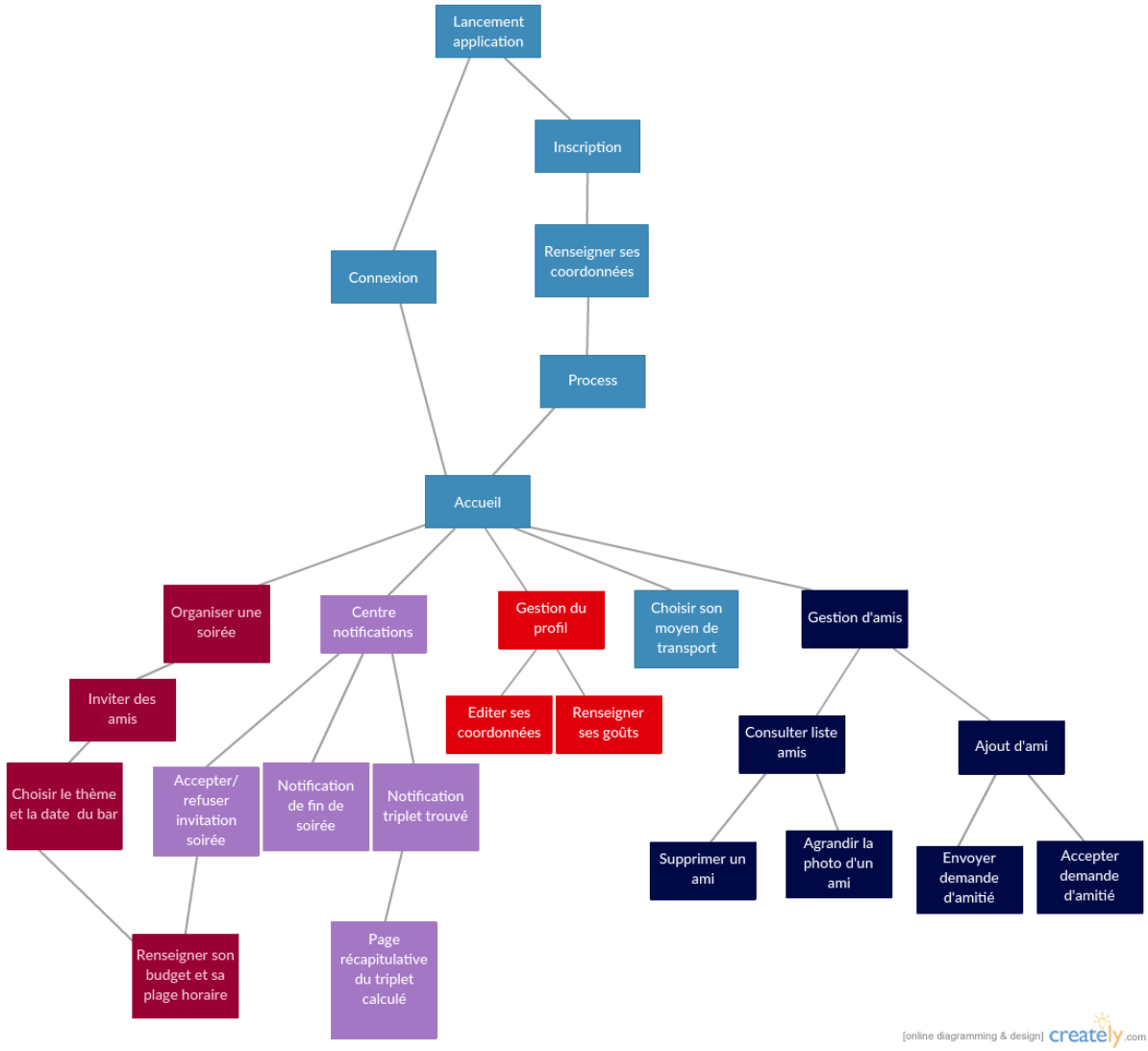
CAHIER DE CONCEPTION

Auteurs :
Koné Aly
Benbakli Sonia
Kazi Aoual Malik

1. Introduction

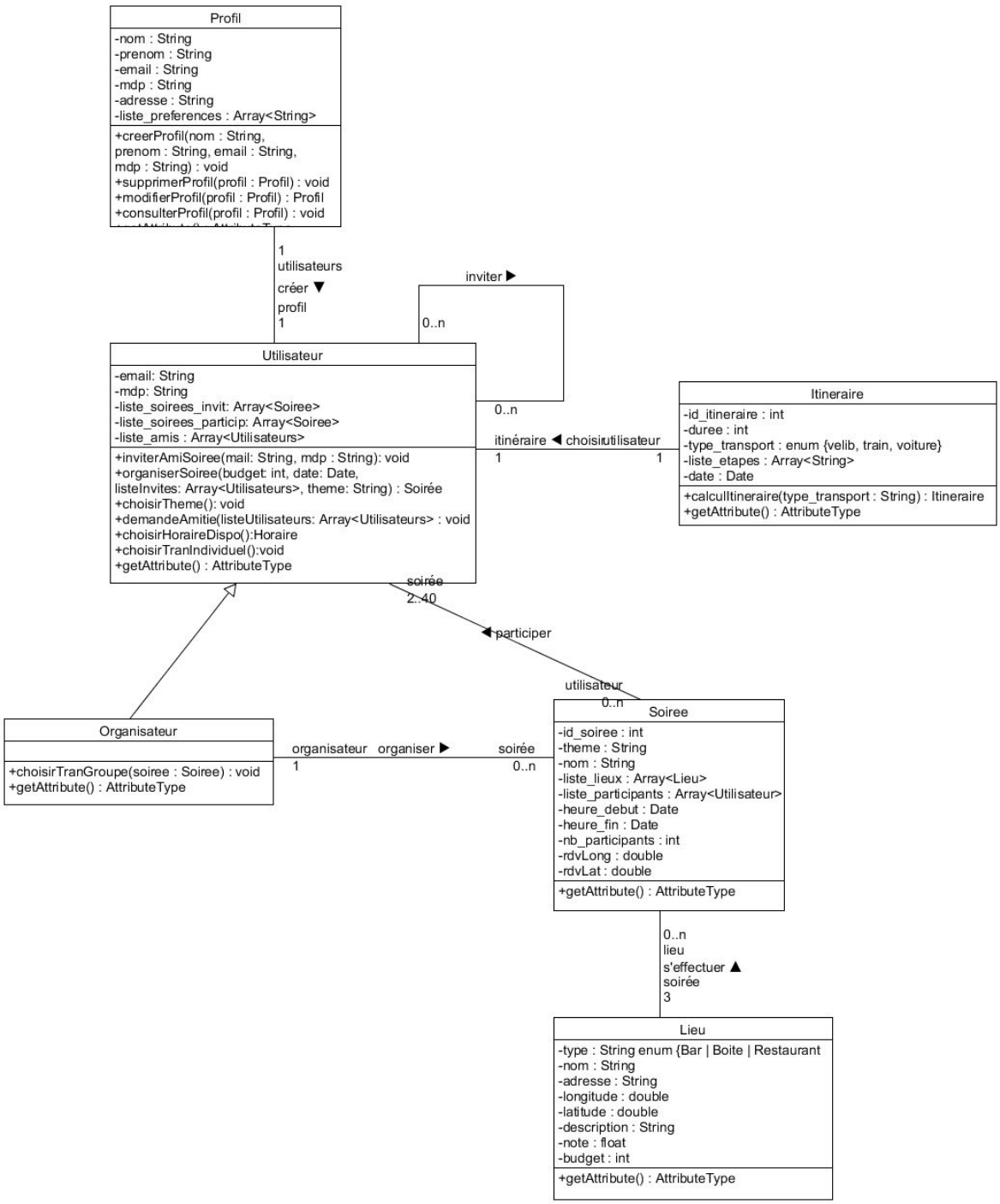
Le document présent est la suite de la phase d'analyse et traite des spécificités techniques de l'application. Il permet de rapporter les résultats de la recherche lors de la phase de **conception** où toutes les informations relatives à l'implémentation sont spécifiées.
Nous demandons aux **codeurs** de prendre soin de **lire** dans la **totalité** le document ci-présent et de respecter les règles qui y sont définies.
Nous nous **engageons** auprès du client sur le fait que toutes les spécifications rédigées dans le cahier des charges sont **parfaitement** retranscrites dans le cahier de conception ici présent.

Fig.1 : Schéma explicatif de la structure de l'application
(repris du cahier des charges)



2. Diagramme des classes

Fig.2 : Diagramme des classes



3. Contraintes à vérifier

Les contraintes listées dans la suite sont tirées du cahier des charges

Le point de rendez-vous

L'application prendra en compte la localisation de chaque personne du groupe. Afin de déterminer une zone qui convient à tout le monde, l'algorithme calcule le centre du cercle circonscrit de diamètre : les deux personnes du groupes les plus éloignées, pondéré par le nombre de personnes présentes dans le demi cercle perpendiculaire à la droite correspondant au diamètre choisi. Ce point sera utilisé pour la recherche du point de rendez-vous au restaurant.

Les horaires de disponibilité

- La plage horaire de disponibilité sera d'au moins 6 heures.
- Lors de l'arrivée en boîte, si l'horaire de fin de disponibilité est inférieur à l'horaire de fermeture de la boîte proposée, l'application enverra une notification à l'utilisateur seul, à son horaire de fin de disponibilité, qui redirigera vers un écran contenant son itinéraire retour. En revanche, si l'horaire de fin de disponibilité de l'usager est supérieure à l'horaire de fermeture de la boîte, l'application enverra une notification à l'utilisateur, à l'heure de fermeture de la boîte afin de le renseigner sur son itinéraire de retour.
- La plage horaire minimum sera 18h-00h.
- L'horaire de début de disponibilité de chacun devra se situer entre 18h et 20h.

Les goûts de chacun

- Afin de choisir le restaurant, l'application oubliera tout d'abord les restaurant qui correspondent à ce que les différents personnes du groupe **n'aiment pas**. Ensuite, le choix s'effectuera sur le goût présent dans la **majorité** des usagers dans le groupe de la soirée.
- S'il s'avère qu'il n'y a **aucun** point commun entre les goûts des différentes personnes du groupe, l'algorithme choisira le restaurant ayant la **meilleure note** parmi tous les restaurants correspondant aux goûts de chacun des participants de la soirée.

Les moyens de transport

- Le trajet en vélo ne sera disponible qu'en vélib' afin de faciliter l'implémentation de l'itinéraire.
- Les trajets de groupe de la soirée ne peuvent pas être choisis par un utilisateur non organisateur de la soirée

La méthode du choix des lieux

Après avoir quitté le bar, les usagers du groupe ne doivent **pas** effectuer un trajet d'une durée **supérieure à 20 min** pour aller au restaurant, de même pour la boîte de nuit. Ainsi, nous limiterons la recherche du restaurant à **6 km autour du bar** et à **6 km autour du restaurant**.
Lors du choix du lieu, l'application liste tout d'abord les lieux correspondant au goût choisi par la majorité. Ensuite, elle sélectionne les endroits correspondant au budget général du groupe. L'application devra choisir parmi cette liste l'endroit ayant été le **mieux noté** par les avis Google.

4. Les tests unitaires

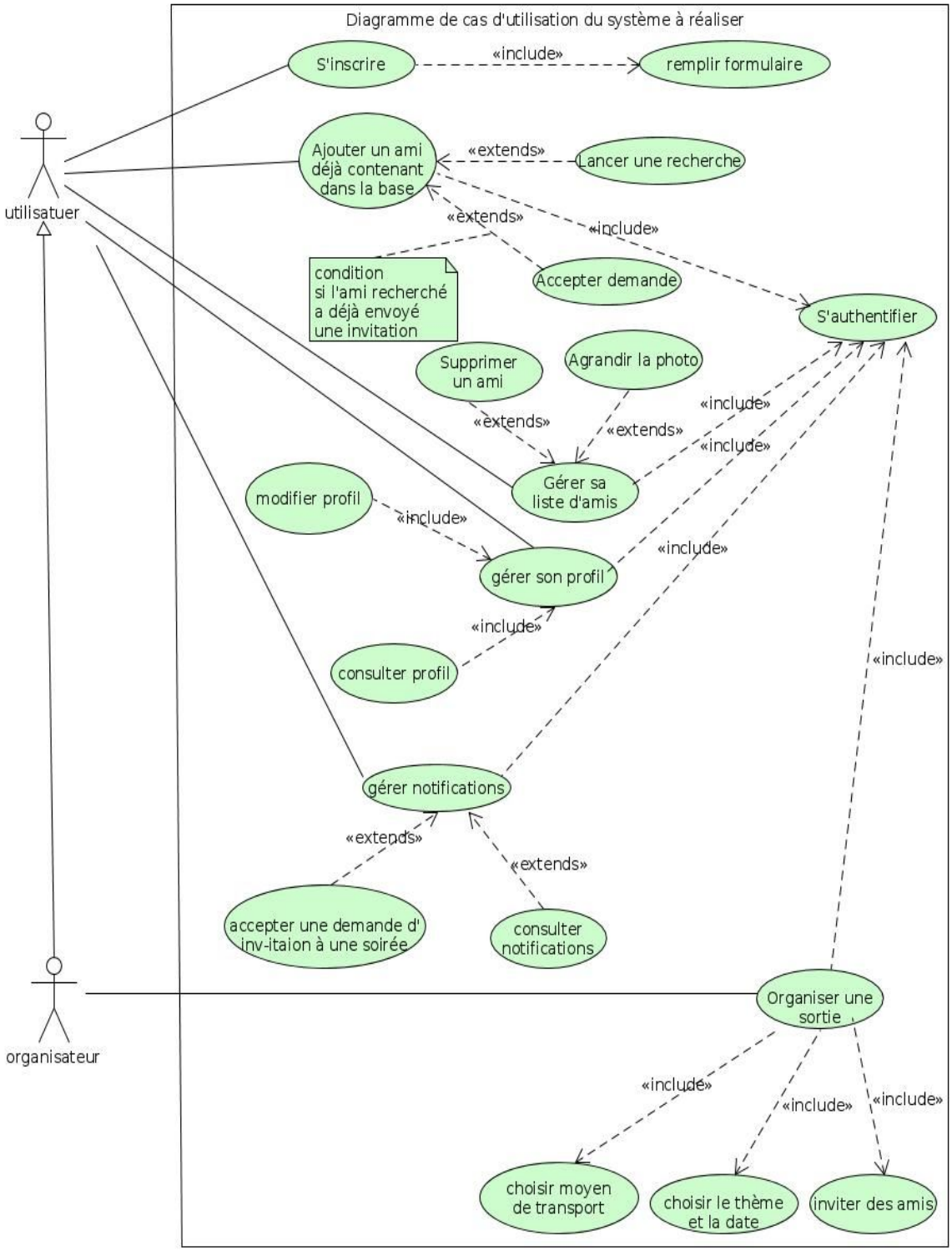
Dans les tableaux ci-dessous, sont listés tous les tests unitaires pertinents à vérifier lors de la phase d'implémentation de l'application.

N°Test	Objectif du test	Données en entrée	Résultat attendu
1	Demander un trajet pour aller au bar en étant en dehors de Paris	**Position :** Ris-Orangis, 91130, France **Arrivée :** 3 rue de la Bourse, 75002, Paris, France	Message d'erreur indiquant que l'application ne peut proposer un trajet que si la personne se situe dans Paris
2	Inscription avec une adresse en dehors de Paris	**Nom :** Tartenpion **Prénom :** Dupond **Mdp :** azertyuiop **Adresse :** 8 rue de Versailles, Orsay, 91400	Message d'erreur indiquant que l'application ne fonctionne qu'avec les résidents de Paris
3	Organiser deux soirées à la même date pour un même utilisateur	- **Soirée 1 :** **Thème :** Bar à chicha **Heure début :** 30/03/2017 18h **Budget :** 3 **Liste participants :** [Aly, Malik, Sonia] - **Soirée 2 :** **Thème :** Bar à bières **Heure début :** 30/03/2017 19h **Budget :** 1 **Liste participants :** [Aly, Gertrude]	Message d'erreur indiquant que l'utilisateur ne peut pas participer à deux soirées le même jour
4	Organiser une soirée	**Nom :** La papaye verte	Message d'erreur

	dans un lieu en dehors de Paris	**Adresse :** 4 Rue Archange, 91400 Orsay	indiquant que la soirée ne peut pas être organiser en dehors de Paris
5	Organiser une soirée à une date strictement antérieure au jour actuel	**Thème :** Bar à chicha **Date actuelle :** 25/03/2017 **Date :** 14/03/2017 **Budget :** 3	Message d'erreur indiquant que la date en entrée est dépassée
6	Choisir un budget pour la soirée en dehors de l'intervalle [1, 4]	**Thème :** Bar à chicha **Date :** 29/12/2017 **Budget :** 5	Message d'erreur indiquant que le budget doit être compris entre 1 et 4
7	Organiser une soirée le jour actuel s'il est plus de 16h	**Thème :** Bar à chicha **Date actuelle :** 27/03/2017 16h15 **Date :** 27/03/2017 **Budget :** 3	Message d'erreur indiquant à l'utilisateur qu'il ne peut pas sélectionner cette date car il est plus de 16h
8	Créer un profil avec un champ manquant	**Nom :** Poule **Prénom :** Hubert **e-mail :** **mdp :** qsdfaqwzsx	Message d'erreur indiquant le champ manquant "Veuillez remplir le champ e-mail"
9	Ajouter un goût inexistant dans la liste des goûts de l'utilisateur	**Liste préférences :** Chinois, Thaïlandais, Fast-food, choukchouka	Message d'erreur indiquant que le goût sélectionné n'est pas valide
10	Sélectionner un thème inexistant dans la liste des thèmes pour une soirée	**Thème :** Bar à galets **Date :** 29/12/2017 **Budget :** 4	Message d'erreur indiquant que le thème sélectionné n'est pas valide
11	Renseigner une horaire de début de disponibilité non conforme à la plage horaire valide (18h - 20h)	**horaire début :** 30/03/2017 6h **horaire fin :** 31/03/2017 00h	Message d'erreur indiquant que l'horaire de début de disponibilité doit être entre 18h et 20h
12	Renseigner une horaire de fin de disponibilité non conforme à la plage horaire valide (minimum 00h j+1 à la date de début de la soirée)	**horaire début :** 30/03/2017 18h **horaire fin :** 30/03/2017 23h	Message d'erreur indiquant que la date indiquée doit être au moins minuit

4. Diagramme des cas d'utilisation

Fig.3 : Diagramme des cas d'utilisation
(déjà présent dans le cahier des charges)

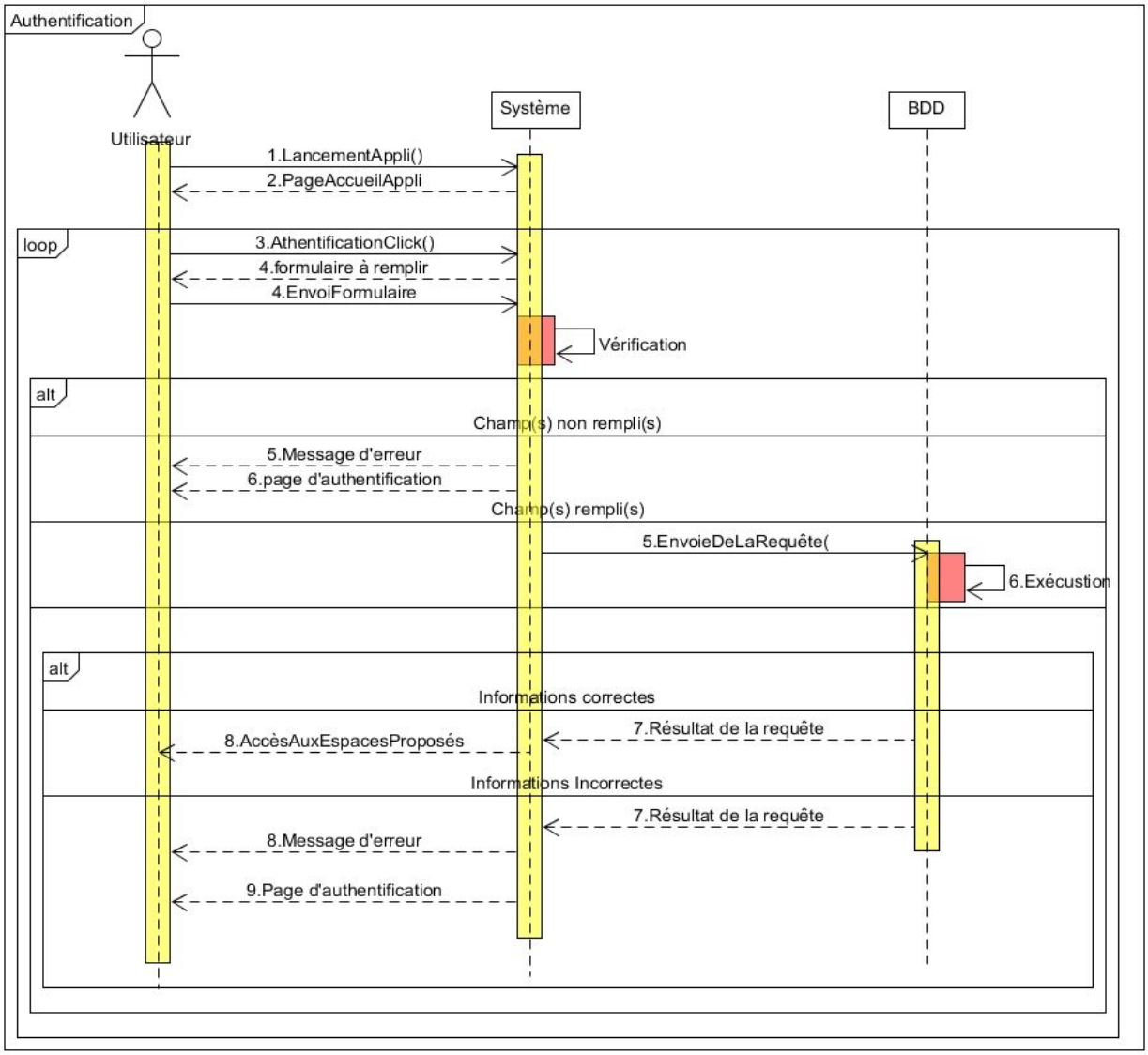


5. Description détaillée des cas d'utilisation avec leur diagramme de séquence

5.1. Authentification

Sommaire d'identification
But : Accès au système **Acteur :** Utilisateur
Description des enchaînements
Le cas d'utilisation commence lorsque l'utilisateur atteint le portail **Pré-condition :** Aucune **Post-condition :** L'utilisateur accède à la page d'accueil **Scénario :** 1. L'utilisateur saisit l'id et le mdp 2. Le système ouvre une session et attribue tous les droits d'accès 3. Le système charge la page d'accueil
Enchaînements alternatifs
1. Affichage d'un message d'information disant que l'id ou le mdp est incorrect 2. Retour à l'étape 1 du scénario

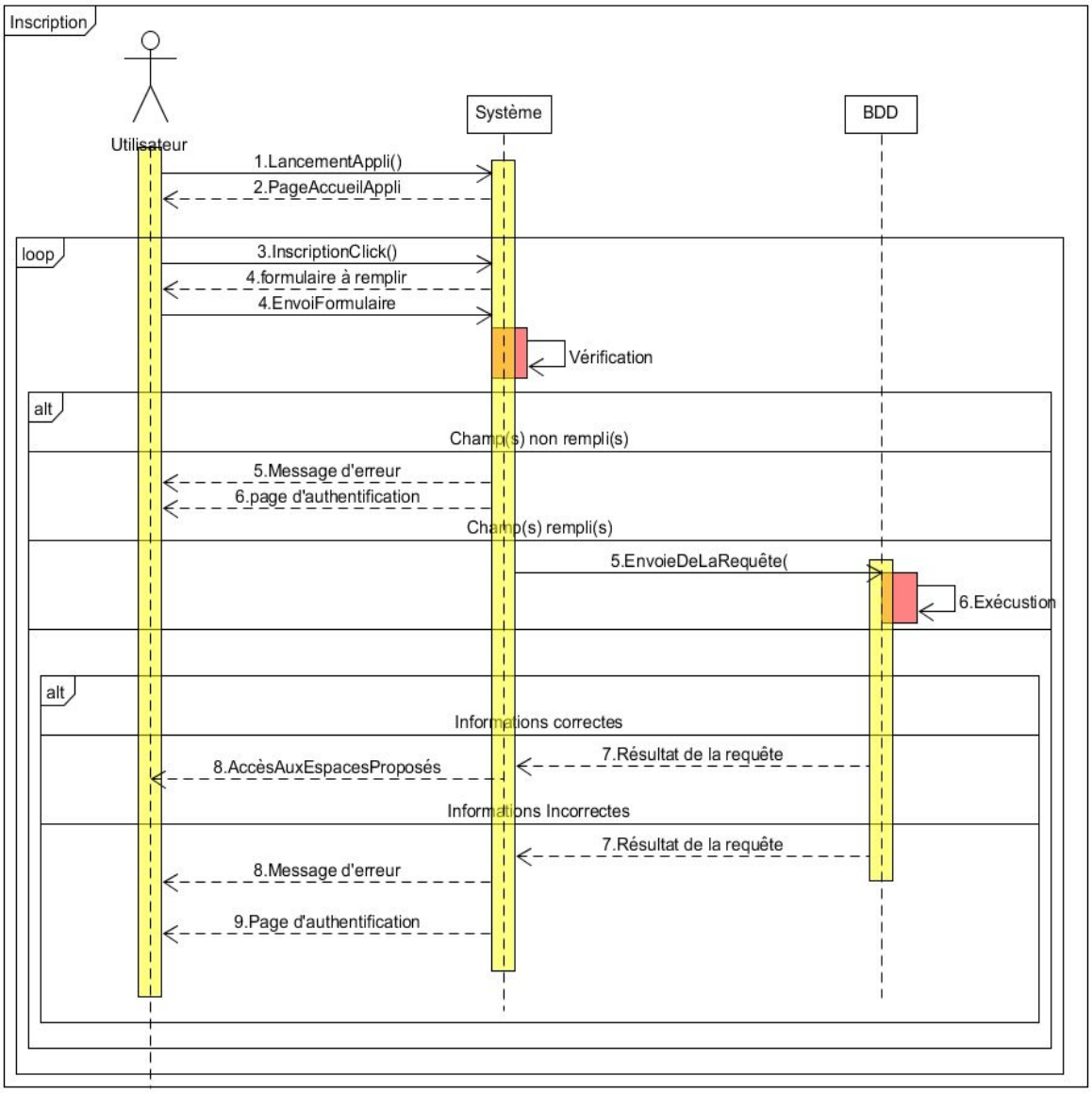
Fig. 4 : Diagramme de séquence du cas : Authentification



5.2. Création profil

Sommaire d'identification
But : Créer un profil **Acteurs :** Utilisateur.
Description des enchaînements
Le cas commence lorsque l'utilisateur clique sur s'inscrire et qu'il sera sur la page d'inscription. **Pré-condition :** Aucune. **Post-condition :** Le profil de l'utilisateur est créé et ajouté dans la base de données du système et l'utilisateur accède à la page d'accueil **Scénario nominal :** 1. L'utilisateur remplit le formulaire 2. Le système vérifie si l'utilisateur n'existe pas déjà 3. Le système crée le profil avec les informations renseignées et le rajoute dans la base de données 4. Le système affiche une confirmation d'inscription 5. Le système affiche la page d'accueil
Enchaînements alternatifs
Si le système détecte le fait que l'utilisateur existe déjà, il affiche un message d'erreur et revient à l'étape 1.

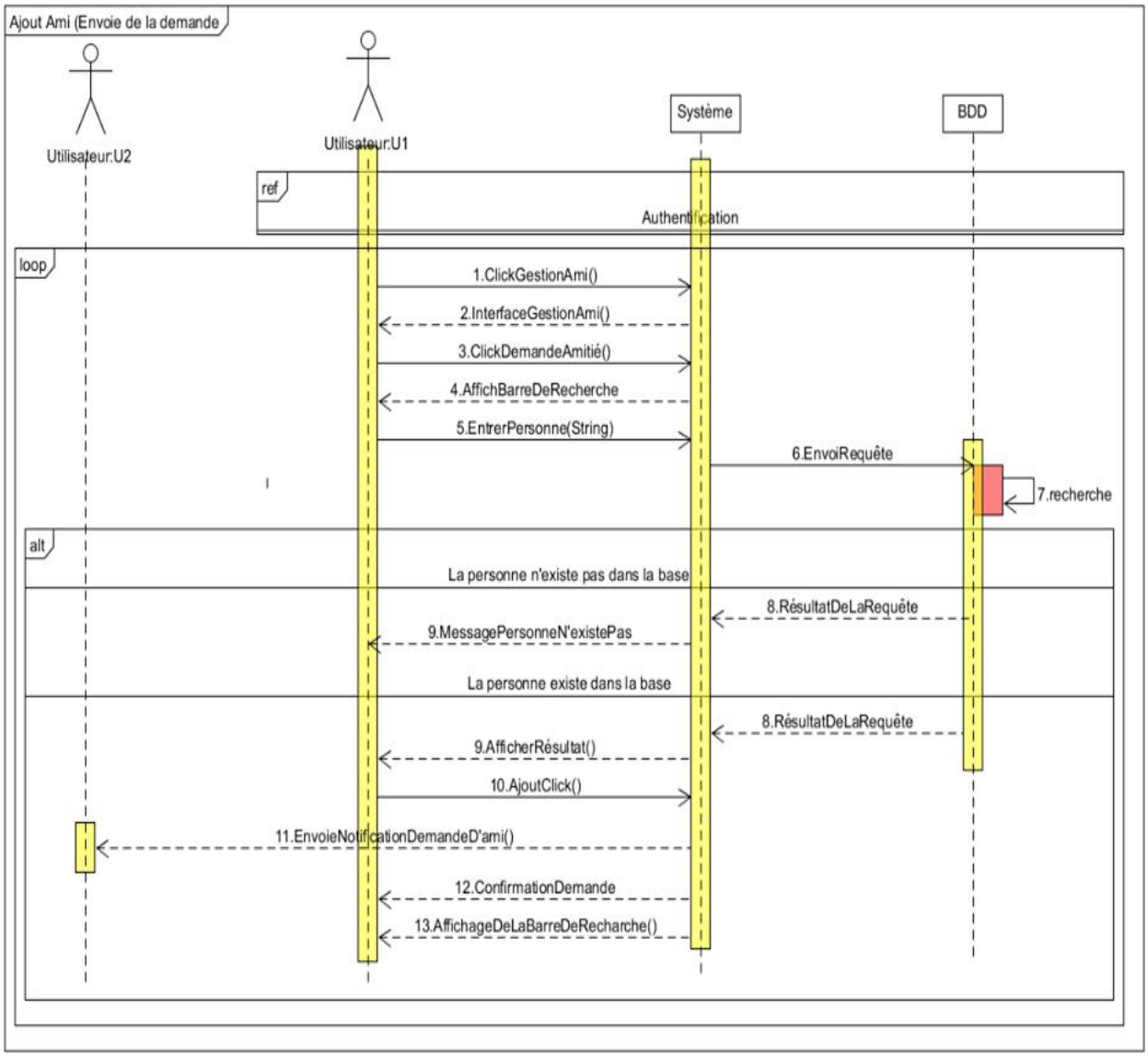
Fig.5 : Diagramme de séquence du cas d'utilisation : Créer profil



5.3. Envoyer une demande d'amitié

Sommaire d'identification
But : Envoyer une demande d'amitié **Acteurs :** Utilisateur 1 (qui envoie une demande à l'utilisateur 2), utilisateur 2.
Description des enchaînements
Ce cas commence lorsque l'utilisateur 1 clique sur l'onglet '+' de la page d'accueil. **Pré-condition :** - L'utilisateur 1 doit être connecté - L'utilisateur 2 que la personne veut ajouter existe bien dans la base de données - Les deux utilisateurs ne doivent pas être amis **Post-condition :** - L'invitation a été envoyée - L'utilisateur 2 a reçu la notification de demande **Scénario nominal :** 1. L'utilisateur 1 clique sur l'onglet '+' de la page d'accueil 2. L'utilisateur 1 clique sur le bouton "ajouter un ami" 3. L'utilisateur 1 recherche l'utilisateur 2 qu'il veut ajouter 4. L'utilisateur 1 clique sur le bouton '+' à côté du résultat trouvé 5. Le système envoie l'invitation à l'utilisateur 2 6. Le système confirme l'envoi de l'invitation
Enchaînements alternatifs
Si l'utilisateur 2 n'existe pas dans la base de données, la recherche ne renvoie aucun résultat.

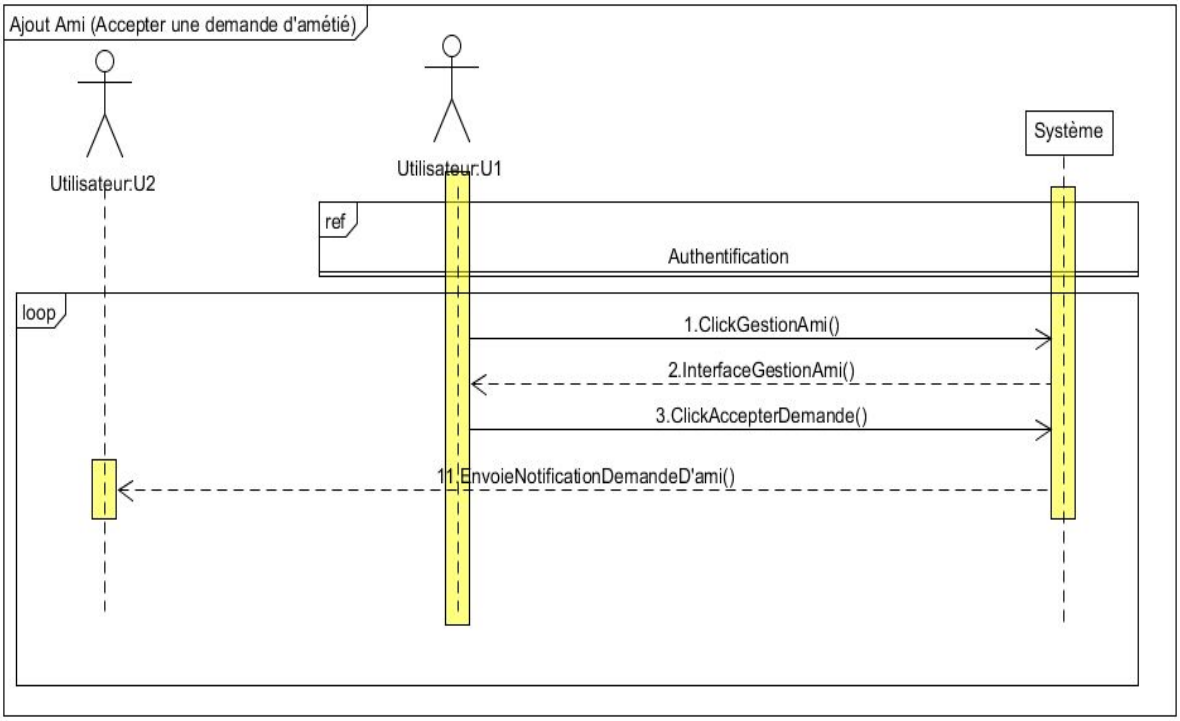
Fig.6 : Diagramme de séquence du cas d'utilisation : Envoyer demande d'amitié



5.4. Accepter une demande d'amitié

Sommaire d'identification
But : Accepter une invitation à rejoindre le cercle d'amis d'un utilisateur **Acteurs :** Utilisateur 1, utilisateur 2 (qui accepte la demande de l'utilisateur 1).
Description des enchaînements
Le cas commence lorsque l'utilisateur 2 clique sur l'onglet "gestion d'amitiés" **Pré-condition :** - L'utilisateur 2 doit être connecté **Post-condition :** - L'utilisateur 2 est bien ajouté dans la liste d'amis de l'utilisateur 1 - L'utilisateur 1 est bien ajouté dans la liste d'amis de l'utilisateur 2 **Scénario nominal :** 1. L'utilisateur 2 clique sur l'onglet de la gestion d'amis 2. L'utilisateur 2 clique sur le bouton confirmer de la notification 3. L'utilisateur 1 reçoit une notification de confirmation d'ajout 4. L'utilisateur 2 est ajouté dans la liste d'amis de l'utilisateur 1 5. L'utilisateur 1 est ajouté dans la liste d'amis de l'utilisateur 2
Enchaînements alternatifs
Si l'utilisateur refuse la demande d'amitié, le système ne fait rien.

Fig.7 : Diagramme de séquence du cas d'utilisation : Accepter demande d'amitié



5.5. Supprimer un ami

Sommaire d'identification

But :
Supprimer un ami de sa liste d'amis

Acteur :
Utilisateur 1 (qui veut supprimer de sa liste l'utilisateur 2), utilisateur 2.

Description des enchaînements

Ce cas commence lorsque l'utilisateur clique sur l'onglet '+' de la page d'accueil

Pré-condition :

- L'utilisateur 2 à supprimer doit être dans la liste d'amis de l'utilisateur 1

Post-condition :

- L'utilisateur 2 est bien supprimé de la liste d'amis de l'utilisateur 1
- L'utilisateur 1 est bien supprimé de la liste d'amis de l'utilisateur 2

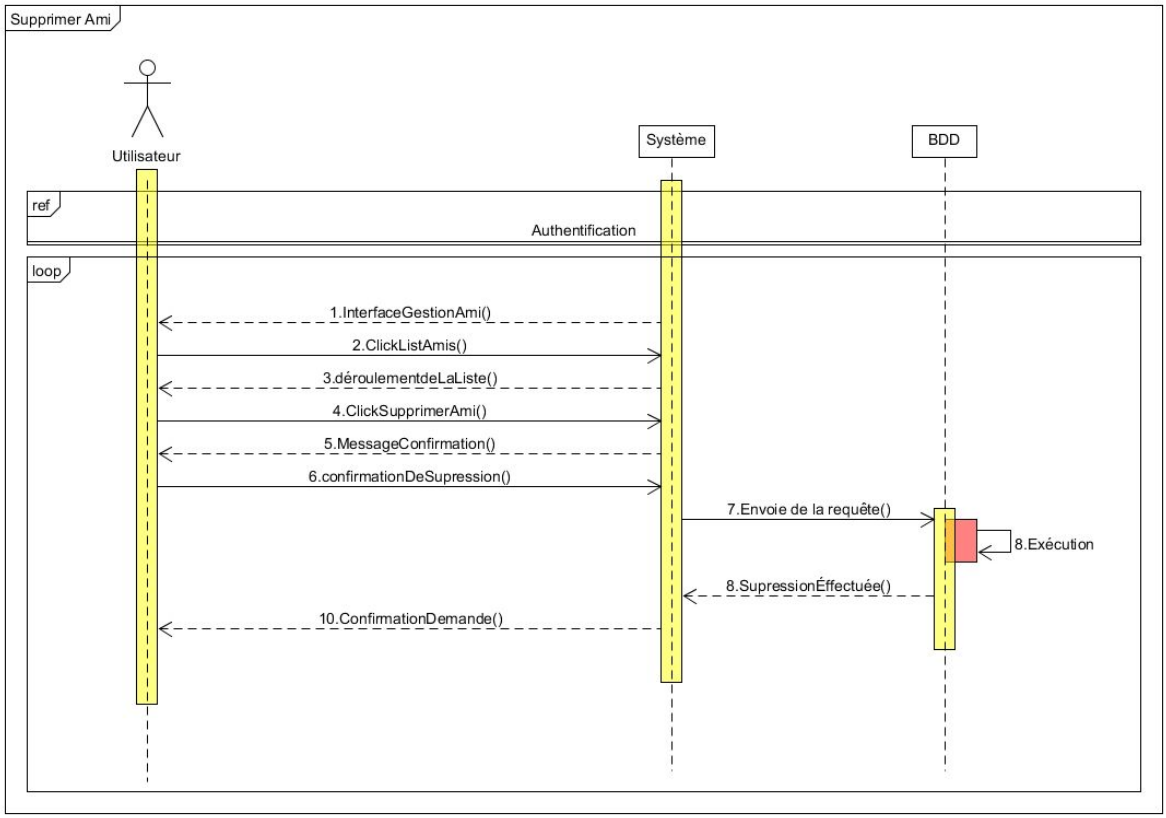
Scénario nominal :

1. L'utilisateur 1 clique sur le bouton '+' de la page d'accueil
2. L'utilisateur 1 clique sur le bouton supprimer à côté de l'utilisateur 2
3. Le système affiche un message demandant à l'utilisateur s'il est sûr de vouloir supprimer cet ami
4. L'utilisateur clique sur "oui"
5. L'utilisateur 2 est supprimé de la liste d'amis de l'utilisateur 1
6. L'utilisateur 1 est supprimé de la liste d'amis de l'utilisateur 2

Enchaînements alternatifs

1. L'utilisateur 1 veut agrandir la photo de l'utilisateur 2
2. L'utilisateur 1 clique sur la photo de l'utilisateur 2
3. La photo de l'utilisateur 2 s'affiche en plein écran

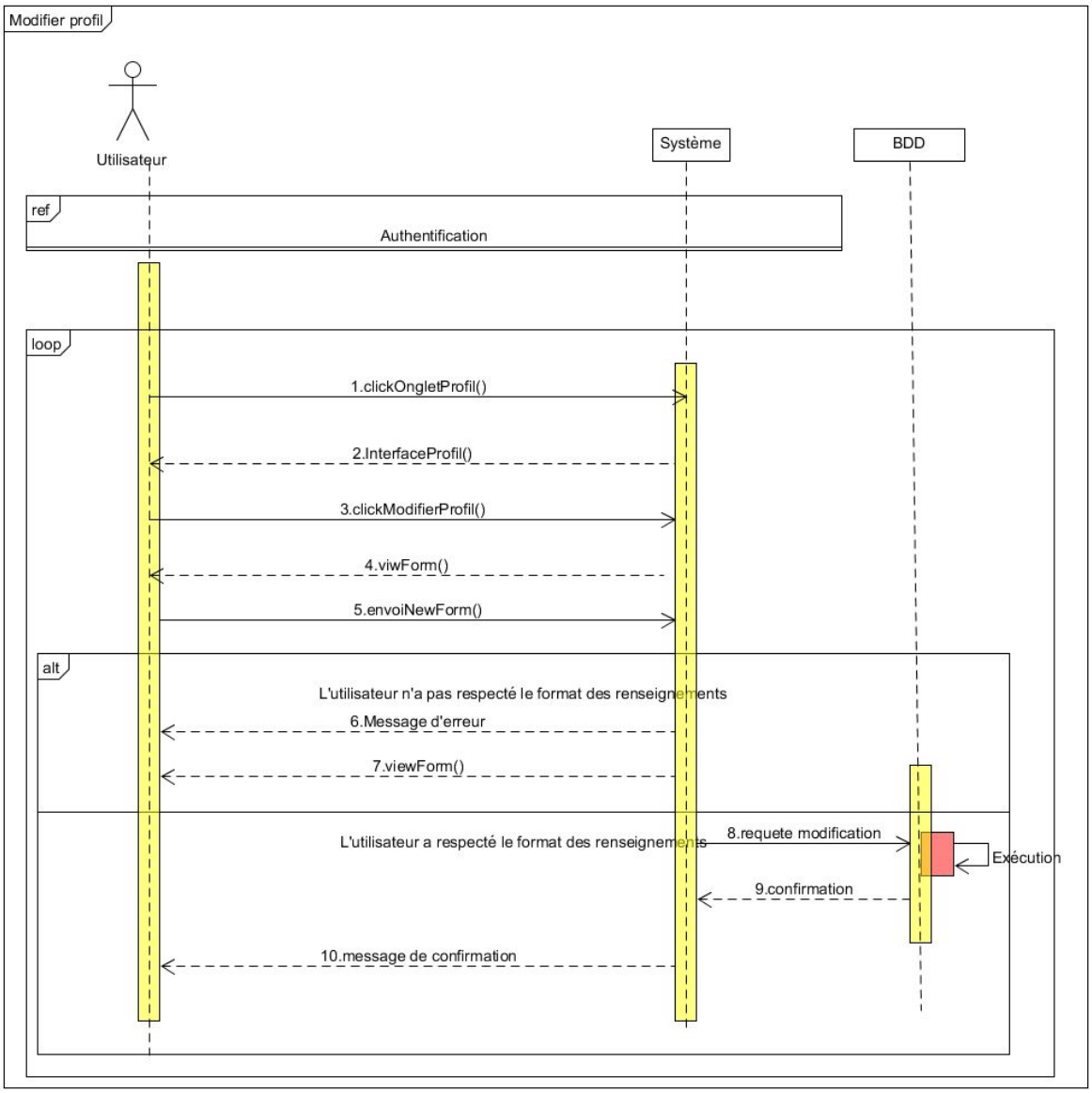
Fig.8 : Diagramme de séquence du cas d'utilisation : Supprimer un ami



5.6. Modifier profil

Sommaire d'identification
But : Modifier ses préférences ou ses coordonnées. **Acteurs :** Utilisateur.
Description des enchaînements
Ce cas commence lorsque l'utilisateur clique sur l'onglet représentant un buste **Pré-condition :** - L'utilisateur doit être connecté **Post-condition :** - Les modifications ont bien été enregistrées **Scénario nominal :** 1. L'utilisateur clique sur l'onglet d'édition de profil 2. L'utilisateur clique sur un bouton "modifier profil" 3. L'utilisateur renseigne les informations qu'il a envie de modifier 4. L'utilisateur clique sur "enregistrer les modification" 5. Le système enregistre les modifications 6. Le système affiche un message de confirmation de modifications
Enchaînements alternatifs
Si le format des renseignement n'est pas respecté, le système affiche un message d'erreur et revient sur le formulaire à l'étape 3.

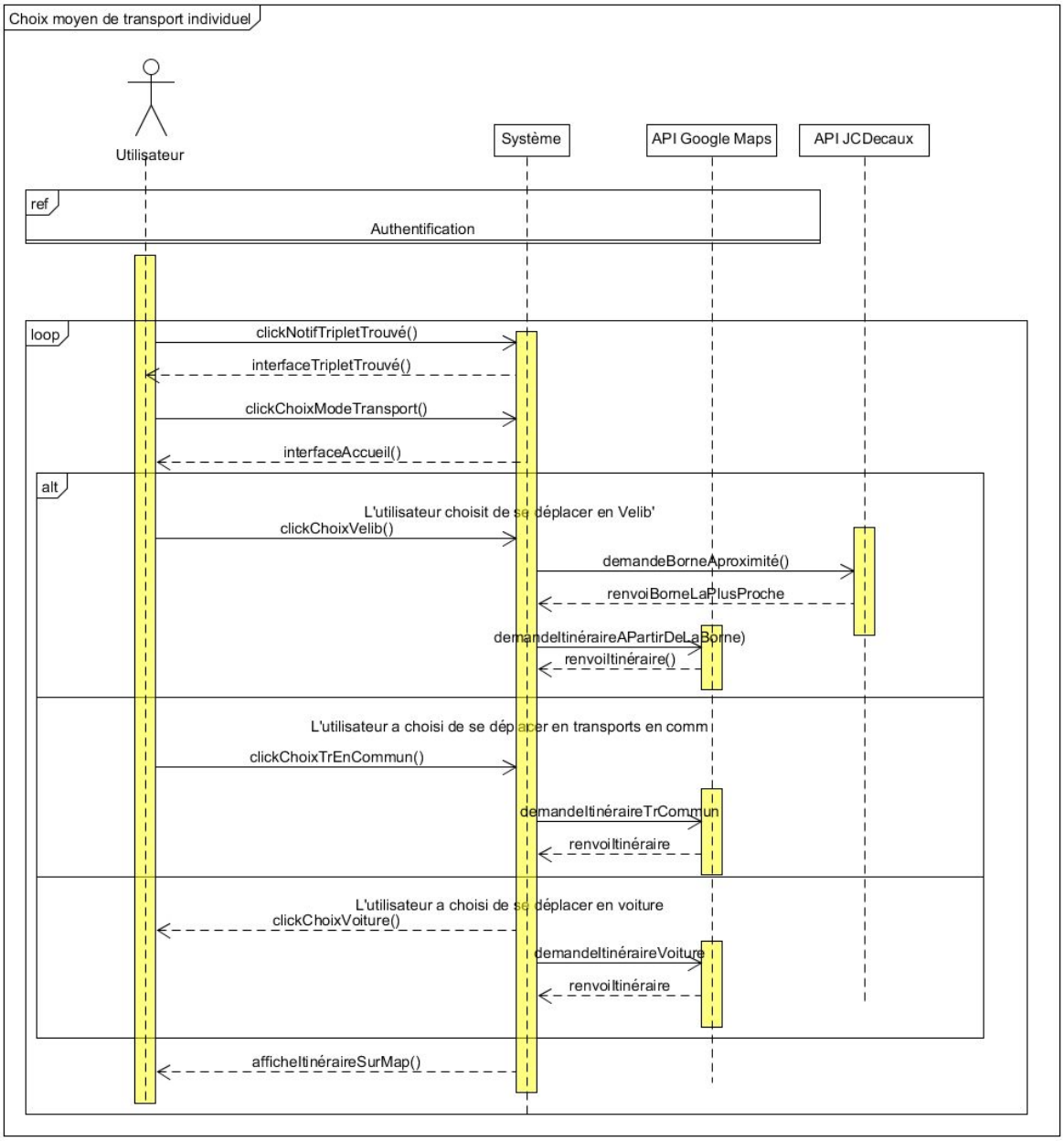
Fig.9 : Diagramme de séquence du cas d'utilisation : Envoyer demande d'amitié



5.7. Choix du moyen de transport individuel

Sommaire d'identification
But : Choisir son moyen de transport pour l'itinéraire d'aller à la soirée et de retour de la soirée **Acteurs :** Utilisateur.
Description des enchaînements
Ce cas commence lorsque le triplet de la soirée a été trouvé **Pré-condition :** - L'utilisateur est invité à la soirée - L'utilisateur doit être connecté - L'utilisateur a accepté l'invitation à la soirée **Post-condition :** - Le trajet est généré et l'itinéraire est affiché sur la page d'accueil **Scénario nominal :** 1. L'utilisateur a reçu la notification du triplet trouvé 2. L'utilisateur clique sur l'onglet par défaut 3. L'utilisateur choisit son moyen de transport en cliquant sur le bouton "démarrer" 4. L'itinéraire est affiché sur l'onglet d'accueil
Enchaînements alternatifs
Si la position de l'utilisateur est en dehors de Paris, le système affiche un message afin de lui indiquer que l'application ne fonctionne que dans Paris intra-muros.

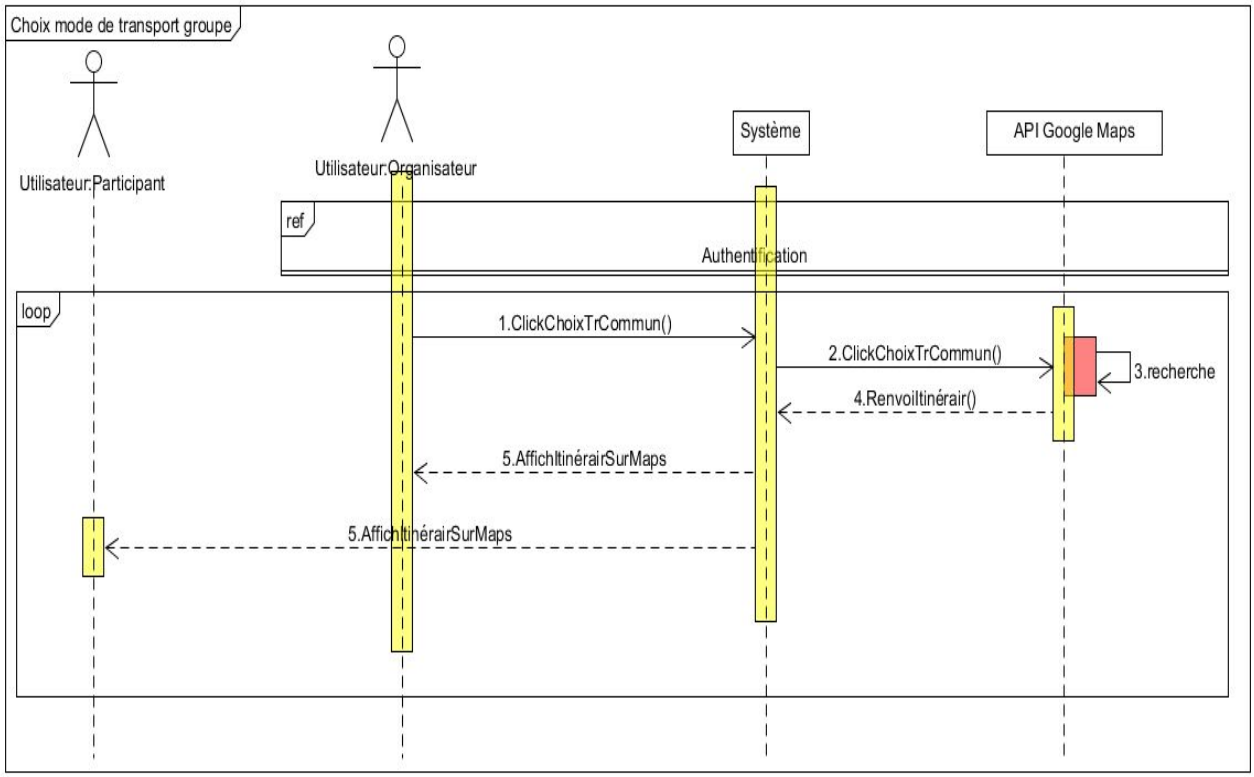
Fig.10 : Diagramme de séquence du cas d'utilisation : Envoyer demande d'amitié



5.8. Choix du moyen de transport de groupe

Sommaire d'identification
But : Choisir le moyen de transport du groupe lors de la soirée entre le bar et le restaurant et le restaurant et la boîte **Acteurs :** L'organisateur de la soirée, les utilisateurs participant à la soirée
Description des enchaînements
Ce cas commence lorsque les utilisateurs sont réunis dans le bar. **Pré-condition :** - L'utilisateur doit être connecté - Le triplet a été généré - Les utilisateurs sont réunis dans le bar ou dans le restaurant **Post-condition :** - Le trajet est généré et l'itinéraire est affiché sur chaque terminal de chaque utilisateur de la soirée **Scénario nominal :** 1. Les participants se sont retrouvés dans le bar ou le restaurant 2. L'organisateur clique sur l'onglet par défaut 3. L'organisateur choisit le moyen de transport de groupe en cliquant sur le bouton "démarrer" 4. L'itinéraire est affiché sur la page d'accueil de chaque utilisateurs participant à la soirée

Fig.11 : Diagramme de séquence du cas d'utilisation : Choix mode de transport en groupe



5.9. Organiser une soirée

Sommaire d'identification

But :
Organiser une soirée avec un triplet (bar, restaurant, boîte)

Acteurs :
L'organisateur de la soirée, les utilisateurs invités à la soirée

Description des enchaînements

Le cas commence lorsque l'utilisateur clique sur le bouton "organiser une soirée" dans l'onglet d'accueil

Pré-condition :

- L'organisateur doit être connecté
- Les plages horaires doivent respecter le format suivant : la plage horaire devra être de 6h minimum, et l'horaire de début de la plage horaire devra se situer entre 18h et 20h
- Le budget de la soirée devra se situer entre 1 et 4
- S'il est plus de 16h, la date sélectionnée devra être au moins un jour après la date de l'organisation de la soirée.
- S'il est avant 16h, la date sélectionnée devra être supérieure ou égale à la date de l'organisation de la soirée.
- Au moins une personne invitée doit avoir accepté l'invitation
- Le nombre de personnes invitées doit être compris entre 2 et 40

Post-condition :

- Le triplet généré devra être au centre du cercle circonscrit des positions des différents participants de la soirée
- Le bar et le restaurant du triplet devront se trouver à une distance de 6 km entre eux. De même entre le restaurant et la boîte
- Le restaurant proposé ne devra pas avoir comme thème celui qu'une personne dans le groupe n'aime pas
- Le budget des endroits proposés devra correspondre à celui que l'organisateur a renseigné
- Les lieux proposés devront se situer dans Paris intra-muros

Scénario nominal :

1. L'utilisateur clique sur "organiser une soirée"
2. L'utilisateur choisit les amis qu'il veut inviter à la soirée en cliquant sur le bouton '+' à côté de chaque ami qu'il veut inviter et clique sur le bouton "Suivant"

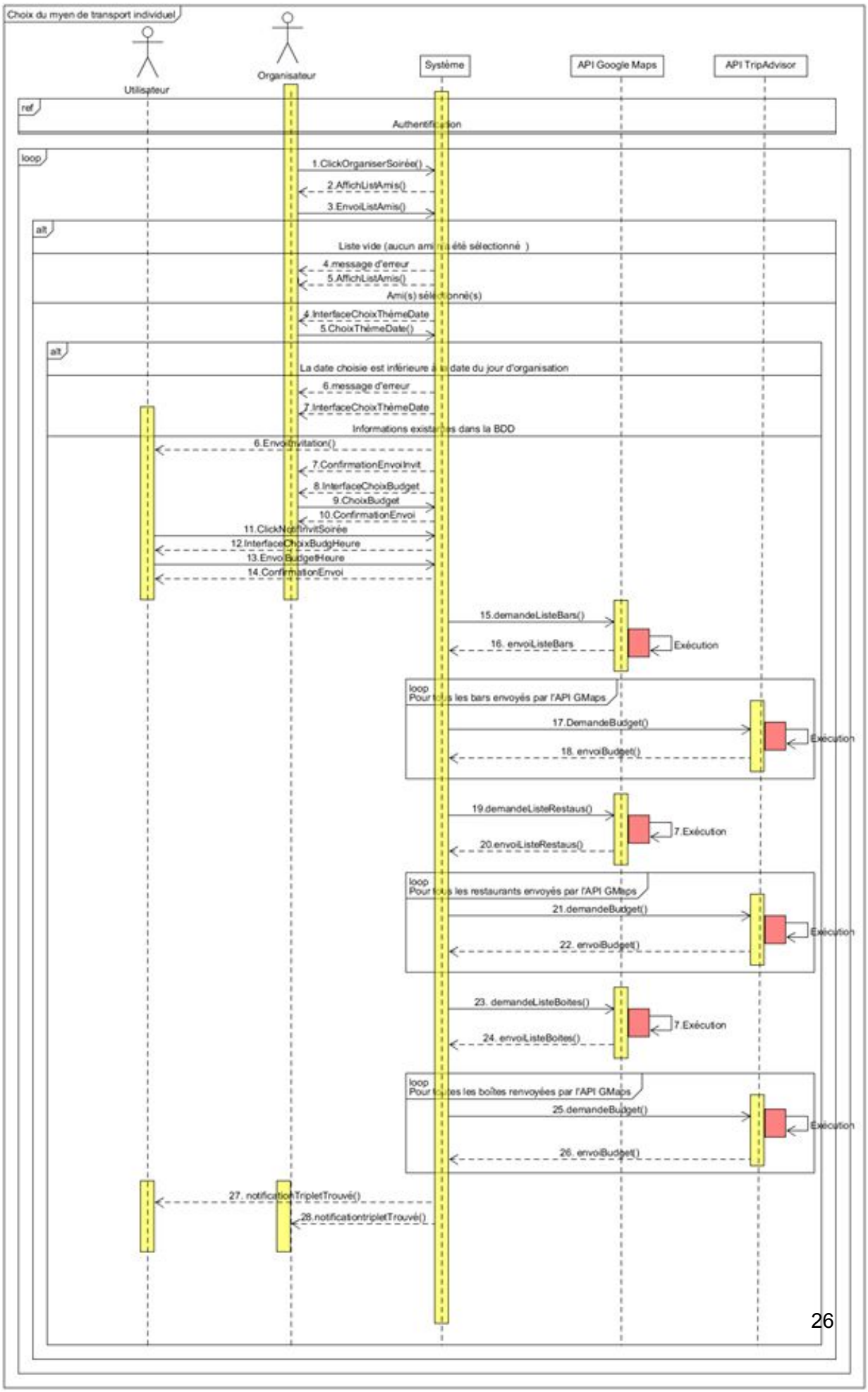
3. L'utilisateur renseigne le thème qu'il veut pour le bar, la date de la soirée ainsi que le budget de la soirée et clique sur "valider"
4. Le système envoie une notification d'invitation à tous les invités
5. L'organisateur est redirigé vers la page de renseignements individuels.
6. L'organisateur renseigne ses horaires de disponibilité
7. Les participants renseignent leurs horaires de disponibilité avant 16h
8. Le triplet est généré à 16h le jour de la soirée ou lorsque tous les invités confirment ou infirment leur venue
9. Le système envoie une notification de triplet trouvé à tous les participants de la soirée

Enchaînements alternatifs

- Si l'organisateur ne reçoit aucune acceptation d'invitation, la soirée est supprimée de la base de données et une notification lui sera envoyée.
- Si l'utilisateur choisit une plage horaires de disponibilités invalide, le système affiche un message d'erreur et demande de renseigner une nouvelle fois sa plage horaires.
- Si l'utilisateur invite plus de 40 personnes, le système affiche un message lui disant que la limite de places dans la soirée est atteinte.
- Si un invité ne répond pas avant 16h du jour de la soirée, l'invité est supprimé de la liste des participants de la soirée. De même si l'invité refuse l'invitation.

En dessous :

Fig. 12 : Diagramme de séquence du cas d'utilisation : Organiser une soirée


Choix du myen de transport individuel
Utilisateur
Organisateur
Système
API Google Maps
API TripAdvisor
ref
Authentification
loop
1.ClickOrganiserSoirée()
2.AffichListAmis()
3.EnvoiListAmis()
alt
Liste vide (aucun ami n'a été sélectionné)
4.message d'erreur
5.AffichListAmis()
Ami(s) sélectionné(s)
4.InterfaceChoixThèmeDate
5.ChoixThèmeDate()
alt
La date choisie est inférieure à la date du jour d'organisation
6.message d'erreur
7.InterfaceChoixThèmeDate
Informations existantes dans la BDD
6.EnvoiInvitation()
7.ConfirmationEnvoiInvit
8.InterfaceChoixBudget
9.ChoixBudget
10.ConfirmationEnvoi
11.ClickNotifInvitSoirée
12.InterfaceChoixBudgHeure
13.EnvoiBudgetHeure
14.ConfirmationEnvoi
15.demandeListeBars()
16. envoiListeBars
Exécution
loop
Pour tous les bars envoyés par l'API GMaps
17.DemandeBudget()
Exécution
18. envoiBudget()
19.demandeListeRestaus()
7.Exécution
20.envoiListeRestaus()
loop
Pour tous les restaurants envoyés par l'API GMaps
21.demandeBudget()
Exécution
22. envoiBudget()
23. demandeListeBoites()
7.Exécution
24. envoiListeBoites()
loop
Pour toutes les boîtes renvoyées par l'API GMaps
25.demandeBudget()
Exécution
26. envoiBudget()
27. notificationTripletTrouvé()
28.notificationtripletTrouvé()

Pseudo code explicatif du cas d'utilisation "organiser une soirée"

```
tantque(1) :
	Si clickOrganiserSoiree() Alors :
		afficheListeAmis();
		Array<Utilisateur> listeInvites = envoiListeAmis();
		Si listeInvites = NULL Alors :
			BEGIN
			afficheMessageErreur();
			afficheListeAmis();
			END;
		afficheInterfaceChoixThemeDateBudget();
		Si clickValider() Alors :
			Soiree s;
			s.setDate() = demandeDate();
			s.setTheme() = demandeTheme();
			s.setNom() = demandeNom();
			s.setListeInvites() = listeInvites;
			s.setOrganisateur() = utilisateurActuel();
			s.setBudget() = choixBudget();
			Pour tout Utilisateur u dans listeInvites :
				u.notificationInvitation();
			Organisateur o = utilisateurActuel();
			Pour tout Utilisateur u dans listeInvites :
				bool reponse = u.attenteReponse();
				Si reponse = NULL à 16h ou !reponse alors :
					listeInvites.remove(u);
			afficherMessageConfirmation();
			interfaceChoixHeure();
			int hDebut = o.choixHeureDebut();
			int hFin = o.choixHeureFin();
			A 16h :
			Array<int> heuresDebut = hDebut;
			Pour tout Utilisateur u dans listeInvites :
				heuresDebut.add(u.choixHeureDebut());
			s.setHeureDebut = calculHeureDebutSoiree(heuresDebut);
			s.setRdv = calculPtRDV();
			Array<Lieu,budget> listeBars = requeteGMaps(s.rdv);
			Pour tout Lieu l dans listeBars faire :
				requeteBudget(l);
			s.setListeLieux() = choixBar(listeBars);
			Array<Lieu,budget>		listeRestaus		=
		requeteGMaps(s.listeLieux(0).longitude, s.listeLieux(0).latitude);
			Pour tout Lieu l dans listeRestaus faire :
				requeteBudget(l);
			s.setListeLieux.add(choixRestaus(listeRestaus);
			Array<Lieu,budget>		listeBoites		=
		requeteGMaps(s.listeLieux(1).longitude, s.listeLieux(1).latitude);
			Pour tout Lieu l dans listeBoite faire :
				requeteBudget(l);
			s.setListeLieux.add(choixBoite(listeRestaus);
```

```
Pour tout Utilisateur u dans listeParticipants faire :
      u.envoiNotificationTripletTrouve(s.listeLieux);
```

6. Modèle relationnel

6.1. Modèle relationnel de données

Nous avons, à partir du diagramme des classes, établi le modèle relationnel de données qui représente la base de données que nous allons utiliser pour la gestion des différentes instances de classes.

- **Utilisateur** (email, mdp, liste_soirees_invit, liste_soirees_particip, liste_amis, #id_itineraire, #id_profil**);**
- **Itinéraire(** id_itineraire, durée, type_transport, liste_etapes, date**);**
- **inviter (**email**);**
- **Soirée (** id_soiree, nom, liste_lieux, liste_participants, heure_debut, heure_fin, theme, nb_participants, rdv_long, rdv_lat, #id_Organisateur**);**
- **Organisateur (**id_organisateur, #id_soiree , #email**);**
- **Profil (**id_profil, nom, prenom, adresse, mdp, liste_preferences, #email**);**
- **Lieu (**id_lieu, type, nom, adresse, longitude, latitude, description, note, budget**);**
- **S'effectuer (** id_soiree, id_lieu**);**
- **Participer (**id_utilisateur, id_soiree**);**
- **Organiser (**id_organisateur, id_soiree**);**

6.2. Dictionnaire de données

Utilisateur

Attribut	Définition de l'attribut	Type	Contraintes sur l'attribut
email	Adresse e-mail identifiant l'utilisateur	VARCHAR(30)	Doit être de format e-mail et unique et not null
mdp	Mot de passe associé à l'e-mail	VARCHAR(30)	Compris entre 8 et 24 caractères
liste_soiree_invit	Liste des soirées auxquelles l'utilisateur est invité	VARCHAR(50)	Pas de contrainte
liste_soiree_	Liste des soirées	VARCHAR**(**50)	Deux soirées dans

particip	auxquelles l'utilisateur participe		la liste ne peuvent avoir la même date
liste_amis	Liste des amis de l'utilisateur	VARCHAR(50)	Pas de contraintes

Organisateur

Attribut	Définition de l'attribut	Type	Contraintes sur l'attribut
id_organisateur	Identifiant de l'utilisateur	INT	Unique et not null

Profil

Attribut	Définition de l'attribut	Type	Contraintes sur l'attribut
id_profil	Identificateur du profil	INT	Unique et not null
nom	Nom de famille de l'utilisateur	VARCHAR(30)	Compris entre 2 et 24 caractères
prenom	Prénom de l'utilisateur	VARCHAR(30)	Compris entre 2 et 24 caractères
adresse	Adresse du domicile de l'utilisateur	VARCHAR(150)	Doit être une adresse dans Paris
mdp	Mot de passe associé à l'e-mail	VARCHAR(30)	Compris entre 8 et 24 caractères
liste_preferences	Liste des préférences de l'utilisateur	VARCHAR(50)	L'utilisateur doit au moins avoir une préférence
email	Adresse e-mail identifiant l'utilisateur	VARCHAR(30)	Doit être de format e-mail et unique.

Lieu

Attribut	Définition de l'attribut	Type	Contraintes sur l'attribut
id_lieu	Identificateur du lieu	INT	Unique et not null
type	Type (bar | restaurant | boîte)	VARCHAR(30)	Aucune
nom	Nom du lieu	VARCHAR(30)	Compris entre 2 et 24 caractères
adresse	Adresse du lieu	VARCHAR(30)	Doit être du format adresse
longitude	Coordonnées longitudinales du lieu	DECIMAL(2,10)	Compris entre -90 et 90
latitude	Coordonnées latitudinales du lieu	DECIMAL(2,10)	Compris entre -90 et 90
description	Description du lieu	VARCHAR(100)	150 caractères maximum
note	Note attribuée au lieu par les utilisateurs Google	DECIMAL(1 , 1)	Compris entre 0 et 5
budget	Budget attribué au lieu par les utilisateurs Yelp	SMALLINT	Compris entre 1 et 4

Soiree

Attribut	Définition de l'attribut	Type	Contraintes sur l'attribut
id_soiree	Identifiant de la soirée	INT	Unique et not null
nom	Nom attribué à la soirée	VARCHAR(30)	Compris entre 3 et 24 caractères
liste_lieux	Liste des lieux de la soirée	VARCHAR(100)	La taille de la liste doit être égale à 3

liste_participants	Liste des participants de la soirée	VARCHAR(100)	La taille de la liste doit être comprise entre 2 et 40
heure_debut	Date de début de la soirée	TIMESTAMP	Doit être comprise entre 18h et 20h
heure_fin	Date de fin de la soirée	TIMESTAMP	Doit être inférieure ou égale à la l'heure de fermeture de la boîte
rdv_long	Coordonnée longitudinale du point de rendez-vous	DECIMAL(2 , 10)	Doit être comprise entre -90 et 90
rdv_lat	Coordonnée latitudinale du point de rendez-vous	DECIMAL(2 , 10)	Doit être comprise entre -90 et 90
theme	Theme du bar de la soirée	VARCHAR(30)	Peut ne pas être renseigné
nb_participants	Nombre de participants à la soirée	INT	Doit être strictement supérieur à 1

Itinéraire

Attribut	**Définition de l'attribut**	**Type**	**Contraintes sur l'attribut**
id_itinéraire	Identificateur de l'itinéraire	INT	Unique et not null
duree	Durée du trajet	INT	Doit être inférieure à 20 minutes
type_transport	Type de transport (Transport en commun | Vélib | voiture)	VARCHAR(50)	Aucune
liste_etapes	Liste des étapes de	VARCHAR(50)	Aucune

	l'itinéraire		

7. Outils de développement

7.1. Logiciels utilisés

- **UMLet :**

La modélisation du projet d'application, mentionné dans le cahier des charges, a été entièrement réalisée à l'aide du logiciel libre UMLet.
Lien de téléchargement : ***http://www.umlet.com/changes.htm***

- **Android Studio:**

Android studio est une distribution spécifique de l'environnement **Eclipse**, contenant notamment le SDK d'Android. L'application sera donc codée principalement sur cette distribution grâce au langage **Java.** Des notions de **JavaScript** seront requises pour la partie dynamique du projet, ainsi que des notions de **HTML** et de **CSS** pour la partie mise en page.
Android Studio propose des outils pour gérer le développement des applications mobile et nous permettra de suivre le développement de l'application de façon **visuelle.**
Lien de téléchargement : ***https://developer.android.com/studio/index.html***
Guide d'utilisation (en) : ***https://developer.android.com/studio/intro/index.html***

- **Gimp**

Gimp est un logiciel d'édition d'images, qui nous permettra de développer la partie **graphique** du projet tout en tenant compte de la charte graphique énoncée dans le cahier des charges.
Lien de téléchargement : ***https://www.gimp.org/***

- **SQLite**

SQLite est le système de gestion des **bases de données** que nous utiliserons lors du développement de ce projet. SQLite est un moteur de bases de données relationnelles qui peut être utilisé indépendamment de l'interface client serveur et est directement intégré aux programmes. Il permet ainsi de stocker la base de donnée dans un fichier externe. SQLite peut être couplé à Java et nous coderons la gestion des bases de données grâce au langage **SQL.**
Lien de téléchargement : ***https://bitbucket.org/xerial/sqlite-jdbc/downloads/*** ***(prendre la dernière version du fichier*** ***sqlite-jdbc-(VERSION).jar***
Utilisation : ajouter le fichier .jar dans le Build path de l'environnement Android Studio et ajouter la ligne de code : `import java.sql.*;`

7.2. API utilisés

Les API Google que nous allons utiliser sont les suivants :

- Google Maps Android API

Cet API permettra d'afficher une carte Google Maps sur l'onglet d'accueil de l'application et peut être utilisé dans Java en important les bibliothèques suivantes :

```
import com.google.android.gms.maps.*;
import com.google.android.gms.maps.model.*;
import android.app.Activity;
import android.os.Bundle;
```

- Google Places API for Android

Cet API nous permettra d'accéder à la base de données de Google contenant tous les Lieux que nous pourrons utiliser afin d'en proposer des triplets et est utilisable sur Android.

- Google Maps Directions API

Cet API nous permettra de calculer les itinéraires des différents trajets lors d'une soirée.
Le guide d'utilisation de cet API se trouve sur le lien plus haut.

- Google Maps Distance Matrix API

Cet API nous permettra d'effectuer les différents calculs de distance sur une carte afin d'implémenter un programme correspondant aux exigences formelles requises dans la cahier des charges ainsi que dans le cahier de conception.

- Yelp API

Cet API nous permettra de trier les endroits de rencontre des utilisateurs par budget et donc de leur fournir une soirée correspondant à leurs moyens. Toute la documentation de l'API se trouve au bout du lien ci-dessus.

- JCDecaux API

Cet API nous permettra, lors du choix du trajet à vélo, de signaler les bornes Vélib' les plus proches. Toute la documentation de l'API est au bout du lien ci-dessus.

Toutes les clefs des API seront communiquées par mail aux développeurs.

8. Sources

Construction du cahier de conception:

- Divers cahier de conception fournis par les proches : exemple d'un projet d'application CGI et exemple de cahier de conception de Publicis Technology ainsi qu'un projet universitaire de Sonia Benbakli.