offre ou Demande» dans la liste.
- *Pré Condition* : cas d'utilisation « Affichage des Offres et Demandes » affiche une liste, cette liste ne doit pas être vide (au moins une offre ou demande doit être créée)
- *Post Condition* : Offre ou Demande affichée
- Scénario nominal :

  1. l'association sélectionne une offre/demande de la liste affichée
  2. la fiche offre/demande apparaît
  3. l'association consulte la fiche

4. l'association retourne à la liste initiale

- *Scénarii alternatifs* :

  4a. l'association supprime la fiche : → cas d'utilisation « Supprimer offre/demande »

3.4.4 Supprimer Offre ou Demande

- *Événement déclencheur* : l'association choisit «Supprimer 1 Offre ou Demande» dans la liste ou dans la fiche
- *Pré Condition* : cas d'utilisation « Affichage des Offres et Demandes» affichent une liste ou « Consulter Offre ou Demande» en cours, la liste ne doit pas être vide (au moins une offre ou demande doit être créée)
- *Post Condition* : intégrité des données, données supprimées en base et envoie de notifications si ok, données non modifiées sinon, mise à jour de la liste

- Scénario nominal :

  1. l'association sélectionne « supprimer »

  2. une fenêtre de confirmation apparaît

  3. l'association confirme la suppression

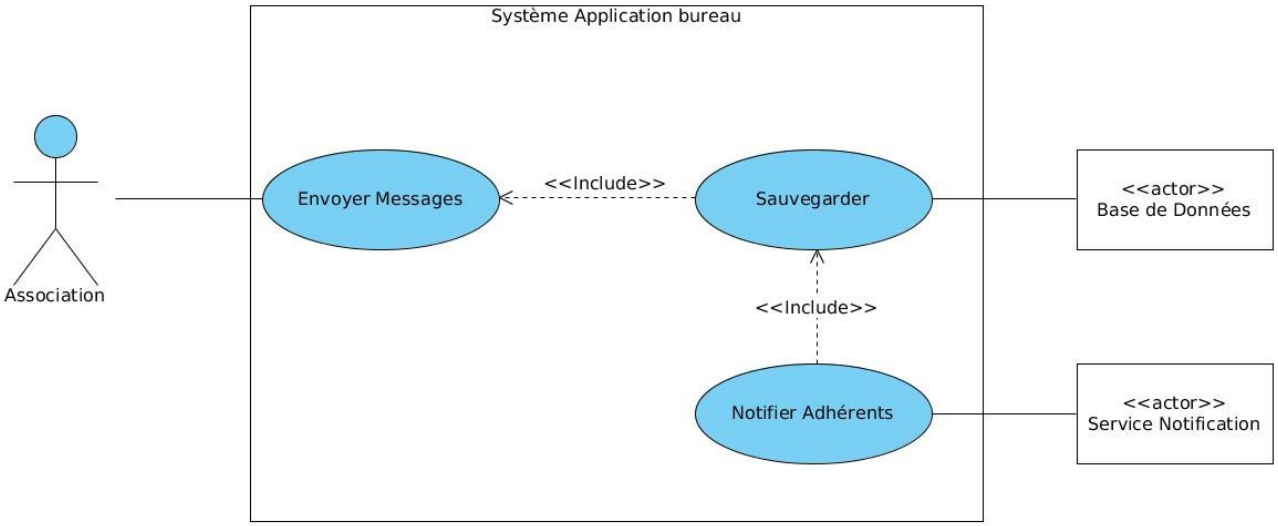
  4. l'association retourne à la liste des offres et demandes ponctuelles

- *Scénarii alternatifs* :

  3a. l'association annule la suppression : retour à l'écran précédent

3.5 Application Web côté bureau : envoyer messages

3.5.1 Diagramme



Cas d'utilisation: application Web côté bureau - Envoyer messages aux adhérents

3.5.2 Envoyer Messages aux adhérents

- *Événement déclencheur* : l'association choisit «Envoyer Message» dans le menu de l'accueil.
- *Pré Condition* : accès au serveur, au moins un adhérent en base
- *Post Condition* : intégrité des données, message enregistré en base de données et envoyé au Service de Notification

- Scénario nominal :

  1. l'association sélectionne «Envoyer Message»
  2. une fenêtre de saisie de message apparait
  3. l'association saisit le message
  4. l'association choisit d'envoyer le message

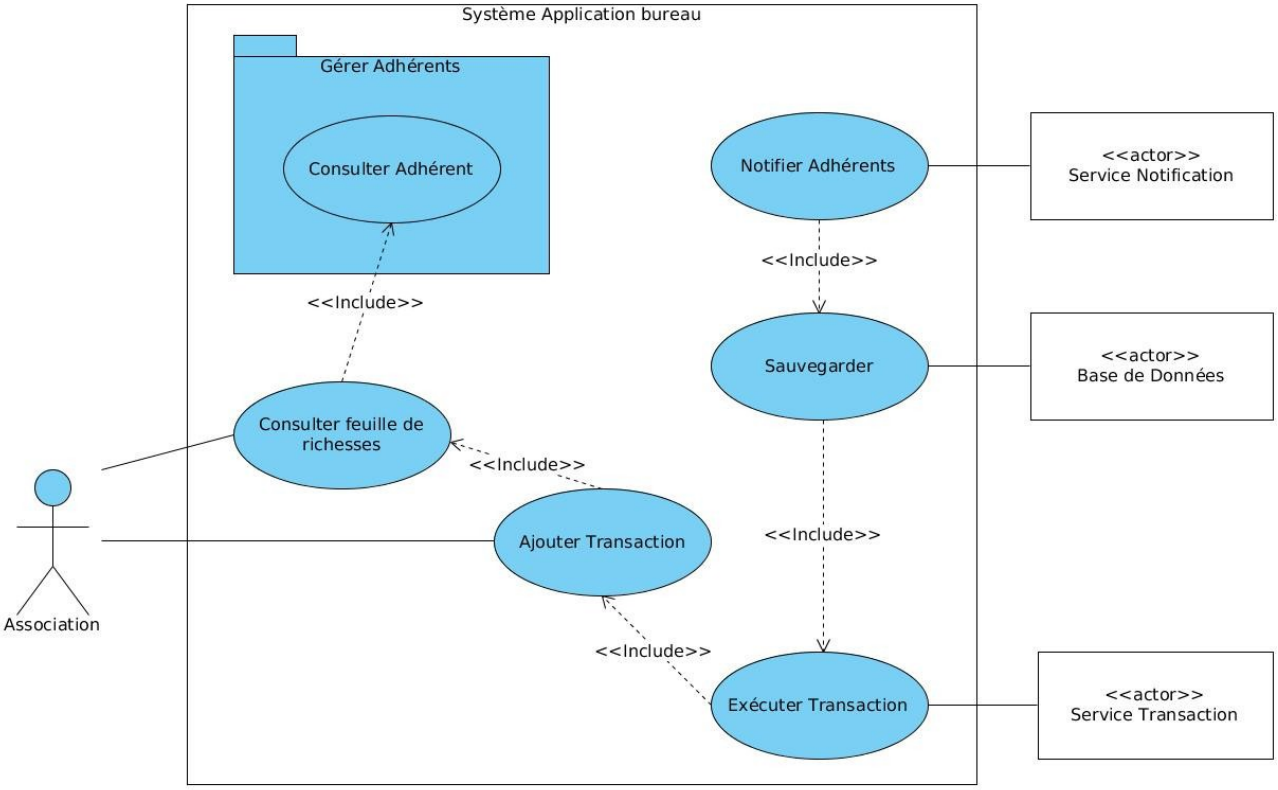
5. une requête de type CREATE est envoyé à la base de donnée pour sauvegarder le message : → cas d'utilisation « Sauvegarder »

- *Scénarii alternatifs* :

  4a. l'association annule la saisie du message: retour à l'écran précédent

3.6 Application Web côté bureau : gérer les feuilles de richesses

3.6.1 Diagramme



Cas d'utilisation: Application Web côté bureau - Gérer feuilles de richesses

3.6.2 Afficher une feuille de richesses

- *Événement déclencheur* : l'association choisit «Gérer Feuilles de Richesses» dans la fiche de consultation de l'adhérent
- *Pré Condition* : accès au serveur, feuille de richesse existante pour l'adhérent
- *Post Condition* : intégrité des données, données non modifiées, Feuille de richesse affichée

- Scénario nominal :

  1. l'association sélectionne «Gérer Feuilles de Richesses» dans le menu

  2. l'application affiche la feuille de richesse

  3. l'association sélectionne « retour »

  4. la fiche de l'adhérent s'affiche

- *Scénarii alternatifs* :

  3a. l'association ajoute une transaction (un paiement entre adhérent : → cas d'utilisation « Ajouter transaction »

  3b. l'association modifie une transaction : → cas d'utilisation «Modifier Transaction »

3.6.3 Ajouter Transaction

- *Événement déclencheur* : l'association choisit «Ajouter Transaction» dans la fiche feuille de richesse
- *Pré Condition* : cas d'utilisation « Consulter feuille de richesses » en cours
- *Post Condition* : envoi d'un message au Service Transaction pour qu'il réalise la transaction effective
- Scénario nominal :

  1. l'association sélectionne « Ajout Transaction » dans la feuille de richesse

  2. un écran de saisie de transaction apparaît

  3. l'association saisit la transaction

  4. l'association valide la transaction : → cas d'utilisation « Exécuter Transaction »

- *Scénarii alternatifs* :

  4a. l'association annule la transaction : retour à la feuille de richesses

3.6.4 Modifier Transaction

- *Événement déclencheur* : l'association choisit «Modifier 1 Transaction» dans la fiche feuille de richesse
- *Pré Condition* : cas d'utilisation « Consulter feuille de richesses » en cours et 1 transaction existe
- *Post Condition* : envoi d'un message au Service Transaction pour qu'il modifie la transaction effective
- Scénario nominal :

  1. l'association sélectionne une transaction dans la feuille de richesse
  2. un écran de saisie de transaction pré rempli apparaît
  3. l'association modifie la transaction
  4. l'association valide la transaction : → cas d'utilisation « Exécuter Transaction »

- *Scénarii alternatifs* :

  4a. l'association annule la modification: retour à la feuille de richesses

3.6.5 Exécuter Transaction

- *Événement déclencheur* : l'association a ajouté ou modifié une transaction dans une feuille de richesse
- *Pré Condition* : cas d'utilisation « Ajouter Transaction» ou « Modifier Transaction » a terminé le scénario nominal
- *Post Condition* : modification des feuilles de richesses des adhérents concernés en base de donnée suivant les règles ***ACID***, le compte de l'un doit être augmenté d'un montant égal à la baisse du compte de l'autre adhérent, la feuille de richesse de l'autre adhérent concerné par la transaction doit être également modifiée.
- Scénario nominal :

  1. un message contenant toutes les informations nécessaires à la transaction est envoyé au service Transaction
  2. le service de transaction construit les requêtes à exécuter en base de données pour réaliser la transaction

3. le service de transaction regroupe ces requêtes dans une opération ayant les caractéristiques ***ACID***

4. le service de transaction envoie l'opération à la base de donnée : → cas d'utilisation « Sauvegarder »

5. le service de transaction reçoit un message comme quoi la sauvegarde a bien été réalisée

6. le service de transaction envoie un message à l'association comme quoi l'opération s'est bien passée

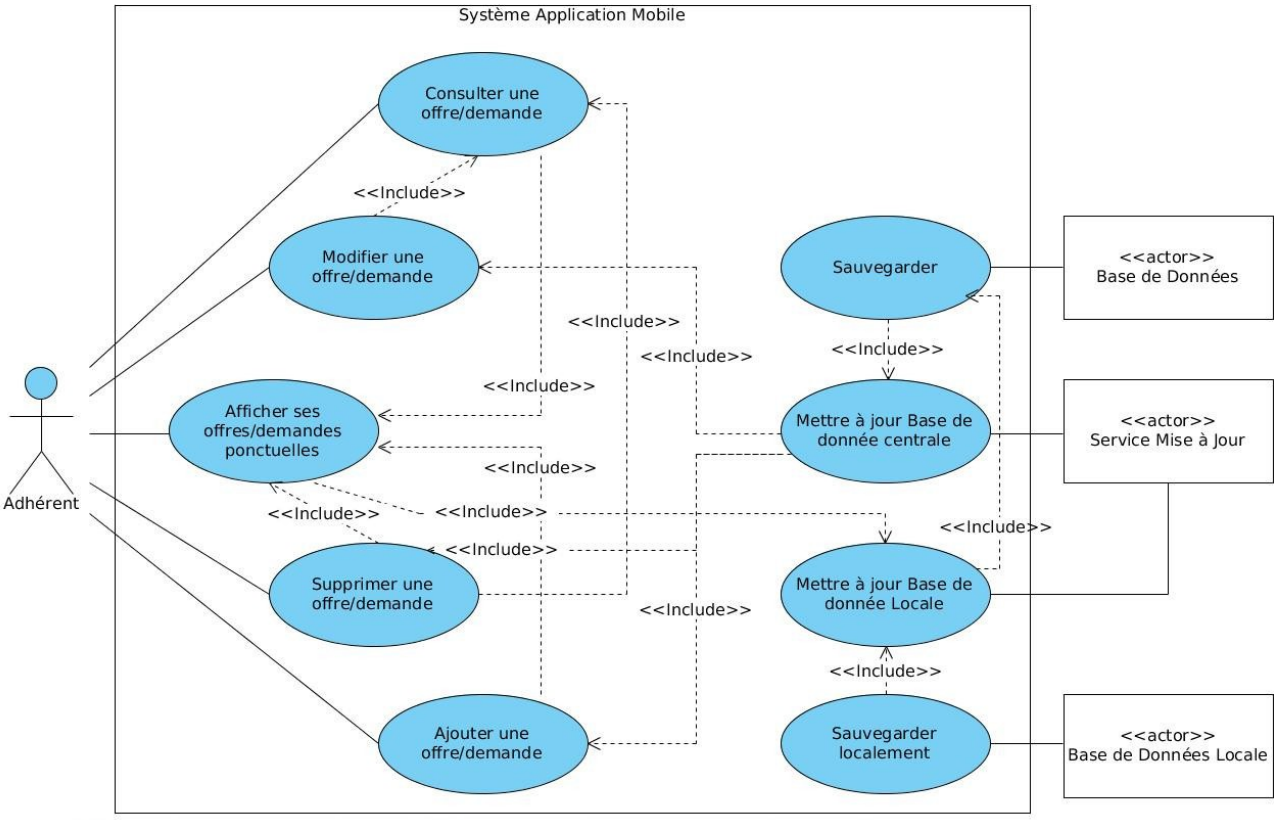
- *Scénarii alternatifs* :

5a. le service de transaction reçoit un message comme quoi la sauvegarde ne s'est pas bien passée

6.a le service de transaction envoie un message d'erreur à l'association

3.7 Application Mobile : gérer ses offres et demandes

3.7.1 Diagramme



Cas d'utilisation: Application Mobile - Gérer ses Offres et Demandes

3.7.2 Afficher ses offres/demandes ponctuelles

- *Événement déclencheur* : l'adhérent choisit «Gérer ses offres/demandes ponctuelles» dans le menu de l'accueil.
- *Pré Condition* : avoir une offre ou demande ponctuelle en cours dans son profil, mise à jour de sa base locale
- *Post Condition* : intégrité des données, données non modifiées
- Scénario nominal :

  1. l'adhérent sélectionne «Gérer ses offres/demandes ponctuelles» dans le menu
  2. l'application affiche un écran avec la liste correspondante
  3. l'adhérent fait dérouler la liste avec son doigt
  4. l'adhérent retourne à l'accueil

- *Scénarii alternatifs* :

  2a. pas d'offre ou demande enregistrées, affichage d'un message informatif

  4a. l'adhérent clique sur une offre/demande: → cas d'utilisation « consulter une offre/demande»

  4b. l'adhérent clique sur « ajouter une offre/demande» : → cas d'utilisation « Ajouter une offre/demande»

  4c. l'adhérent clique sur « supprimer » : → cas d'utilisation « Supprimer une offre/demande»

3.7.3 Ajouter une Offre/Demande

- *Événement déclencheur* : l'adhérent choisit «Ajouter une offre/demande» dans l'écran d'affichage de la liste
- *Pré Condition* : cas d'utilisation « Afficher ses offres/demandes» terminé
- *Post Condition* : intégrité des données, données ajoutées en base centrale et envoie de notifications si ok puis en base locale, données non modifiées sinon,
- Scénario nominal :

  1. l'adhérent sélectionne «Ajouter une Offre/demande» dans le menu
  2. l'application affiche un formulaire de saisie d'offre/demande

3. l’adhérent saisit le formulaire

4. l’adhérent valide le formulaire

5. une fenêtre de confirmation apparaît

6. l’adhérent confirme

7. mise à jour de la base de donnée centrale: → cas d’utilisation «Mise à jour Base de donnée centrale»

8. sauvegarde en base de donnée : → cas d’utilisation « Sauvegarder »

9. notification des Adhérents : → cas d’utilisation « Notifier Adhérents »

10. retour à l’écran affichant la liste des offres et demandes modifiés

- *Scénarii alternatifs* :

  4a. l’adhérent annule retour à la liste

  6a. l’adhérent annule la confirmation de création : retour à la saisie du formulaire

  7a. pas d’accès à internet : message à envoyer au service de mise à jour mis en attente dans une file et sauvegarde de la modification en base locale : → cas d’utilisation « Sauvegarder Localement »

  8a. la sauvegarde ne réussit pas

  9a. la notification n’est pas envoyée

  10a. un message d’erreur de création de l’adhérent apparaît, retour à la liste

3.7.4 Consulter une offre/demande

- *Événement déclencheur* : l’adhérent choisit «Consulter une offre/demande» dans l'écran d'affichage de la liste
- *Pré Condition* : cas d’utilisation « Affichage de ses offres/demandes » affichent une liste des offres/demandes, cette liste ne doit pas être vide (au moins une offre/demande doit être créé)
- *Post Condition* : fiche de l’offre/demande affichée
- Scénario nominal :

  1. l’adhérent sélectionne une offre/demande de la liste affichée

2. la fiche offre/demande apparaît

3. l'adhérent consulte la fiche

4. l'adhérent retourne à la liste initiale

- *Scénarii alternatifs* :

  4a. l'association supprime la fiche : → cas d'utilisation « Supprimer une offre/demande»

  4b. l'association modifie la fiche : → cas d'utilisation « Mettre à jour une offre/demande»

3.7.5 Supprimer une offre/demande

- *Événement déclencheur* : l'adhérent choisit «Supprimer une offre/demande» dans l'écran d'affichage de la liste ou dans la fiche de consultation de l'offre/demande
- *Pré Condition* : cas d'utilisation « Consulter une offre/demande» en cours ou affichage de la liste terminée
- *Post Condition* : intégrité des données, données supprimées en base et envoie de notifications si ok, données non modifiées sinon
- Scénario nominal :

  1. l'adhérent sélectionne « supprimer »

  2. une fenêtre de confirmation apparaît

  3. l'adhérent confirme la suppression

  4. l'adhérent retourne à la liste

  5. mise à jour de la base de donnée centrale: → cas d'utilisation «Mise à jour Base de donnée centrale»

  6. sauvegarde en base de donnée : → cas d'utilisation « Sauvegarder »

  7. notification des Adhérents : → cas d'utilisation « Notifier Adhérents »

  8. retour à l'écran affichant la liste des offres et demandes modifiés

- *Scénarii alternatifs* :

  3a. l'association annule la suppression : retour à l'écran précédent

  5a. pas d'accès à internet : message à envoyer au service de mise à jour mis en attente dans une file et sauvegarde de la modification en base locale : → cas d'utilisation « Sauvegarder Localement »

3.7.6 Mettre à Jour une offre/demande

- *Événement déclencheur* : l'adhérent choisit «Modifier une offre/demande» dans l'écran d'affichage de la liste ou dans la fiche de consultation de l'offre/demande
- *Pré Condition* : cas d'utilisation « Consulter une offre/demande» en cours
- *Post Condition* : intégrité des données, données modifiées en base et envoie de notifications si ok, données non modifiées sinon
- Scénario nominal :

  1. l'adhérent sélectionne «Mettre à Jour une offre/demande» dans le menu

  2. l'application affiche un formulaire de saisie offre/demande pré rempli avec les données de la fiche

  3. l'adhérent saisit le formulaire

  4. l'adhérent valide le formulaire

  5. une fenêtre de confirmation apparaît

  6. l'adhérent confirme

  7. mise à jour de la base de donnée centrale: → cas d'utilisation «Mise à jour Base de donnée centrale»

  8. sauvegarde en base de donnée : → cas d'utilisation « Sauvegarder »

  9. notification des Adhérents : → cas d'utilisation « Notifier Adhérents »

  10. retour à l'écran affichant la liste des offres et demandes modifiés

- *Scénarii alternatifs* :

  4a. l'adhérent annule retour à la liste

  6a. l'adhérent annule la confirmation de modification: retour à la saisie du formulaire

7a. pas d'accès à internet : message à envoyer au service de mise à jour mis en attente dans une file et sauvegarde de la modification en base locale : → cas d'utilisation « Sauvegarder Localement »

8a. la sauvegarde ne réussit pas

9a. la notification n'est pas envoyée

10a. un message d'erreur de création de l'offre/demande apparaît, retour à la liste

3.7.7 Mettre à Jour base de donnée centrale

- *Événement déclencheur* : une modification en base de donnée est sollicitée par l'adhérent par l'intermédiaire de l'application mobile
- *Pré Condition* : cas d'utilisation ayant pour but de modifier la base de données s'étant déroulés suivant le scénario nominal
- *Post Condition* : envoi d'une requête à la base de donnée
- Scénario nominal :

  1. le service de mise à jour reçoit un message comme quoi il faut modifier la base de donnée centrale

  2. le service de mise à jour traduit le message en requête de type CREATE ou UPDATE ou DELETE l'opération est transactionnelle

  3. le service de mise à jour regroupe les requêtes en une opération ***ACID***

  4. le service de mise à jour envoie l'opération à la base de donnée : → cas d'utilisation « Sauvegarder »

  5. le service de mise à jour reçoit un message comme quoi la sauvegarde a bien été réalisée

  6. le service de mise à jour envoie un message à l'adhérent comme quoi l'opération s'est bien passée

- *Scénarii alternatifs* :

  5a. le service de mise à jour reçoit un message comme quoi la sauvegarde ne s'est pas bien passée

  6.a le service de mise à jour envoie un message d'erreur à l'adhérent

3.7.8 Sauvegarder

- *Événement déclencheur* : l'adhérent confirme une modification en base de donnée
- *Pré Condition* : le service de mise à jour envoie une requête
- *Post Condition* : intégrité des données, données modifiées en base et envoie de notifications si ok, données non modifiées sinon,
- Scénario nominal :

  1. une requête de type CREATE ou UPDATE ou DELETE est exécutée en base de donnée

  2. l'opération est transactionnelle

  3. l'opération réussit

  4. une réponse comme quoi une modification de la base de données s'est bien déroulée est envoyé au service de notification : → cas d'utilisation « Notifier Adhérent »

  5. une réponse comme quoi l'opération s'est bien déroulée est envoyé au service de mise à jour.

- *Scénarii alternatifs* :

  3a. l'opération ne réussit pas

  4a. une réponse comme quoi l'opération n'a pu être réalisée est envoyée au service de mise à jour : transfert de l'erreur à l'adhérent

3.7.9 Mettre à Jour base de donnée locale

- *Événement déclencheur* : le service de mise à jour vérifie si de nouvelles données sont présentes en base centrale
- *Pré Condition* : de nouvelles données sont présentes en base centrale
- *Post Condition* : envoi d'une requête à la base de donnée locale
- Scénario nominal :

  1. le service de mise à jour récupère les nouvelles données de la base centrale

2. le service de mise à jour traduit le message en requête l'opération est transactionnelle

3. le service de mise à jour envoie l'opération aux bases de données locales: → cas d'utilisation « Sauvegarder Localement»

4. le service de mise à jour reçoit un message comme quoi la sauvegarde a bien été réalisée

- *Scénarii alternatifs* :

  4a. le service de mise à jour reçoit un message comme quoi la sauvegarde ne s'est pas bien passée

3.7.10 Sauvegarder Localement

- *Événement déclencheur* : l'adhérent confirme une modification en base de donnée ou le service de mise à jour met à jour la base locale
- *Pré Condition* : base locale accessible
- *Post Condition* : intégrité des données, données modifiées en base
- Scénario nominal :

  1. une requête de type CREATE ou UPDATE ou DELETE est exécutée en base de donnée locale

  2. l'opération est transactionnelle

  3. l'opération réussit

- *Scénarii alternatifs* :

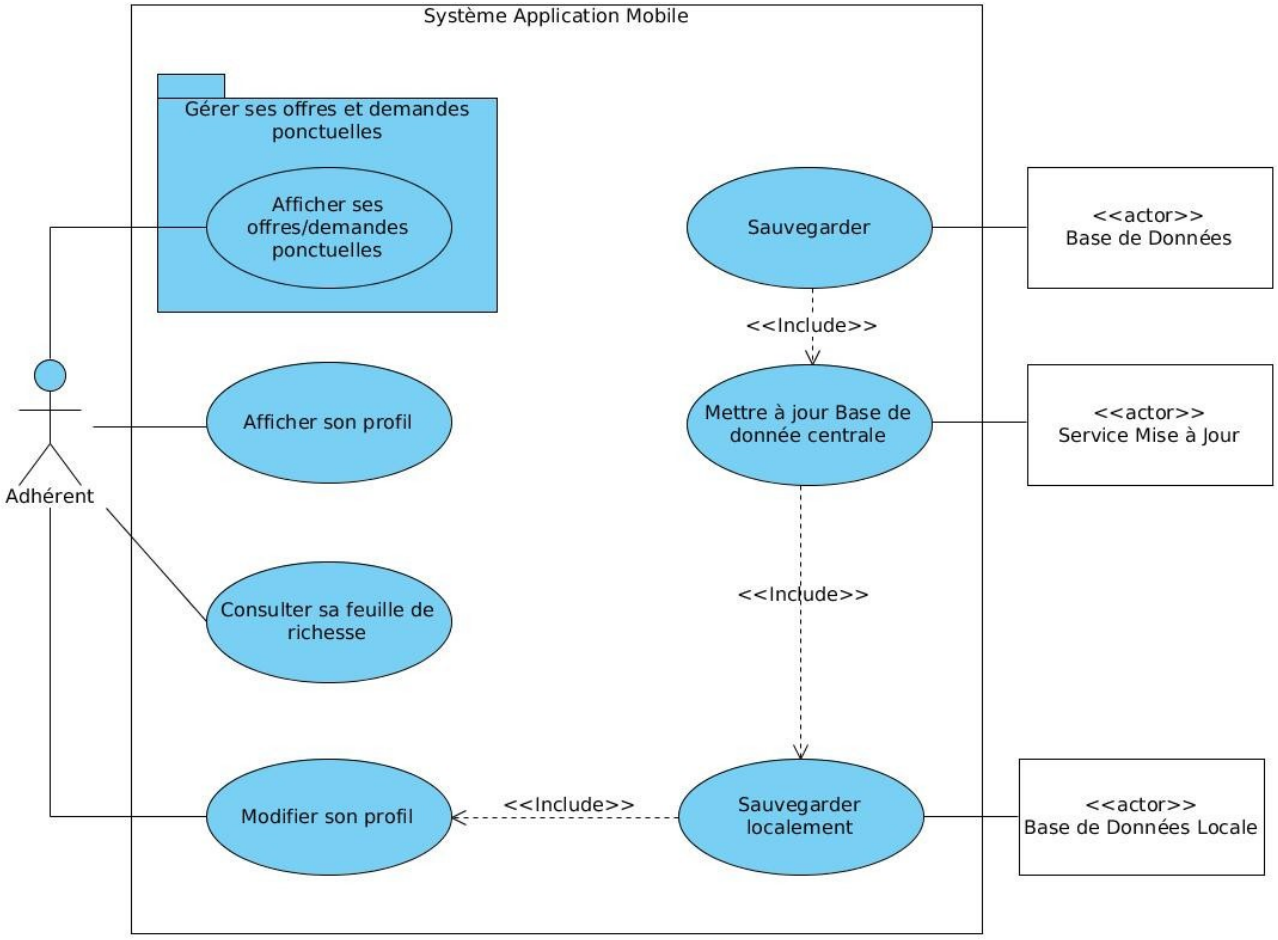
  3a. l'opération ne réussit pas

  4. une réponse comme quoi l'opération n'a pu être réalisée est envoyée au service de mise à jour : l'opération est stockée dans une file

  5. tant que la file n'est pas vide, les opérations sont retentées

3.8 Application Mobile : gérer son profil

3.8.1 Diagramme



Cas d'utilisation: Application Mobile - gérer son profil

3.8.2 Afficher son profil

- *Événement déclencheur* : l'adhérent choisit «Gérer son profil» dans le menu de l'accueil.
- *Pré Condition* : profil créé par l'association
- *Post Condition* : intégrité des données, données non modifiées
- Scénario nominal :

  1. l'adhérent sélectionne «Gérer son profil» dans le menu

2. l’application affiche un écran avec son profil

3. l’adhérent consulte son profil

4. l’adhérent retourne à l’accueil

- *Scénarii alternatifs* :

  4a. l’adhérent clique sur afficher ses offres/demandes: → cas d’utilisation « afficher ses offres/demandes»

  4b. l’adhérent clique sur « feuille de richesse» : → cas d’utilisation « Consulter sa feuille de richesse »

  4c. l’adhérent clique sur «modifier» : → cas d’utilisation « modifier son profil»

3.8.3 Consulter sa feuille de richesse

- *Événement déclencheur* : l’adhérent choisit ««« feuille de richesse » dans son profil.
- *Pré Condition* : cas d’utilisation « Affichage de son profil » en cours, feuille de richesse créée
- *Post Condition* : feuille de richesse affichée
- Scénario nominal :

  1. l’adhérent sélectionne « feuille de richesse » dans son profil

  2. la feuille de richesse apparaît

  3. l’adhérent consulte la feuille

  4. l’adhérent retourne à son profil

- *Scénarii alternatifs* :

  Sans Objet

3.8.4 Modifier son profil

- *Événement déclencheur* : l’association choisit «Modifier son profil» dans l'écran d'affichage du profil
- *Pré Condition* : cas d’utilisation « Afficher son profil» en cours

- *Post Condition* : intégrité des données, données modifiées en base et envoie de notifications si ok, données non modifiées sinon
- *Scénario nominal :*

  1. l'adhérent sélectionne «Mettre à Jour » dans son profil

  2. l'application affiche un formulaire de saisie de son profil pré rempli avec les données du profil existant

  3. l'adhérent saisit le formulaire

  4. l'adhérent valide le formulaire

  5. mise à jour de la base de donnée centrale: → cas d'utilisation «Mise à jour Base de donnée centrale»

  6. sauvegarde en base de donnée : → cas d'utilisation « Sauvegarder »

  7. notification des Adhérents : → cas d'utilisation « Notifier Adhérents »

  8. retour à l'écran affichant la liste des offres et demandes modifiés

- *Scénarii alternatifs* :

  4a. l'adhérent annule retour à la liste

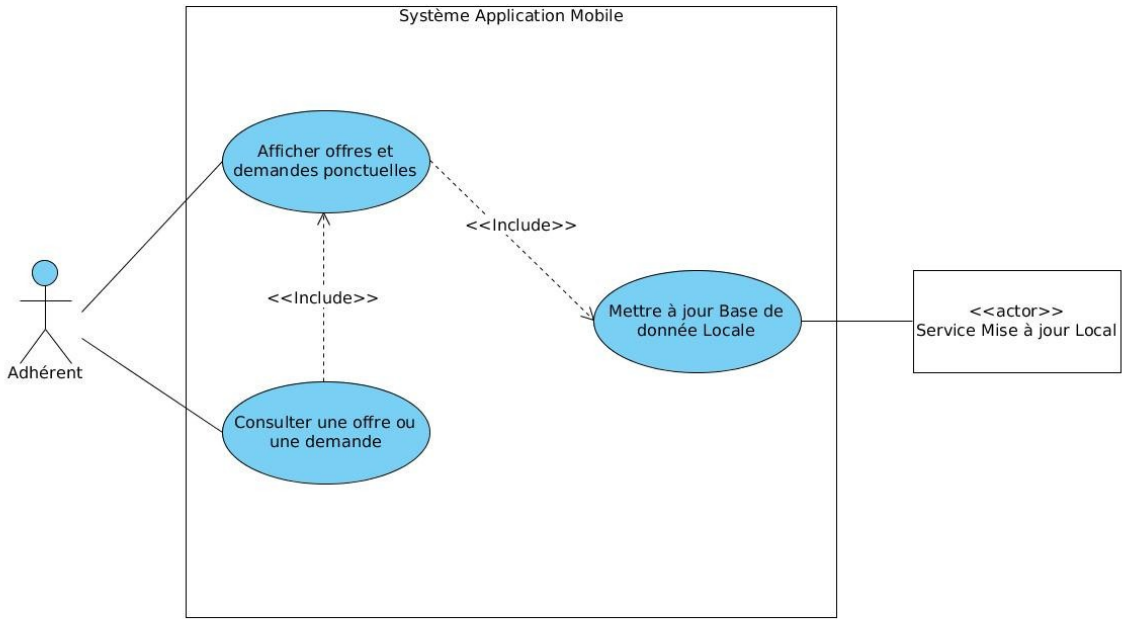
  5a. pas d'accès à internet : message à envoyer au service de mise à jour mis en attente dans une file et sauvegarde de la modification en base locale : → cas d'utilisation « Sauvegarder Localement »

  6a. la sauvegarde ne réussit pas

  7a. la notification n'est pas envoyée

3.9 Application Mobile : consulter les offres et demandes ponctuelles

3.9.1 Diagramme



Cas d'utilisation: Application Mobile - consulter les offres et demandes

3.9.2 Afficher Offres/Demandes

- *Événement déclencheur* : l'adhérent choisit «Afficher Offres/Demandes » dans le menu de l'accueil.
- *Pré Condition* : offres/demandes non vide sinon message en conséquence
- *Post Condition* : intégrité des données, données non modifiées
- Scénario nominal :

  1. l'adhérent sélectionne «Afficher Offres/Demandes ponctuelles» dans le menu
  2. l'application affiche un écran avec la liste des offres et demandes
  3. l'adhérent consulte la liste en la faisant défiler avec le doigt
  4. l'adhérent retourne à l'accueil

- *Scénarii alternatifs* :

  4a. l'adhérent sélectionne une offre/demande : → cas d'utilisation « Consulter une offre ou une demande »

3.9.3 Consulter une Offre ou une Demande

- *Événement déclencheur* : l'adhérent sélectionne une offre/demande de la liste
- *Pré Condition* : aucune
- *Post Condition* : intégrité des données, données non modifiées
- Scénario nominal :

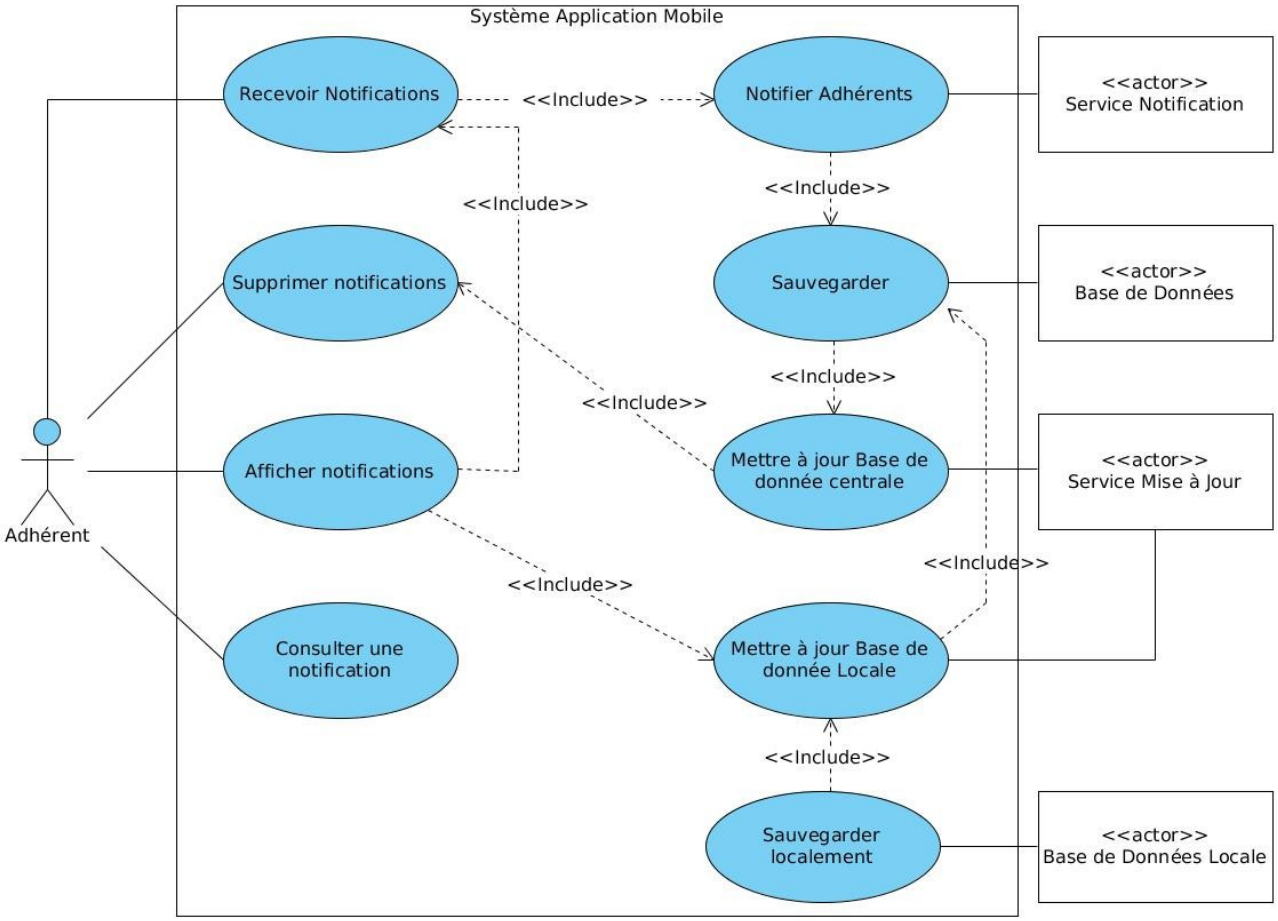
  1. l'adhérent sélectionne une offre ou une demande dans le menu
  2. l'application affiche un écran de l'offre ou la demande
  3. l'adhérent consulte l'offre ou la demande
  4. l'adhérent retourne à l'accueil

- *Scénarii alternatifs* :

  Sans Objet

3.10 Application Mobile : recevoir et consulter ses notifications

3.10.1 Diagramme



Cas d'utilisation: Application Mobile - Recevoir et Consulter ses notifications

3.10.2 Recevoir Notifications

- *Événement déclencheur* : le service de notification interroge la base centrale
- *Pré Condition* : avoir accès à internet, de nouvelles notifications sont présentes
- *Post Condition* : notifications signalées sur le smartphone de l'adhérent
- Scénario nominal :

  1. le service de notification envoie le message de notification à la base locale
  2. la base locale enregistre les nouvelles notifications
  3. la base locale envoie les nouvelles notifications à l'application

4. l'application affiche un icône sur le smartphone de l'adhérent

- *Scénarii alternatifs* :

  sans objet

3.10.3 Afficher ses Notifications

- *Événement déclencheur* : l'adhérent choisit de cliquer sur l'icône de notification dans le menu de l'accueil.
- *Pré Condition* : avoir une notification signalée sur son application, cas d'utilisation « Recevoir notifications » terminé
- *Post Condition* : notifications affichées
- Scénario nominal :

  1. l'adhérent sélectionne «Consulter ses notifications» dans le menu
  2. l'application affiche un écran avec la liste correspondante
  3. l'adhérent fait dérouler la liste avec son doigt
  4. l'adhérent retourne à l'accueil

- *Scénarii alternatifs* :

  4a. l'adhérent clique sur une notification: → cas d'utilisation « consulter une notification»

  4b. l'adhérent sélectionne des notifications pour les supprimer : → cas d'utilisation « Supprimer notifications»

3.10.4 Consulter une notification

- *Événement déclencheur* : l'adhérent sélectionne une notification de la liste
- *Pré Condition* : liste des notifications affichée et non vide
- *Post Condition* : notification affichée
- Scénario nominal :

  1. l'adhérent sélectionne une notification dans la liste

2. l'application affiche un écran sur la notification

3. l'adhérent consulte la notification

4. l'adhérent retourne à la liste

- *Scénarii alternatifs* :

  4a. l'adhérent supprime la notification : → cas d'utilisation « Supprimer notifications »

3.10.5 Supprimer notifications

- *Événement déclencheur* : l'adhérent supprime la notification dans la fiche ou a sélectionné des notifications dans la liste
- *Pré Condition* : cas d'utilisation « consulter une notification» ou « afficher notifications » en cours
- *Post Condition* : intégrité des données, données supprimées en base si ok, données non modifiées sinon
- Scénario nominal :

  1. l'adhérent sélectionne « supprimer »

  2. une fenêtre de confirmation apparaît

  3. l'adhérent confirme la suppression

  4. l'adhérent retourne à la liste

  5. mise à jour de la base de donnée locale: → cas d'utilisation «Mise à jour Base de donnée locale»

  6. sauvegarde en base de donnée locale: → cas d'utilisation « Sauvegarder Localement»

  7 retour à l'écran affichant la liste avec message de confirmation de suppression

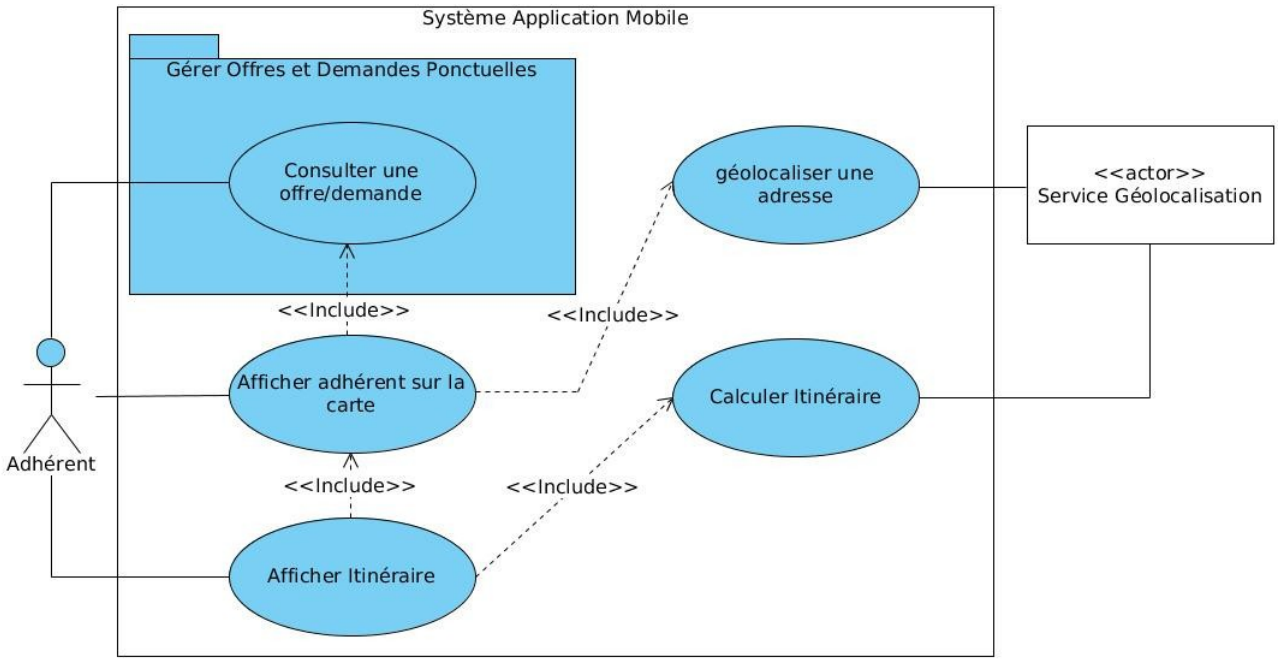
- *Scénarii alternatifs* :

  3a. l'association annule la suppression : retour à l'écran précédent

  7a. problème de suppression en base locale : envoie d'un message d'erreur

3.11 Application Mobile : géolocaliser un adhérent

3.11.1 Diagramme



Cas d'utilisation: Application Mobile - géolocaliser un adhérent

3.11.2 Afficher adhérent sur la carte

- *Événement déclencheur* : l'adhérent sélectionne géolocaliser lorsqu'il consulte une offre/demande d'un adhérent
- *Pré Condition* : la liste des offres/demandes ponctuelles n'est pas vide, la géolocalisation du smartphone est activée, accès à internet
- *Post Condition* : carte avec adresse adhérent positionnée affichée
- Scénario nominal :

  1. l'adhérent sélectionne «géolocaliser» dans la fiche offre/demande

  2. l'adresse de l'adhérent concerné est envoyée au service de géolocalisation ; → cas d'utilisation « Géolocaliser une adresse »

  3. l'adhérent voit s'afficher une carte avec la position de l'adhérent concerné par l'offre/demande

4. l'adhérent retourne à la fiche offre/demande

- *Scénarii alternatifs* :

  4a. l'adhérent choisit de calculer un itinéraire à partir de sa position jusqu'à celle de l'adhérent concerné : → cas d'utilisation : « Afficher itinéraire sur la carte »

3.11.3 Afficher itinéraire sur la carte

- *Événement déclencheur* : l'adhérent sélectionne « calculer itinéraire » lorsqu'il consulte la carte avec la position d'un adhérent
- *Pré Condition* : le cas d'utilisation « Afficher adhérent sur la carte » est terminé
- *Post Condition* : carte avec itinéraire affichée
- Scénario nominal :

  1. l'adhérent sélectionne «calculer itinéraire» sur la carte

  2. l'adresse de l'adhérent concerné et celle du smartphone sont envoyées au service de géolocalisation ; → cas d'utilisation « Calculer un itinéraire »

  3. l'adhérent voit s'afficher une carte avec l'itinéraire de sa position jusqu'à l'adhérent concerné par l'offre/demande

  4. l'adhérent retourne à la fiche offre/demande

- *Scénarii alternatifs* :

  Sans Objet

3.11.4 Géolocaliser une adresse

- *Événement déclencheur* : une demande est envoyée au service de géolocalisation par l'adhérent
- *Pré Condition* : la liste des offres/demandes ponctuelles n'est pas vide, la géolocalisation du smartphone est activée, accès à internet
- *Post Condition* : envoie des informations nécessaires à l'affichage d'une carte avec adresse adhérent positionnée
- Scénario nominal :

  1. le service de géolocalisation reçoit une adresse

2. le service de géolocalisation transforme l'adresse en coordonnées GPS

3. le service de géolocalisation envoie ces coordonnées à l'application mobile

- *Scénarii alternatifs* :

  2a. le service de géolocalisation n'arrive pas à récupérer les coordonnées

  3a. envoie d'un message d'erreur à l'application mobile

3.11.5 Calculer un itinéraire

- *Événement déclencheur* : une demande est envoyée au service de géolocalisation par l'adhérent
- *Pré Condition* : le cas d'utilisation « Afficher adhérent sur la carte » est terminé, la géolocalisation du smartphone est activée, accès à internet
- *Post Condition* : envoie des informations nécessaires à l'affichage d'une carte avec l'itinéraire demandé
- Scénario nominal :

  1. le service de géolocalisation reçoit deux adresses

  2. le service de géolocalisation calcule un itinéraire entre ces 2 adresses

  3. le service de géolocalisation envoie l'itinéraire à l'application mobile

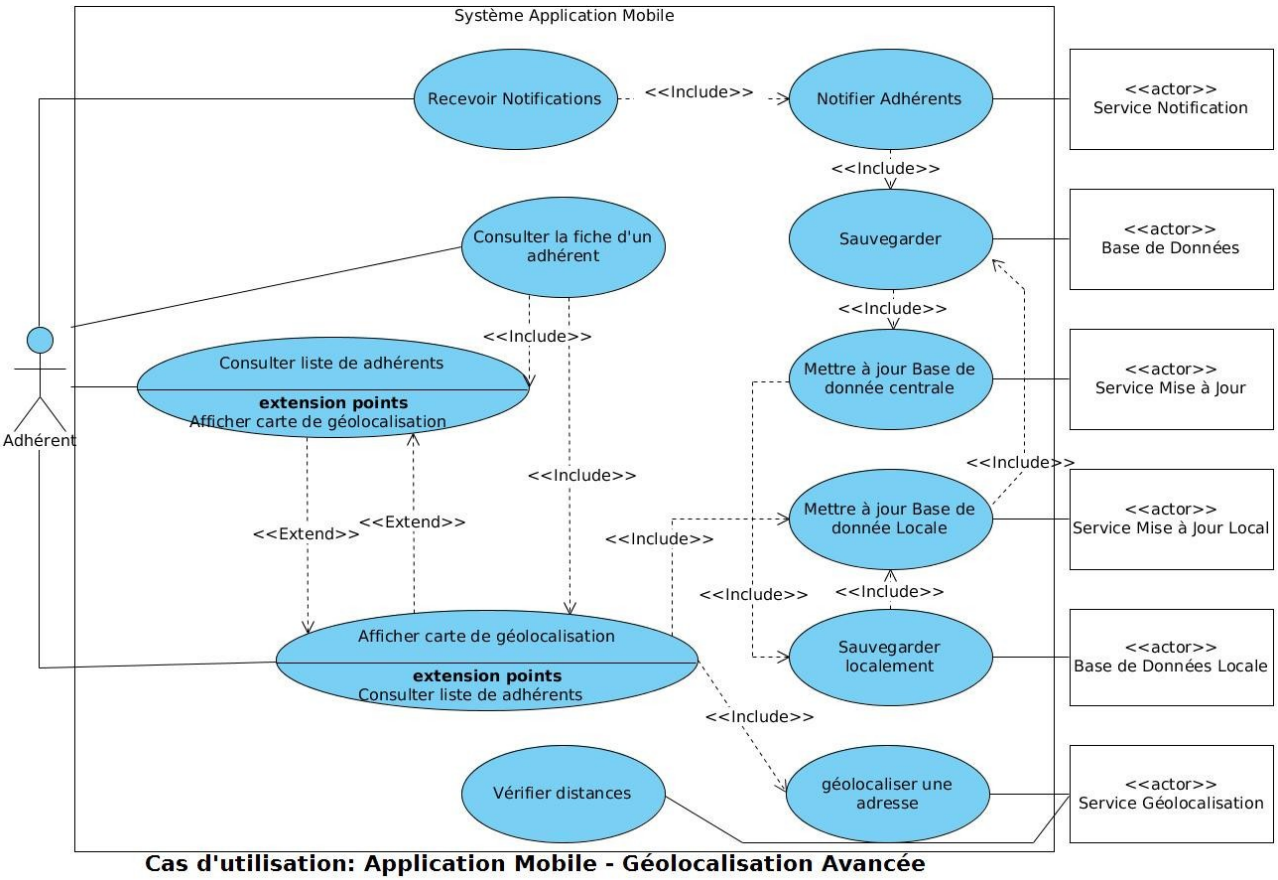
- *Scénarii alternatifs* :

  2a. le service de géolocalisation n'arrive pas à calculer l'itinéraire

  3a. envoie d'un message d'erreur à l'application mobile

3.12 Application Mobile : géolocalisation avancée

3.12.1 Diagramme



Cas d'utilisation: Application Mobile - Géolocalisation Avancée

3.12.2 Afficher carte de géolocalisation

- *Événement déclencheur* : l'adhérent sélectionne « carte géolocalisation » à l'accueil
- *Pré Condition* : la géolocalisation du smartphone est activée, accès à internet, les filtres appliqués retournent des adhérents
- *Post Condition* : carte avec adresse adhérent positionnée affichée
- Scénario nominal :

  1. l'adhérent sélectionne « carte géolocalisation » à l'accueil

  2. l'application mobile sélectionne une liste d'adhérent en fonction des filtres appliqués

3. l'application envoie la liste des adresses correspondantes aux adhérents sélectionnés au service de géolocalisation : → cas d'utilisation « géolocaliser une adresse »

4. l'application récupère les coordonnées

5. un écran géolocalisant tous les adhérents sélectionnés, ainsi que sa position apparaît

6. l'adhérent retourne à l'accueil

- *Scénarii alternatifs* :

  2a. l'adhérent choisit de désactiver les filtres, l'application envoie la liste de tous les adhérents au service de géolocalisation

  6a. l'adhérent peut consulter les adhérents sélectionnés sous forme de liste : → cas d'utilisation « Consulter la liste des adhérents »

  6a. l'adhérent peut zoomer, dé zoomer , sélectionner un adhérent : → cas d'utilisation « Consulter la fiche d'un adhérent »

3.12.3 Consulter la liste des adhérents

- *Événement déclencheur* : l'adhérent sélectionne « liste des adhérents » sur la carte de géolocalisation, ou s'affiche directement lorsque l'adhérent choisit la carte de géolocalisation et qu'il n'y a pas d'accès internet
- *Pré Condition* : les filtres appliqués retournent des adhérents
- *Post Condition* : liste des adhérents affichée non modifiée
- Scénario nominal :

  1. l'adhérent sélectionne « liste des adhérents » sur la carte

  2. l'application mobile affiche la liste des adhérents sélectionnés par les filtres

  3. l'adhérent peut faire défiler la liste avec les doigts

  4. l'adhérent retourne à la carte de géolocalisation (à l'accueil si pas accès internet)

- *Scénarii alternatifs* :

  2a. l'adhérent choisit de désactiver les filtres, l'application affiche la liste de tous les adhérents

  4a. l'adhérent peut zoomer, dé zoomer , sélectionner un adhérent : → cas d'utilisation « Consulter la fiche d'un adhérent »

3.12.4 Consulter la fiche d'un adhérent

- *Événement déclencheur* : l'adhérent sélectionne la fiche d'un adhérent sur la carte de géolocalisation, ou sur la liste des adhérents
- *Pré Condition* : les filtres appliqués retournent des adhérents
- *Post Condition* : fiche adhérent affichée non modifiée
- Scénario nominal :

  1. l'adhérent sélectionne un adhérent sur la carte ou sur la liste

  2. l'application mobile affiche la fiche adhérent

  3. l'adhérent consulte la fiche

  4. l'adhérent retourne à l'écran précédent

- *Scénarii alternatifs* :

  Sans Objet

3.12.5 Recevoir notifications

- *Événement déclencheur* : l'adhérent passe à proximité d'un adhérent respectant les filtres
- *Pré Condition* : accès internet, géolocalisation du mobile activée, avoir renseigné dans les filtres une distance à partir de laquelle on souhaite être notifié et lise des adhérents filtrés non vide
- *Post Condition* : notification envoyée sur le mobile
- Scénario nominal :

  1. l'application mobile envoie sa position au service de géolocalisation et la liste des adhérents filtrés : → cas d'utilisation « Vérifier Distances »

2. l’adhérent reçoit la notification

- *Scénarii alternatifs* :

  2a. l’adhérent ne passe jamais à proximité, il ne reçoit pas de notification

3.12.6 Vérifier Distances

- *Événement déclencheur* : accès internet, distance renseignée dans filtre et liste des adhérents filtrés non vide
- *Pré Condition* : aucune
- *Post Condition* : message envoyé au service de notification
- Scénario nominal :

  1. l’application mobile envoie sa position au service de géolocalisation et la liste des adhérents filtrés

  2. le service de géolocalisation calcule la distance entre les adhérents et le mobile

  3. le service de géolocalisation vérifie si une des distances est inférieure au seuil indiqué

  4. l’adhérent passe à proximité d’un adhérent respectant les conditions

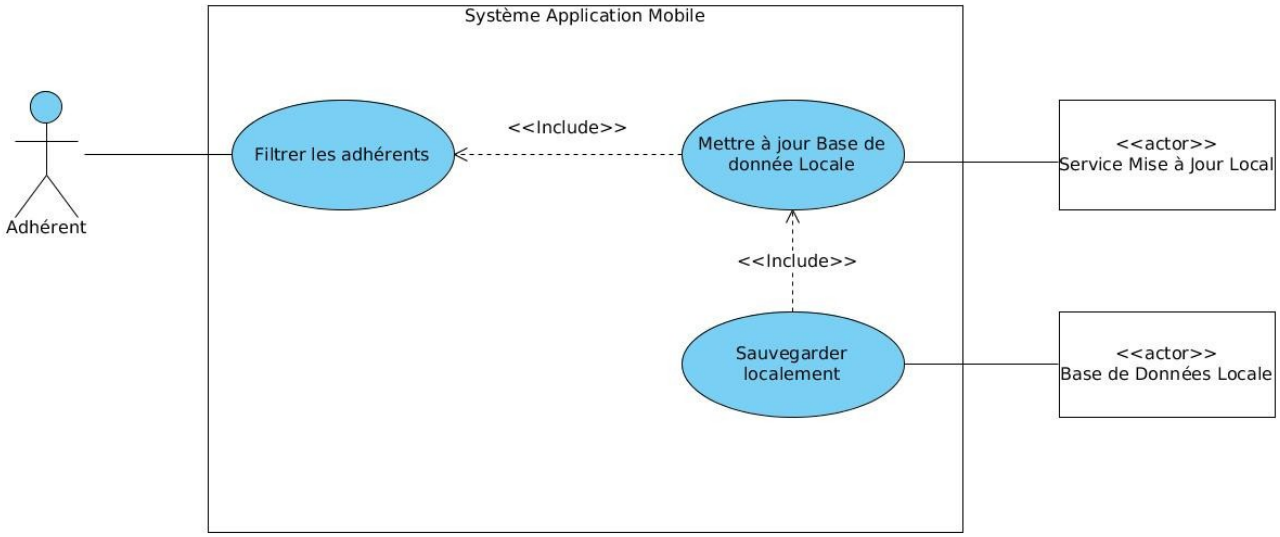
  5. le service de géolocalisation envoie un message au service de notification : → cas d’utilisation « Notifier Adhérent »

- *Scénarii alternatifs* :

  4a. l’adhérent ne passe jamais à proximité, aucun message n’est envoyée

3.13 Application Mobile : filtrage des adhérents

3.13.1 Diagramme



Cas d'utilisation: Application Mobile - Filtrage des adhérents

3.13.2 Filtrer adhérents

- *Événement déclencheur* : l'adhérent sélectionne « filtrer adhérents » à l'accueil
- *Pré Condition* : aucune
- *Post Condition* : base de donnée locale modifiée, intégrité des données
- Scénario nominal :

  1. l'adhérent sélectionne« filtrer adhérents »
  2. un écran de saisie des filtres s'affiche
  3. l'adhérent saisie ses filtres
  4. l'adhérent valide
  5. les filtres modifiés sont envoyés au service de mise à jour local : → cas d'utilisation « mise à jour de la base locale »
  6. le service de mise à jour local envoie une requête à la base de donnée locale : → cas d'utilisation « Sauvegarder Localement »
  7. un message est envoyé à l'adhérent comme quoi tout s'est bien passé

- *Scénarii alternatifs* :

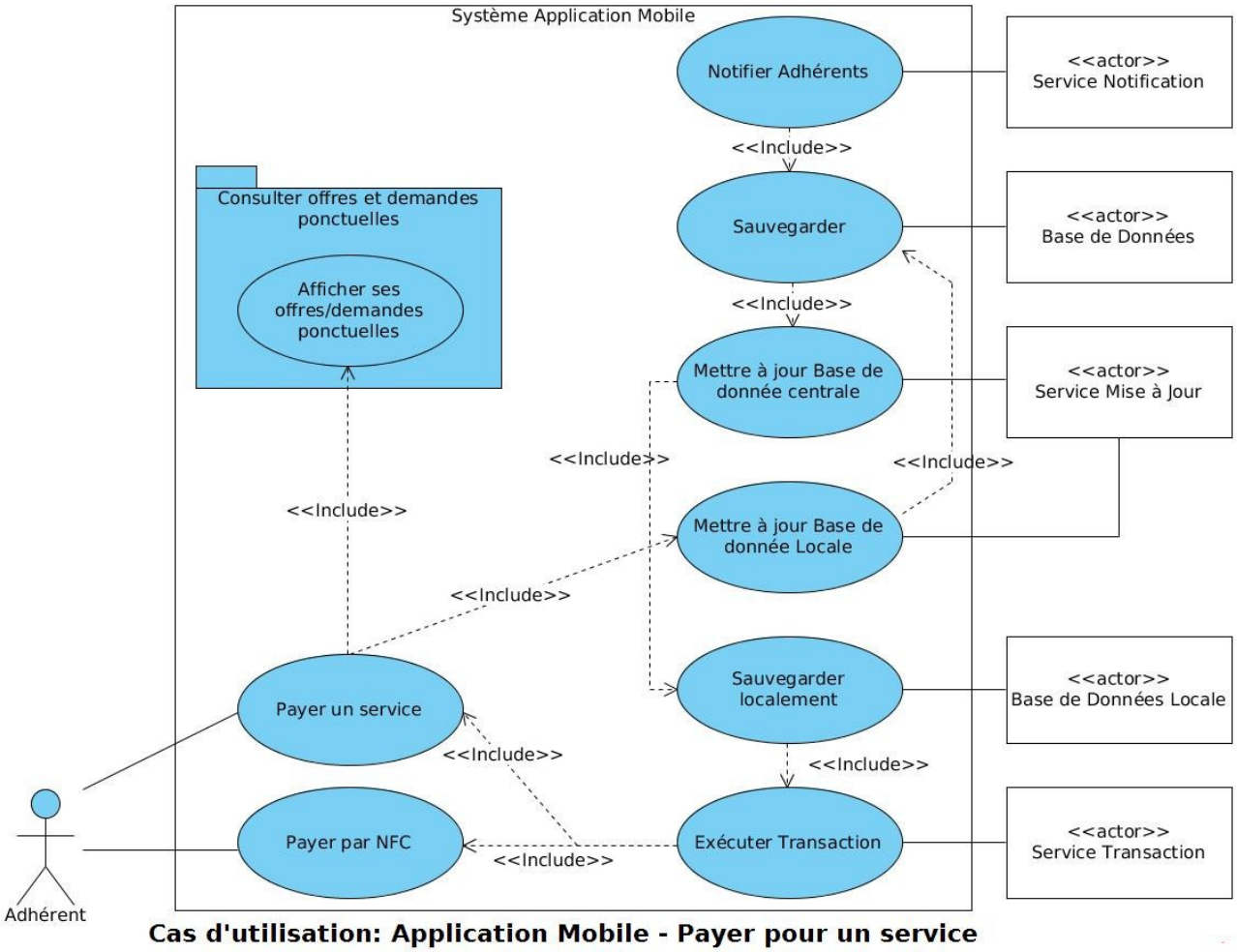
  4a l'adhérent annule, retour à l'accueil

  6a. la sauvegarde s'est mal passée

  7a. envoie d'un message d'erreur à l'adhérent

3.14 Application Mobile : payer pour un service

3.14.1 Diagramme



Cas d'utilisation: Application Mobile - Payer pour un service

3.14.2 Payer un service

- *Événement déclencheur* : l'adhérent choisit «payer» dans le menu de la liste des offres/demandes ou dans la fiche offre/demande
- *Pré Condition* : soit l'adhérent qui paye est à l'origine d'une demande, soit l'adhérent qui est payé est à l'origine d'une offre

- *Post Condition* : le compte de l'adhérent qui a payé sera diminué d'un montant égal à l'augmentation du compte de l'adhérent qui aura été payé, l'offre/demande concernée est supprimée
- Scénario nominal :

  1. l'adhérent qui paye sélectionne payer dans le menu de la liste des offres/demandes ou dans la fiche offre/demande

  2. un écran de paiement apparaît

  3. l'adhérent qui paye saisit le formulaire de paiement

  4. l'adhérent valide le paiement : → cas d'utilisation « Exécuter Transaction »

  5. tout s'est bien passé, les adhérents reçoivent une notification

- *Scénarii alternatifs* :

  4a. l'adhérent annule le paiement : retour à l'écran précédent

  5a. le paiement n'a pas pu se faire : envoi d'un message d'erreur

3.14.3 *Exécuter Transaction*

- *Événement déclencheur* : l'adhérent valide un paiement
- *Pré Condition* : accès internet, cas d'utilisation « Payer un service» a terminé le scénario nominal
- *Post Condition* : modification des feuilles de richesses des adhérents concernés en base de donnée suivant les règles ***ACID***, le compte de l'un doit être augmenté d'un montant égal à la baisse du compte de l'autre adhérent
- Scénario nominal :

  1. un message contenant toutes les informations nécessaires à la transaction est envoyé au service de transaction

  2. le service de transaction regroupe ces requêtes dans une opération ayant les caractéristiques ***ACID***

  3. le service de transaction envoie l'opération à la base locale → cas d'utilisation « Sauvegarder localement »

4. la base locale fait appel au service de mise à jour et lui transmet toutes les informations de la transaction : → cas d'utilisation « Mise à jour base centrale »

5. le service de mise à jour de l'adhérent qui a reçu le paiement interroge la base centrale : → cas d'utilisation « Mise à jour base locale »

6. le service de notification des 2 adhérents concernés par le paiement interroge la base centrale : cas d'utilisation « Notifier adhérent »

- *Scénarii alternatifs* :

  un enregistrement en base s'est mal passé. Nouvelle tentative.

3.14.4 Payer par NFC

- *Événement déclencheur* : l'adhérent choisit «payer» dans le menu de la liste des offres/demandes ou dans la fiche offre/demande
- *Pré Condition* : soit l'adhérent qui paye est à l'origine d'une demande, soit l'adhérent qui est payé est à l'origine d'une offre, ***NFC*** activé sur les 2 smartphones
- *Post Condition* : le compte de l'adhérent qui a payé sera diminué d'un montant égal à l'augmentation du compte de l'adhérent qui aura été payé, l'offre/demande concernée est supprimée
- Scénario nominal :

  1. l'adhérent qui paye sélectionne payer dans le menu de la liste des offres/demandes ou dans la fiche offre/demande

  2. un écran de paiement apparaît

  3. l'adhérent qui paye saisit le formulaire de paiement

  4. les adhérents échangent par ***NFC*** : → cas d'utilisation « Exécuter Transaction »

  5. tout s'est bien passé, les adhérents reçoivent une notification

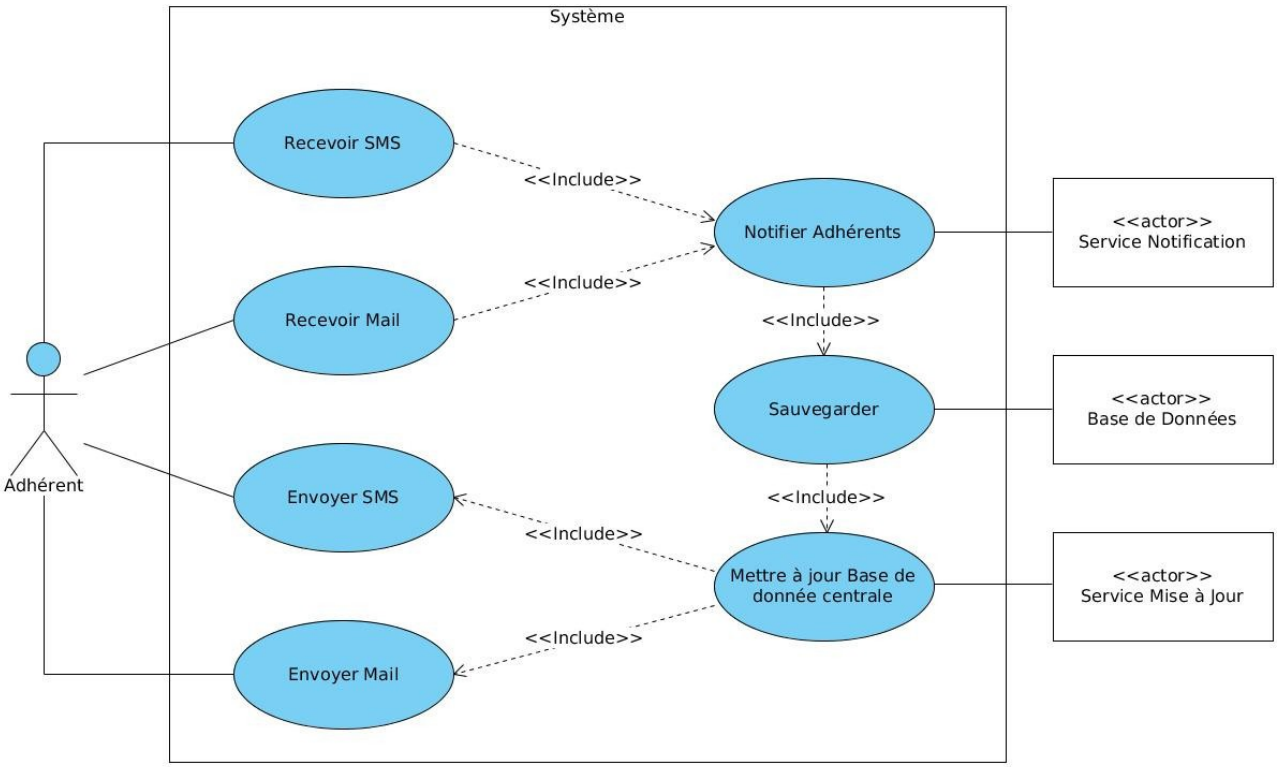
- *Scénarii alternatifs* :

  4a. l'adhérent annule le paiement : retour à l'écran précédent

  5a. le paiement n'a pas pu se faire : envoi d'un message d'erreur

3.15 Cas d'utilisation Envoyer et Recevoir SMS et Mail

3.15.1 Diagramme



Cas d'utilisation: SMS et Mail - Envoyer et Recevoir

3.15.2 Envoyer SMS

- *Événement déclencheur* : l'adhérent saisie un SMS sur son portable
- *Pré Condition* : aucune
- *Post Condition* : le SMS est enregistré en base de donnée centrale, tous les adhérents sont notifiés
- Scénario nominal :

  1. l'adhérent saisie un SMS sur son portable
  2. l'adhérent envoie le SMS au numéro de l'association
  3. la base centrale enregistre le SMS : → cas d'utilisation « Sauvegarder »

4. le service de notification reçoit un message de la base de données : → cas d'utilisation «Notifier Adhérents»

- *Scénarii alternatifs* :

  3a. problème de sauvegarde : pas de notification

3.15.3 Envoyer Mail

- *Événement déclencheur* : l'adhérent saisie un mail sur son navigateur
- *Pré Condition* : aucune
- *Post Condition* : le mail est enregistré en base de donnée centrale, tous les adhérents sont notifiés
- Scénario nominal :

  1. l'adhérent saisie un mail sur son navigateur

  2. l'adhérent envoie le mail à l'adresse de l'association

  3. la base de données sauvegarde le mail: → cas d'utilisation « Sauvegarder »

  4. le service de notification reçoit un message de la base de données : → cas d'utilisation «Notifier Adhérents»

- *Scénarii alternatifs* :

  3a. problème de sauvegarde : pas de notification

3.15.4 Recevoir SMS

- *Événement déclencheur* : le service de notification envoi une notification
- *Pré Condition* : aucune
- *Post Condition* : le SMS est reçu par les adhérents concernés
- Scénario nominal :

  1. le service de notification reçoit un nouveau message

  2. le service de notification récupère le numéros des adhérents concernés par le message : → cas d'utilisation « Notifier Adhérent »

  3. le service de notification informe les adhérents concernés en leur envoyant un SMS

4. les adhérents concernés reçoivent le SMS

- *Scénarii alternatifs* :

  4a. les adhérents ne reçoivent pas le SMS

3.15.5 Recevoir Mail

- *Événement déclencheur* : le service de notification envoi une notification
- *Pré Condition* : aucune
- *Post Condition* : le mail est reçu par les adhérents concernés
- Scénario nominal :

  1. le service de notification reçoit un nouveau message

  2. le service de notification récupère les adresses mails des adhérents concernés par le message : → cas d'utilisation « Notifier Adhérent »

  3. le service de notification informe les adhérents concernés en leur envoyant un mail

  4. les adhérents concernés reçoivent le mail

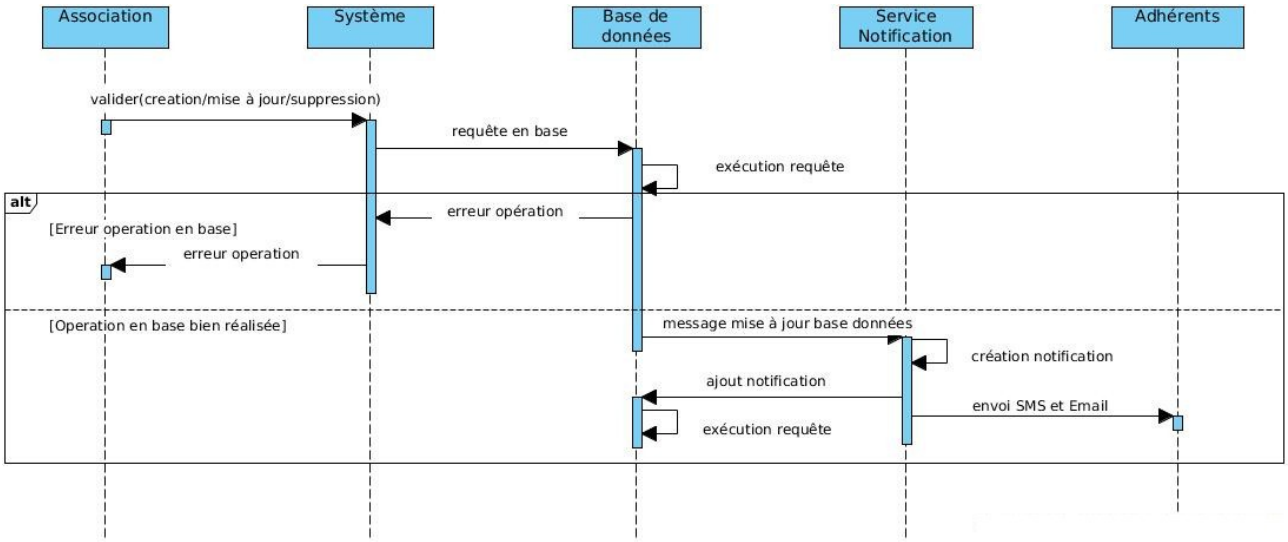
- *Scénarii alternatifs* :

  4a. les adhérents ne reçoivent pas le mail

4 Diagrammes de Séquences Système

Les diagrammes de séquences ci après sont des diagrammes simplifiés ayant pour but de montrer le principe de comportement des différents services aux actions de l'utilisateur

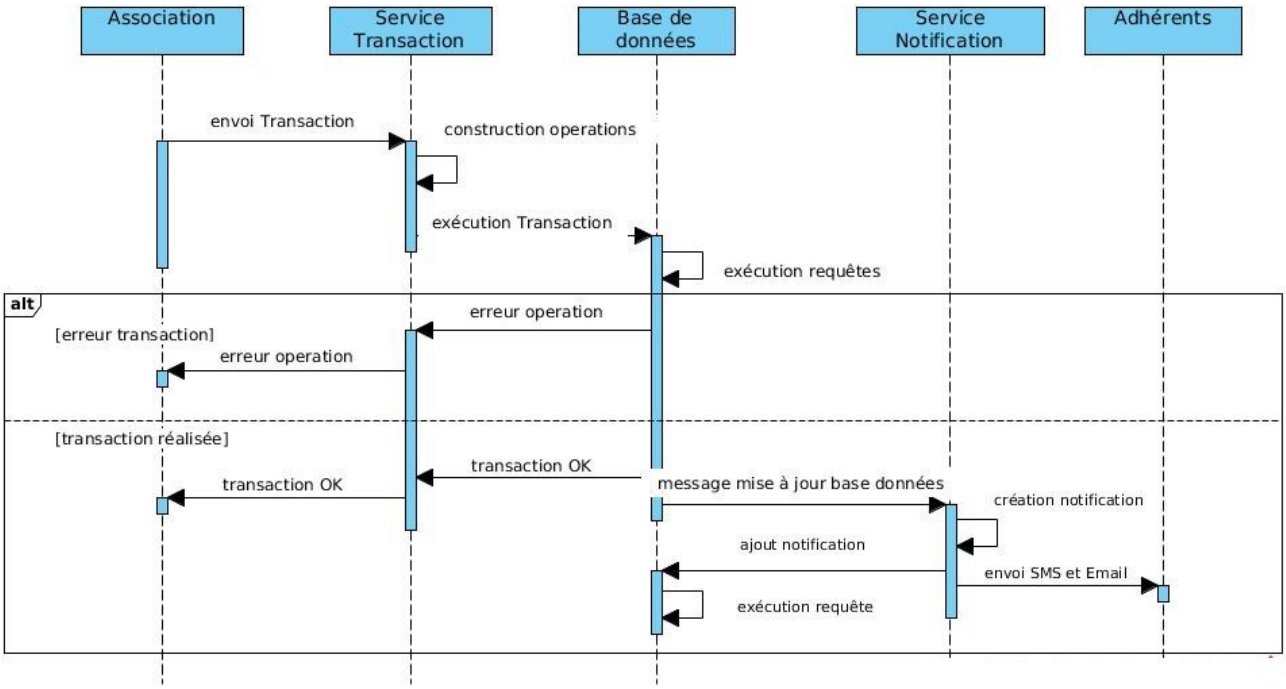
4.1 Service Notification – Application Bureau



L'association valide une opération qui doit faire l'objet d'une notification. Cette opération fait d'abord l'objet d'un enregistrement en base de données. Cette dernière informe le service de notification qui crée une notification contenant le message et les adhérents qui devront la recevoir. Le service va ensuite enregistrer cette notification en base et envoyer un SMS et Email aux adhérents concernés.

Les applications mobile iront ensuite interroger la base pour savoir s'il n'y a pas de nouvelles notifications.

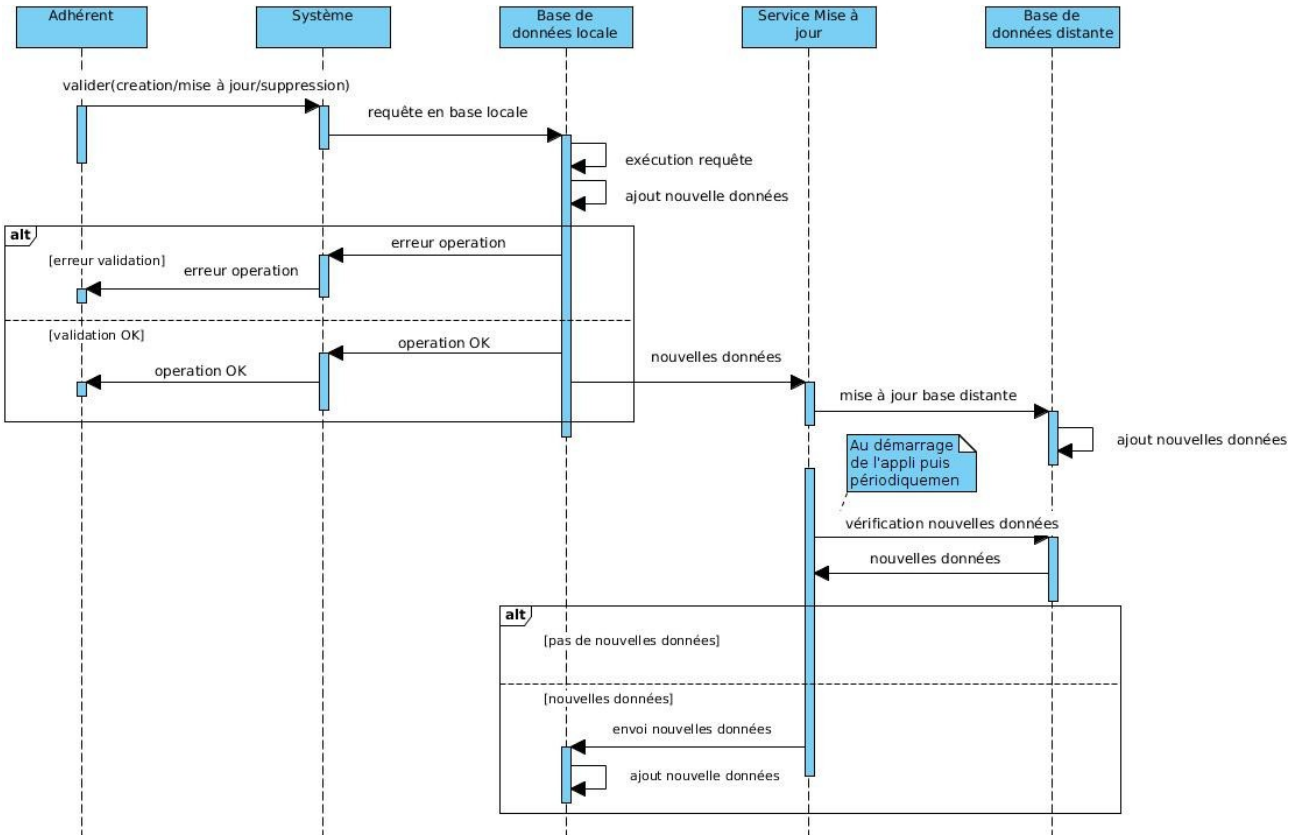
4.2 Service Transaction – Application Bureau



L'association effectue une transaction entre 2 adhérents. Le service transaction construit l'opération et la base de données enregistre les opérations.

Le service notification va ensuite créer une notification qui va concerner les 2 adhérents (voir ci avant)

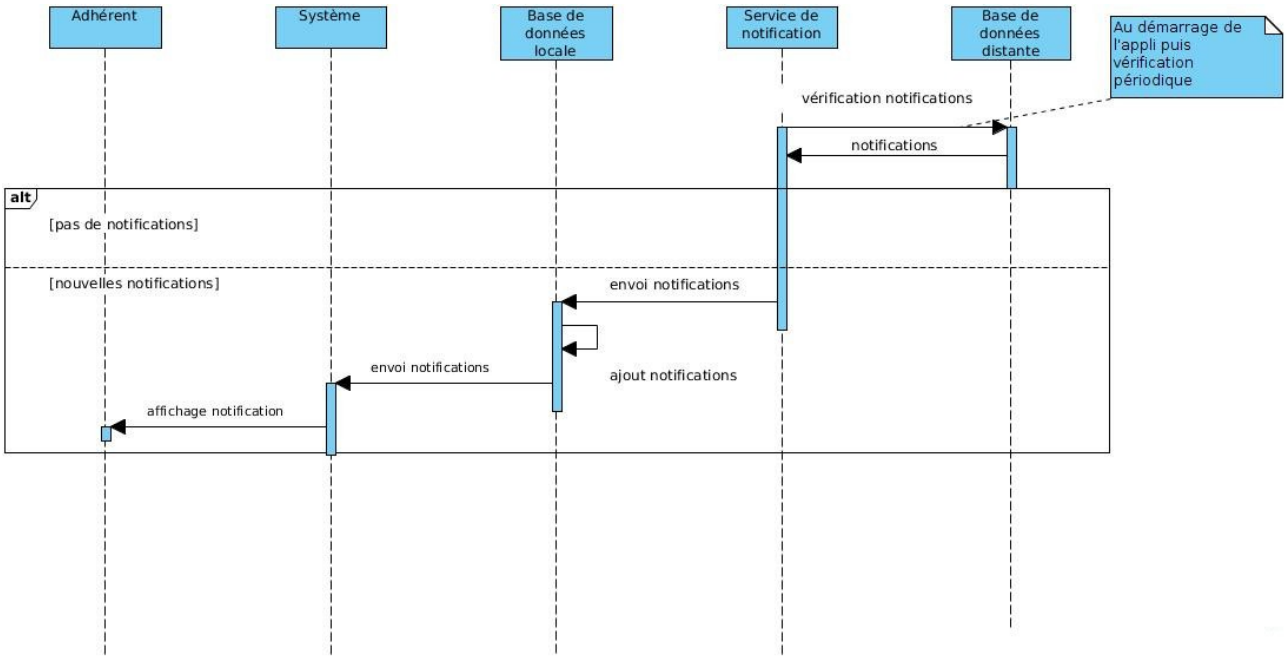
4.3 Service Mise à jour – Application Mobile



La mise à jour de la base centrale se fera quand une opération de modification de la base locale d'un adhérent sera effectuée. Lorsque le mobile aura accès à internet, le service de mise à jour postera les nouvelles données à la base centrale.

La mise à jour de la base locale se fera quand une opération de modification de la base centrale sera effectuée. Lorsque le mobile aura accès à internet, le service de mise à jour vérifiera si des nouvelles données sont présentes dans la base centrale. Si tel est le cas, il les récupère pour les transmettre à la base locale.

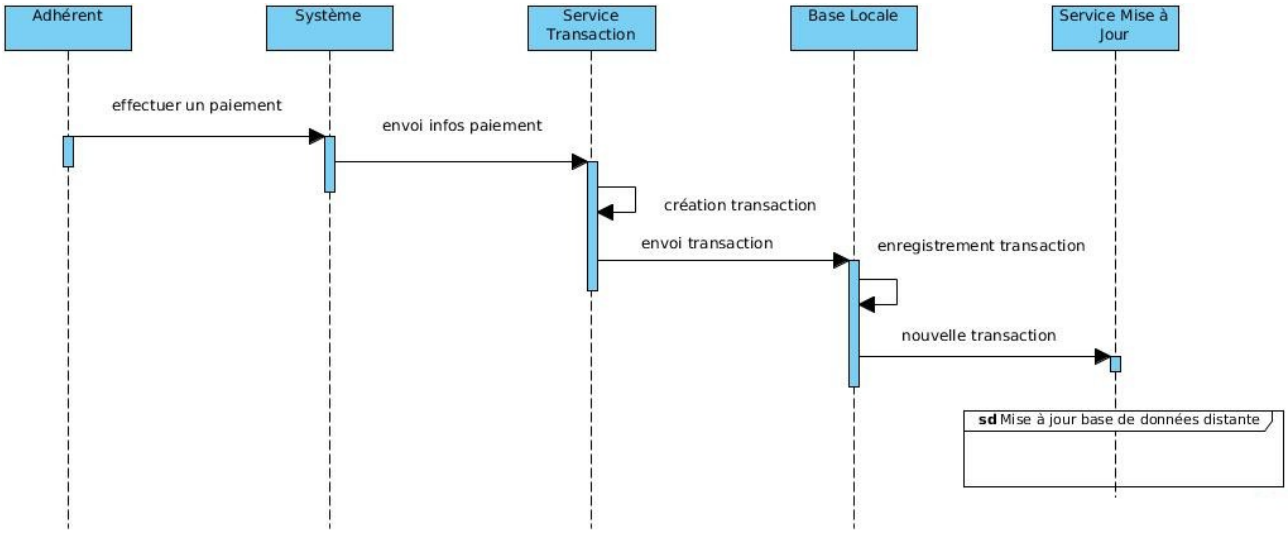
4.4 Service Notification – Application Mobile



Le service de notification de chaque application mobile, va vérifier périodiquement si de nouvelles notifications sont présentes en bases de données centrale.

Si tel est le cas, il va les récupérer pour les transmettre à la base de donnée locale. Puis elles seront affichées sur le mobile de l'adhérent.

4.5 Service transaction – application mobile



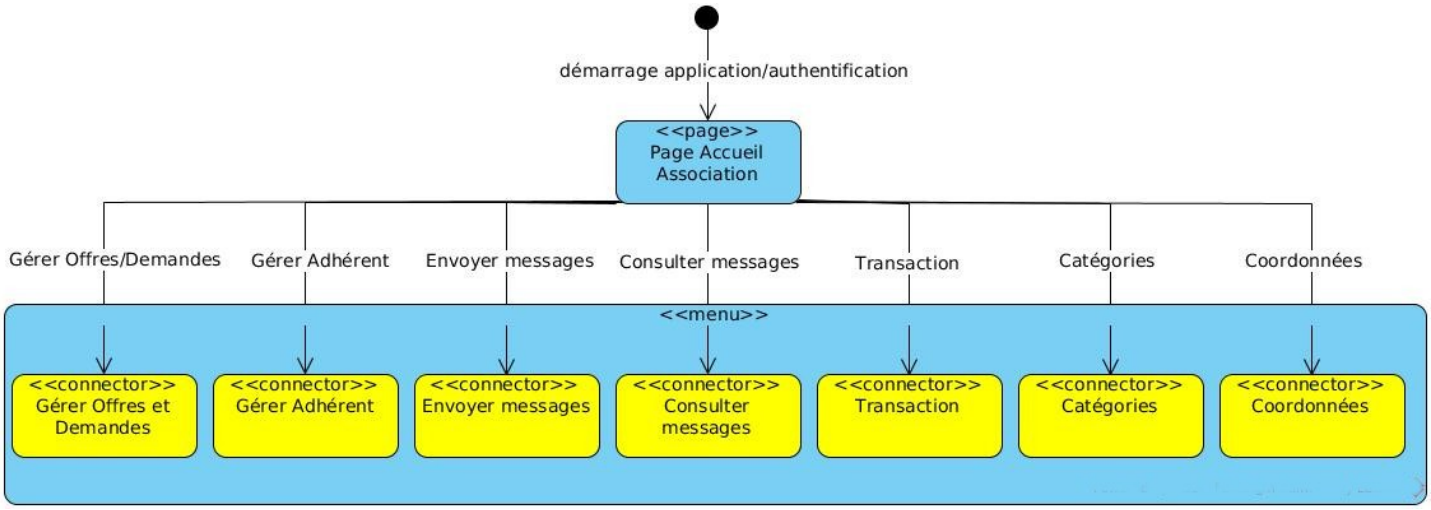
Lorsqu'un adhérent effectue un paiement, le service de transaction va construire une transaction et l'enregistrer en base locale. Le service de mise à jour se chargera d'en informer la base centrale distante.

Si l'adhérent utilise le paiement via le ***NFC***, c'est le service de transaction de celui qui a reçu le paiement qui construira la transaction.

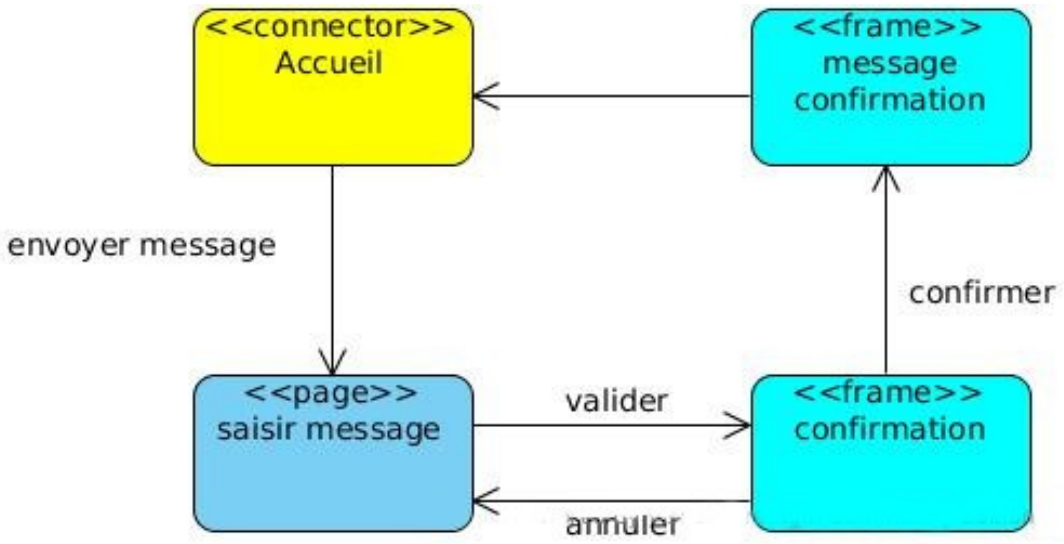
5 Diagramme de Navigation

5.1 Application Bureau

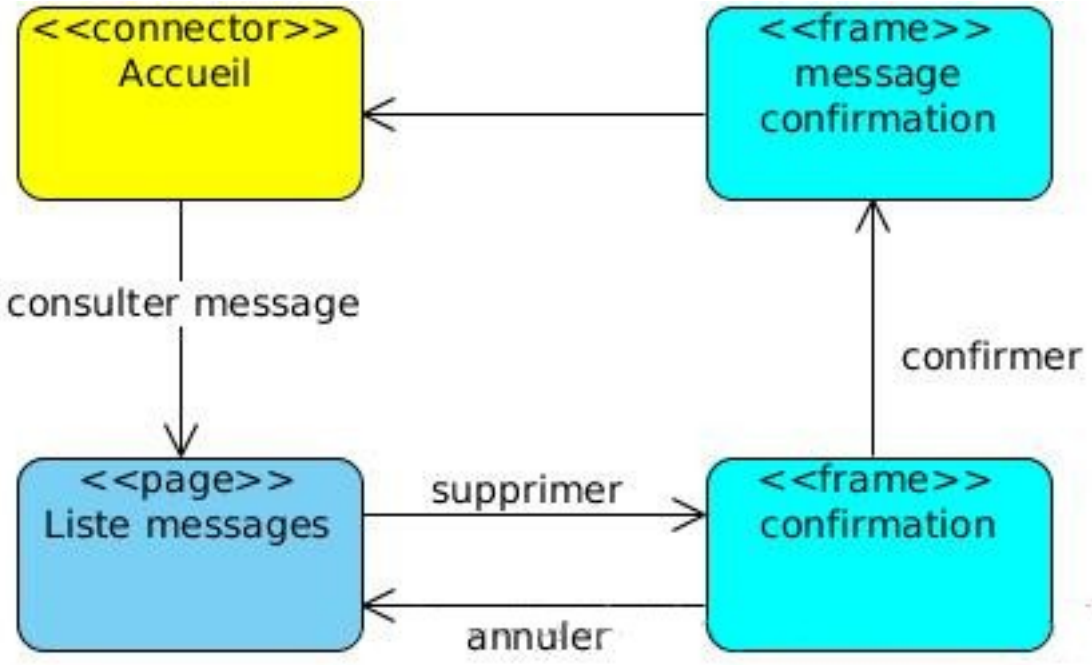
5.1.1 Diagramme navigation Accueil Association



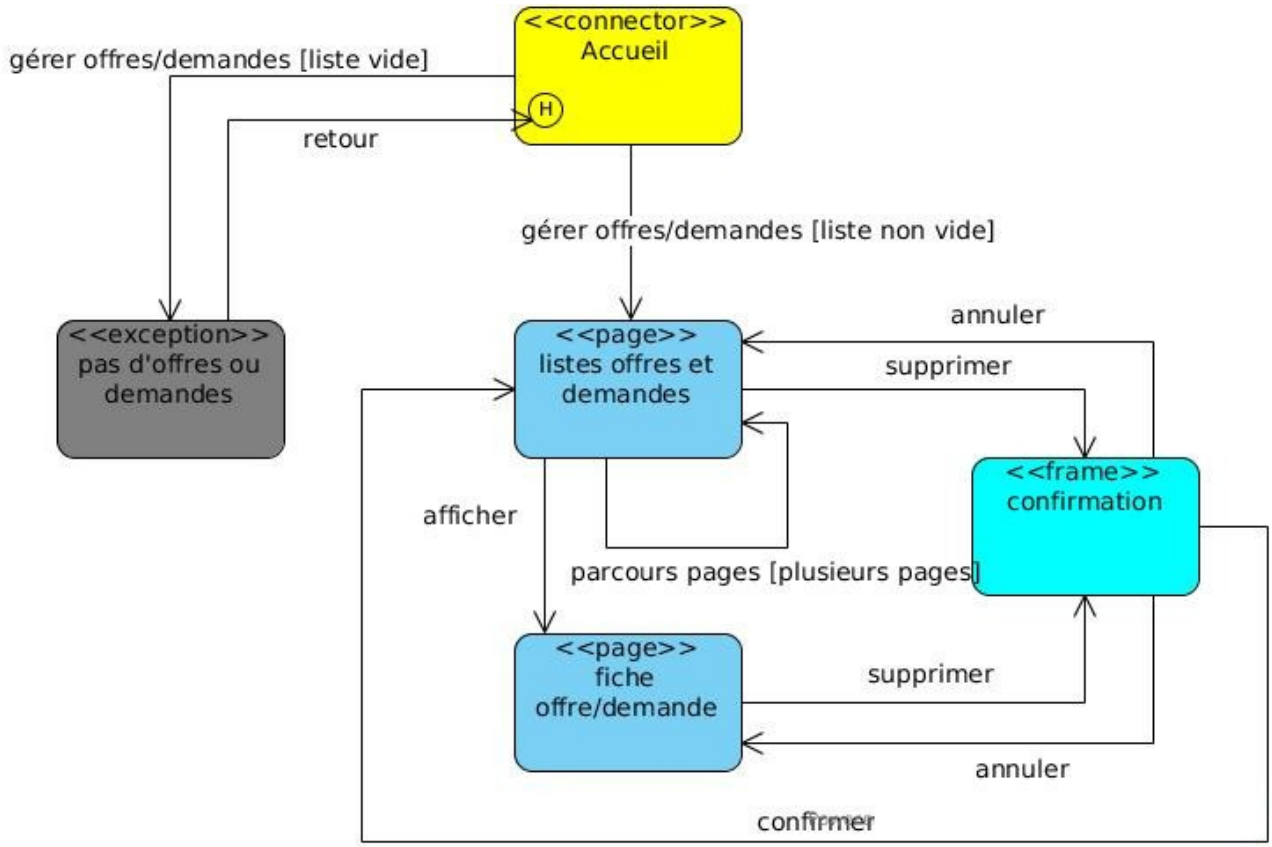
5.1.2 Diagramme navigation Envoyer messages



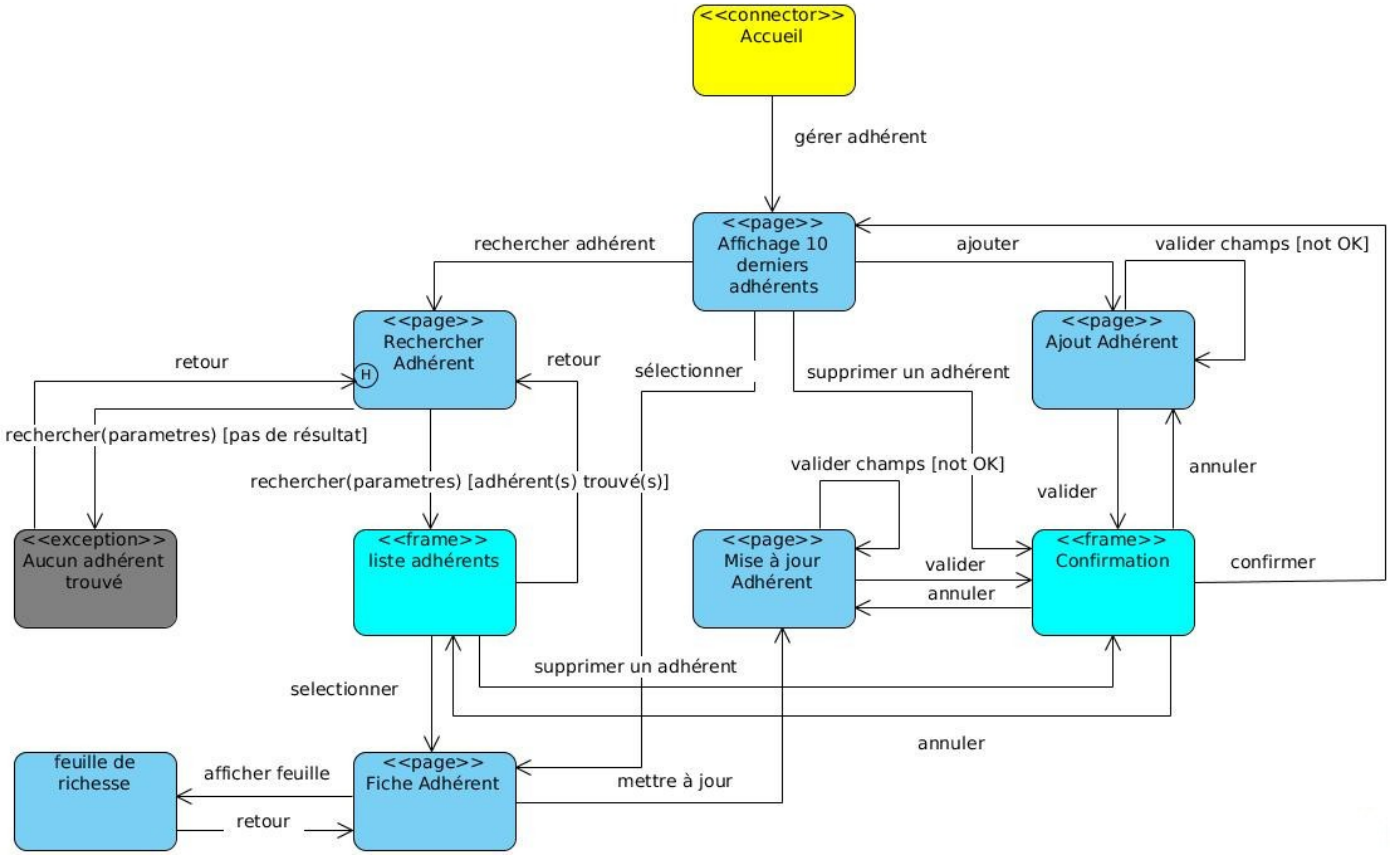
5.1.3 Diagramme navigation Consulter Messages



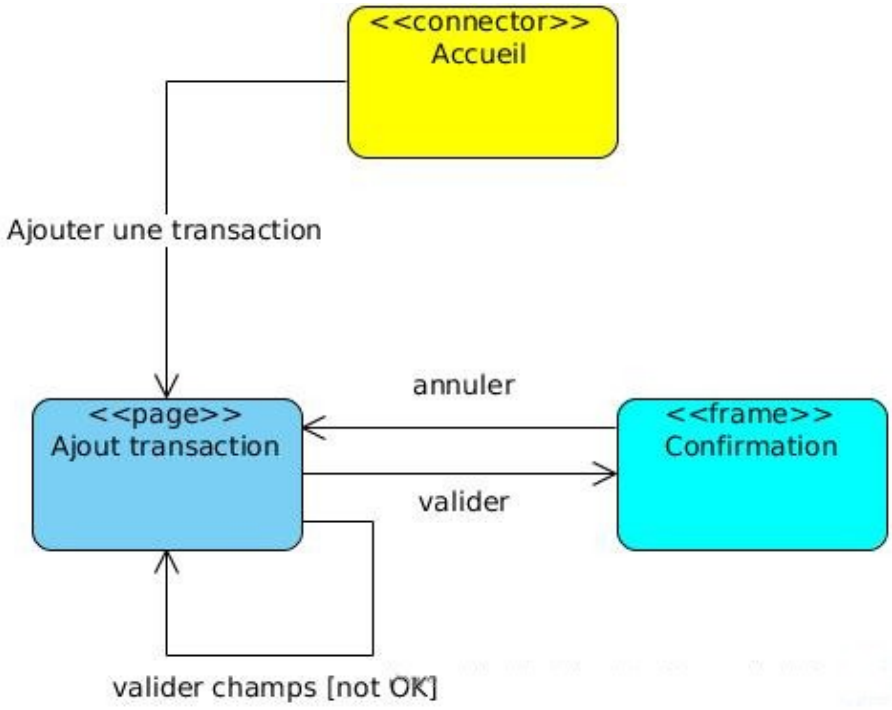
5.1.4 Diagramme navigation Gérer Offres et Demandes



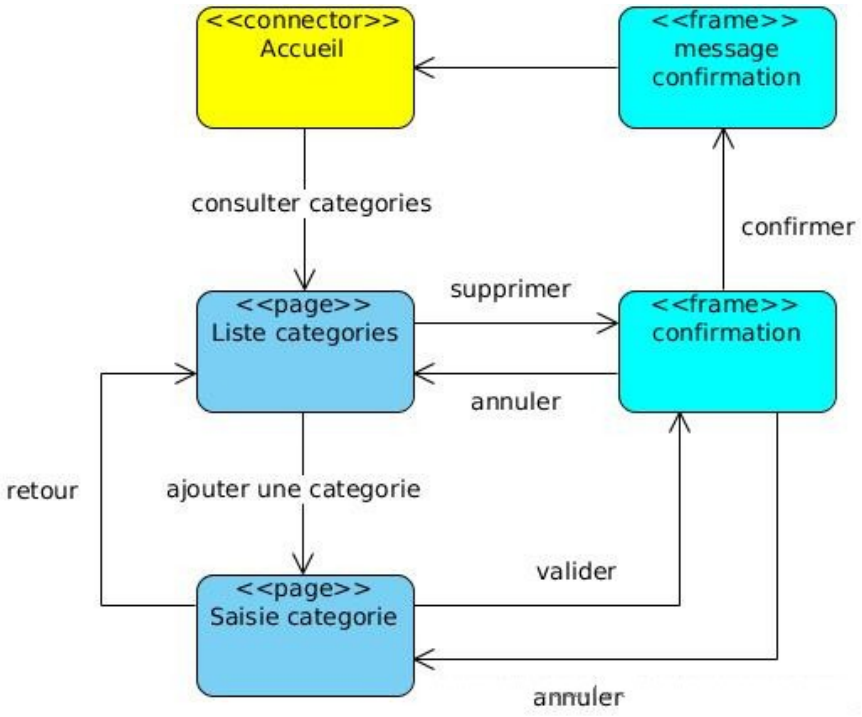
5.1.5 Diagramme navigation Gérer Adhérents



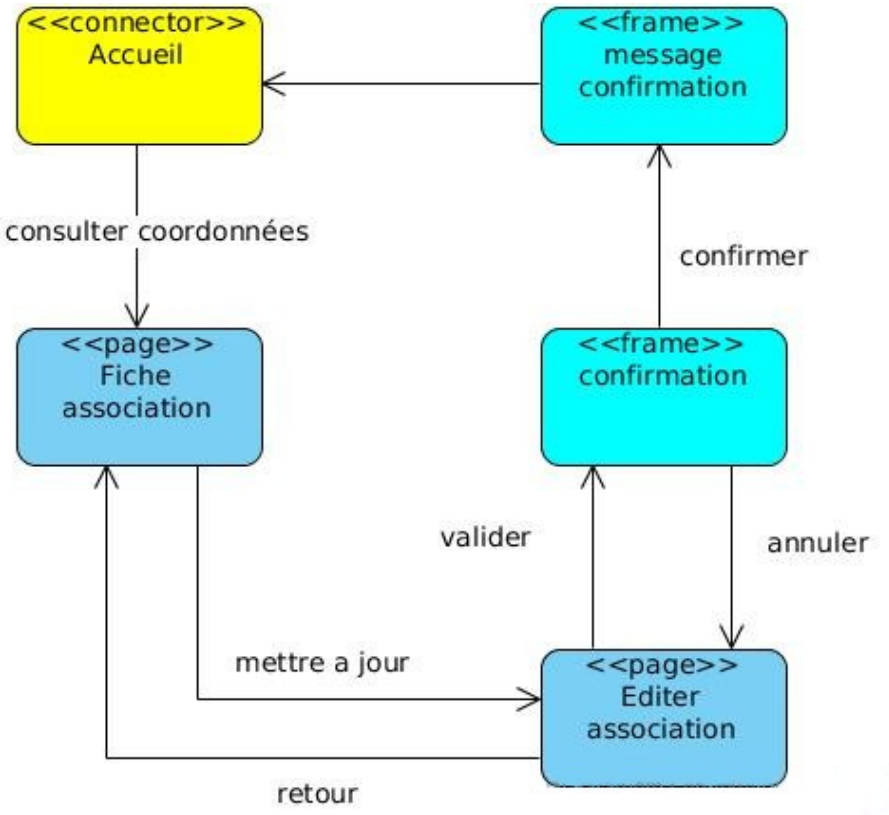
5.1.6 Diagramme navigation Transaction



5.1.7 Diagramme navigation catégories

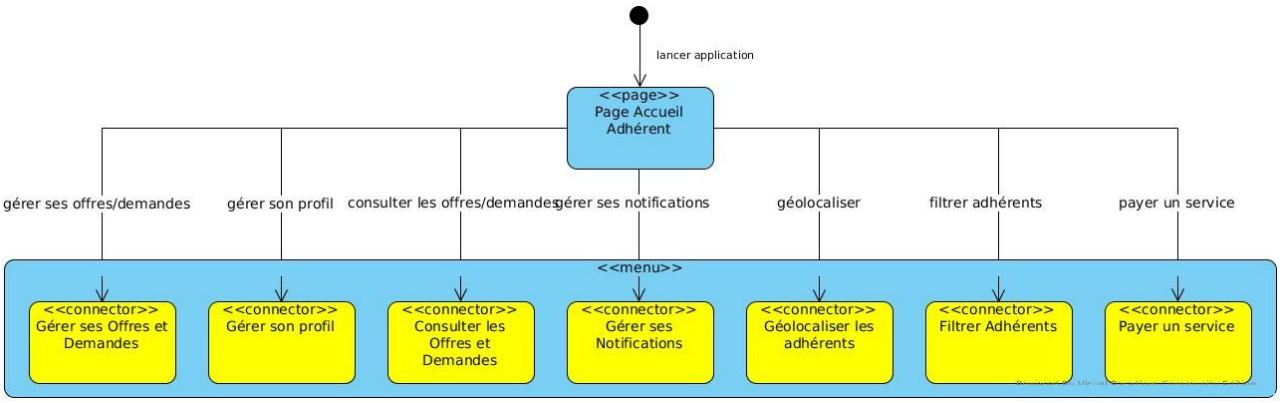


5.1.8 Diagramme navigation Consulter coordonnées

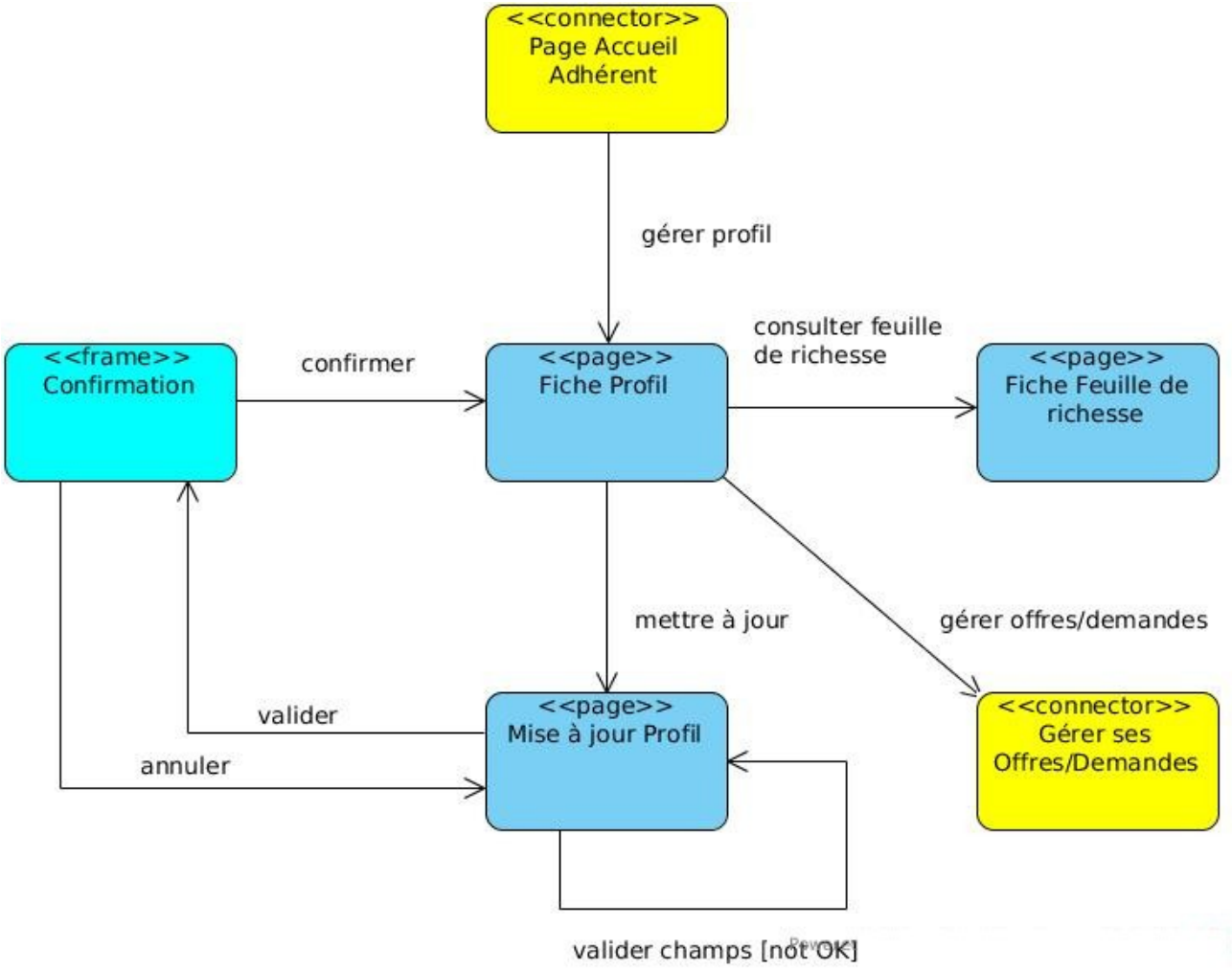


5.2 Application Mobile

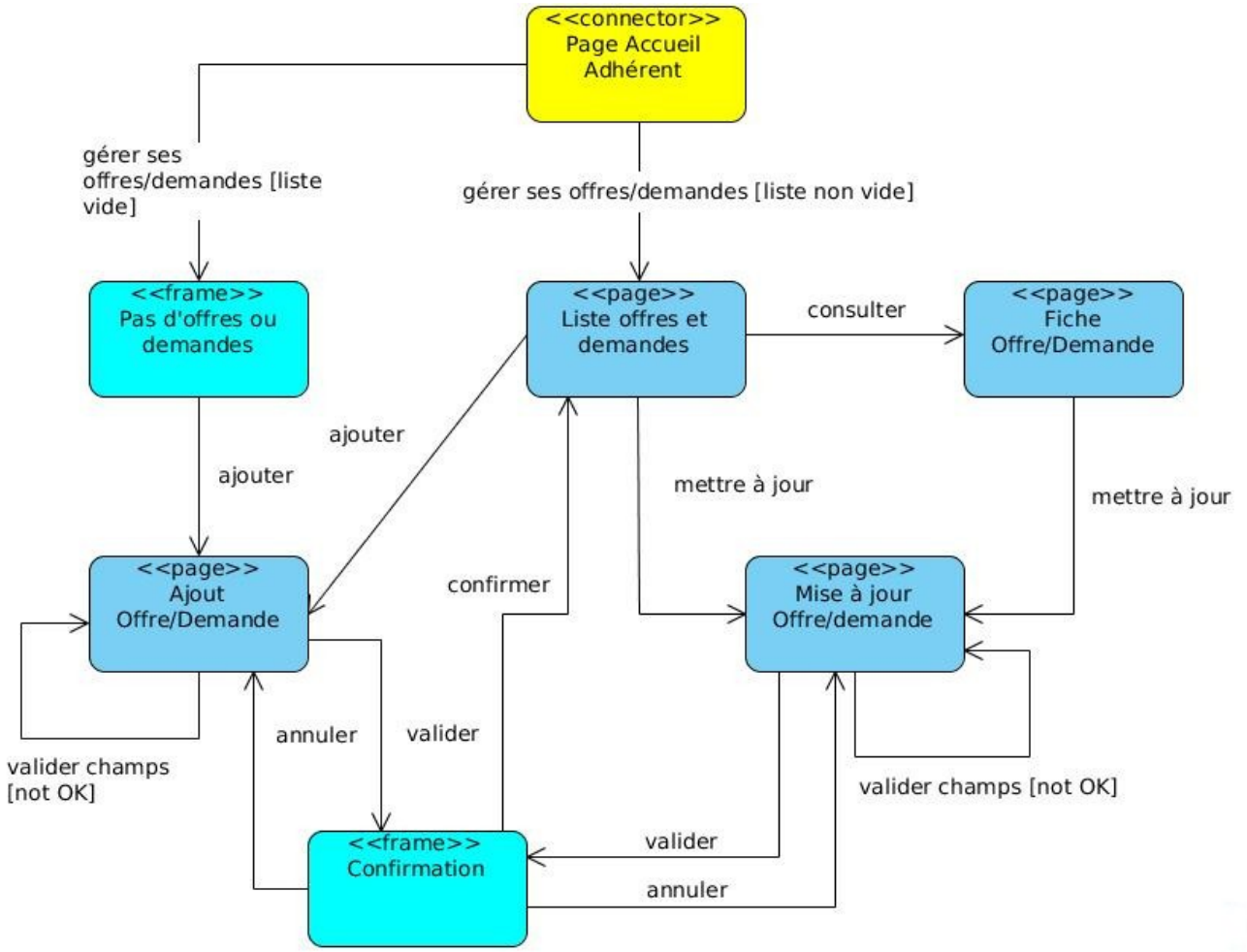
5.2.1 Diagramme navigation Accueil Adhérent



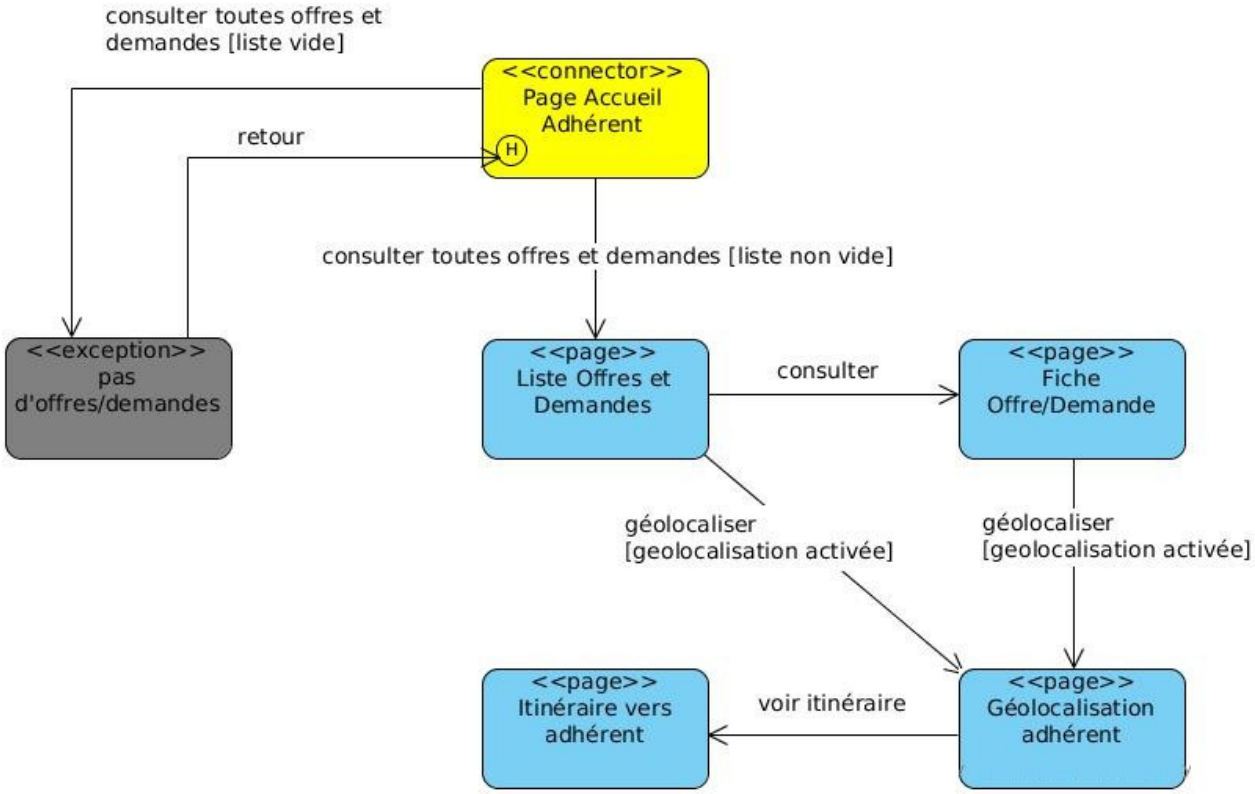
5.2.2 Diagramme navigation gérer son profil



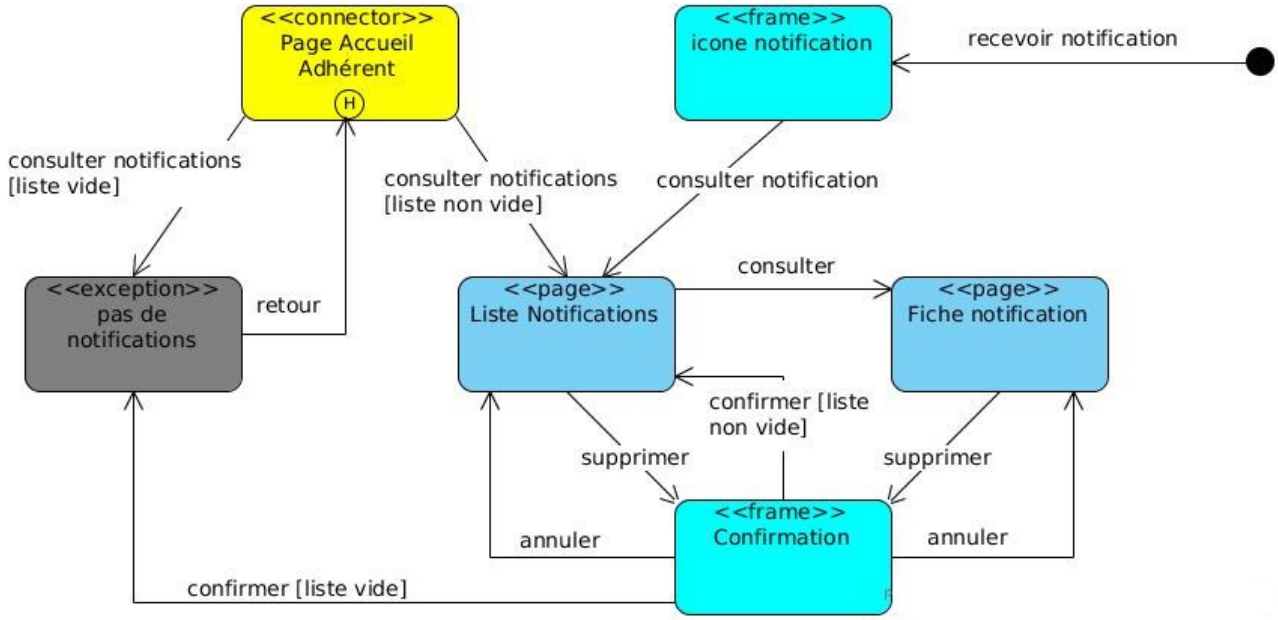
5.2.3 Diagramme navigation gérer ses offres et demandes



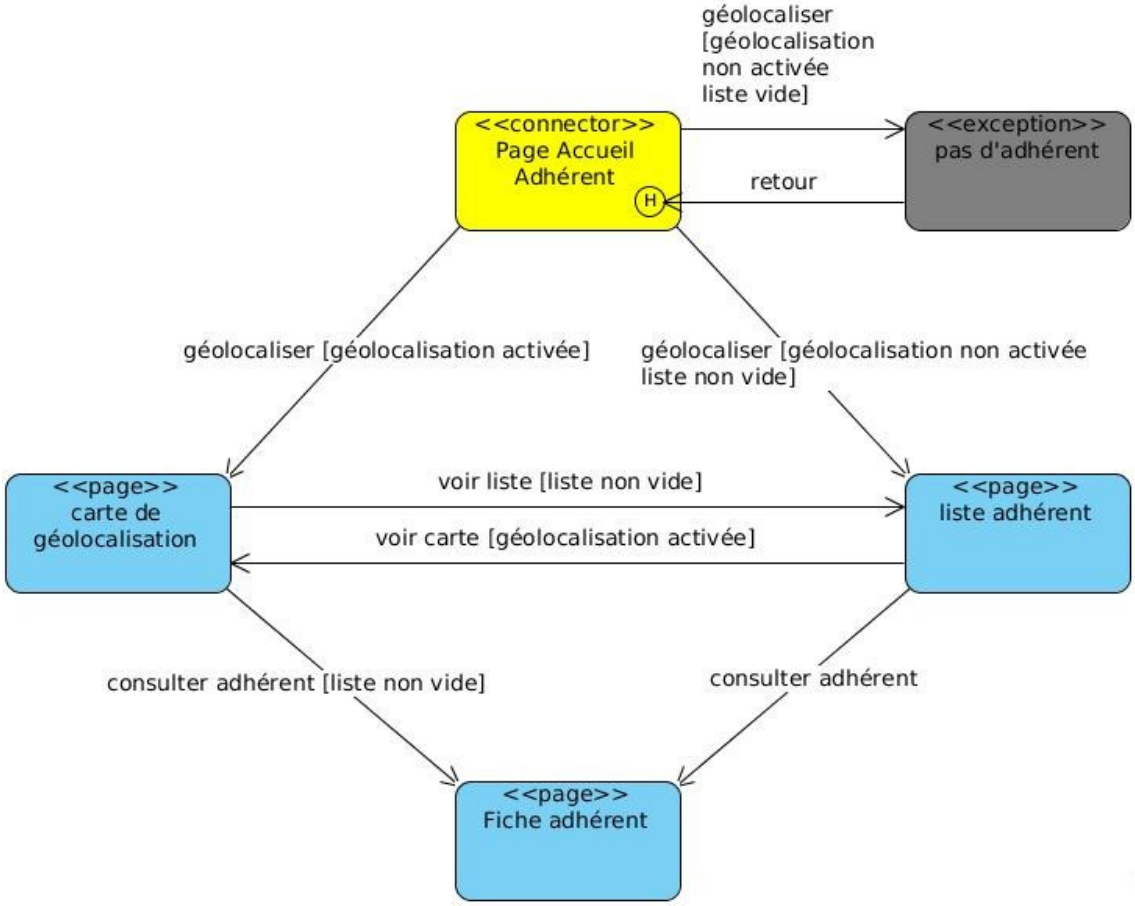
5.2.4 Diagramme navigation consulter toutes les offres et demandes



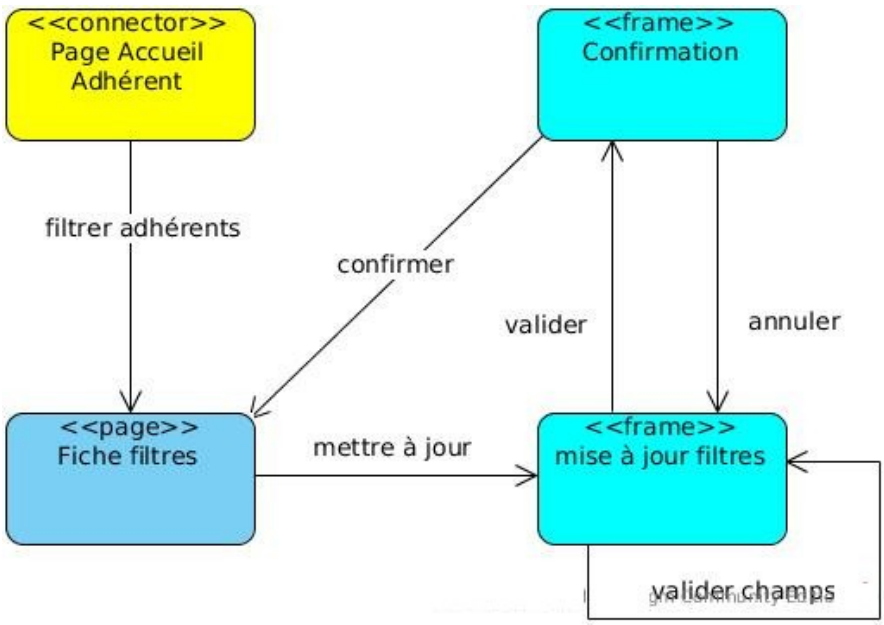
5.2.5 Diagramme navigation gérer ses notifications



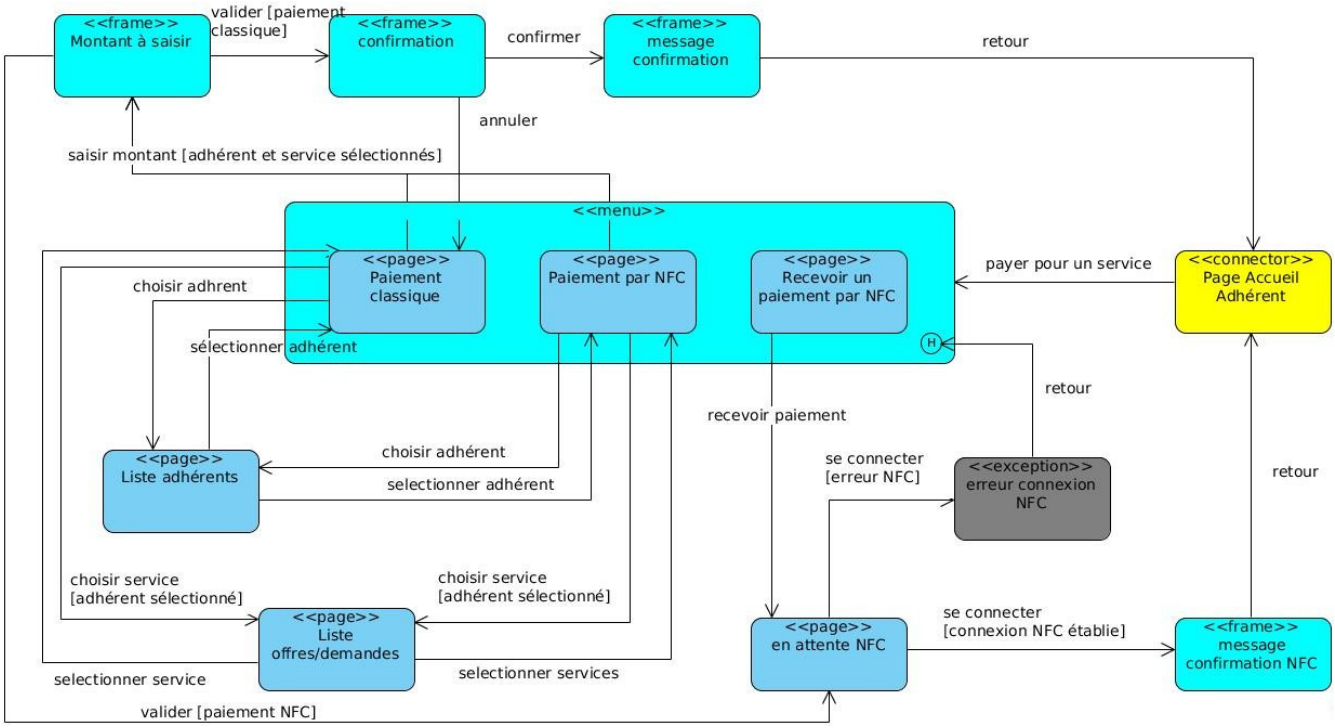
5.2.6 Diagramme navigation géolocaliser



5.2.7 Diagramme navigation filtrer adhérents



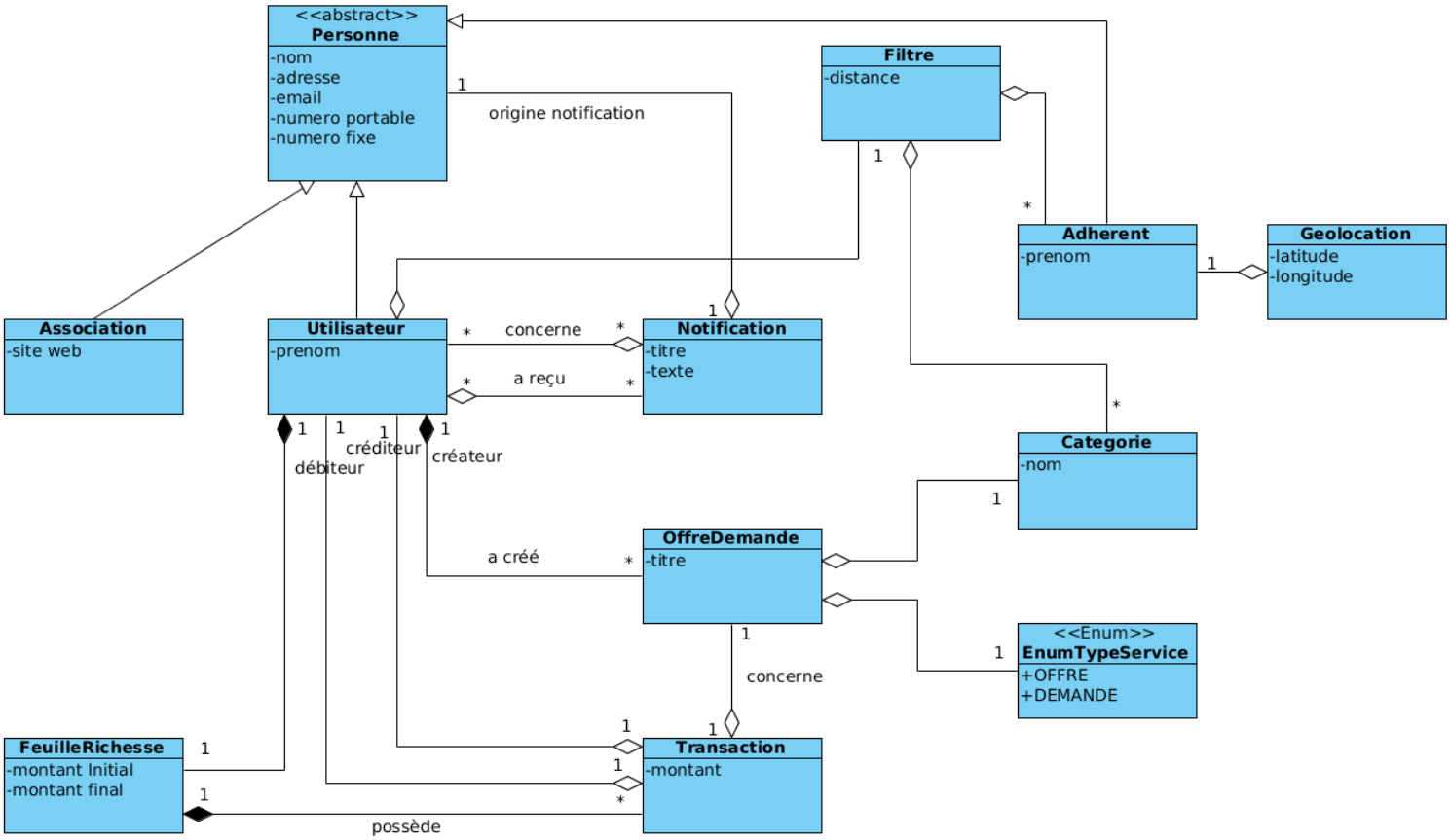
5.2.8 Diagramme navigation payer un service



6 Diagramme de Classes

Le diagramme de classe ci après représente les principaux modèles de données qui seront sauvegardés en base de données.

6.1 Diagramme



6.2 Détail des classes

Personne : classe abstraite représentant les utilisateurs du système. Elle possède un *nom* une *adresse* postale, une adresse *mail*, un numéro de *portable* et un numéro de téléphone *fixe*.

Association : réalisation de **Personne**, représente les utilisateurs de l'application Bureau. Elle possède un attribut supplémentaire à savoir un *lien* vers son site web.

Utilisateur: réalisation de **Personne**, représente les utilisateurs de l'application Mobile. Elle possède en plus un *prénom*. Elle possède également une liste d'**OffreDemande** qu'elle a créé, une liste de **Notification** qu'elle a reçu, une **FeuilleRichesse** représentant son compte et un **filtre** pour les notifications et les autres **adhérents**.

Adhérent : réalisation de **Personne**, représente les autres membres de l'association. Elle possède en plus un *prénom.*

Notification : représente les notifications a envoyé aux adhérents (possède attribut liste d'**Adherent**). Elle est caractérisée également par un *titre* et un *texte* de notification.

OffreDemande : représente un service que peuvent s'échanger les adhérents. Les services peuvent être de type *Offre* ou *Demande*. Ils sont caractérisé par un *titre* et une *catégorie*. Ils sont forcément créés par un **Adherent**.

FeuilleRichesse : représente le compte en monnaie virtuelle. Elle possède un *solde initial* et *final*. Elle possède également une liste de **Transaction**. Un **Adherent** possède forcément une et une seule **FeuilleRichesse**.

Transaction : représente l'opération de paiement d'un adhérent vers un autre adhérent pour un service rendu. C'est pourquoi elle possède un *montant*, un **Adherent** *créditeur*, un **Adherent** *débiteur* et une **OffreDemande**.

Categorie : représente les domaines selon lesquels les offres et demandes peuvent être regroupées

Filtre : représente les critères pour filtrer les notifications à recevoir et les adhérents à afficher. Ces critères peuvent être la distance maximale entre le domicile de l'utilisateur et les autres adhérents, les catégories des offres et demandes et une liste d'adhérent.

Geolocation : représente la géolocalisation d'un adhérent suivant la latitude et longitude de son domicile.

7 Interfaces REST

L'accès au serveur par les applications mobiles se fera par des services web de type ***REST***. L'échange se fera par le protocole ***HTTPS*** et l'information sera sérialisée au format ***JSON***.

L'accès au serveur d'application par l'application bureau se fera également par des services web de type ***REST***. L'échange se fera par le protocole ***HTTPS*** et l'information sera sérialisée au format ***JSON***.

7.1 Authentification

Méthode	URI	Description
GET	/member/get/{mobileId}/ {cellNumber}	Récupération de l'id de l'utilisateur mobile lors de la première connexion
GET	/association/authentication/{login} {password}/{number}	Authentification pour accéder à l'application bureau. *Number* correspond au nombre d'adhérent à afficher à l'accueil

7.2 Mise à jour

Méthode	URI	Description
GET	/member/get/{id}/update	Récupération de la liste des tables à mettre à jour
PUT	/member/put/{id}/dateLastUpdate	Modification de la date de dernière mise à jour de l'utilisateur

7.3 Association

Méthode	URI	Description
GET	/association	Récupération des informations relatives à l'association dans l'application mobile
PUT	/association/put	Mise à jour des coordonnées de l'association dans l'application bureau

7.4 Offres et Demandes

Méthode	URI	Description
GET	/supplyDemand/get/all	Récupération de toutes les offres et demandes des adhérents
GET	/supplyDemand/get/type/{type}	Récupération de toutes les offres ou de toutes les demandes des adhérents (dépend du *type* précise)
GET	/supplyDemand/get/{id}	Récupération de l'offre/demande avec *id* comme identifiant
POST	/supplyDemand/post	Création d'une offre/demande
PUT	/supplyDemand/put	Modification d'offre/demande
DELETE	/supplyDemand/delete/{id}	Suppression de l'offre/demande avec *id* comme identifiant
DELETE	/supplyDemand/delete/get/{id}	Suppression de l'offre/demande avec *id* comme identifiant puis récupération de la liste des offres/demandes résultantes

7.5 Utilisateur – profil

Méthode	URI	Description
GET	/member/get/{id}/member	Récupération du profil utilisateur
PUT	/member/put	Modification du profil utilisateur

7.6 Utilisateur – notifications

Méthode	URI	Description
GET	/member/get/{id}/notification	Récupération des notifications de l'utilisateur
DELETE	/member/delete/{id}/notification	Suppression des notifications de l'utilisateur

7.7 Utilisateur – feuille de richesse

Méthode	URI	Description
GET	/member/get/{id}/wealthSheet	Récupération de la feuille de richesse de l'utilisateur

7.8 Adhérents

Méthode	URI	Description
GET	/member/get/all	Récupération de tous les adhérents
DELETE	/member/delete/{id}	Suppression de l'adhérent dont l'identifiant vaut *id* en base de donnée
GET	/member/get/last/{number}	Récupération de la liste des derniers adhérents consultés ou modifiés (nombre d'adhérents affichés = *number*
GET	/member/get/attributes	Récupération de la liste des adhérents correspondant aux attributs
GET	/member/get/{id}	Récupération de l'adhérent dont l'identifiant vaut *id* en base de donnée
POST	/member/post	Ajout d'un nouvel adhérent
PUT	/member/put/member	Mise à jour d'un adhérent

7.9 Transactions

Méthode	URI	Description
POST	/transaction	Création d'une transaction
POST	/transaction/post/get	Création et récupération d'une transaction dans l'application bureau
GET	/member/get/{id}/transaction	Récupération des transactions de l'utilisateur d'identifiant *id*

7.10 Catégories

Méthode	URI	Description
GET	/category	Récupération de toutes les catégories d'offres et demandes proposées par l'association
POST	/category/post	Ajout d'une catégorie au niveau de l'application bureau
DELETE	/category/delete/ {id}	Suppression de la catégorie dont l'identifiant vaut *id* en base de donnée

7.11 Messages

Méthode	URI	Description
GET	/message/get/all	Récupération de toutes les messages de l'association
POST	/message	Ajout d'un message au niveau de l'application bureau
DELETE	/message/delete/ {id}	Suppression d'un message dont l'identifiant vaut *id* en base de donnée

8 Design Pattern

8.1 Application Bureau

- **Model View Controller (MVC)** : Ce patron est utilisé pour gérer plus efficacement l'interaction homme-machine.

  - Le modèle contiendra les données à afficher et représentera les objets métiers.

  - La vue représentera un écran à afficher.

  - Le contrôleur gérera les actions utilisateurs.

- **Composite** : Ce patron sera utilisé pour pouvoir créer des widgets complexes à partir de widget de base. Ce patron est facilement utilisable avec ***GWT***, où nos classes widgetComposite devront étendre la classe « Composite » . On peut ensuite rajouter tous les composants héritant de « Widget » dans le constructeur de notre classe.
- **Singleton** : Ce patron sera utilisé pour avoir une seule instance des différentes pages et leurs modèles.
- **Strategy** : Ce patron sera utilisé pour pouvoir agencer de plusieurs manières une page composée de plusieurs modules.
- **Validator** : Ce patron sera utilisé pour faciliter la gestion des erreurs de saisie dans les formulaires.
- **Proxy** et **Callback** : Avec ***GWT***, la communication avec le serveur s'opère via un protocole propriétaire de type ***RPC***. Les appels sont asynchrones. Le développeur doit définir une interface de service qui représente le proxy. L'implémentation de cette interface se fait dans le package *server* (voir chapitre 2.2.1). Il doit également implémenter l'interface « CallbackAsync » pour traiter le résultat de l'appel asynchrone.

8.2 Application Mobile

8.2.1 Model View Controller (MVC)

L'implémentation JAVA du patron de conception ***Observateur/Observé*** dans la package ***java.util*** est utilisé :

- Des classes représentant le *modèle* des activités seront implémentées. Elles seront *observables* en étendant la classe JAVA '*Observable*' . Dans chacune de leur méthode qui modifient la valeur des données, elles font appel aux méthodes de la superclasse *setChanged() et notifyObservers()* pour indiquer aux observateurs que les données ont été modifiées.
- Les classes représentant les activités implémentent l'interface JAVA '*Observer*'. Elles implémentent donc la méthode *update(Observable observable, Object object)* qui sera appelée à chaque modification des données de l'observable et qui affichera ces nouvelles valeurs. Elles possèdent un modèle comme attribut, ce modèle ajoute l'activité dans sa liste d'observateur.
- Les activités implémentent également l'interface '*View.OnClickListener*' du package ***android.view*** pour réagir aux clics de l'utilisateur. A la mise au 1° plan de l'activité et/ou à un clic de l'utilisateur, la méthode correspondante du modèle de l'activité est appelée, cette dernière change des données qui seront ensuite mise à jour dans l'affichage.

8.2.2 Data Access Object (DAO)

Une classe '*DaoFactory*' permettra l'accès à la base de donnée locale ***SQLITE*** en étendant la classe '*SQLiteOpenHelper*' du package ***android.database.sqlite***.

Elle permettra également de charger les fichiers de type *SharedPreferences* utilisés par l'application.

Et pour finir elle fournit les différents ***Dao*** qui possèdent les différentes méthodes de gestion de la base de donnée.

Les ***Dao*** seront représentés par des interfaces.

On aura un ***Dao*** pour les opérations relatives aux données de l'association, des catégories, des messages d'erreur, des adhérents, du profil utilisateur, des offres et demandes, des transactions.

8.2.3 Divers

- **Adapter** : Ce patron est utilisé par ***Android*** pour afficher des composants graphiques, des *ListView* dans notre cas.
- **Validator** : Il sera utilisé pour la validation de toutes saisies effectuées par l'utilisateur. Il combinera les patrons de conception *Strategy et Composite*. Pour chaque champ à vérifier on applique une stratégie de vérification. Cette stratégie peut être réalisée par la vérification d'une seule contrainte (feuille) ou par la vérification de plusieurs (composite)

8.3 Serveur d'Application

- **Layers**: Le but de ce patron est de décomposer l'architecture en plusieurs couches afin d'avoir un couplage faible entre les fonctionnalités de l'application. Ainsi le changement d'implémentation dans une couche n'entraînera pas de modification dans une autre couche.

- **Dependency Injection** : Ce patron sera réalisé avec les conteneurs ***EJB*** . Après l'instanciation du conteneur, il permet d'injecter des services pour obtenir une architecture souple et modulaire.

- **Singleton** : Ce patron sera réalisé avec les conteneurs ***EJB*** . Il permet de s'assurer qu'une seule instance d'une classe ne soit créée. Il sera utilisé pour l'accès à la base de donnée, pour qu'un seul objet crée la connexion.

- **Interceptor** : Ce patron sera réalisé avec les conteneurs ***EJB***. Il permet de découpler les opérations à réaliser lors d'un événement. Il sera utilisé pour lancer les opérations de création de notification, lorsqu'une opération de modification de la base de donnée sera réalisée.

- **Factory et Strategy :** Le patron factory permet de fabriquer dynamiquement des objets. Le patron strategy permet de permuter dynamiquement des algorithmes utilisés dans une application. Ces 2 patrons seront utilisés conjointement pour fabriquer des notifications en fonction d'un contexte qui sera déterminé par l'objet de la modification en base de donnée.

- **Publish Suscribe** : Ce patron permet de publier un message d'un éditeur à un médiateur. En fonction du type du message, le médiateur le transmettra à ceux qui ont souscrit à ce type. Dans notre cas, il sera utilisé pour déterminer les adhérents qui devront recevoir les notifications créées par le mécanisme décrit ci avant.

- **Chain of Responsability** : Ce patron permet à un certain nombre de classe de répondre à une requête sans savoir si les autres classes peuvent y répondre, ce qui permet un couplage faible entre les classes. Dans notre cas, il sera utilisé pour transmettre la notification aux adhérents sélectionnés par le mécanisme ci avant, de différentes façon : par SMS, par mail ou par l'application mobile. Il permettra de rajouter de nouveaux moyens de transmission en ne modifiant pas ou très peu le code existant.

  Nota : les 4 derniers patrons seront utilisés par l'acteur *Service de Notifications* défini dans les use cases.