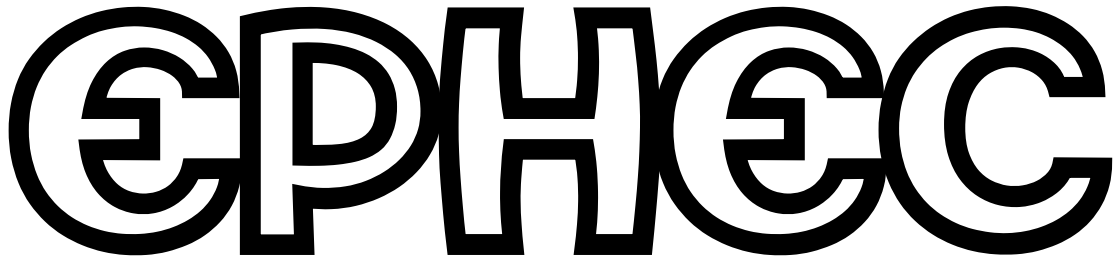


Ecole Pratique des Hautes Etudes Commerciales



Institut d'Enseignement Supérieur Economique de Type Court

**NEARYSPORT: UNE APPLICATION MOBILE POUR
ORGANISER DES ÉVÉNEMENTS SPORTIFS ENTRE
AMIS**

Lionel QUIRYNEN

**Rapporteur(s)
J. HECQUET**

Travail de Fin d'Études présenté
en vue de l'obtention du diplôme de
BACHELIER EN INFORMATIQUE DE GESTION

Année académique 2018-2019

Ce travail de fin d'études n'aurait pas été possible sans les conseils avisés d'un grand nombre de personnes.

Je tiens à remercier mon relecteur Jean-Paul Hecquet pour ses différents conseils quant à la rédaction de ce TFE et ses indications d'amélioration un peu plus « fonctionnelles » qui m'ont grandement aidé pour le développement de mon application.

Un tout grand merci à Antoine Corbisier pour ses réponses toujours très rapides et complètes, il a également su se montrer très disponible pour la consultation des TFE.

Je tiens également à remercier mes parents et plus particulièrement mon père qui m'a beaucoup conseillé et qui a gentiment accepté de relire ce TFE, ce document n'aurait pas été pareil sans lui.

Enfin j'adresse mes remerciements à ma famille et à Catharina Van Dun pour leur soutien au cours de ces derniers mois et durant la rédaction de ce TFE.

Chapitre I : Introduction	1
1.1 Contexte général du projet	1
1.1.1 Sport	1
1.1.2 Marché mobile	1
1.2 Présentation générale du projet	1
Chapitre II : Analyse fonctionnelle	3
2.1 Use Cases	3
2.1.1 Brève description des fonctionnalités	3
2.1.2 Admin vs Participant d'un évènement	4
2.2 Diagramme de classes conceptuel	5
2.3 Diagramme de navigation entre écrans	6
2.4 Règles fonctionnelles	7
2.4.1 Connexion	7
2.4.2 Inscription utilisateur	7
2.4.3 Oubli de mot de passe	7
2.4.4 Accès Evènements	7
2.4.5 Création évènement	8
2.4.6 Profil utilisateur	8
2.4.7 Messagerie	8
2.4.8 Invitations	8
Chapitre III : Architecture générale	9
3.1 Introduction	9
3.1.1 Architecture technique du système : multi-couches	9
3.1.2 Technologies employées	9
3.2 Analyse des scénarios de l'application	10
3.2.1 Inscription Utilisateur	11
3.2.2 Connexion Utilisateur	12
3.2.3 Mot de passe oublié	13
3.2.4 Inscription évènement	15
3.2.5 Accès aux évènements	17
Chapitre IV : Base de données	18
4.1 Schéma Base de données	18
4.2 Tables : Contraintes et clefs	19

4.2.1 Users	19
4.2.2 Profil	19
4.2.3 ImageProfil	19
4.2.4 Events	20
4.2.5 Participations	20
4.2.6 GroupEventMessage	20
4.2.7 Messages	21
4.2.8 Invitations	21
4.2.9 ForgotPassword	21
4.3 Stored Procedure	22
4.3.1 Choix SP	22
4.3.2 Liste SP	22
4.4 Triggers	25
4.4.1 On cascade Delete	25
4.4.2 Liste Triggers	26
Chapitre V : Web API	27
5.1 Introduction	27
5.1.1 Le choix REST	27
5.1.2 Asp.net, IIS	28
5.1.3 Entity Framework	28
5.2 Arborescence API	29
5.3 Sécurité et Autorisation	32
5.3.1 Hashage	32
5.3.2 Json Web Token	35
5.3.3 Attribut customisé : Authorize filter	37
5.4 Règles Business Côté Serveur	38
5.4.1 EventController	38
5.4.2 ForgotPasswordController	38
5.4.3 LoginController	38
5.4.4 ParticipationController	38
5.4.5 ProfilController	39
Chapitre VI : Interface Utilisateur Android	40
6.1 Le choix Xamarin (native)	40

6.1.1 Introduction	40
6.1.2 Xamarin native vs Xamarin Forms	40
6.2 Vues Android	42
6.2.1 Introduction	42
6.2.2 Vue connexion	43
6.2.3 Vue inscription	45
6.2.4 Vue mot de passe oublié (envoyer e-mail, confirmer code, mise à jour mot de passe)	46
6.2.5 Vue fil d'actualité évènements	47
6.2.6 Vue mes évènements	49
6.2.7 La vue création évènement	50
6.2.8 La vue Mon profil	53
6.2.9 La vue détails évènement	54
6.2.10 La vue Participants	56
6.2.11 La vue Messagerie évènement	58
6.2.12 La vue Invitations	60
6.3 Services Android	62
6.3.1 Qu'est-ce qu'un service Android ?	62
6.3.2 Implémentation d'un service	62
6.4 Permissions	64
6.5 Image de profil	65
Chapitre VII : Futures possibilités de développement	66
7.1 Développement IOS	66
7.2 Développement Web	66
7.3 Multi-Langues	66
7.4 Partage d'évènements	66
7.5 Notions « d'amitié »	67
7.6 Messagerie individuelle	67
7.7 OAuth2	67
Chapitre VIII : Conclusion	68
Bibliographie	69
Annexes	71
Annexe I : Documentation Swagger	71

Annexe II : Diagramme de séquences : Image de profil	73
Annexe III : Invitation	74

Chapitre I : Introduction

1.1 Contexte général du projet

1.1.1 Sport

Les bénéfices du sport ne sont plus à prouver : pratiquer une activité physique régulière permet en effet de protéger contre la survenue des maladies cardiovasculaires, protéger contre certains cancers, faciliter la stabilité de la pression artérielle, réduire les risques d'obésité, ... Et pourtant, il n'est pas toujours facile de se bouger après le travail ou une longue journée de cours, de pratiquer une activité physique régulière. En Belgique, une enquête menée par *Jean Tafforeau de l'Institut Scientifique de Santé Publique*, démontre qu'en moyenne 26% des belges ne pratiquent pas d'activités sportives et sont donc à risque par manque d'activité physique. 58% ne pratiquent qu'une activité physique légère (moins de 4h/semaine).

Pour tenter de motiver les sédentaires à pratiquer une activité physique, de nombreux cours collectifs ont vu le jour ces dernières années, et certains ont rencontré un succès retentissant. On peut notamment nommer le projet « Je cours pour ma forme » qui, en Belgique, a réussi -fin 2016- à rassembler plus de 25 000 inscrits. Mais malgré le succès de ces cours collectifs, certains n'ont pas les moyens financiers pour y participer.

L'application que je vais développer tentera d'apporter un semblant de solution à ce problème, en essayant de réunir des amis/inconnus autour d'une session sportive, payante ou non.

1.1.2 Marché mobile

Fin 2018, l'on comptait 4,9 milliards d'utilisateurs mobiles partout dans le monde. Et parmi ces nombreux utilisateurs, la GSMA (Gsm Association) affirme que 54% de ces utilisateurs possèdent un smartphone. Sur ces 54% possédant un smartphone, Android est le grand gagnant, avec une part de marché oscillant entre 70 et 80% sur ces deux dernières années.

Avec une moyenne de 5h passées/jour sur son smartphone en 2017, les utilisateurs sont hyper connectés. Lorsqu'une nouvelle application voit le jour, le public potentiel est énorme.

1.2 Présentation générale du projet

Il n'est pas toujours facile de trouver des amis pour nous accompagner à nos sessions sportives, les horaires de chacun étant différents. Il n'est pourtant pas toujours nécessaire de connaître la personne pour passer un bon moment autour

d'un match de tennis/minifoot/basket,... Et pratiquer une activité physique avec des inconnus peut déboucher sur de nouvelles amitiés.

J'ai donc eu pour idée de créer une application android permettant à un utilisateur connecté de voir tous les évènements sportifs organisés par d'autres utilisateurs de l'application autour de sa position. Il aura la possibilité de s'inscrire à ces évènements ainsi que d'en créer.

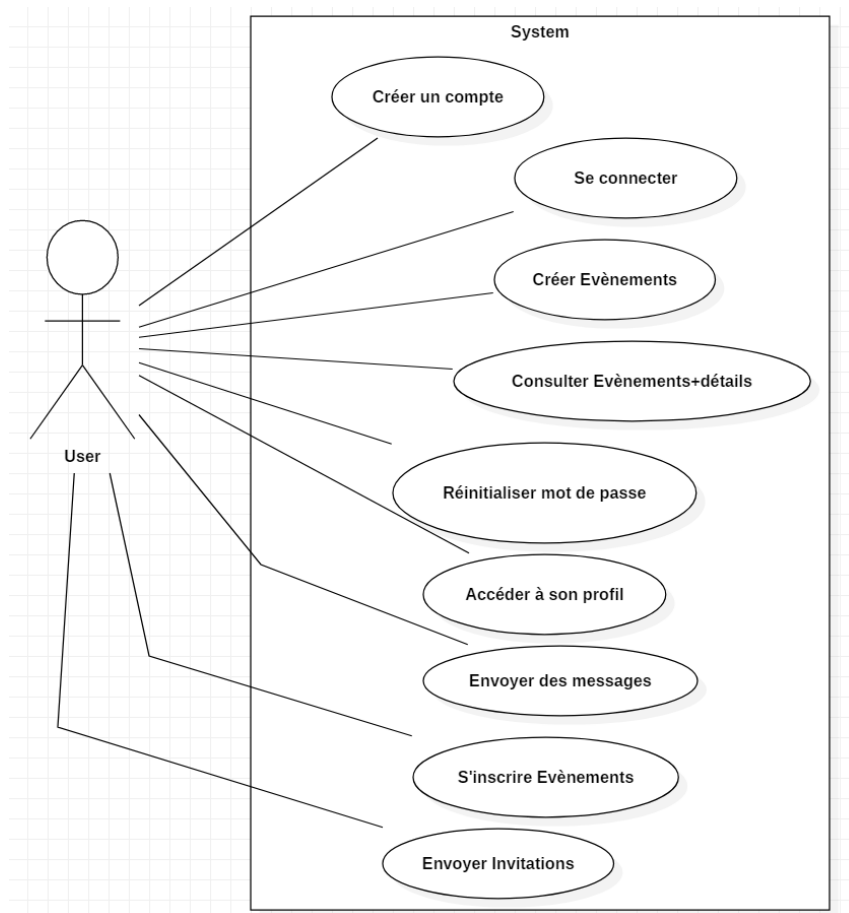
Chapitre II : Analyse fonctionnelle

2.1 Use Cases

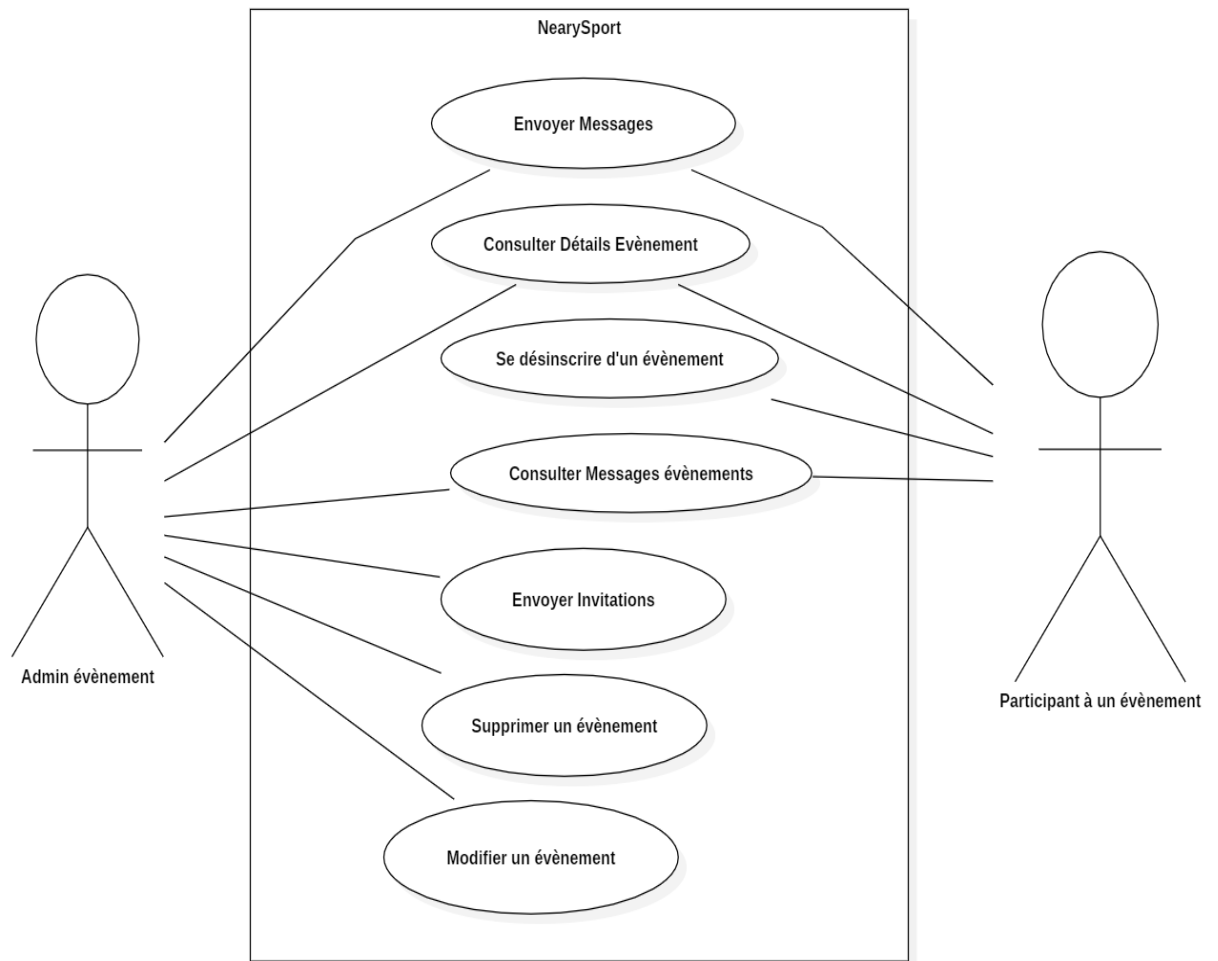
2.1.1 Brève description des fonctionnalités

Dans l'application, un utilisateur aura de nombreuses possibilités, en voici une liste non exhaustive, un diagramme de Use Case les reprend également ci-dessous :

- Créer un compte
- Se connecter à l'application
- Créer un évènement
- Accéder au fil d'actualité avec tous les évènements
- Accéder aux détails d'un évènement
- Accéder à ses évènements
- Réinitialiser un mot de passe oublié
- Accéder à son profil
- Envoyer des messages dans un évènement où il participe
- S'inscrire à un évènement
- Envoyer des invitations à un évènement qu'il administre



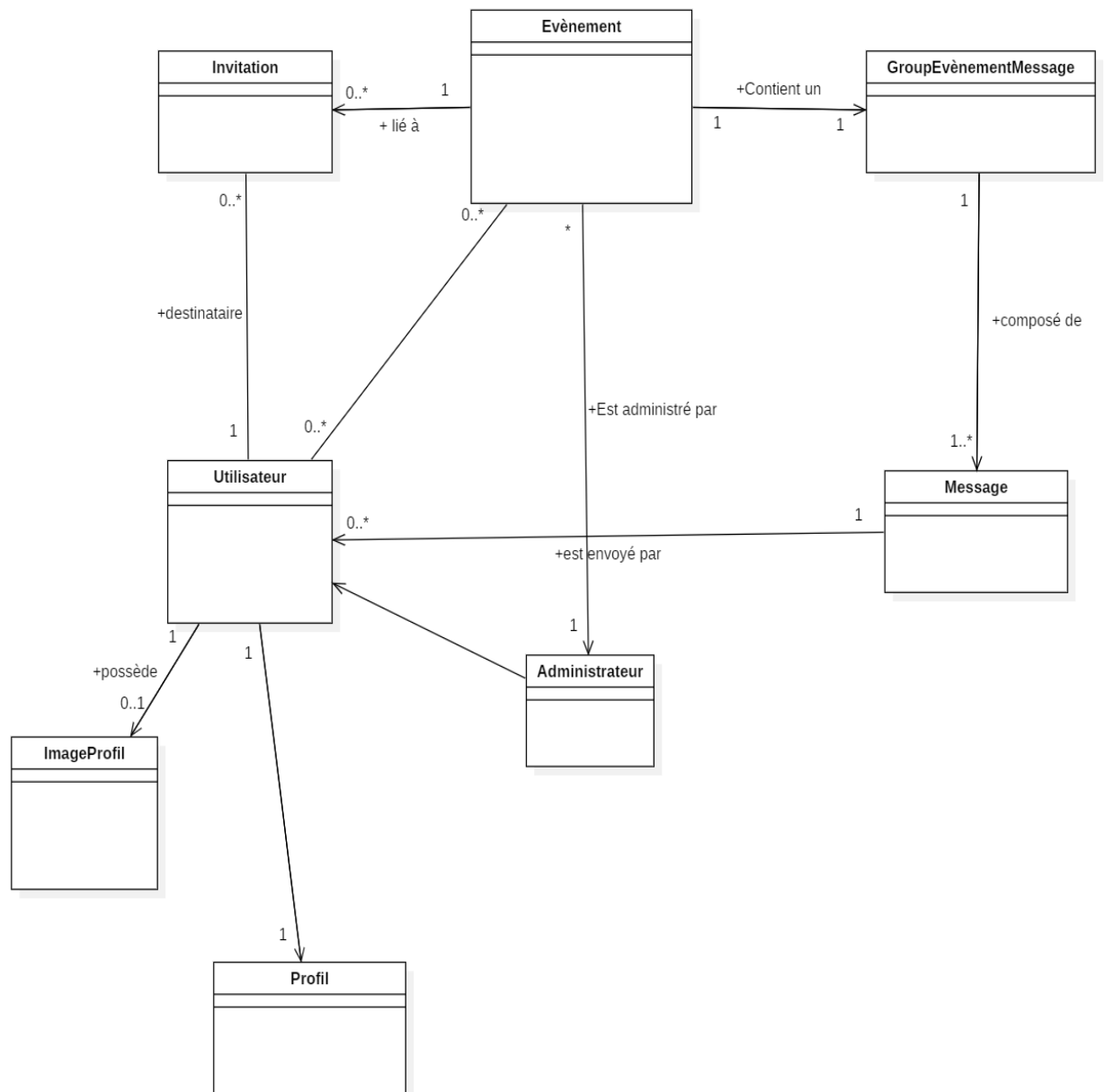
2.1.2 Admin vs Participant d'un évènement



Comme on peut le voir sur le diagramme de Use Case ci-dessus, un administrateur n'a pas la possibilité de se désinscrire de son évènement. En effet si celui-désire quitter son évènement, il doit le supprimer, toutes les informations liées à cet évènement seront donc également supprimées (Invitations, messages et participations) via des triggers ou les clefs étrangères.

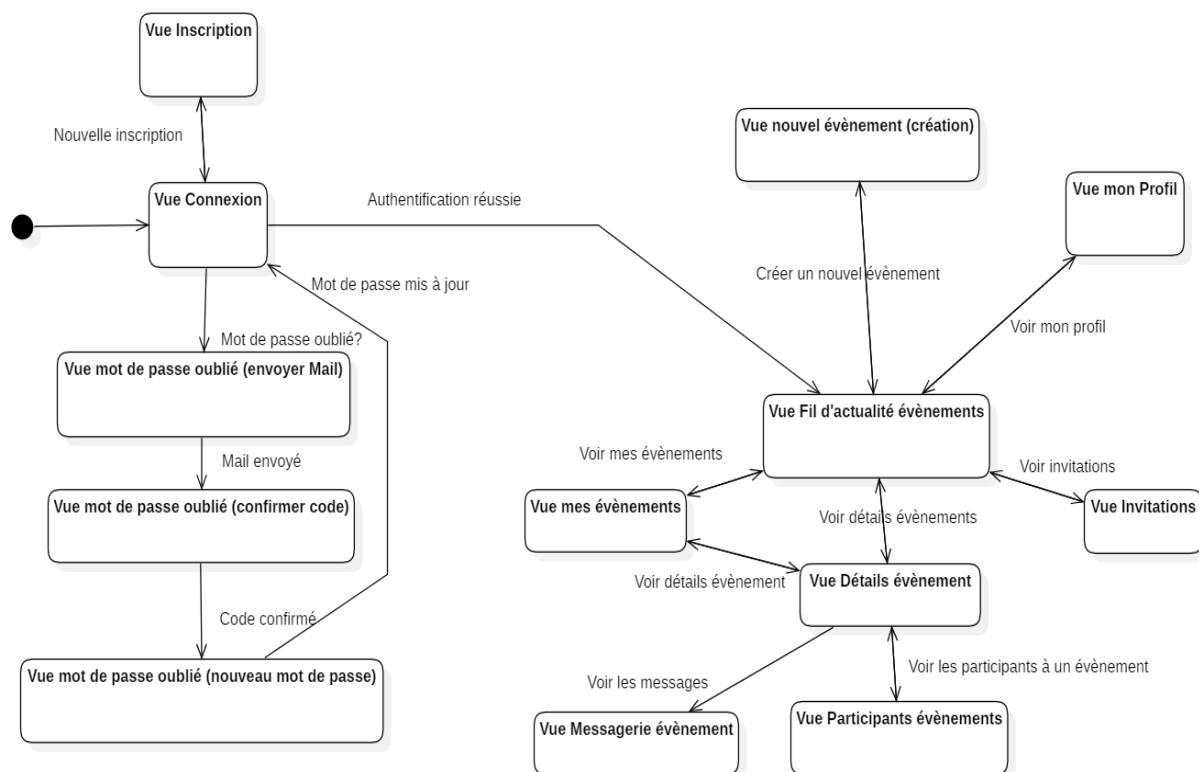
2.2 Diagramme de classes conceptuel

Ci-dessous, un diagramme conceptuel de classes utilisées dans ce projet.



2.3 Diagramme de navigation entre écrans

Pour permettre d'y voir plus clair, un diagramme d'états a été réalisé pour définir l'arborescence de l'application et ses différents écrans.



2.4 Règles fonctionnelles

Voici une liste des règles fonctionnelles qui seront implémentées dans l'application :

2.4.1 Connexion

- Pour pouvoir se connecter, il faut posséder une connexion internet valide.
- Les champs username et password doivent être remplis.
- Pour pouvoir se connecter, il faut posséder un compte.
- Le mot de passe doit faire entre 8 et 15 caractères
- Si un mot de passe ou un nom d'utilisateur erroné est rentré, l'utilisateur recevra un message d'erreur approprié.

2.4.2 Inscription utilisateur

- Le mot de passe doit avoir un format de minimum 8 caractères et 15 maximum.
- Pour pouvoir s'enregistrer, tous les champs doivent être remplis.
- L'adresse Mail doit avoir un format valide
- L'adresse ne doit pas être supérieure à 50 caractères.
- Les deux champs « Password » et « Confirm Password » doivent être identiques.
- Les champs Username et E-mail doivent être uniques : s'ils sont déjà utilisés, un message d'erreur apparaîtra.

2.4.3 Oubli de mot de passe

- Pour recevoir un code, il faut avoir un compte lié avec l'adresse e-mail rentrée.
- Le code reçu sur l'adresse e-mail n'est plus valide au bout de 5 minutes.

2.4.4 Accès Evènements

- Pour pouvoir accéder au fil d'actualité, l'utilisateur doit être connecté.
- L'utilisateur ne peut pas s'inscrire deux fois au même évènement.
- Les évènements dont la date est dépassée n'apparaîtront pas dans le fil d'actualité.
- Les évènements affichés dans le fil d'actualité sont triés sur la date et le sport de l'évènement.
- Si l'utilisateur décide de quitter un évènement alors que la limite de désinscription avait été atteinte, il recevra un avertissement, et s'il décide de quitter, sa fiabilité¹ s'en verra impactée.
- L'utilisateur ne peut plus accéder à des évènements expirés depuis plus de trois jours.
- L'utilisateur ne pourra pas modifier l'évènement s'il n'est pas administrateur de celui-ci.

¹ Règle business détaillée en 6.2.7

- L'utilisateur n'a pas besoin d'être inscrit pour accéder aux détails de l'évènement.
- L'utilisateur n'a pas besoin d'être inscrit pour accéder aux participants de l'évènement.
- Un évènement privé ne peut être consulté que si l'utilisateur fait partie des invités de l'évènement.
- L'utilisateur doit être inscrit pour accéder à la messagerie de l'évènement.

2.4.5 Création évènement

- Une limite de résignation permet à l'administrateur de définir une limite avant laquelle un utilisateur peut quitter l'évènement sans qu'il y ait un impact sur la fiabilité de ce dernier.
- L'adresse doit être composée d'une ville, rue, code postal et peut être plus précise via la description de l'évènement.
- L'évènement doit avoir une heure de fin postérieure à l'heure de début.
- On ne peut pas créer un évènement dans le passé.
- L'évènement ne peut pas avoir un cout négatif.
- L'évènement ne peut pas avoir un nombre de participants négatif.
- La limite de désinscription ne peut pas être supérieure à la date de création de l'évènement.

2.4.6 Profil utilisateur

- Le nom d'utilisateur et la fiabilité ne peuvent pas être modifiés.

2.4.7 Messagerie

- L'utilisateur doit être connecté et être un participant de l'évènement pour pouvoir envoyer un message et avoir accès à la messagerie.
- Si l'utilisateur quitte l'évènement, tous ses messages sont supprimés.
- Les messages sont triés dans l'ordre d'envoi.

2.4.8 Invitations

- Pour envoyer une invitation à un évènement, l'utilisateur doit être administrateur de cet évènement.
- Pour envoyer une invitation à un utilisateur, ce dernier ne peut pas déjà être inscrit à cet évènement.
- Une fois une invitation acceptée, la participation n'est actée que s'il reste de la place à cet évènement.
- Pour supprimer une invitation, il faut être l'administrateur de l'évènement.

Chapitre III : Architecture générale

3.1 Introduction

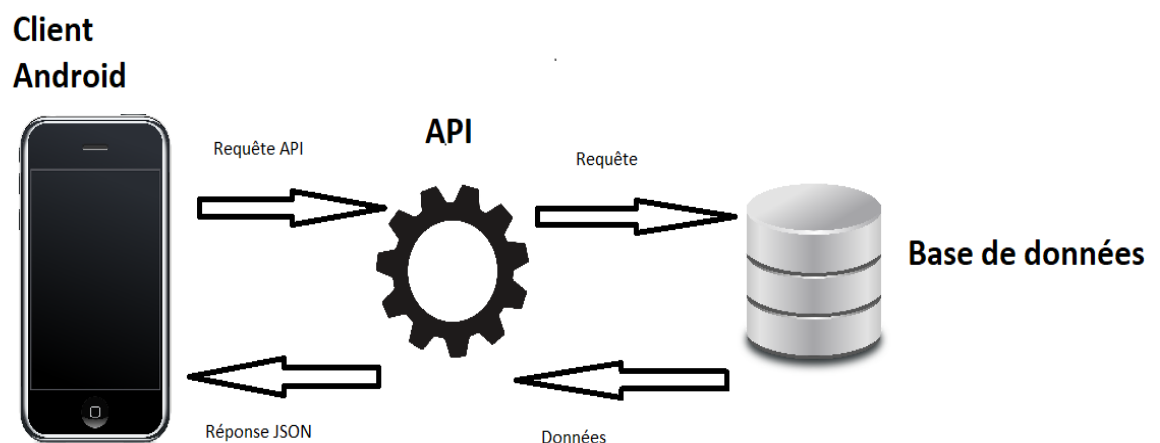
3.1.1 Architecture technique du système : multi-couches

L'application est découpée en 3 couches :

- Interface client Android
- Web Api
- Base de données

L'application ne communiquera jamais directement avec la base de données.

L'application passera toujours, via une Web API JSON, par une couche Server, qui implémentera ces APIs en contactant la base de données (voir schéma ci-dessous).



3.1.2 Technologies employées

Différentes technologies ont été utilisées pour le projet :

1. **Android** : "Pour le client Android, le framework Xamarin Android a été choisi, ce choix sera justifié plus tard.
2. **Web API** : Pour la partie Web API, le framework ASP.Net a été choisi et est complété par l'ORM Entity Framework pour communiquer avec la base de données.
3. **Base de données** : Pour le système de gestion de base des données, SQL Server a été choisi.
4. **Gestionnaire de versions** : Un gestionnaire de versions est un logiciel permettant de suivre les différentes versions de fichiers (généralement projets informatiques). Il permet de voir ce qui a été modifié et de savoir qui a effectué la modification. Etant donné que pour ce projet je travaillais seul, la dernière utilisation n'était pas nécessaire. Cela m'a néanmoins permis d'avoir une sauvegarde fiable de mon projet et de voir quels étaient les

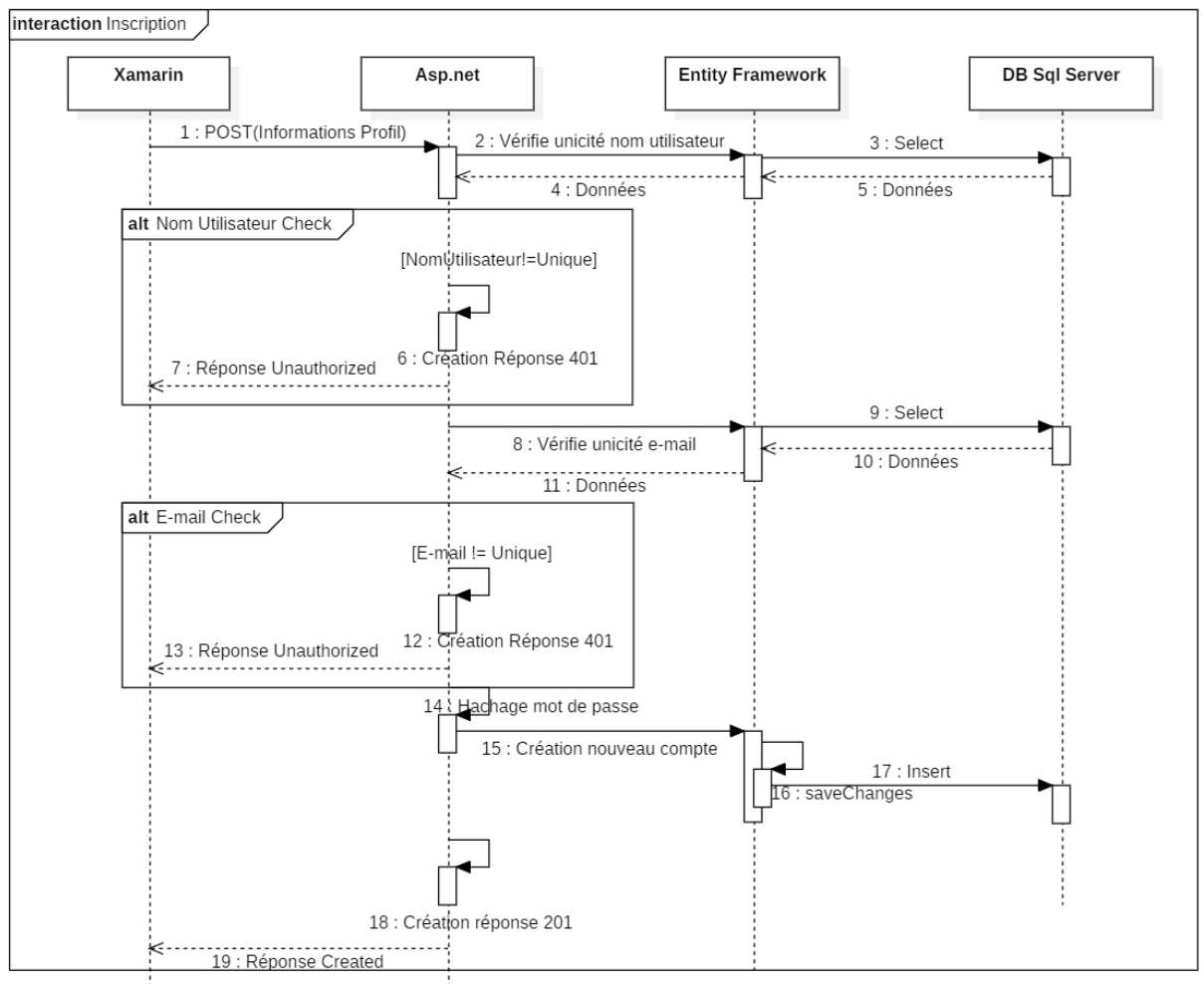
derniers changements que j'avais effectué. Pour la rédaction de ce TFE, mon choix s'est porté sur GitLab.

3.2 Analyse des scénarios de l'application

Pour permettre d'y voir plus clair au niveau des interactions entre les différentes couches, des diagrammes de séquence ont été réalisés. Les parties jugées les plus intéressantes sont décrites ci-dessous via des diagrammes de séquence réalisés à l'aide du logiciel StarUML, le reste des diagrammes se trouve dans les annexes.

Un diagramme de Use Cases a également été réalisé pour montrer quelles sont les différences et possibilité d'interactions en tant qu'administrateur d'un évènement et en tant que participant (voir 2.1.2).

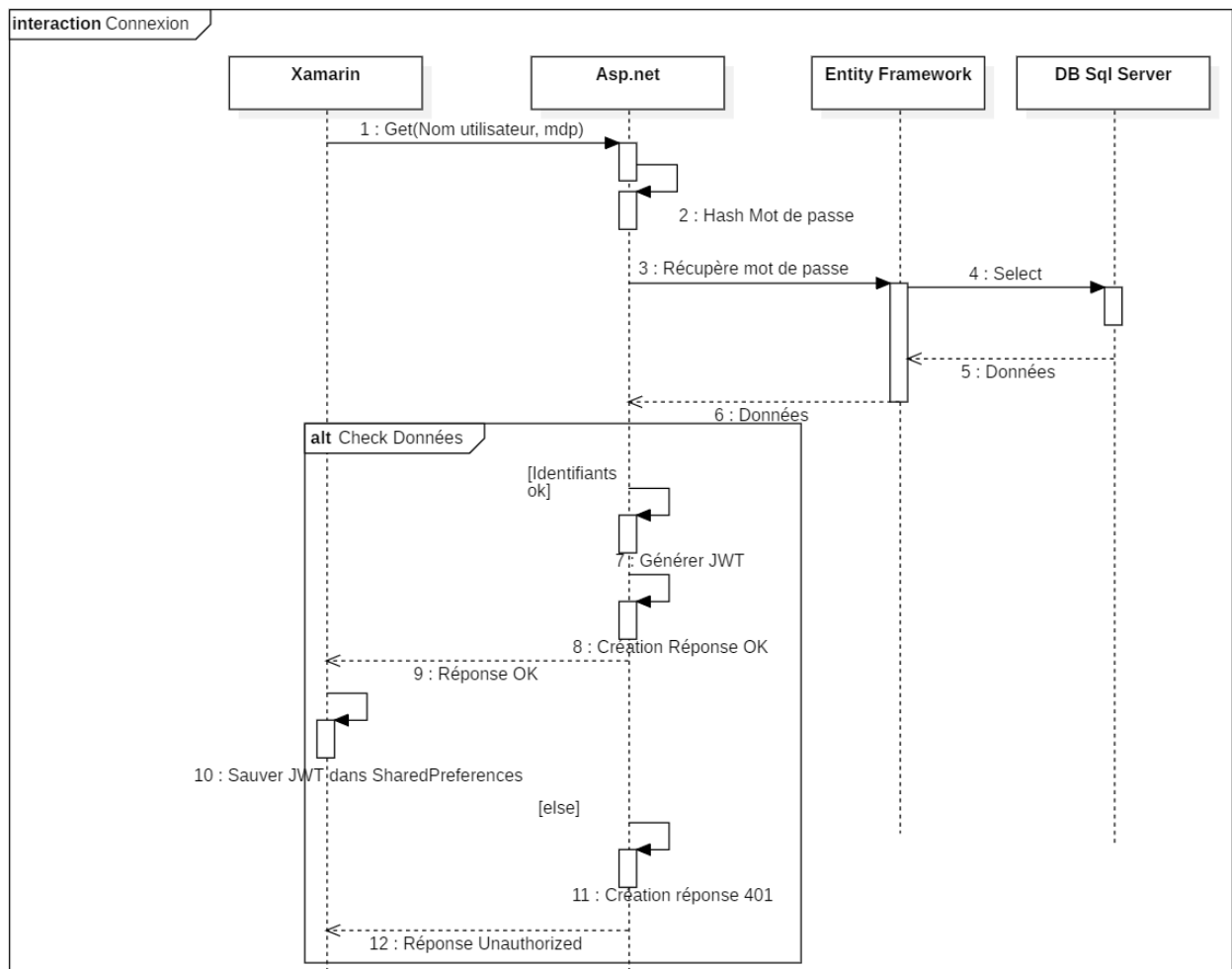
3.2.1 Inscription Utilisateur



Le processus ci-dessus représente l'inscription d'un utilisateur :

- L'utilisateur envoie ses informations
- Le serveur vérifie via EF que l'e-mail et le nom d'utilisateur sont bien disponibles.
- Si les identifiants ne sont pas disponibles, une réponse 401 est renvoyée avec un message approprié.
- Si les identifiants sont disponibles, le mot de passe est haché.
- Une fois le mot de passe haché, le compte est inséré dans la base de données.
- Une réponse 201 est renvoyée à l'utilisateur.

3.2.2 Connexion Utilisateur

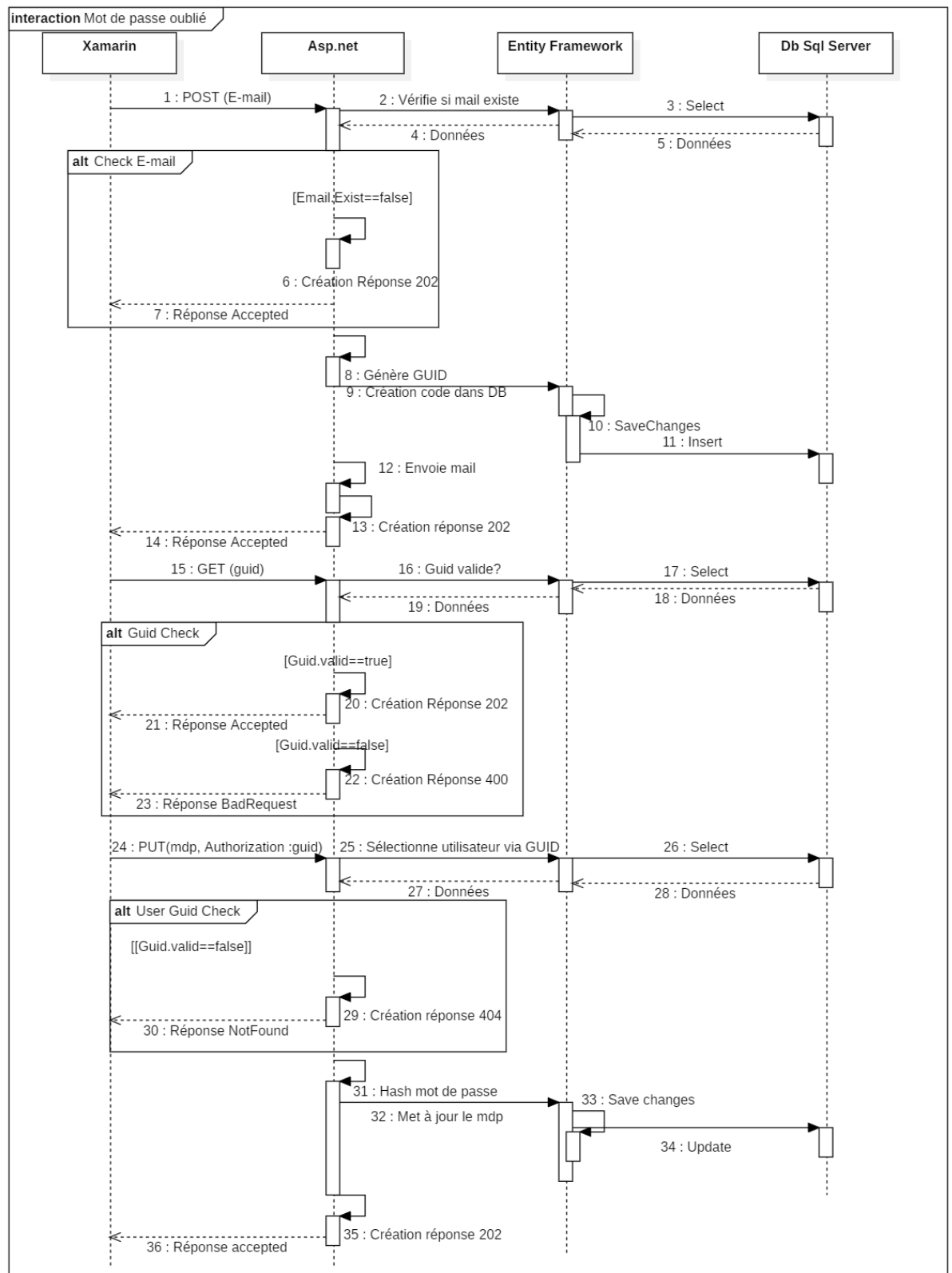


Légende : L'étape 2) représente le hash du mot de passe, ce procédé est expliqué plus en détails dans la Partie 5.3.1. L'étape 7) concernant la création d'un Json Web Token est également expliquée plus en détails en 5.3.2.

Le processus ci-dessus représente la connexion d'un utilisateur :

- L'utilisateur se connecte en entrant son mot de passe et nom d'utilisateur
- Le mot de passe est haché
- Le mot de passe correspondant au nom d'utilisateur entré est récupéré dans la base de données en passant par EF
- Si les identifiants sont corrects, un JWT (correspondant à un « ID de session ») est généré et une réponse OK est renvoyée à l'utilisateur avec le JWT.
- Si les identifiants sont incorrects, une réponse 401 unauthorized est renvoyée à l'utilisateur avec un message approprié.

3.2.3 Mot de passe oublié



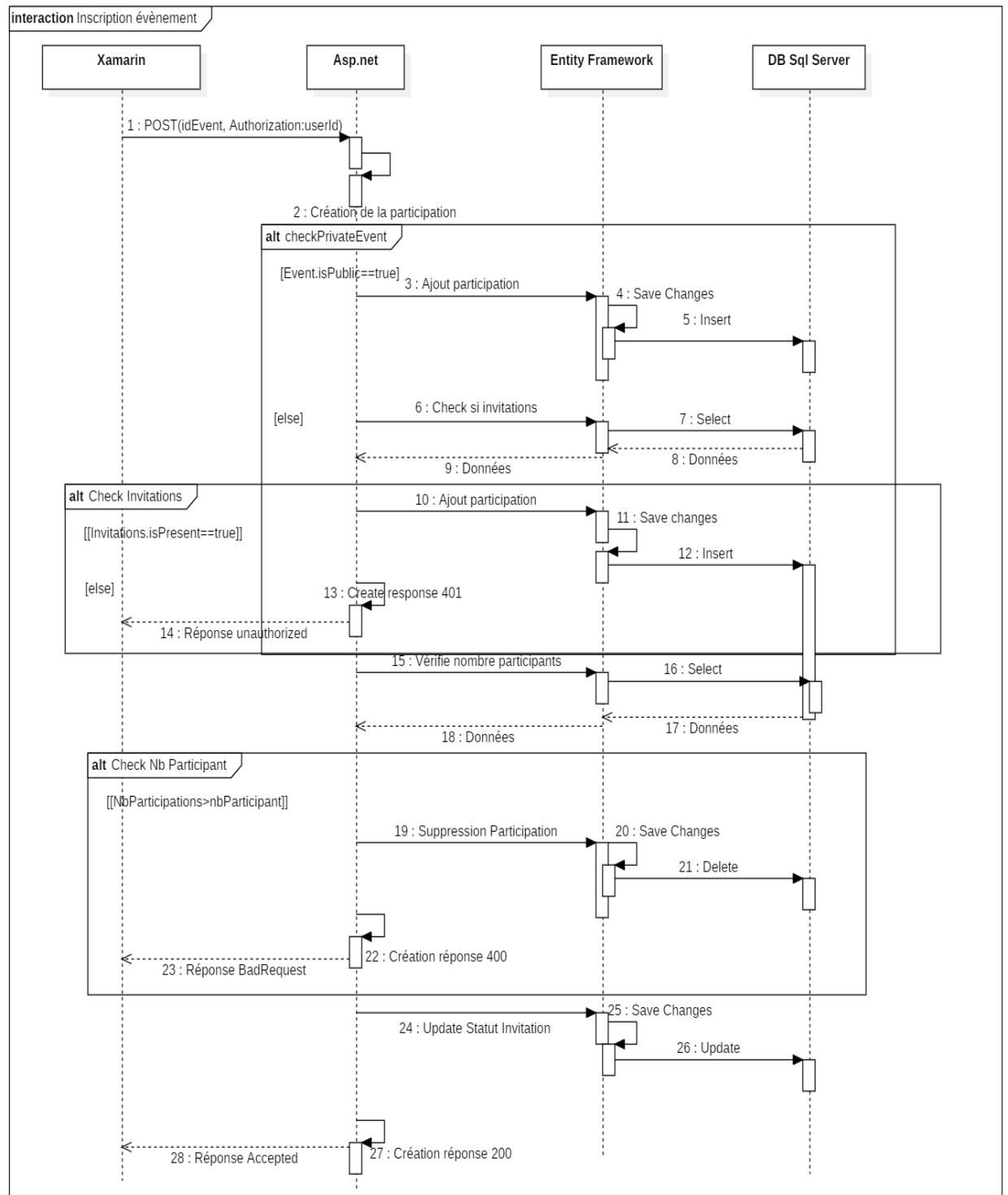
Le processus de la page précédente représente l'oubli de mot de passe et sa « récupération » :

- L'utilisateur rentre son e-mail lié au compte avec lequel il a oublié son mot de passe.
- Le système vérifie si l'email rentré est bien lié à un compte
- Si l'e-mail n'existe pas, une réponse 202 Accepted est renvoyée².
- Si l'e-mail existe bien, un GUID³ est généré et envoyé à l'e-mail rentré.
- Une réponse 202 est ensuite envoyée à l'utilisateur.
- L'utilisateur rentre le code reçu par e-mail et confirme le code.
- Le code est reçu par le système et est vérifié comme valide ou non.
- Si celui-ci est valide, une réponse 202 est renvoyée à l'utilisateur sinon une réponse 400 est renvoyée.
- Si l'utilisateur a obtenu une réponse 202, il a la possibilité de rentrer un nouveau mot de passe, ce dernier est envoyé dans la requête avec également le code précédemment confirmé dans le header Authorization.
- Le code est une nouvelle fois vérifié, et vérifie de quel utilisateur vient le GUID.
- Si le GUID est confirmé, le mot de passe est haché et ensuite mis à jour, une réponse 202 est renvoyée.
- Si le GUID est refusé, une réponse 404 est renvoyée à l'utilisateur.

² Une réponse 202 est renvoyée même si l'utilisateur a entré une mauvaise adresse e-mail, question de sécurité.

³ GUID est un identificateur global unique.

3.2.4 Inscription évènement



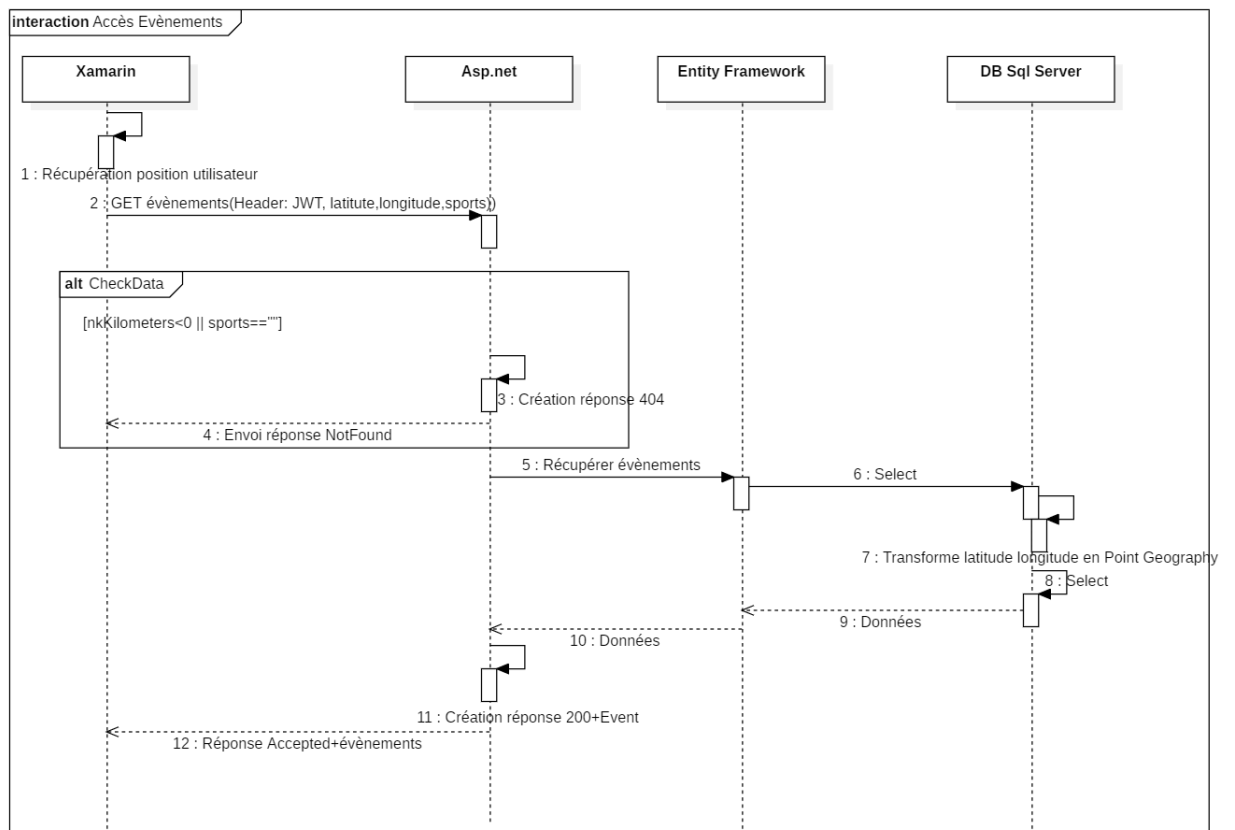
Explications à la page suivante

Le processus ci-dessus décrit l'inscription d'un utilisateur à un évènement :

- L'utilisateur s'inscrit à un évènement, la requête contient le userId dans le header et l'idEvent.
- La participation est construite au niveau API.
- Si l'évènement est public, l'ajout de la participation dans la base de données est fait via EF⁴.
- Si l'évènement est privé, le système vérifie via EF qu'une invitation pour cet évènement se trouve bien dans la base de données.
- Si une invitation est présente, une participation est ajoutée dans la base de données.
- Une fois l'ajout fait dans la base de données, un trigger s'enclenche et change le statut de l'invitation (en attente-> acceptée).
- Si l'invitation n'était pas présente dans la base de données, une réponse 401 unauthorized est renvoyée à l'utilisateur.
- Une vérification du nombre de participants à l'évènement est faite après ajout, toujours via EF.
- Si le nombre de participants dépasse celui permis par l'évènement, la participation est supprimée et une réponse 400 est renvoyée à l'utilisateur avec un message approprié.
- Si la condition précédente n'est pas remplie, une réponse 200 est renvoyée à l'utilisateur.

⁴ EF : Entity Framework

3.2.5 Accès aux évènements



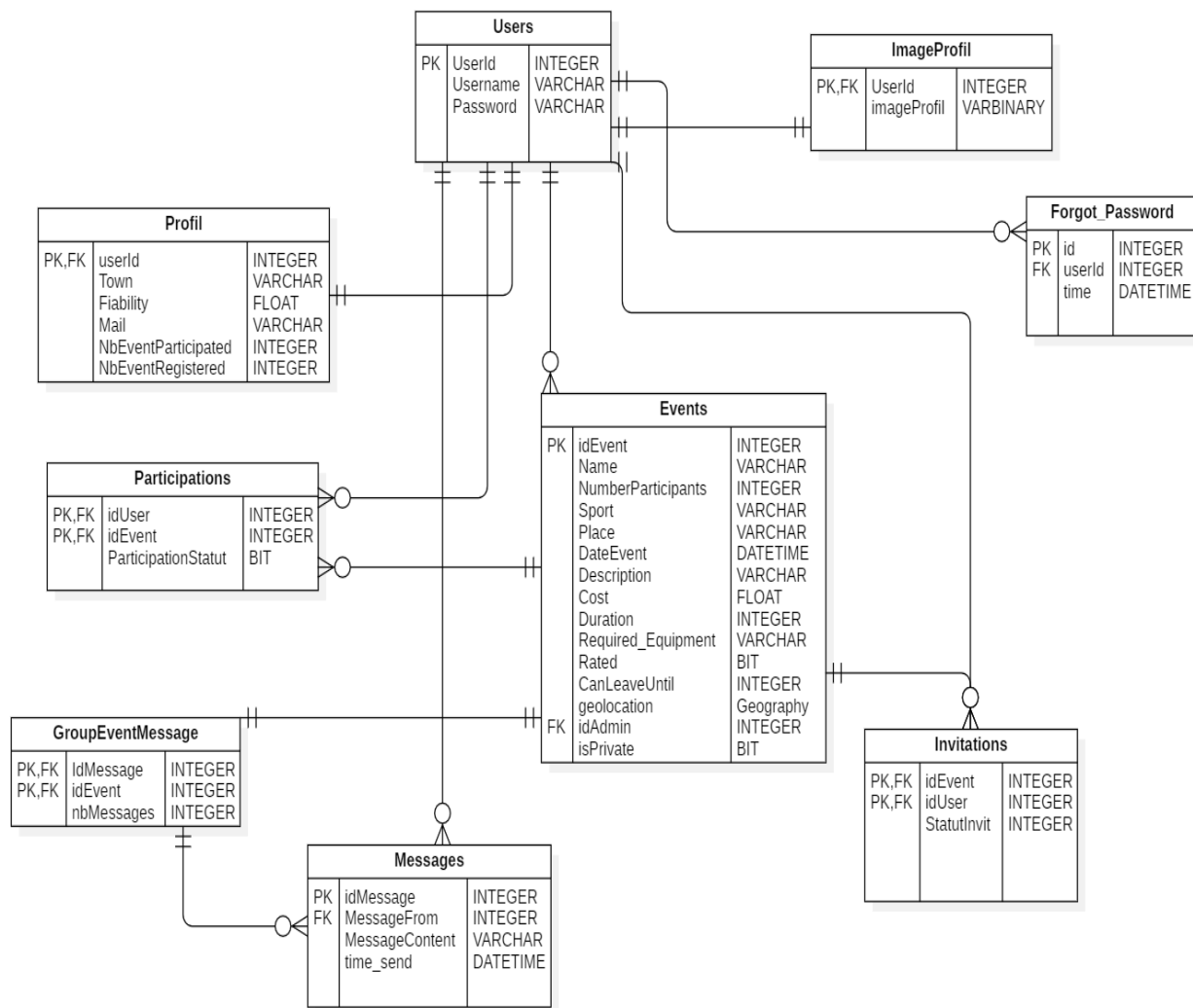
Le processus ci-dessus décrit l'accès aux évènements :

- Dans un premier temps, la position de l'utilisateur est récupérée par l'application Android.
- La latitude, longitude, sports et JWT sont envoyés dans les headers de la requête.
- Si le nombre de kilomètres est inférieur à 0 ou les sports inexistant, une réponse 404 NotFound est renvoyée à l'utilisateur.
- Les évènements sont récupérés via EF.
- Lors de la récupération de ces évènements, les latitudes et longitudes sont transformés en Point géographique, une méthode fournie par Sql Server permet de comparer la distance entre deux points géographique, c'est donc cette méthode qui est utilisée pour filtrer les évènements.
- Les données sont renvoyées à l'utilisateur avec une réponse 202 Accepted et la liste des évènements disponibles autour de lui.

Chapitre IV : Base de données

4.1 Schéma Base de données

Ci-dessous, le schéma de la base de données Sql Server, réalisé à l'aide du logiciel StarUml.



4.2 Tables : Contraintes et clefs

Ci-dessous sont repris, un tableau incluant toutes les clefs et contraintes pour les différentes tables composant la base de données ainsi qu'une brève explication pour le rôle des tables et des champs qui les constituent.

4.2.1 Users

Clef primaire	userId (int autoincrement)
Clef étrangère	/
Contrainte d'intégrité	/
Contrainte d'unicité	Username (varchar)

La table Users représente un utilisateur. Ce dernier est défini par un id, un nom d'utilisateur et un mot de passe. Cette table a été séparée de Profil pour garder l'aspect authentification de cette table, tandis que Profil est plus « fonctionnelle ».

4.2.2 Profil

Clef primaire	userId (int)
Clef étrangère	userId référence userId de Users
Contrainte d'intégrité	NbEventParticipated mis à 0 lors d'un insert NbEventRegistered mis à 0 lors d'un insert
Contrainte d'unicité	Email (varchar)

La table Profil représente un profil d'un utilisateur. Ce profil est défini par un id, une adresse e-mail, une ville, une fiabilité (est-ce que l'utilisateur est venu aux événements auxquels il s'est inscrit ?), un nombre d'événements auxquels l'utilisateur s'est inscrit et un nombre d'événements auxquels l'utilisateur a participé.

4.2.3 ImageProfil

Clef primaire	userId (int)
Clef étrangère	userId référence userId de Users
Contrainte d'intégrité	/
Contrainte d'unicité	/

La table imageProfil représente l'image de profil d'un utilisateur. Elle est définie par un id (correspondant au userId à qui l'image appartient), et une image sauvee sous le format varbinary.

4.2.4 Events

Clef primaire	idEvent (int autoincrement)
Clef étrangère	idAdmin référence userId de Users
Contrainte d'intégrité	NotNegCost : vérifie que le cout ne soit pas négatif NotNegNbParticipants : vérifie que le nombre de participants ne soit pas négatif DF_CanLeaveUntil : contrainte par défaut qui met le champ à 0 DF_Rated : contrainte par défaut qui met le champ à 0
Contrainte d'unicité	/

La table Events représente un évènement. Cet évènement est constitué de nombreux champs : Un id, un nom, une adresse, un nombre de participants, un sport, une date, une description, un coût, une durée, un équipement requis pour y participer, s'il a déjà été noté ou non, une limite pour quitter ce dernier, une géolocalisation basée sur l'adresse, un admin, et enfin si celui-ci est privé ou non.

4.2.5 Participations

Clef primaire	userId (int), idEvent (int)
Clef étrangère	idEvent référence idEvent de Events, userId référence userId de Users
Contrainte d'intégrité	/
Contrainte d'unicité	/

La table Participation représente une participation à un évènement. Celle-ci est constituée d'un userId (l'utilisateur qui participe), un idEvent (l'évènement auquel il participe), et enfin un statutParticipation (si l'utilisateur s'est bien rendu à l'évènement).

4.2.6 GroupEventMessage

Clef primaire	idMessage (int), idEvent (int)
Clef étrangère	idEvent référence idEvent de Event
Contrainte d'intégrité	/
Contrainte d'unicité	/

La table GroupEventMessage représente un message lié à un évènement. Ce dernier est représenté par un idEvent (l'Event auquel le message est lié) ,un idMessage (l'id du message en question) et un nombre de messages (le nombre de messages envoyés dans ce groupe de chat).

4.2.7 Messages

Clef primaire	idMessage(int)
Clef étrangère	messageFrom référence userId de Users
Contrainte d'intégrité	/
Contrainte d'unicité	/

La table Messages représente un message. Ce dernier est constitué d'un id, d'un émetteur (représenté par l'id de l'utilisateur qui l'a envoyé), d'un contenu et d'une date.

4.2.8 Invitations

Clef primaire	userId (int), idEvent (int)
Clef étrangère	idEvent référence idEvent de Events userId référence idUser de Users
Contrainte d'intégrité	/
Contrainte d'unicité	/

La table Invitations représente une invitation à un évènement. Cette dernière est constitué d'un idUser(utilisateur à qui est destinée cette invitation), idEvent (Evènement auquel l'utilisateur est invité) et d'un statut invitation (représenté par un integer → 0 pour refusée, 1 pour acceptée, 2 pour en attente).

4.2.9 ForgotPassword

Clef primaire	Id(int)
Clef étrangère	userId référence userId de Users
Contrainte d'intégrité	/
Contrainte d'unicité	/

La table ForgotPassword aide à la récupération d'un mot de passe oublié. Elle est constitué d'un id (GUID, identificateur global unique) qui est le code qui permettra à l'utilisateur de mettre à jour son mot de passe, d'un userId (l'utilisateur qui a oublié son mot de passe), et une date (le moment où l'utilisateur a fait cette demande de réinitialisation -> ex : 26/04/2019 08 :42 05sec).

4.3 Stored Procedure

4.3.1 Choix SP

En choisissant de travailler avec Entity Framework, deux possibilités se sont offertes :

- Travailler avec LINQ to SQL
- Travailler via des Stored Procedure

J'ai choisi de favoriser l'utilisation des Stored Procedure. J'ai dû peser le pour et le contre car les deux possibilités sont attrayantes pour différentes raisons. J'ai finalement opté pour le choix avec la meilleure performance. En effet les Stored Procedure sont plus rapides au niveau de l'exécution que LINQ : Les query de LINQ sont recompilés chaque fois tandis que les Stored Procedure réutilisent le cache du plan d'exécution, ce qui les rend plus performantes.

4.3.2 Liste SP

Ci-dessous, la liste de toutes les Stored Procedure appelées par l'API et une brève description de leur fonctionnement :

SPROC NAME	SPROC DESCRIPTION
CHECKEXIST	Stored Procedure qui vérifie si le nom d'utilisateur rentré lors de l'inscription d'un nouvel utilisateur est déjà pris
CHECKMAILEXIST	Stored Procedure qui vérifie si l'email rentré lors de l'inscription d'un nouvel utilisateur a déjà été utilisé par un autre utilisateur
CHECKMESSAGES	Stored Procedure qui retourne les messages plus récents qu'un message donné en paramètre
CREATEUSER	Stored Procedure qui effectue un insert dans la table User avec le nom d'utilisateur et le password
DELETEALLPARTICIPATIONS	Stored Procedure qui supprime toutes les participations pour un évènement donné
DELETEMESSAGE	Stored Procedure qui supprime un message pour un idMessage donné
DELETEEVENT	Stored Procedure qui supprime l'évènement dont l'id est rentré en paramètre
DELETEINVITATION	Stored Procedure qui supprime une invitation
DELETEPARTICIPATION	Stored Procedure qui supprime la participation d'un utilisateur à un évènement, l'userId et l'idEvent sont rentrés en paramètres

DELETEPROFIL	Supprimer le profil, le compte utilisateur, et toutes les participations pour un userId rentré en paramètre
GETALLEVENTSFILTER	Stored Procedure qui sélectionne tous les évènements sportifs à venir, groupés par date, sport et autour d'une certaine position donnée en paramètre
GETEVENTSBYID	Stored Procedure qui sélectionne les champs d'un évènement pour un idEvent donné en paramètre
GETIDFORGOTPASSWORD	Stored Procedure qui sélectionne si le code donné en paramètre est bien présent dans la Database
GETIMAGEPROFIL	Stored Procedure qui sélectionne une image de profil pour un utilisateur donné en paramètre
GETMESSAGEBYID	Stored Procedure qui sélectionne un message pour un idMessage donné en paramètre
GETMESSAGESEVENT	Stored Procedure qui sélectionne les 30 derniers messages pour un event donné
GETMYEVENTS	Stored Procedure qui sélectionne les évènements de l'utilisateur
GETMYEVENTSNOTIFICATION	Stored Procedure qui sélectionne les évènements de l'userId rentré en paramètre qui peuvent faire l'objet d'une notification
GETMYEVENTSNOTIFICATIONADMIN	Stored Procedure qui sélectionne les évènements que l'userId rentré en paramètre doit noter
GETPARTICIPATIONBYID	Stored Procedure qui sélectionne une participation pour un utilisateur et un évènement rentré en paramètres
GETPARTICIPATIONSEVENT	Stored Procedure qui sélectionne toutes les participations pour un évènement donné en paramètre
GETPROFIL	Stored Procedure qui sélectionne le profil d'un utilisateur
GETUSERIDFROMEMAIL	StoredProcedure qui sélectionne le userId d'un utilisateur avec une adresse mail rentrée en paramètre
GETUSERIDFROMGUID	StoredProcedure qui sélectionne le userId d'un utilisateur avec un GUID rentré en paramètre
SELECTFIRSTUSERS	StoredProcedure qui sélectionne les 10 premiers utilisateurs comprenant une

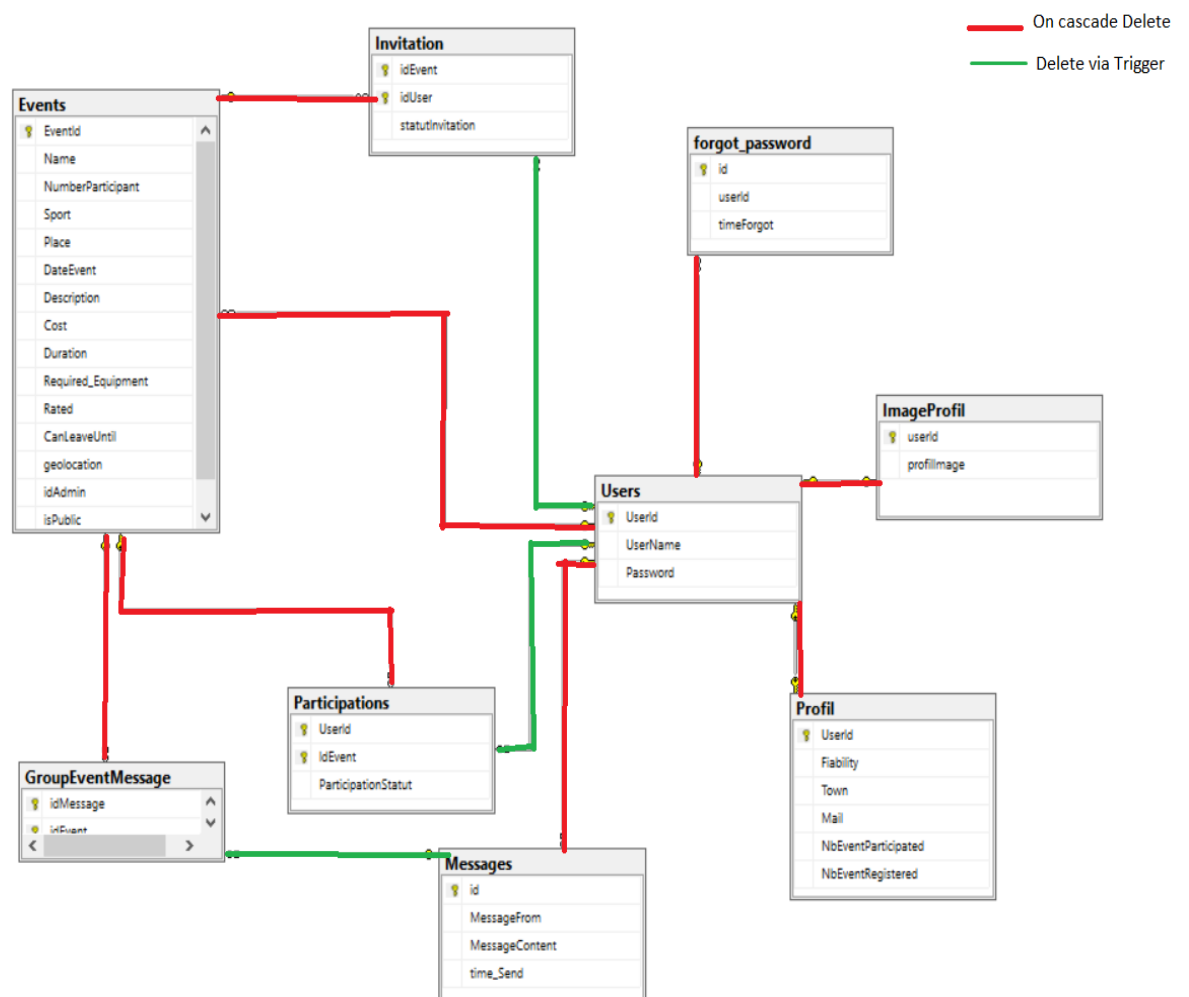
	chaîne de caractère donné en paramètre dans leur username
SELECTINVITATIONSFROMEVENT	Stored Procedure qui sélectionne les invitations envoyées pour un évènement particulier
SELECTMYINVITATIONS	Stored Procedure qui sélectionne les invitations reçues pour un utilisateur donné
UPDATEINVITATION	Stored Procedure qui met à jour une invitation
UPDATEEVENT	Stored Procedure qui met à jour un évènement
UPDATESTATUTPARTICIPATION	Stored Procedure qui met à jour le statut d'une participation pour un utilisateur et un évènement donné
UPDATEEMAIL	Stored Procedure qui met à jour l'email de l'utilisateur
UPDATEIMAGEPROFIL	Stored Procedure qui met à jour l'image de profil d'un utilisateur
UPDATEVILLE	Stored Procedure qui met à jour la ville de l'utilisateur
VALIDATELOGIN	Stored Procedure qui vérifie si le nom d'utilisateur et le mot de passe rentrés lors de la connexion correspondent bien à un utilisateur inscrit
UPDATEPASSWORD	Stored Procedure qui met à jour le mot de passe de l'utilisateur

4.4 Triggers

4.4.1 On cascade Delete

Une clef étrangère avec la mention « On cascade Delete » signifie que lors de la suppression d'un élément dans la table parent, l'élément correspondant dans la table enfant est supprimé. Sql Server empêche l'utilisation de « Cascade Delete » plusieurs fois dans la même table, l'alternative proposée par Microsoft est l'utilisation de triggers couplée à des clefs étrangères.

J'ai donc opté pour un seul « chemin » de suppression automatique via les clefs étrangères et compléter ce chemin par l'utilisation de triggers. Le schéma ci-dessous reprend les explications ci-dessus :



4.4.2 Liste Triggers

Des triggers ont été utilisés pour effectuer des traitements lors de différents Update/Insert/Delete sur certaines tables. Ci-dessous, une liste reprend leurs explications.

- Table Events
 - CheckCanLeaveUntil : Lors de l'insertion d'un évènement, un trigger est enclenché et vérifie que la limite de résignation n'est pas supérieure à la date de l'évènement.
 - DateCheck : Lors de l'insertion ou la modification d'un évènement, un trigger est enclenché et vérifie que la date de l'évènement n'est pas antérieure à la date d'aujourd'hui.
- Table GroupEventMessage
 - DeleteGroupMessage : Lors de la suppression d'un GroupEventMessage, tous les messages liés à ce groupe sont également supprimés.
- Table Participations
 - RatedFiability : Lors de la clôture d'une participation, un trigger est enclenché et vérifie la valeur du statut participation, en fonction de la valeur de ce dernier, il met la fiabilité à jour ainsi que les statistiques NBEventParticipated et NbEvent Registered dans la table profil.
 - RatedFiabilityOnDelete : Lors de la suppression d'une participation, un trigger est enclenché et vérifie met la fiabilité à jour ainsi que les statistiques NBEventParticipated et NbEventRegistered dans la table profil en fonction de la limite de désinscription de l'évènement.
- Table Users
 - DeleteUser : Lors de la suppression d'un profil, toutes les invitations, messages et participations liées à ce profil sont supprimées.

Chapitre V : Web API

5.1 Introduction

5.1.1 Le choix REST

REST (Representational State Transfert) est un standard qui a été créé en 2000 par Roy Fielding (informaticien américain notamment cofondateur d'Apache).

Les API REST sont basées sur HTTP (HyperText Transfer Protocol). L'échange repose sur des requêtes entre le client et le serveur. Le client envoie une requête http, et le serveur renvoie une réponse sous un format de données spécifiques (JSON/XML).

Une API REST doit respecter certains principes, elle doit en effet être :

- **Sans état** : L'état de la session n'est pas conservé, il est transmis au serveur à chaque nouvelle requête du client. La requête d'un client doit contenir toutes les informations nécessaires pour que le serveur puisse y répondre correctement.
- **Avec cache** : Les réponses peuvent être mises en cache par les clients ou les serveurs intermédiaires, cela permet une optimisation du nombre de données transférées.
- **Interface uniforme** : L'interface uniforme se doit de respecter elle-même 4 contraintes :
 - Identification des ressources dans les requêtes, chaque ressource est identifiée dans les requêtes.
 - Manipulation des ressources par des représentations, le client peut modifier ou supprimer la ressource grâce aux représentations fournies.
 - Messages autodescriptifs, chaque message est suffisamment explicite pour que le client sache comment l'interpréter.
 - HATEOAS (Hypermedia As The Engine of Application State), permet de naviguer entre les différentes actions disponibles sur une ressource (optionnel).
- **En couches** : Le client ne sait pas s'il est connecté directement au serveur final ou à un serveur intermédiaire.
- **Orienté client/serveur** : Les responsabilités sont divisées entre le client et le serveur. Découper l'interface utilisateur et la logique business côté serveur permet d'assurer une meilleure indépendance entre les évolutions/améliorations des 2 couches.
- **Code à la demande (facultatif)** : Du code est renvoyé par le serveur au client, ce qui permet au client d'être plus léger et générique.

Le choix de respecter (le plus possible) le protocole d'API REST en lieu et place de SOAP m'a semblé plus approprié pour mon cas de figure, avec notamment la

possibilité d'utiliser JSON. D'autres différences reprises dans le tableau ci-dessous m'ont aidé à confirmer ce choix.

#	SOAP	REST
1	Un protocole basé sur XML	Protocole architectural
2	Utilise seulement XML	Utilise Xml ou JSON pour envoyer et recevoir des données
3	Invoque les services en appelant des méthodes RPC	Invoque les services via le chemin URL
4	Ne retourne pas de contenu lisible par un humain	Le résultat retourné est lisible sous format XML ou JSON
5	Performance moins bonne que REST	Meilleure performance que SOAP, moins d'utilisation CPU,..

5.1.2 Asp.net, IIS

Pour développer/implémenter ma Web API, mon choix s'est porté sur Asp.net. Ce dernier est un framework développé dans le début des années 2000 par Microsoft. Il permet de développer des web services ainsi que des applications web.

Pour l'hébergement de mon API, j'ai dans un premier temps choisi IIS (serveur web de microsoft, qui supporte nativement Asp.net). Une fois l'API développée, la solution sera déployée sur un serveur azure.

5.1.3 Entity Framework

Entity Framework est un ORM (Object Relational Mapper), autrement dit un outil qui permet de simplifier le « mapping » entre les objets et les tables et colonnes dans la base de données.

Entity framework permet :

- De générer une base de données à l'aide d'un modèle existant
- De générer un modèle à partir d'une base de données existante

La deuxième approche a été choisie pour ce projet.

5.2 Arborescence API

La table ci-dessous décrit les différents contrôleurs de l'API :

CONTROLLER	VERB	ENTRY POINT	DESCRIPTION
EVENT	GET	api/Events/Around	Retourne tous les évènements groupés par Sport autour de la position de l'utilisateur.
	GET	api/Events/{id}	Retourne un évènement via Id
	GET	api/Events/{idEvent}/Participation	Retourne toutes les participations liées à un évènement
	PUT	api/Events	Met à jour un évènement existant
	POST	api/Events	Crée un nouvel évènement
	DELETE	api/Events/{id}	Supprime un évènement via Id
LOGIN	GET	api/Login	Check si un utilisateur est présent dans la Base de données avec les données userid – password et retourne un JWT de session si oui
	PUT	api/Login	Met à jour un mot de passe existant
	POST	api/Login/CreateProfil	Ajoute un nouvel utilisateur
	PUT	api/Login/CreateNewPassword	Met à jour un mot de passe existant lorsque l'utilisateur l'avait oublié
	DELETE	api/Login	Supprime un utilisateur existant

PARTICIPATIONS	GET	api/MyParticipations	Retourne les évènements auxquels l'utilisateur participe
	PUT	api/MyParticipations/{idEvent}	Met à jour le statut d'une participation
	POST	api/MyParticipations	Ajoute une nouvelle participation
	DELETE	api/MyParticipations/{id}	Supprime une participation existante pour un évènement donné
PROFIL	GET	api/Profil	Retourne le profil d'un utilisateur
	PUT	api/Profil	Met à jour le profil d'un utilisateur
	POST	api/Profil	Créer un nouveau profil utilisateur
	DELETE	api/Profil	Supprime un profil utilisateur existant
FORGOTPASSWORD	POST	api/ForgotPassword/{Email}	Créer un nouveau GUID pour l'utilisateur
	GET	api/ForgotPassword/{guid}	Checke si le guid est bien présent dans la Base de données
NOTIFICATION	GET	api/Notifications	Retourne les notifications de l'utilisateur
IMAGE	GET	api/Image	Retourne l'image de profil de l'utilisateur
	GET	api/Image/{id}	Retourne l'image de profil pour un userId donné
	PUT	api/Image	Met à jour l'image de profil pour un utilisateur donné
	DELETE	api/Image	Supprime l'image de profil de l'utilisateur
MESSAGES	POST	api/Messages	Ajoute un message

MESSAGESGROUP	POST	api /MessagesGroup	Ajoute un message dans le groupEventMessage
	GET	api/MessagesGroup/{idEvent}	Retourne les messages liés à l'évènement rentré en paramètre
	GET	api/MessagesGroup/Async/{idEvent}/{idMessage}	Retourne les 30 premiers messages d'un évènement
INVITATIONS	POST	api/Invitations	Ajoute une nouvelle invitation pour un utilisateur et un évènement
	GET	api/Invitations	Retourne une invitation pour un utilisateur
	GET	api/Invitations/{idEvent}	Retourne toutes les invitations pour un évènement donné
	PUT	api/Invitations	Met à jour une invitation
	DELETE	api/Invitations/{idEvent}/{userId}	Supprime une invitation

Plusieurs captures d'écran ont été mise en annexe, ces dernières représentent une documentation Swagger générée par l'API.

5.3 Sécurité et Autorisation

5.3.1 Hashage

« Quand on développe des sites web ou des applications web, il est important de toujours garder en tête que tout le monde va tenter de vous nuire. Vous devriez toujours vous comporter comme si votre code était vulnérable à une attaque. Vous pourriez penser « Personne n’aura jamais accès à ma base de données », mais ça c’est n’est pas la bonne attitude à avoir »⁵.

Introduction au Hashage

Le hashage est l’une des pratiques de sécurité les plus répandues qui doit être effectuée lors de la persistance de mots de passe. Sans cette manœuvre de sécurité, chaque mot de passe présent dans la base de données peut être immédiatement utilisé et volé si le stockage est compromis. Beaucoup de personnes utilisent la même combinaison d’e-mail/mot de passe partout sur internet, et exposer leur mot de passe via notre application laisserait les « hackers » l’utiliser sur d’autres applications web. Il est donc primordial de prendre des mesures de sécurité.

En appliquant un hashage sur le mot de passe avant de le stocker dans notre base de données, un « hacker » aura beaucoup plus de mal pour découvrir le mot de passe original.

Comment ça marche ?

Un hashage est une fonction qui va générer une représentation de notre mot de passe. Quand un utilisateur va s’inscrire dans notre application, le mot de passe sera stocké sous la forme du « hash » généré et non pas le mot de passe rentré par l’utilisateur. Quand on entre un mot de passe dans une fonction de « hashage », cela produira toujours le même résultat. Quand un utilisateur va rentrer son mot de passe pour tenter de se connecter, ce mot de passe va être « hashé » et comparé au mot de passe « hashé » présent dans la base de données : si les deux sont identiques, l’utilisateur a rentré le bon mot de passe.

Malheureusement le « hashage » d’un mot de passe n’est pas suffisant. En effet, pour contourner cette sécurité, il suffit de générer une table de « hash » avec la combinaison de lettres, numéros et symboles. Il suffit ensuite de comparer le « hash » que l’on veut cracker et voir si cela correspond.

Salting

Pour compliquer la tâche des « hackers », il faut coupler le « hashage » avec ce que l’on appelle le grain de sel (« salting »). Le « salting » consiste à ajouter un peu

⁵ Brown, Philippe. *Why do we need to hash and salt passwords ?*. Mise à jour le 21 janvier 2013. [En ligne]. <https://culttt.com/2013/01/21/why-do-you-need-to-salt-and-hash-passwords/> [consulté le 30 mars 2019].

de données au mot de passe avant de le « hasher ». Par exemple, on peut ajouter au mot de passe rentré par l'utilisateur une chaîne de caractères avant d'effectuer le « hashage ». Le « salting » est très important, cela ajoute une difficulté supplémentaire à cracker le mot de passe. Les tables de « hashes » deviennent inutilisables/improductives avec l'ajout du « salting ».

Algorithmes de Hashage

Il y a de nombreux algorithmes de « Hashage ». Parmi les plus populaires, on retrouve MD5 et la famille d'algorithmes SHA.

MD5 (Message Digest 5) a été inventé en 1991 par Ronald Rivest, il est désormais considéré comme absolument dépassé et n'est plus considéré comme sûr, il est en effet possible de casser les MD5 par force brute⁶.

Dans la famille d'algorithmes SHA, l'on retrouve la SHA-1, SHA-2, SHA-3. SHA-1 publié en 1993 par la NSA a été jugé obsolète. En effet, le 23 février 2017, Google a publié un communiqué pour annoncer que la fonction de « hashage » SHA-1 avait été cassée par « collision⁷ ». L'utilisation de SHA-2 ou SHA-3 est désormais recommandée.

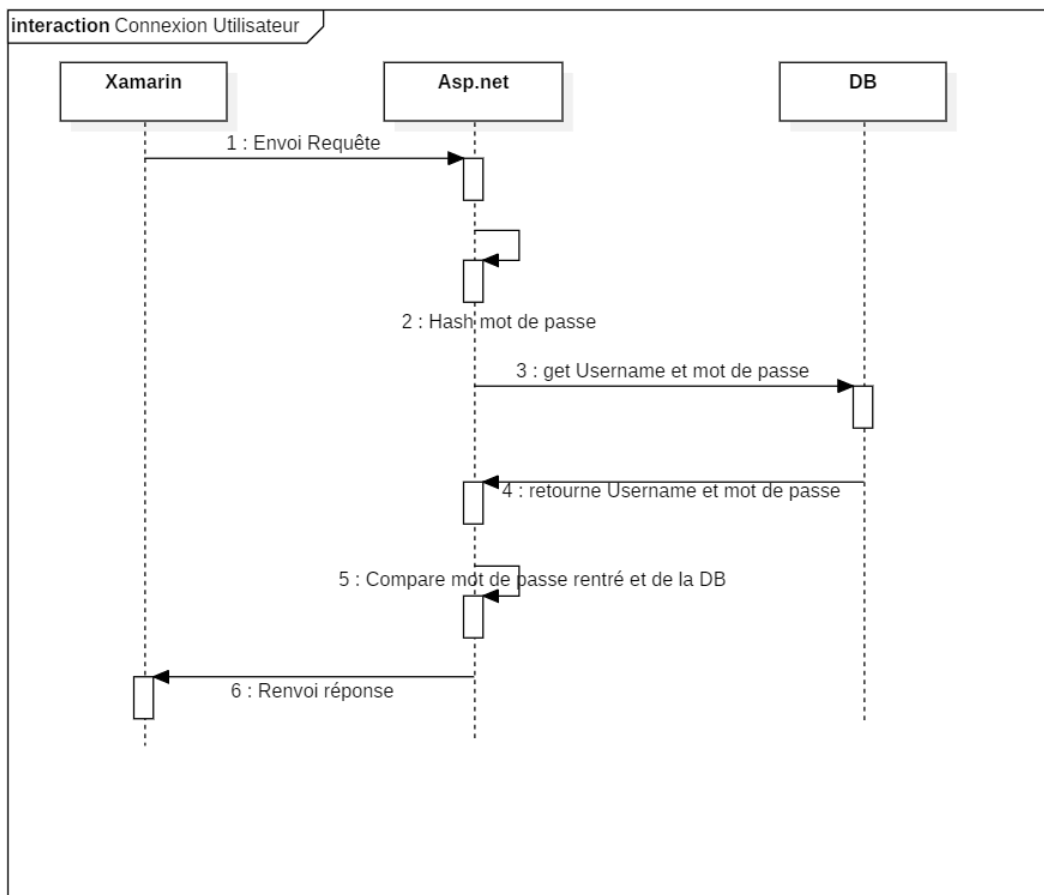
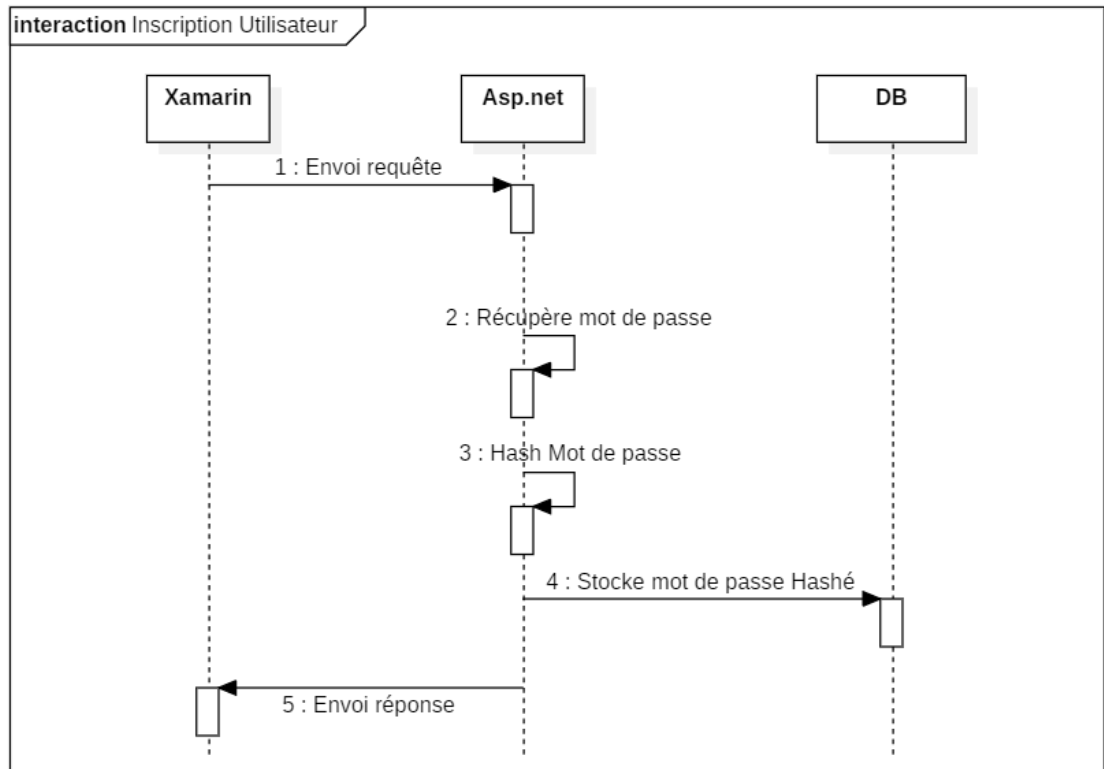
Explication NearySport

La partie « hashage » se déroule au niveau de l'API asp.net. Le mot de passe utilisateur est rentré niveau client, reçu côté serveur et est ensuite traité.

L'algorithme de Hashage choisi est SHA256, il fait partie de la famille d'algorithmes SHA-2, et a été développé par la NSA, il est utilisé en symbiose avec la classe fournie par asp.net : `Rfc2898DeriveBytesRfc2898DeriveBytes`. Ci-dessous, deux diagrammes reprenant les différentes étapes impliquant le « hashage » dans l'application mobile.

⁶ Force brute : méthode utilisée en cryptanalyse pour trouver un mot de passe en testant, un à un, toutes les combinaisons possibles.

⁷ Collision : Attaque de collision est une attaque qui tente de trouver deux entrées qui produisent le même résultat.

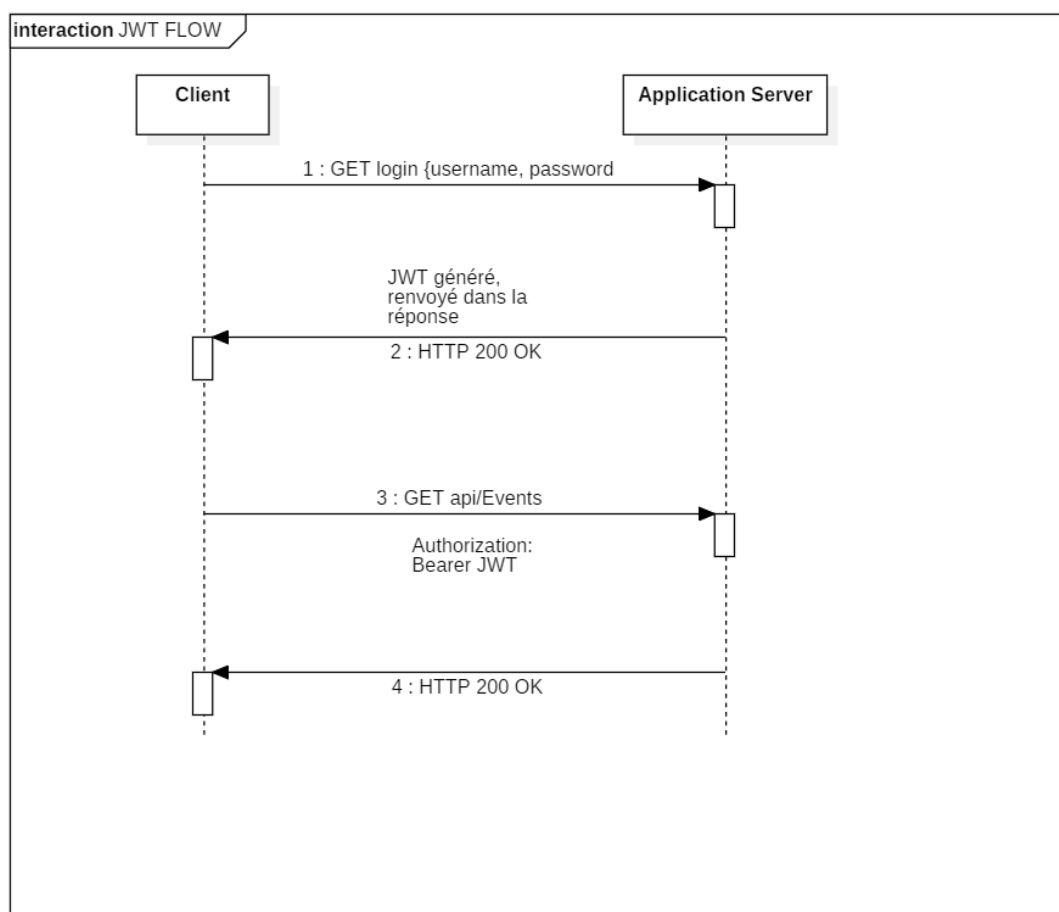


5.3.2 Json Web Token

Introduction

Dans les applications Web, il faut pouvoir gérer l'authentification d'un utilisateur. Bien souvent cette authentification était gérée via les cookies avec une session stockée côté serveur. Mais pour respecter le principe d'API REST (stateless, sans session) il faut éviter cette approche.

Les Json web tokens sont donc une alternative intéressante, après une authentification réussie, le serveur génère un hash de plus de 100 caractères qui servira de signature pendant une certaine durée. Le token sera ensuite utilisé dans chaque requête envoyée par le client vers le serveur. Ci-dessous, une représentation de l'utilisation d'un JWT via un diagramme de séquence :



JWT

Un Json Web Token (JWT) comporte trois parties, séparées par un point :

1. Header : Contient l'algorithme utilisé pour "hasher" le contenu, ainsi que le type de jeton, le tout encodé en base64 via une clef privée.
2. Contenu (Payload) : Contient toutes les informations associées à ce JWT : dans certains cas, ces données seront échangées entre le client et le serveur,

mais c'est plutôt ici un contenu "opaque" pour le client : le JWT est généré côté server et est validé sur ce même server.

3. Signature : La signature est composée de la concaténation entre le payload et le header, le tout encodé avec l'algorithme contenu dans le header.

JWT permet donc d'échanger des informations entre 2 plateformes, mais le rôle de JWT dans ce cas d'authentification est concentré sur le server : le "contrat pour le client" est de toujours passer ce JWT (obtenu par la requête d'authentification) dans le header Authorization de toutes les requêtes suivantes

Utilisation API

Dans l'API que j'ai développée, deux JSON WEB TOKEN sont générés et ont une structure un petit peu différente :

1. JWT pour authentification et appels API : Ce Json Web Token est généré après une authentification réussie de l'utilisateur. Il est échangé entre le client et le serveur pour les différentes requêtes.
 - a. Header : algorithme : HS256, type de jeton : JWT
 - b. Contenu : Expiration : 3 jours, userId
2. JWT pour notifications : Ce Json Web Token est généré après une authentification réussie de l'utilisateur. Il est échangé lors d'un service Android (voir service Android).
 - a. Header : algorithme : HS256, type de jeton : JWT
 - b. Contenu : Expiration : 6 mois, userId

Chaque requête contiendra le JWT dans le header Authorization :

```
client.DefaultRequestHeaders.Authorization =  
    new AuthenticationHeaderValue("Bearer", _token);
```

Stockage JWT

Le Json Web Token, une fois généré, aura besoin d'être stocké chez le client pour que celui-ci puisse l'envoyer avec chacune de ses requêtes.

Pour ce faire, le JWT sera stocké dans les SharedPreferences⁸.

⁸ SharedPreferences : Espace de stockage propre à chaque application Android avec un système clef/valeur.

5.3.3 Attribut customisé : Authorize filter

Un filtre d'authentification/autorisation permet d'authentifier/autoriser une requête HTTP. Ce filtre peut être appliqué sur une action ou sur un contrôleur. Un filtre sera donc appliqué sur chaque contrôleur pour filtrer les requêtes HTTP.

```
[JWTAuthenticationFilter]
public class ParticipationsController : ApiController
{
}
```

Ce filtre se charge de valider le JWT envoyé dans le header Authorization de la requête HTTP :

- Le filtre vérifie que le schéma d'authentification est bien « Bearer »⁹
- Le filtre va ensuite valider le JWT en vérifiant que :
 - L'algorithme est correct
 - Le JWT n'a pas expiré
 - La signature est correcte

Ensuite, l'information de userid peut être utilisée dans le traitement de la requête pour distinguer par exemple si le userid authentifié est bien l'administrateur de l'événement.

Si l'utilisateur n'est pas autorisé, il recevra un code de réponse 401 (Unauthorized).

Il est néanmoins possible de permettre à certains entry Point d'être appelé par des anonymes, pour ce faire il faut procéder comme ci-dessous :

```
[System.Web.Http.AllowAnonymous]
public HttpResponseMessage Xamarin_CreateNewPassword(string password)
{
}
```

⁹ Bearer représente le « porteur » du jeton, affirme le fait que l'utilisateur est bien authentifié.

5.4 Règles Business Côté Serveur

Certaines « règles » ont été implémentées côté serveur pour préserver l'intégrité des données et vérifier que les différents appels soient effectués par des utilisateurs autorisés.

5.4.1 EventController

- Lors de la création d'un nouvel évènement, la limite de départ fixé par l'administrateur ajoutée à la date de l'évènement doit être inférieure ou égale à `DateTime.Now` (date actuelle).
- Lors de la création d'un nouvel évènement, l'heure de début et de fin se doivent d'être différentes, la date ne peut pas déjà être passée.
- Lors de la création d'un nouvel évènement, si l'adresse n'est pas correcte, une erreur est renvoyée à l'utilisateur (via google Maps API).
- Lors de la création d'un nouvel évènement, l'adresse rentrée par l'utilisateur est transformée en latitude longitude via l'utilisation de Google Maps Geocoding.
- Lors de la sélection des évènements, le nombre de kilomètres autour desquels chercher des évènements doit être positif, le nombre de sports filtrés ne peut pas être null.
- Lors de la mise à jour d'un évènement, l'utilisateur ayant fait appel à cet entry point doit être l'administrateur de l'évènement.
- Lors de la suppression d'un évènement, l'utilisateur ayant fait appel à cet entry point doit être l'administrateur de l'évènement.

5.4.2 ForgotPasswordController

- Pour recevoir un mail lors de la récupération de mot de passe, l'utilisateur doit avoir rentré un mail existant dans la base de données.
- Pour que le code reçu dans l'e-mail soit toujours valide, il faut le confirmer au maximum 5min après la réception de l'e-mail.

5.4.3 LoginController

- Lors de l'inscription d'un nouvel utilisateur, le nom d'utilisateur rentré ne peut pas être déjà utilisé par un autre utilisateur.
- Lors de l'inscription d'un nouvel utilisateur, l'e-mail rentré ne peut pas déjà être utilisé par un autre utilisateur.
- Lors de la suppression d'un utilisateur, la personne devant être à l'origine de cette demande doit être l'utilisateur pour qui cette action est destinée.
- Lors de l'inscription d'un nouvel utilisateur, le mot de passe rentré doit faire entre 8 et 15 caractères et le nom d'utilisateur au minimum 3 caractères.

5.4.4 ParticipationController

- Lors de l'ajout d'une nouvelle participation dans un évènement, celui-ci ne peut pas déjà être rempli. Si c'est le cas, la participation est supprimée.

- Lors de la mise à jour du statut d'une participation, l'utilisateur mettant à jour le statut doit être administrateur de l'évènement pour lequel il effectue cette mise à jour.
- Seul l'utilisateur peut supprimer sa propre participation à un évènement.

5.4.5 ProfilController

- Lors de la suppression d'un profil utilisateur, la personne devant être à l'origine de cette demande doit être l'utilisateur pour qui cette action est destinée.
- Lors de la mise à jour d'un profil utilisateur, la personne devant être à l'origine de cette demande doit être l'utilisateur pour qui cette action est destinée.
- Lors de l'ajout d'un profil, l'e-mail rentré ne doit pas déjà être utilisé par un autre utilisateur, et l'e-mail doit satisfaire le standard utilisé (être valide, pas une chaîne de caractères random, vérifié via .net).

Chapitre VI : Interface Utilisateur Android

6.1 Le choix Xamarin (native)

6.1.1 Introduction

Lorsque l'on évoque le développement d'applications mobiles, beaucoup de gens pensent que les seules possibilités sont les langages natifs Java et Objectif-C. Pourtant, de nombreuses plateformes ont vu le jour au cours de ces dernières années, permettant une approche « cross-platform »¹⁰. Parmi ceux-ci, l'on compte notamment Ionic (TypeScript), React Native (JavaScript/React) ou encore Xamarin (C#). Ci-dessous, un tableau reprenant les différences des frameworks cités précédemment :

Différences	Xamarin	React Native	Ionic
Code	C#	JavaScript	JavaScript/TypeScript
Portabilité	IOS, Android, Windows, Mac OS	IOS, Android	IOS, Android
Code réutilisabilité	75% code Native 98% Xamarin Forms	70% code	98% code
UI	Native UI	Native UI	HTML, CSS
Communauté	Grande communauté	Grande communauté	Grande communauté
Compilation Android	AOT/JIT	JIT	JIT
Compilation IOS	AOT	Interpreter	JIT/WKWebView

Le tableau ci-dessus reprend les principales différences entre les frameworks cross-platform. AOT = Compilation anticipée, traduit un langage évolué en langage machine avant l'exécution du programme. JIT = Compilation à la volée, traduction du byte-code en code machine au moment de l'exécution. Interpreter = « traduit » le programme pas à pas, généralement cela cause des performances un peu moins optimisées.

Mon choix s'est porté sur Xamarin, vu que j'ai déjà une base solide en C# : le développement sera donc plus rapide et plus facile que sur les autres frameworks où il aurait fallu me familiariser avec le langage utilisé par le framework.

6.1.2 Xamarin native vs Xamarin Forms

Xamarin offre la possibilité à ses développeurs le choix entre Xamarin native (Android, IOS, Windows phone) et Xamarin Forms. Xamarin forms offre la possibilité d'utiliser le même code (back-end¹¹ et front-end¹²) pour les 3 plateformes tandis que Xamarin native ne permet que de réutiliser que le code back-end, une interface utilisateur devant être développée pour chaque plateforme (75% de code partagé). Un schéma ci-dessous permet d'y voir plus clair :

¹⁰ Un logiciel dit « cross-platform » est un logiciel conçu pour fonctionner sur différentes plateformes.

¹¹ Back-End : partie non-visible de l'iceberg, typiquement base de données, serveur,...

¹² Front-end : partie visible de l'iceberg en développement, ce que l'on voit à l'écran

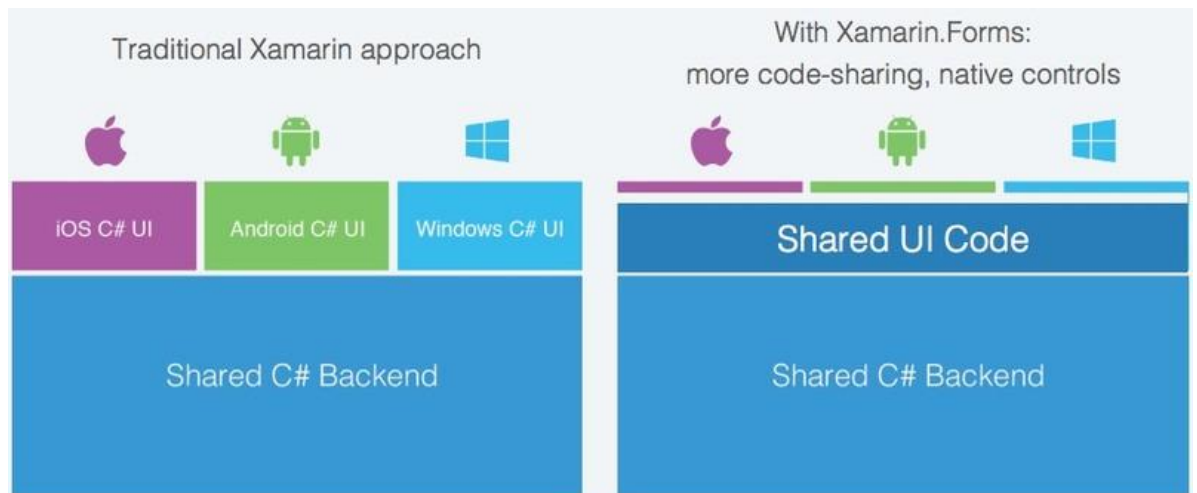


Tableau comparatif Xamarin : https://applikeysolutions.com/uploads_production/ckeditor/attachme/nts/179/content_xamarin-comparison.jpg [Consulté le 25 mars 2019]

Outre l'explication ci-dessus, certaines différences sont aussi à prendre en compte dans le choix du framework. Xamarin Forms permet un gain de temps et le code est plus facile à mettre à jour étant donné qu'il n'y a qu'un seul code à changer. Mais il y a également certains désavantages : la taille de l'application est plus volumineuse et une performance réduite face à une interface utilisateur native.

Il est vrai que partager une interface utilisateur et un code back-end pour plusieurs plateformes est très attrayant, mais la performance me semble encore plus importante, notamment pour l'expérience utilisateur, j'ai donc choisi de travailler avec Xamarin Native et plus particulièrement Xamarin Android.

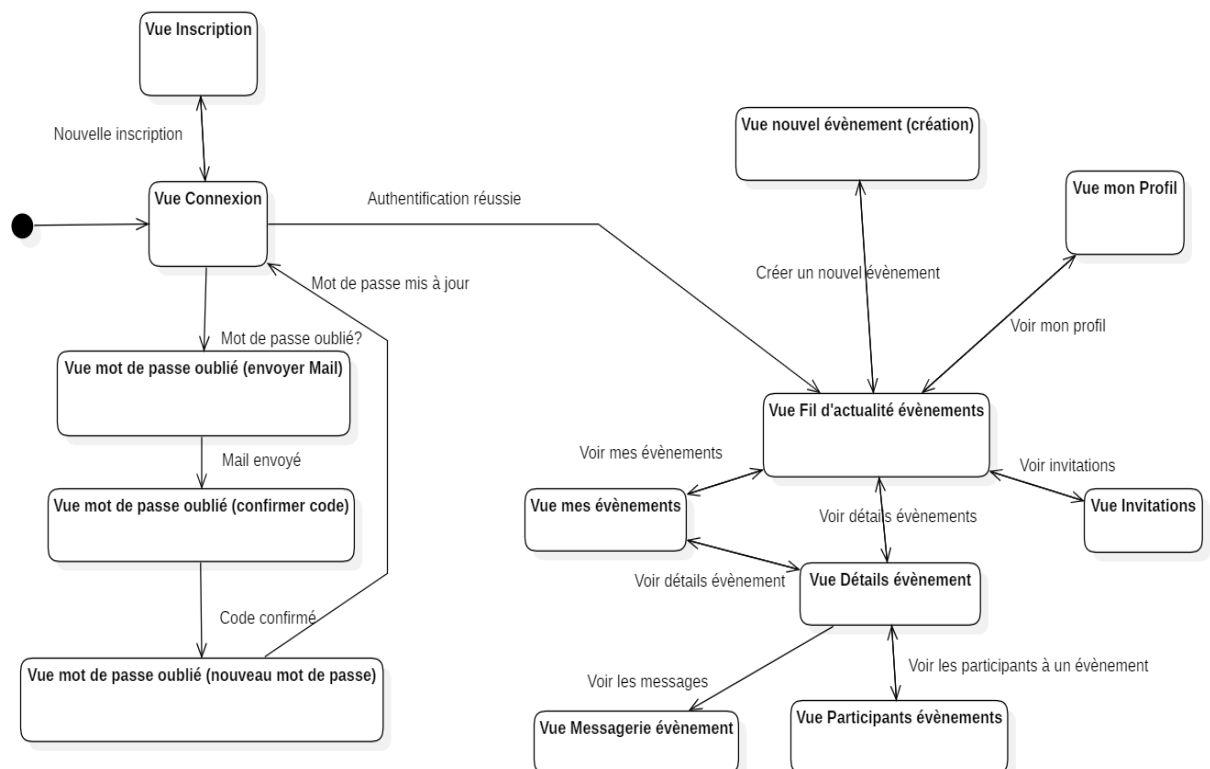
6.2 Vues Android

6.2.1 Introduction

De nombreuses vues ont été pensées et implémentées pour cette application :

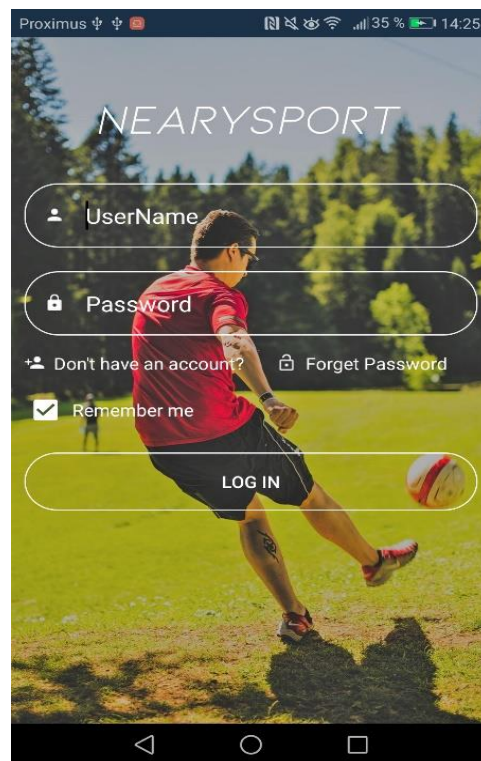
- La vue Connexion
- La vue Inscription
- La vue Mot de passe oublié (envoyer mail)
- La vue Mot de passe oublié (confirmer code)
- La vue Mot de passe oublié (nouveau mot de passe)
- La vue Fil d'actualité évènements
- La vue Nouvel évènement
- La vue mon Profil
- La vue mes évènements
- La vue Invitations
- La vue Participants évènements
- La vue Messagerie évènements
- La vue Détails évènements

Je reprends le diagramme fonctionnel de navigation d'écrans.

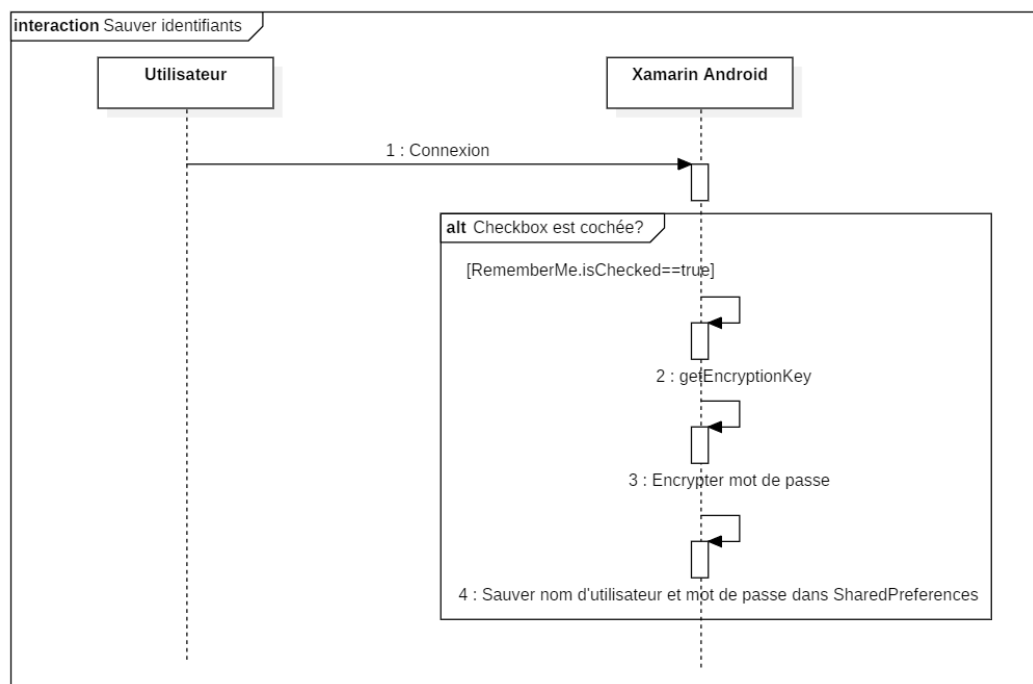


Chacune de ces vues sont détaillées ci-dessous avec des screenshots.

6.2.2 Vue connexion



Cette vue permet à l'utilisateur de se connecter s'il a déjà un compte existant. Il y rentre son nom d'utilisateur et mot de passe. Il peut également choisir de cocher la checkBox pour que l'application se souvienne de son nom d'utilisateur/mot de passe. Ses identifiants seront sauvés dans SharedPreferences et y seront cryptés. Le processus de sauvegarde de mot de passe est décrit ci-dessous via un diagramme de séquences.



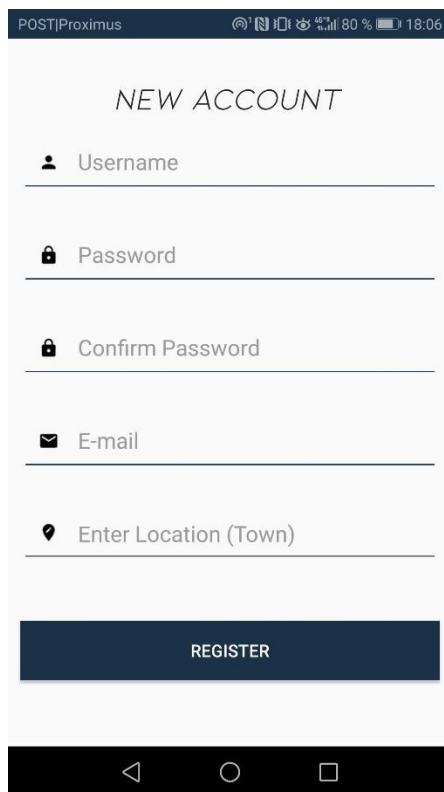
Si c'est la première visite de l'utilisateur, il peut cliquer sur le label « Don't have an account ? », il sera redirigé vers une autre vue de l'application où il aura la possibilité de se créer un compte. Si l'utilisateur a oublié son mot de passe, il peut cliquer sur le label « Forget Password », il sera dirigé ensuite vers un écran où il aura la possibilité d'envoyer un mail de récupération de mot de passe sur sa boîte mail (le processus complet sera décrit plus bas).

Si la connexion de l'utilisateur est correcte, celui-ci sera redirigé vers la vue fil d'actualité, un Json Web Token lui sera renvoyé et ce dernier sera crypté et stocké dans les SharedPreferences du téléphone de la même manière que pour la fonction Remember me. Le processus de création de connexion d'un utilisateur est repris dans la section 3.2 reprenant les principaux scénarios de l'application.

Règles techniques :

- La clef d'encryption nécessaire au cryptage est stockée dans le répertoire Resources/String

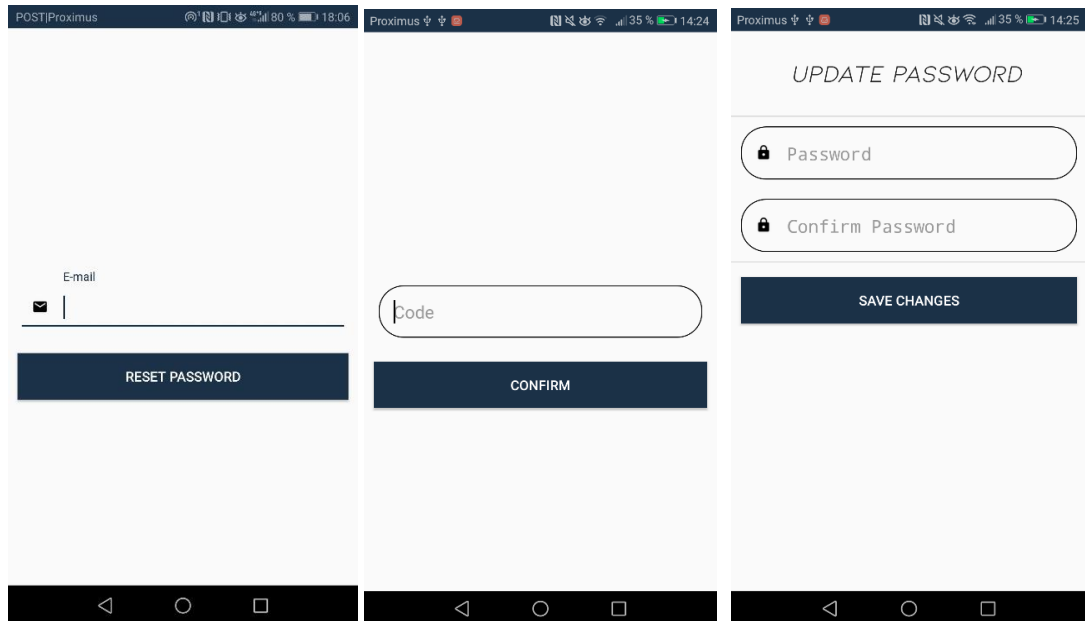
6.2.3 Vue inscription



La vue est lancée lorsque l'utilisateur clique sur le label « Don't have an account ? » présent sur la vue connexion. L'utilisateur sera donc obligé d'insérer un nom d'utilisateur, un mot de passe comportant 8 caractères minimum et 15 maximum. Il devra également saisir une adresse e-mail valide. Il aura la possibilité d'avoir l'aide de l'API de Google maps(alternative envisageable : OPENSTREETMAPS) qui l'aidera à trouver la ville de l'utilisateur.

Si l'inscription se déroule de manière correcte, l'utilisateur est redirigé vers la vue connexion où il pourra renseigner ses identifiants. Le processus est décrit de manière plus détaillée via un diagramme de séquences en section 3.2.1.

6.2.4 Vue mot de passe oublié (envoyer e-mail, confirmer code, mise à jour mot de passe)



Cette vue en comporte en fait trois. La réinitialisation du mot de passe s'effectue en 3 étapes. Dans un premier temps l'utilisateur rentre son e-mail avec laquelle son compte est lié.

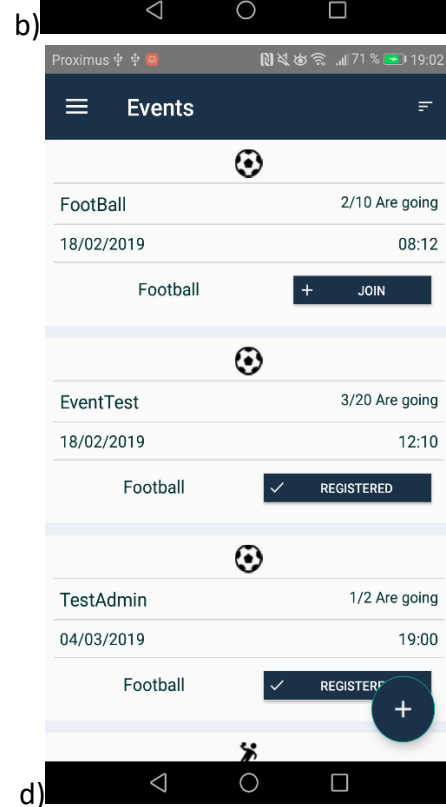
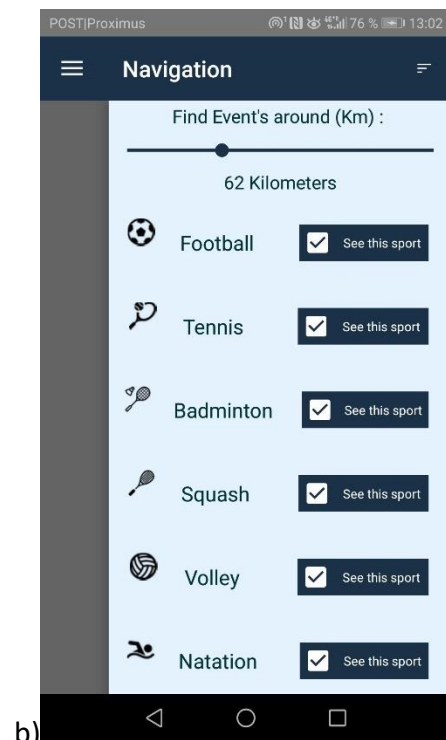
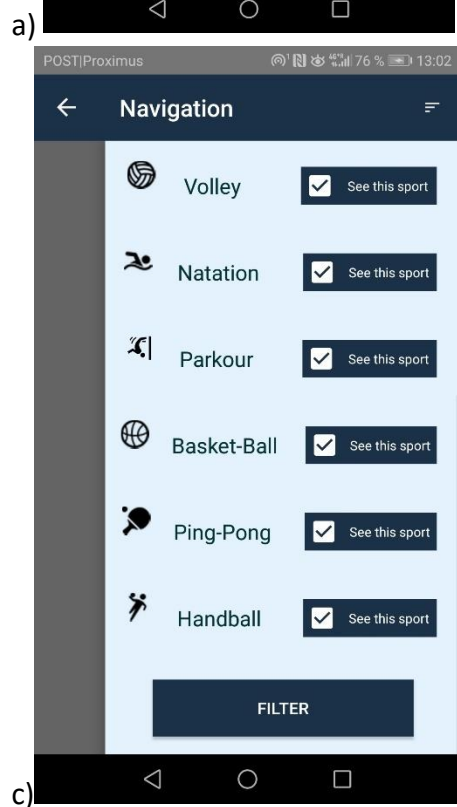
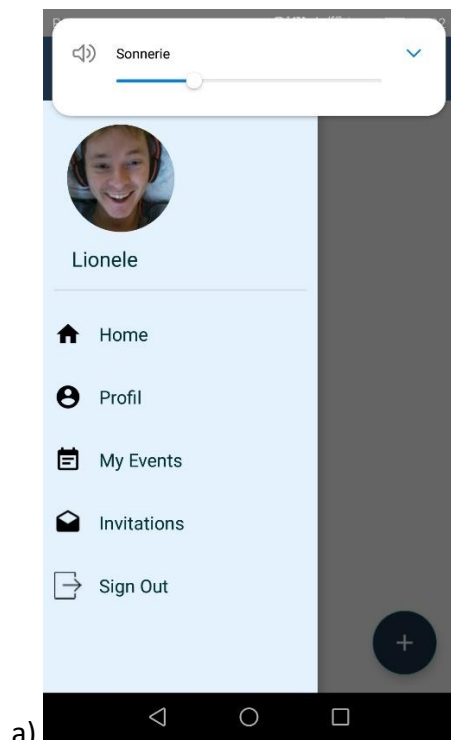
Une fois le mail envoyé, il reçoit un code sur sa boîte e-mail (si celle-ci existe bien dans le système), et il est renvoyé sur la vue où il doit confirmer le code reçu (le code expire au bout de 5 minutes). Si le code rentré par l'utilisateur est valide, il est ensuite redirigé vers la 3^e et dernière vue qui va lui permettre de choisir un nouveau mot de passe pour son compte utilisateur.

Le processus de ces différentes vues est détaillé en section 3.2.3.

Règles techniques :

- Pour être redirigé vers la vue de confirmation de code, l'utilisateur n'a pas besoin d'une adresse mail présente dans la base de données. Cela empêche les personnes mal intentionnées d'accéder à une adresse mail.
- Les champs Password et Confirm Password doivent être identiques et correspondre au standard de mot de passe imposé dans l'application (Entre 8 et 15 caractères).

6.2.5 Vue fil d'actualité évènements



La vue Fil d'actualité permet d'accéder aux autres vues de l'application. En cliquant sur le burger (figure a), l'utilisateur pourra en effet accéder à une listview lui proposant de :

- Revenir au fil d'actualité
- Accéder à son profil
- Accéder aux évènements qui lui sont propres
- Accéder aux invitations
- Se déconnecter

En cliquant sur le Floating Action Button (figure d), l'utilisateur sera redirigé vers une vue où il pourra créer un évènement.

S'il clique sur le bouton filtre (en haut à droite de l'écran, figure b,c), il pourra filtrer le nombre de kilomètres autour duquel il veut chercher les évènements et quels sports il veut voir s'afficher dans le fil d'actualité.

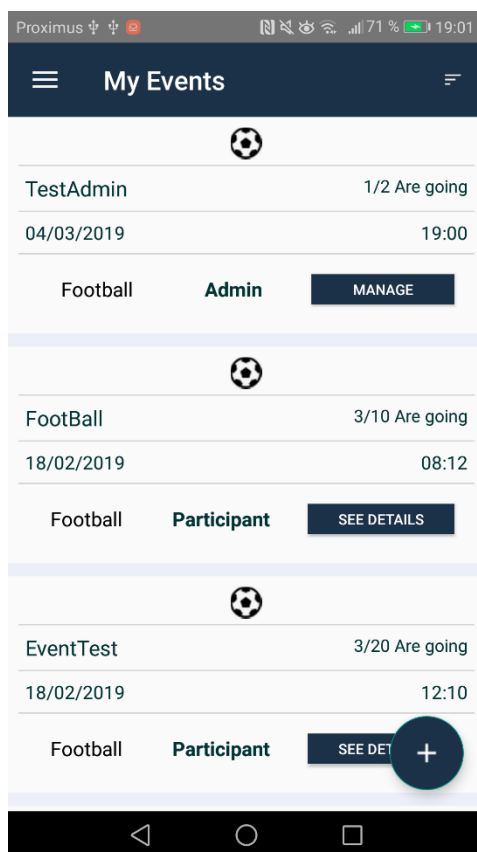
La vue permettra surtout d'accéder aux évènements disponibles dans le fil d'actualité. En cliquant sur le bouton join, il aura la possibilité de rejoindre l'évènement. Le bouton changera donc son texte en « registered », s'il clique de nouveau sur le bouton, il sera désinscrit de l'évènement. En cliquant sur l'évènement il pourra accéder aux détails de l'évènement. En effectuant un slide vers le bas lorsque l'on se trouve au début de la liste, un refresh sera effectué sur le fil d'actualité.

Le processus de recherche d'évènements est détaillé en section 3.2.5.

Règles techniques :

- Les évènements sont chargés dizaine par dizaine, lorsque la fin de la listview est atteinte, d'autres évènements sont chargés s'il en reste.
- L'utilisateur peut faire varier le nombre de kilomètres autour duquel il veut chercher un évènement, si celui-ci met la barre à 0, le serveur augmentera ce chiffre de 1 pour avoir un rayon effectif autour duquel chercher.
- Si l'utilisateur décoche tous les sports, un message approprié lui sera renvoyé.

6.2.6 Vue mes événements



La vue mes événements permet à l'utilisateur de consulter les événements auxquels il participe ou auxquels il a participé récemment (encore 3 jours après la date de l'évènement). Il aura également la possibilité de naviguer dans les différents menus via la listview disponible sur la gauche (cliquer sur le burger).

En cliquant sur l'un des événements (ou sur le bouton SEE DETAILS) il accèdera à la vue détails événement. Le bouton changera son libellé pour « MANAGE » si l'utilisateur est Administrateur de l'évènement.

En cliquant sur le Floating Action Button, il accèdera à la vue Création Evenement.

Règles techniques :

- Les événements sont triés par rôle utilisateur (Admin/Participant).
- L'utilisateur doit être connecté pour accéder à ses événements.

6.2.7 La vue création évènement

The image displays two screenshots of a mobile application interface for creating a new event. The left screenshot shows the top half of the form with the title 'NEW EVENT' in a dark blue header. Below the header, there are input fields for 'Event's Name', 'Number of Particip...', and 'Cost'. A 'Sport' section features a football icon and the text 'Football'. Below this is a date picker labeled 'Pick a Date : ' with a 'DD/MM/YYYY' format. At the bottom of this section are 'Begin' and 'End' buttons, and a 'Resignation's Limit' field. The right screenshot shows the bottom half of the form. It starts with a duration selector set to '1 Hour'. The 'Address' section includes fields for 'Town', 'Zip Code', and 'Street Address'. Below this is a 'Required Equipment' field with a football icon and the text 'Ball, racket,...'. A 'Description' field follows. There is a checkbox labeled 'Is Private' which is checked. At the bottom is a large dark blue button labeled 'CREATE EVENT'.

Cette vue permet à l'utilisateur de créer un nouvel évènement.

L'utilisateur accède à cette vue en cliquant sur le floating action button présent dans la vue mes évènements ou à partir du fil d'actualité. Il pourra y choisir le nom, le nombre de participants, le coût de l'évènement, le sport, la date (jour et heure de début/fin), la limite de résignation¹³, l'adresse, l'équipement requis et la description de l'évènement, ainsi que si l'évènement est privé ou public.

Règles techniques :

- L'évènement doit obligatoirement avoir tous les champs cités précédemment remplis.
- L'adresse doit être valide, si lors de la création de l'évènement, le serveur ne valide pas l'adresse, un message d'erreur sera renvoyé.
- Un `dialogTimePicker`¹⁴ Fragment sera lancé pour aider l'utilisateur à choisir la date de son évènement.

¹³ Limite de résignation : Limite avant laquelle l'utilisateur peut quitter l'évènement sans que cela ait un impact sur sa fiabilité.

¹⁴ Un `dialog Time Picker` est un composant Android permettant à l'utilisateur de sélectionner une date et une heure via l'interface utilisateur.

- Une dropdown list¹⁵ contient tous les sports possibles pour créer un évènement.
- Une dropdown list contient toutes les limites de résignation possible pour l'évènement.

Fiabilité et limite de résignation

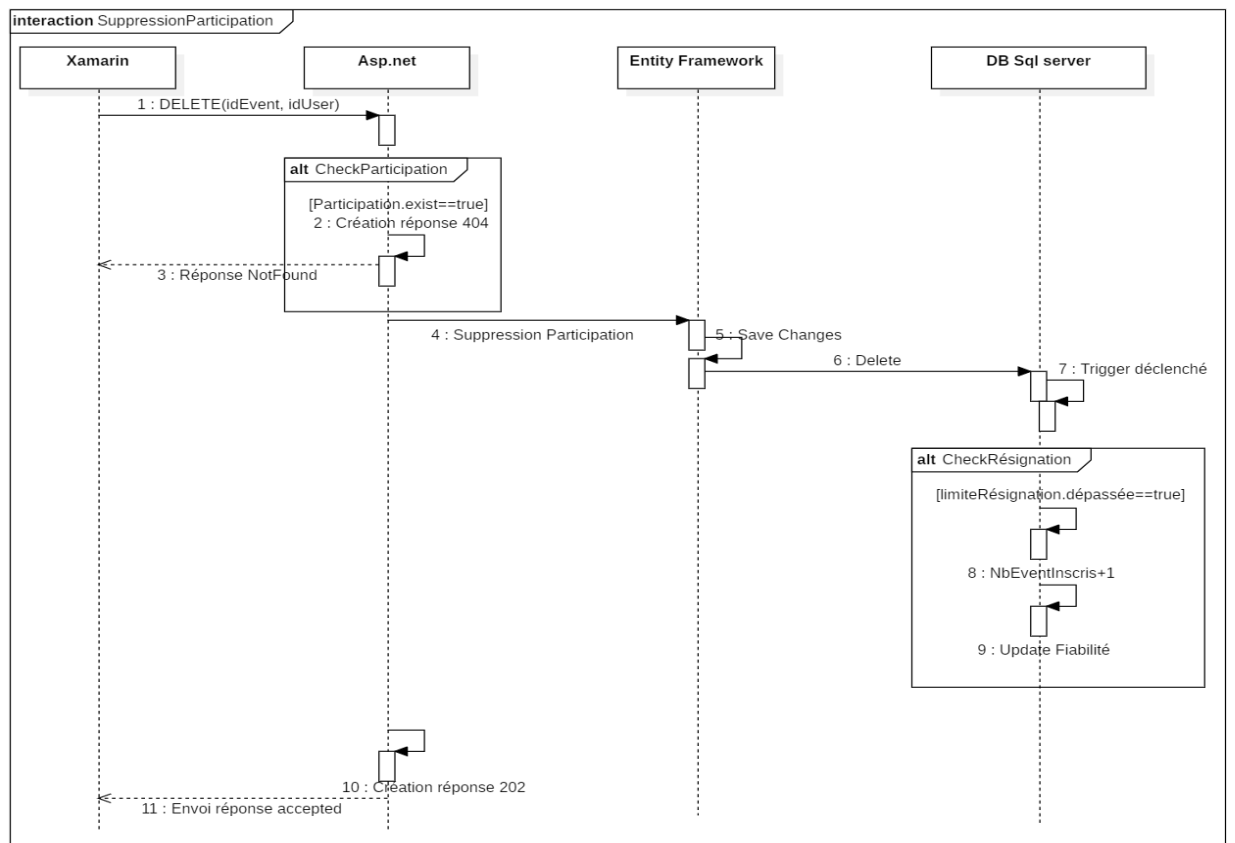
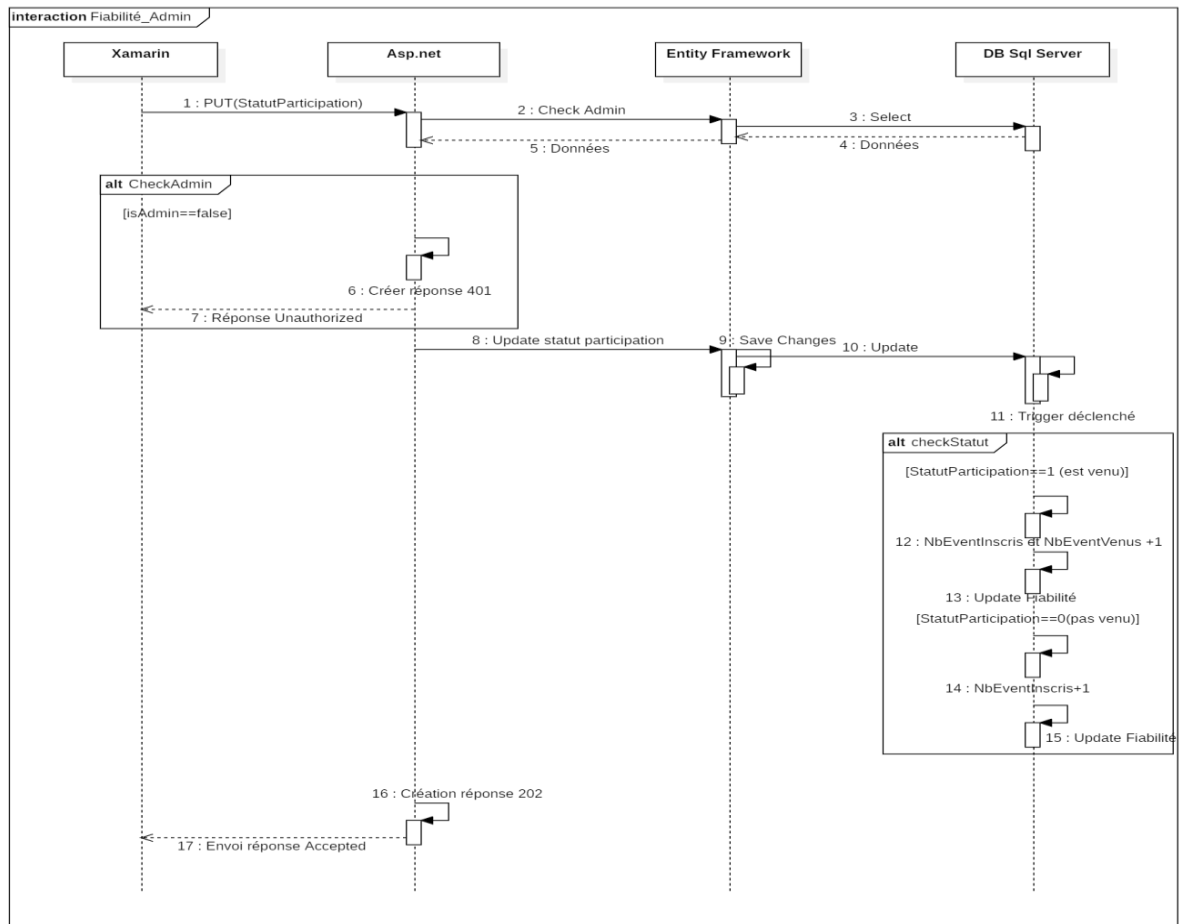
La fiabilité d'un utilisateur désigne si l'utilisateur est « fiable » et si celui-ci se présente aux évènements auxquels il s'est inscrit. L'utilisateur se voit attribuer une fiabilité maximum de 10 (sur 10) lors de la création de son compte. La fiabilité est calculée sur un total de 10, 0 étant la valeur la plus basse. Son résultat est obtenu via la division entre le nombre d'évènements où l'utilisateur est venu, et le nombre d'évènements auxquels il s'est inscrit (Exemple : l'utilisateur Damiaan s'est inscrit à 5 évènements mais n'est venu qu'à un d'entre eux, sa fiabilité sera donc de 2/10).

Cette fiabilité peut être affectée par :

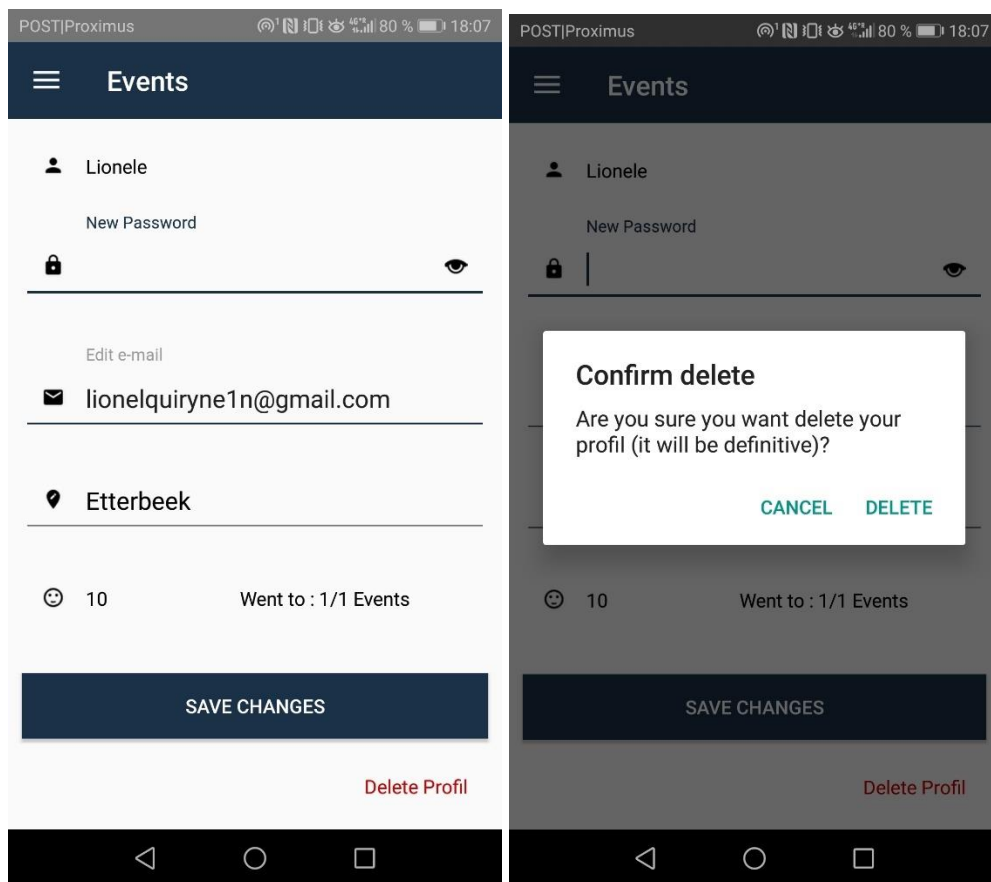
- Les évènements où il s'est inscrit : si l'utilisateur s'inscrit à un évènement et ne vient pas, l'administrateur aura pour devoir à la fin de cet évènement de préciser les absents. Si cela est fait correctement, l'utilisateur se verra ajouter un évènement auquel il n'est pas venu (et donc baisser sa fiabilité, sauf si elle est déjà à 0). Mais si l'utilisateur se présente bien à l'évènement inscrit, il se verra ajouter un évènement inscrit et un évènement où il est venu (et donc augmenter sa fiabilité, sauf si elle est déjà maximale)
- La désinscription à un évènement où la limite de résignation a été dépassée : par exemple, imaginons que l'utilisateur soit inscrit à un évènement pour lequel la limite de résignation est de trois jours. Deux jours avant l'évènement, l'utilisateur décide de se désinscrire, il recevra un message d'avertissement l'informant que s'il décide effectivement de se désinscrire, sa fiabilité sera impactée. En effet, un évènement auquel il n'est pas venu sera ajouté à sa fiabilité.

Les processus expliqués ci-dessus sont détaillés dans des diagrammes de séquence ci-dessous : Le premier diagramme reprend la notation effectuée par l'administrateur à la fin d'un évènement, le deuxième diagramme reprend la suppression d'une participation par l'utilisateur.

¹⁵ Une dropdown list est un composant graphique Android (présent dans beaucoup d'autres langages également), permettant à l'utilisateur de choisir un élément dans une liste.



6.2.8 La vue Mon profil



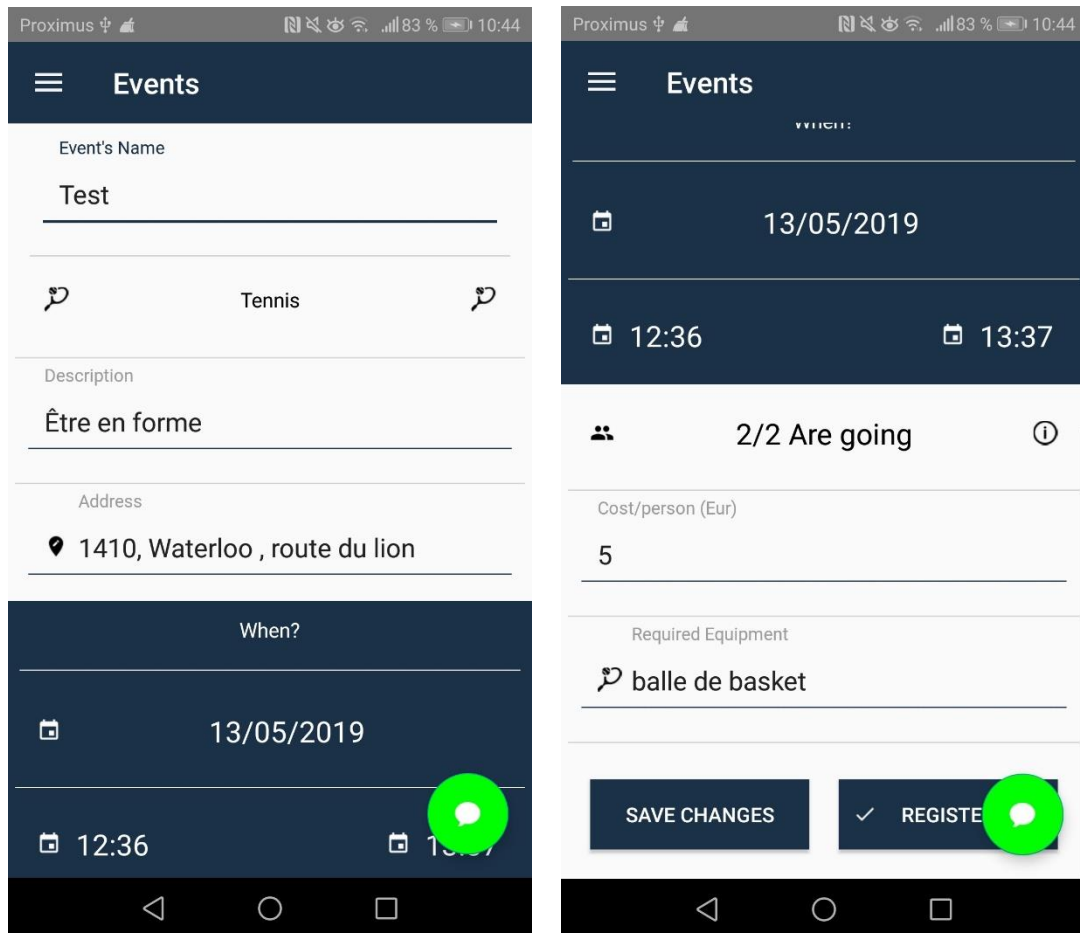
La vue Profil, accessible depuis la listview de navigation, permet de consulter son profil (Nom d'utilisateur, mot de passe, e-mail, emplacement défini et la fiabilité) et de modifier certains champs de celui-ci. Les champs éditables sont le mot de passe, l'adresse e-mail, et la ville. Elle permet également de changer de vue via la listview de navigation.

Si un utilisateur décide de supprimer son profil, il en a la possibilité via le label Delete Profil. En cliquant dessus, un avertissement sera lancé pour savoir si c'est bien la volonté de l'utilisateur. S'il confirme ce choix, son profil sera supprimé (supprimant ainsi les évènements, participations, messages et invitations liés à cet utilisateur, ce processus de suppression est réalisé à l'aide de triggers décrits précédemment).

Règles techniques :

- Quand l'on modifie le mot de passe, un champ invisible se rend visible et nous demande de confirmer le nouveau mot de passe.
- Lors de la suppression d'un profil, toutes les informations liées à l'utilisateur (participations évènements et profil) sont supprimées.
- L'adresse E-mail doit être valide si celle-ci est modifiée.
- Le mot de passe doit contenir entre 8 et 15 caractères si celui-ci est modifié.
- L'adresse ne peut contenir que 50 caractères maximum.

6.2.9 La vue détails évènement



La vue Details Events est accessible depuis les vues mes évènements et fil d'actualité. Elle apparaît à l'écran lorsque l'utilisateur appuie sur un évènement présent dans les deux vues. Elle permet d'afficher plus de détails concernant cet évènement. L'utilisateur peut également décider de s'y inscrire ou de se désinscrire. En fonction du rôle de l'utilisateur, certains champs pourront être modifiés si celui est administrateur. Un bouton deviendra visible pour qu'il puisse sauver les changements apportés à l'évènement (comme sur l'image ci-dessus).

L'utilisateur gardera la possibilité de changer de menu via la listview de navigation sur la gauche de l'écran.

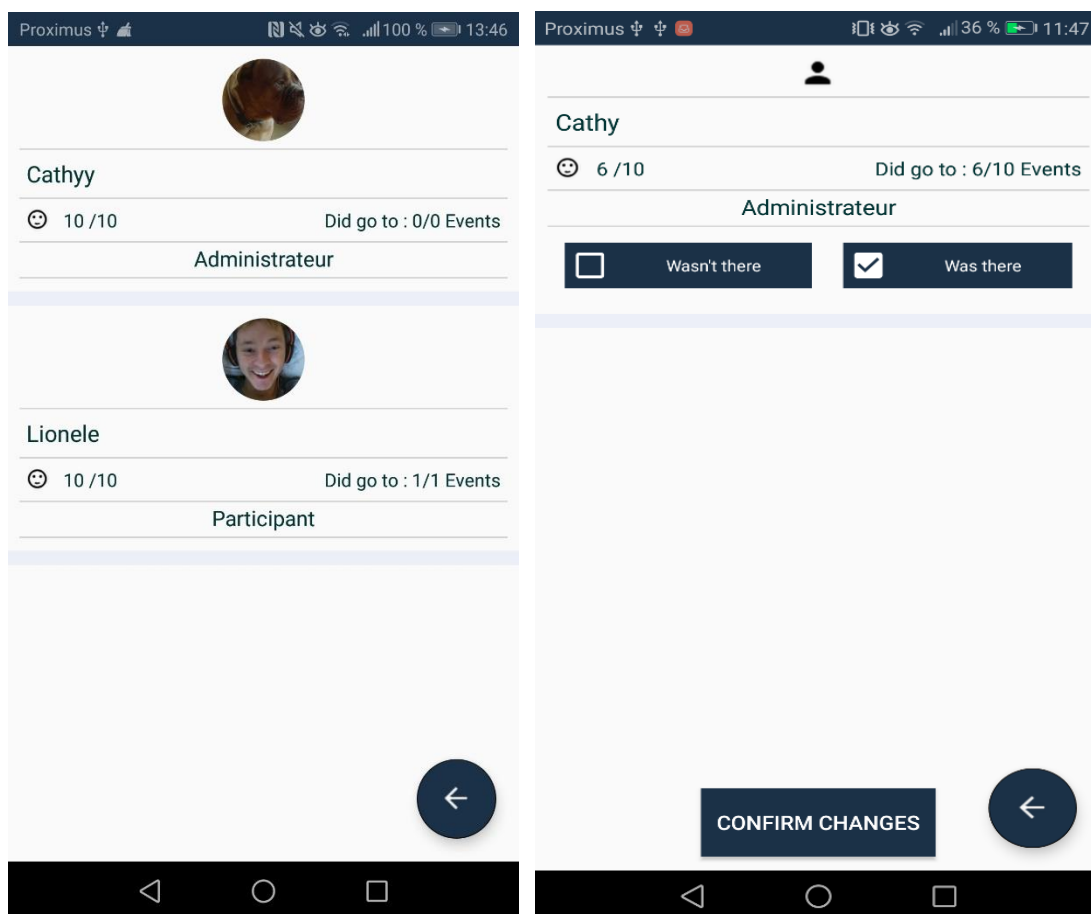
Si l'utilisateur clique sur le point info à droite du nombre de participants, il pourra accéder à la vue Participants de l'évènement, il pourra donc voir qui sont les participants de l'évènement.

Si l'utilisateur clique sur le floating action button représentant une bulle de messagerie, il accèdera à la vue Messagerie. L'utilisateur n'a accès à cette vue que s'il est inscrit à l'évènement consulté.

Règles techniques :

- Les mêmes règles que lors de la création d'un évènement entrent en ligne de compte si l'administrateur veut modifier celui-ci.
- Le bouton « save changes » n'est visible que si l'utilisateur est administrateur de l'évènement consulté.
- Si l'utilisateur n'est pas encore inscrit, le bouton « registered » sera affiché avec le text « join » à la place.
- Lors d'un click effectué sur l'adresse de l'évènement, si l'application google Maps est installée sur le téléphone de l'utilisateur, cette dernière sera ouverte avec l'adresse comme paramètre rentrant de l'application.

6.2.10 La vue Participants



Cette vue est accessible via la vue Détails évènement. Elle permet de voir quels sont les utilisateurs participants à l'évènement.

L'utilisateur peut voir quel est le rôle occupé par les différents participants dans l'évènement en question. Il peut également observer la fiabilité de chaque participant et le nombre d'évènements auxquels ceux-ci ont réellement participer, il peut également voir la photo de profil d'un utilisateur.

En cliquant sur le Floating Action Button, l'utilisateur peut revenir à la vue Détails évènement de l'évènement que celui-ci consultait.

Dans le cas où l'utilisateur qui consulte cette vue est administrateur et que l'évènement est terminé, celui va pouvoir accéder à des checkbox cachées (voir ci-dessus). L'Administrateur pourra donc notifier au système les participants qui sont venus, et ceux qui se sont désistés. Cela impactera la fiabilité des utilisateurs.

Règles techniques :

- Pour pouvoir accéder aux checkbox pour noter la fiabilité, l'utilisateur doit être administrateur et l'évènement doit être terminé.
- L'utilisateur devra cocher toutes les checkbox et une fois l'évènement noté, il ne sera plus accessible pour personne.

- Une notification sera envoyée à l'administrateur dès qu'il aura la possibilité de noter la fiabilité. Ce service est détaillé en section 5.3.
- Si l'utilisateur n'a pas de photo de profil, une image par défaut est affichée.
- Les utilisateurs sont affichés dizaines par dizaines, dès que le bas de la listview est atteint, les participants sont chargés s'il en reste encore. Cela permet d'éviter un temps de chargement trop long.

6.2.11 La vue Messagerie événement



Cette vue est accessible par les utilisateurs d'un événement via la vue Détails événement. Elle permet de communiquer aux autres participants de l'évènement.

L'utilisateur peut envoyer des messages via le TextView (« Write a message ») et peut les envoyer au groupe via le bouton situé en bas à droite de l'écran. Les utilisateurs de l'évènement voient les messages les plus récents en bas de l'écran, avec le nom de l'utilisateur qui a envoyé le message ainsi que l'heure et la photo.

Règles techniques :

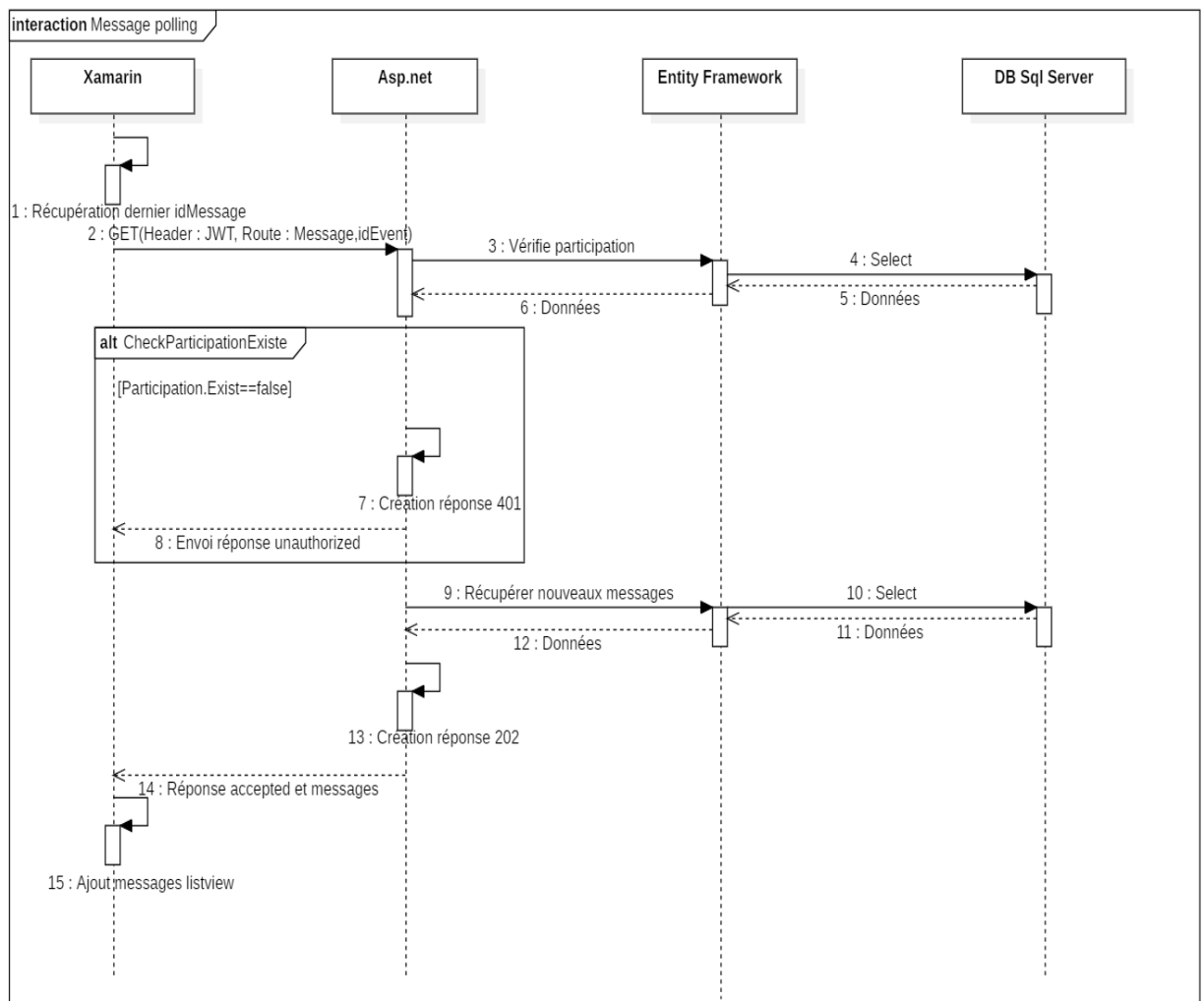
- Les messages sont triés dans l'ordre d'envoi.
- Les messages sont chargés par 50 pour limiter le temps de chargement. Une fois arrivé en haut de la messagerie, des nouveaux messages sont chargés s'il en reste.
- Les messages sont mis à jour dans le groupe de la messagerie si l'utilisateur se trouve sur la vue en question et qu'il y a des nouveaux messages qui ont été envoyés (explication détaillée ci-dessous).

Mise à jour des messages : Polling

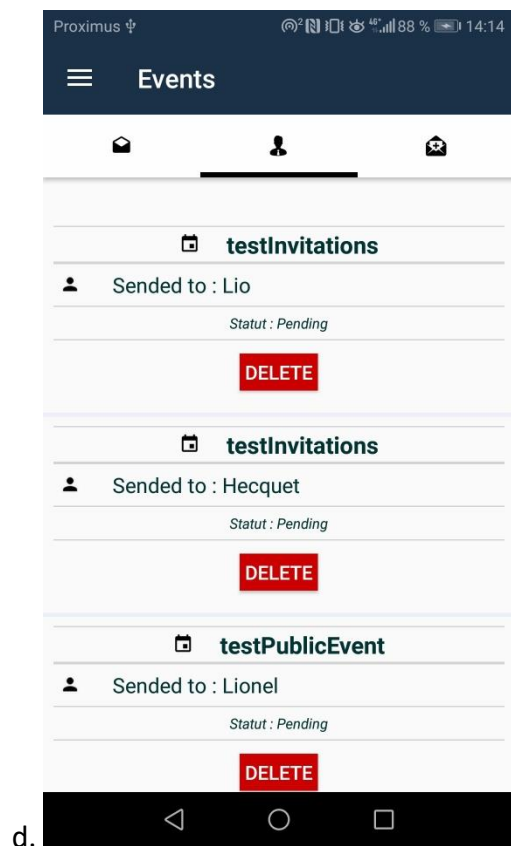
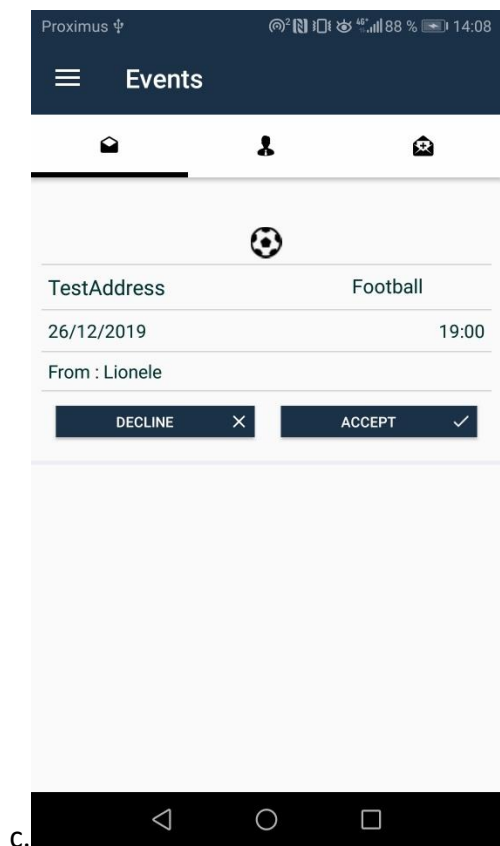
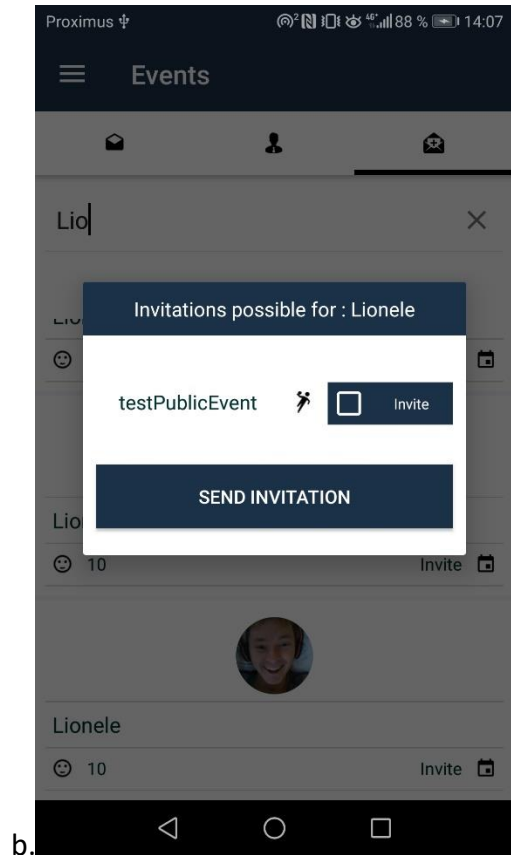
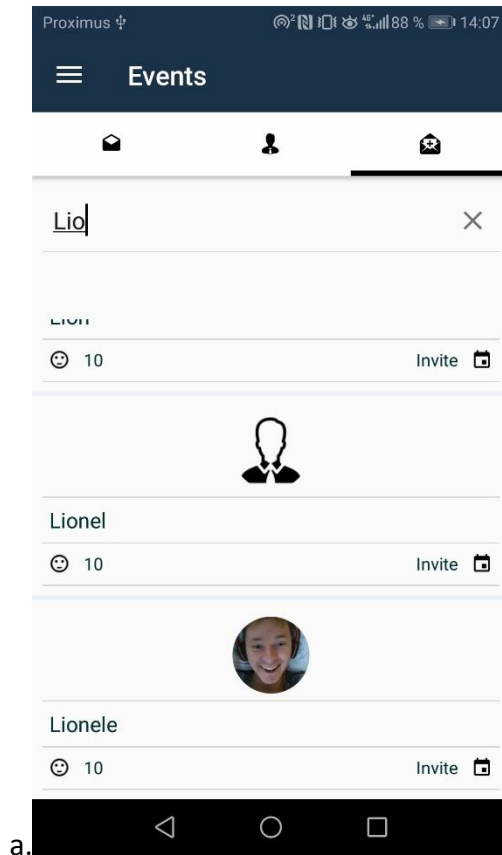
Lorsque l'utilisateur se trouve dans la vue messagerie, celui-ci doit être capable de recevoir les messages envoyés par d'autres utilisateurs. Pour ce faire, le polling va être utilisé.

Le polling (ou « attente active » en français) est une technique de programmation où l'on va interroger le serveur pour vérifier une condition, un état ou pour récupérer des données. Cette technique peut s'avérer très chronophage pour l'utilisation du CPU mais dans notre cas de figure, le polling ne sera effectué que quand l'utilisateur se trouve sur la vue, cela ne devrait donc pas poser trop de problèmes.

L'implémentation de notre polling s'effectuera dans un thread à part à l'aide de la classe BackgroundWorker mise à disposition par le framework .net. Le diagramme ci-dessous détaille l'implémentation :



6.2.12 La vue Invitations



La vue Invitations est composée de trois écrans :

- Envoyer des invitations (a + b)
- Recevoir des invitations (c)
- Gérer les invitations envoyées (d)

La vue (a+b) permet de rechercher des noms d'utilisateurs pour leur envoyer une invitation à un évènement que l'utilisateur administre.

La vue (c) permet de voir les invitations reçues. En cliquant sur l'invitation, elle nous emmène à la vue Détails Evènement. Elle permet également d'accepter ou de refuser l'invitation. Si l'invitation est acceptée, une participation est directement ajoutée à l'évènement (s'il reste une place). Si l'invitation est refusée, celle-ci est automatiquement supprimée.

La vue (d) permet de gérer les invitations envoyées pour un évènement et donc de les supprimer si l'administrateur a assez de participants ou a changé d'avis.

6.3 Services Android

6.3.1 Qu'est-ce qu'un service Android ?

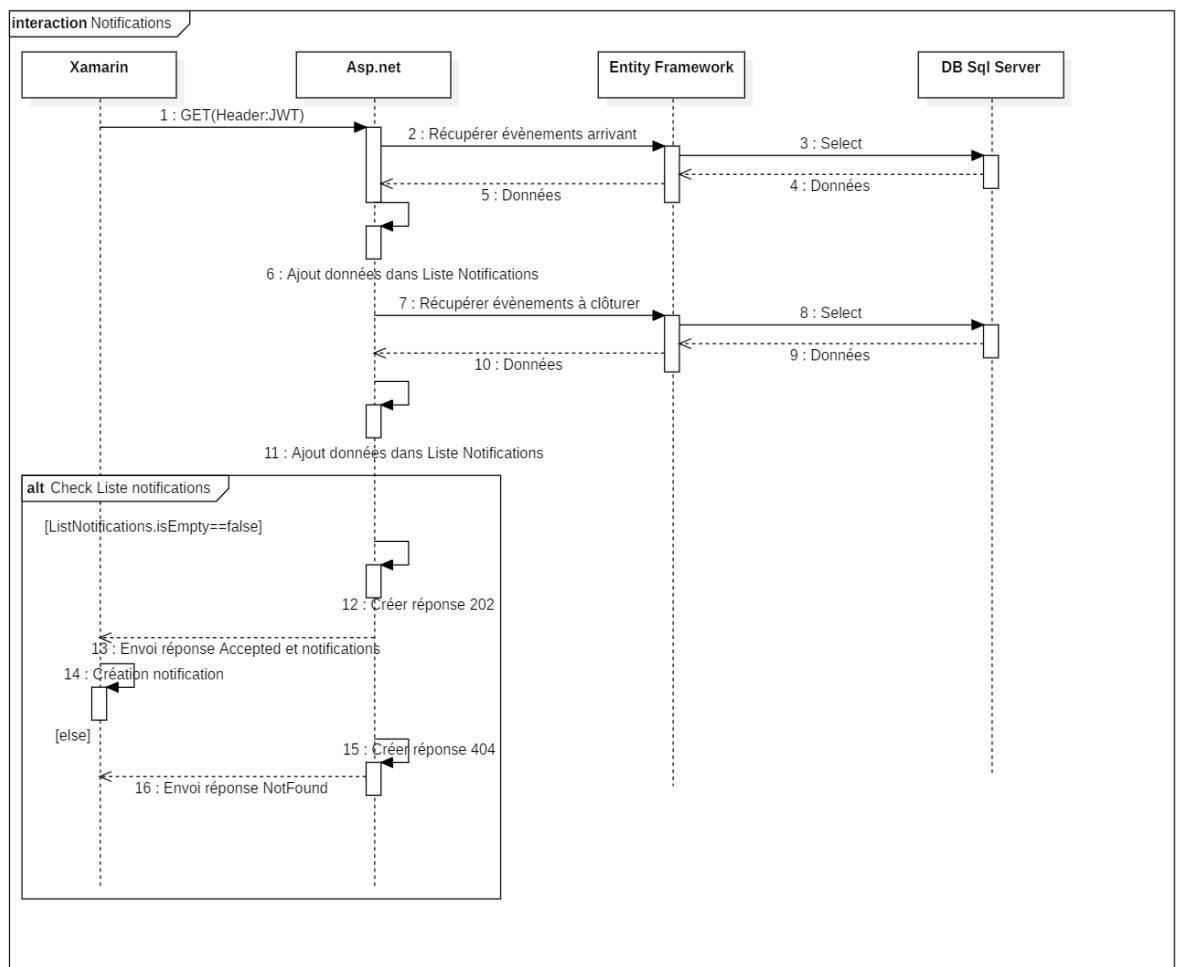
Un service est un composant Android (Thread indépendant de l'utilisateur) qui s'exécute en arrière-plan pour des opérations de longue durée. Il continue de tourner même si l'application est détruite.

6.3.2 Implémentation d'un service

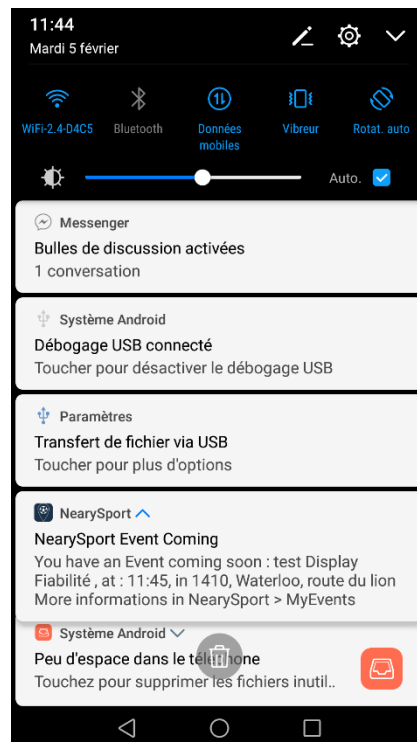
Un service a été implémenté. Ce dernier se lance périodiquement avec un intervalle d'une heure. Il est chargé de notifier l'utilisateur quand celui-ci a un évènement prévu dans l'heure et quand l'utilisateur doit clôturer un évènement qu'il a administré.

La vérification de nouveautés à afficher s'appuie sur une sécurisation plus « light » que l'authentification à l'App : un autre JWT, avec une durée de vie plus longue ; cela ne présente pas un souci puisque les données en output de l'API consultée par le service de notification ne renvoient que du texte.

Un diagramme de séquences ci-dessous décrit le fonctionnement détaillé de ce service.



La figure ci-dessous montre une notification envoyée à l'utilisateur :



Si l'utilisateur décide de cliquer sur la notification, il sera redirigé vers la page principale de l'application (la vue Connexion).

La notification ci-dessus montre un événement à venir, dans le cas d'un événement à clôturer, l'administrateur de l'évènement recevra des notifications chaque heure tant que celui-ci n'aura pas clôturer l'évènement correctement.

6.4 Permissions

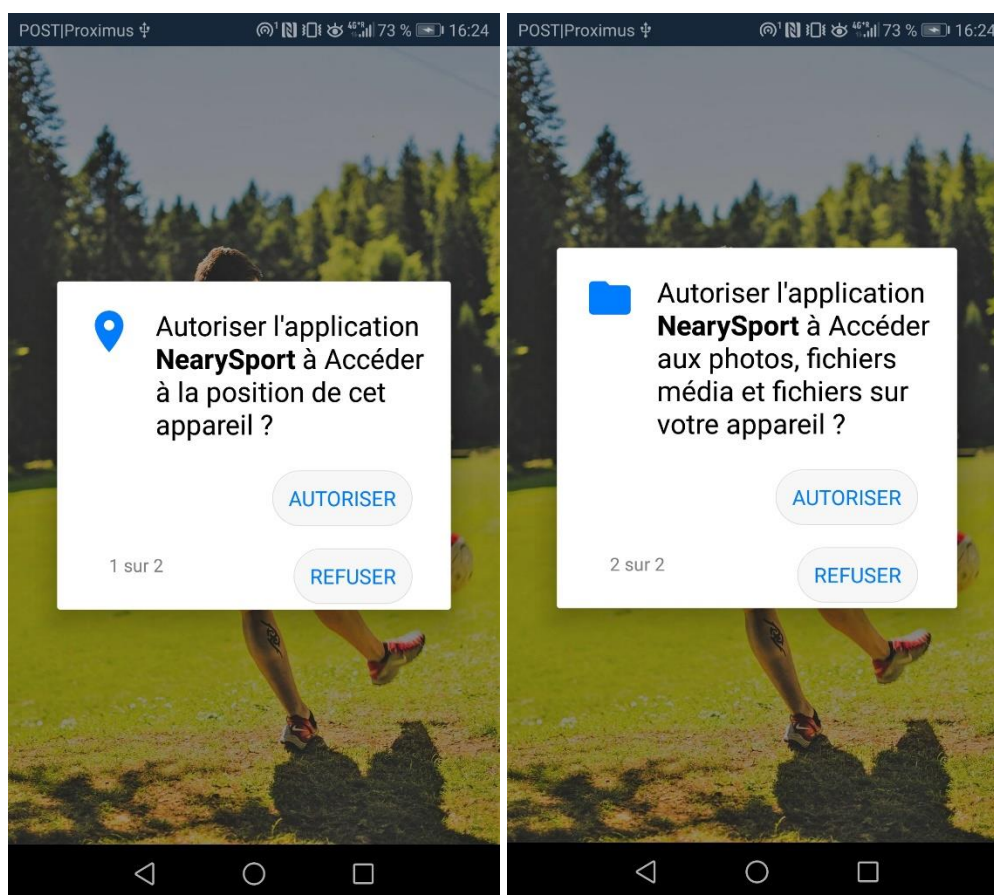
Pour pouvoir utiliser certaines fonctionnalités Android, une application a parfois besoin de demander des permissions à l'utilisateur pour pouvoir utiliser ces fonctionnalités.

Pour ce faire, il faut lister les permissions requises dans le manifest Android. Il faut ensuite demander l'approbation à l'utilisateur.

Pour pouvoir fonctionner correctement, l'application a besoin de l'approbation de l'utilisateur pour :

- Utiliser la géolocalisation de l'appareil : Pour chercher les événements dans les alentours de l'utilisateur.
- Lire le contenu du téléphone : Pour pouvoir changer la photo de profil de l'utilisateur.

Ces permissions sont demandées au démarrage de l'application :

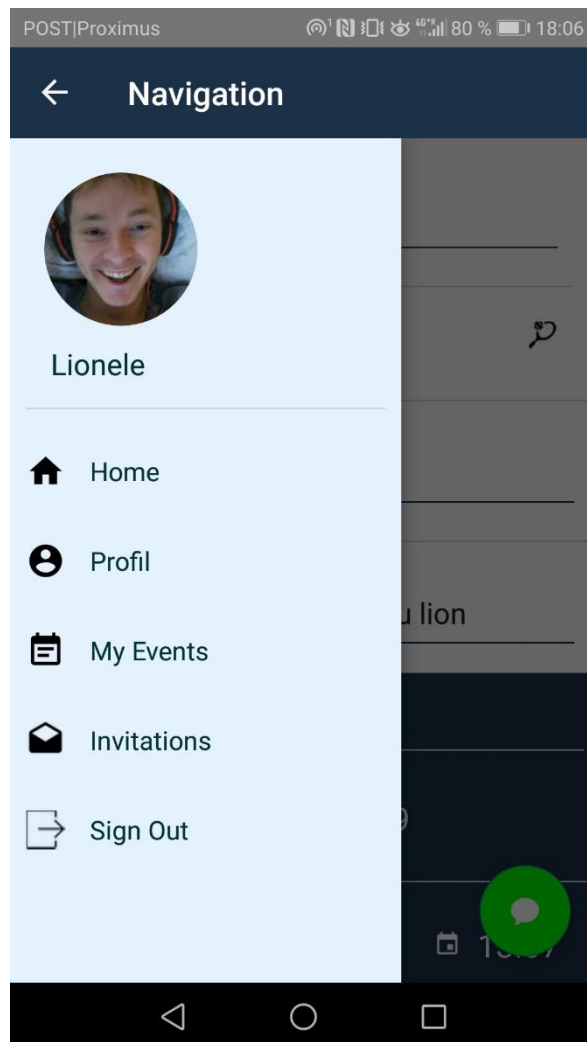


Si la première autorisation est refusée, l'application se ferme avec un message approprié pour expliquer à l'utilisateur pourquoi la permission est requise. Si la deuxième autorisation est refusée, elle sera demandée à nouveau si l'utilisateur veut pouvoir changer sa photo de profil par la suite.

6.5 Image de profil

L'utilisateur a la possibilité de changer sa photo de profil pour que celle-ci soit visible par les autres utilisateurs et que l'on puisse plus facilement l'identifier. Au début de la création du compte, il s'en voit attribuer une par défaut (stockée dans l'application).

Pour modifier sa photo de profil, l'utilisateur doit avoir un compte valide et être connecté, il doit ensuite se rendre sur une vue qui possède un menu de navigation sur la gauche (fil d'actualité, mon profil, ...). Il doit ensuite cliquer sur l'image de profil, si la permission n'a pas été acceptée, elle sera demandée. Une fois cette permission acceptée, il sera redirigé vers le contenu de son téléphone où il pourra valider une photo choisie par ses soins. Ci-dessous, un exemple de photo de profil utilisateur :



Chapitre VII : Futures possibilités de développement

Dans le cadre de ce TFE, le développement est axé sur la partie Android de l'application avec un scope bien défini. Cela n'empêche pas des futures possibilités de développement, ci-dessous, une liste non exhaustive de ces développements.

7.1 Développement IOS

L'application a été développée en Android dans un premier temps. Etant donné que l'utilisation de Xamarin permet de partager le back-end, le développement de l'application en Xamarin IOS sera plus rapide.

Le seul « souci » majeur du développement ios (Xamarin ou Objectif-c), c'est l'obligation d'avoir un mac pour le développement de l'application et un Iphone pour tester l'application.

Une autre possibilité est de louer des machines mac dans le cloud juste pour le temps du développement.

7.2 Développement Web

Quelle que soit la technologie front-end utilisée (Angular Js, React js ou même JQuery), le développement Web pourra également partager une grosse partie du back-end utilisé par les applications mobiles.

7.3 Multi-Langues

L'application a été développée en anglais, car celle-ci est généralement parlée par une grande majorité de personnes dans le monde. Mais il est toujours préférable d'avoir l'application disponible dans la langue maternelle de l'utilisateur.

Microsoft met notamment à disposition un outil « Multilingual App Toolkit » (Qui est disponible en tant que Nuget dans Visual Studio) et qui permet de traduire du contenu dans de nombreux langages.

7.4 Partage d'évènements

Pour pouvoir toucher un plus grand public, de nombreuses applications permettent à leurs utilisateurs de partager le contenu de l'app avec des applications extérieures (typiquement WhatsApp, Facebook, Gmail, Messenger).

C'est une fonctionnalité qui pourrait s'avérer intéressante et qui pourrait ramener plus de participants pour un évènement.

7.5 Notions « d'amitié »

Le rajout d'une fonctionnalité permettant de rajouter des « amis » et de voir les évènements auxquels ceux-ci participent, ou pouvoir directement envoyer des invitations à un évènement personnel à ceux-ci me paraît être intéressant et un plus pour l'application.

7.6 Messagerie individuelle

Pour le moment, l'application permet seulement d'interagir avec des personnes inscrites au même évènement que nous. L'ajout d'une messagerie privée entre utilisateurs serait également un plus pour l'application.

7.7 OAuth2

Certains utilisateurs n'aiment pas devoir se créer un compte sur une nouvelle application. Certains par manque de confiance envers l'application, d'autres par fainéantise.

Beaucoup d'applications et de sites web permettent donc de s'authentifier via un « Third party » comme Google, Facebook, Twitter, ...

C'est un plus pour l'application et Microsoft met à disposition des développeurs un système facilitant cette approche via un outil nommé Xamarin.Auth, le schéma ci-dessous reprend le fonctionnement général.

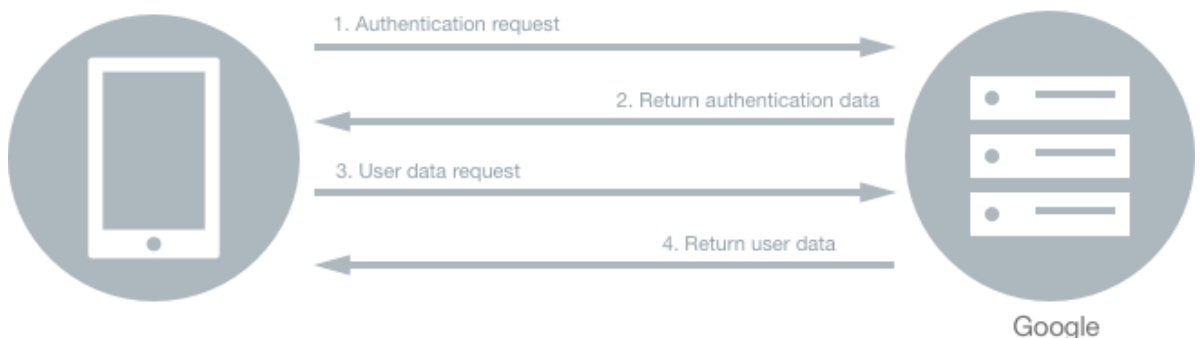


Tableau flow Xamarin.Auth : <https://docs.microsoft.com/en-us/xamarin/xamarin-forms/data-cloud/authentication/oauth>

Chapitre VIII : Conclusion

Avec l'élaboration de travail de fin d'études, j'ai voulu proposer une application offrant la possibilité d'organiser des événements sportifs avec des amis ou avec des inconnus. Bien que de nombreuses autres idées qui me sont passées par la tête n'aient pas eu le temps d'être développées, je pense que l'application soumise offre une bonne base et permet de réaliser l'essentiel et l'objectif de base qu'était l'organisation d'événements sportifs avec d'autres utilisateurs de l'application.

Une fois ce travail terminé, je me rends compte que les possibilités de développement sont énormes. Si le temps me le permet, je pense continuer à travailler sur ce projet et notamment la partie Apple, pour éventuellement le publier sur les stores Android et Apple.

Outre le semblant de fierté que j'éprouve après avoir développé ma première application « full-stack » Android, ce travail de développement et d'écriture, m'a permis d'apprendre et d'améliorer ma maîtrise de nombreuses technologies. En effet, avant la rédaction de ce TFE, je ne connaissais Xamarin que de nom et je ne m'étais encore jamais plongé dans le développement Android, c'est maintenant chose faite et je pense désormais en maîtriser les bases. J'ai également pu améliorer mes connaissances en Sql Server et Asp.net, les projets précédemment réalisés à l'aide de ces technologies n'étaient pas d'aussi grande envergure, ni aussi poussés.

Bibliographie

Pour la rédaction de ce TFE, de nombreux sites ont été consultés pour me renseigner un maximum. Voici la liste des articles consultés :

Anderson, Rick. Dykstra, Tom. Smith, Steve. *Filters in Asp.net Core*. Mis à jour le 02/08/2019 [En ligne]. <https://docs.microsoft.com/en-us/aspnet/core/mvc/controllers/filters?view=aspnetcore-2.2> [Consulté le 3 avril 2019]

BAILLY, Michael. *Les Web Services, Pourquoi ? Comment ?* Mis à jour le 5/10/16 [En ligne]. <https://blog.linagora.com/developpement-et-exploitation-de-web-services-rest/> [Consulté le 25 mars 2019]

BROWN, Philippe. *Why do we need to hash and salt passwords ?*. Mise à jour le 21 janvier 2013. [En ligne]. <https://culttt.com/2013/01/21/why-do-you-need-to-salt-and-hash-passwords/> [consulté le 30 mars 2019].

BURNS, Amy. *Introduction au développement mobile*. Mis à jour le 28/03/2017 [En ligne] <https://docs.microsoft.com/fr-fr/xamarin/cross-platform/get-started/introduction-to-mobile-development> [Consulté le 3 avril 2019]

G, Jérôme. *MD5 obsolète*. Mis à jour le 31 décembre 2008 [En ligne] <https://www.generation-nt.com/microsoft-hachage-md5-securite-actualite-209801.html> [Consulté le 3 avril 2019]

HERPIN, Julien. *Introduction aux Json Web tokens*. [En ligne] <https://www.ekino.com/articles/introduction-aux-json-web-tokens> [Consulté le 3 avril 2019]

LOUVIAUX, Alain. *L'incroyable succès de « Je cours pour ma forme ». 25.820 inscrits en 2016 en Belgique*. Mis à jour le 22 avril 2017 [En ligne]. <https://tinlot.blogs.sudinfo.be/archive/2017/04/21/l-incroyable-succes-de-je-cours-pour-ma-forme-25-820-inscri-222543.html> [Consulté le 23 mars 2019]

REESE, Emily. *Utilisez des API REST dans vos projets Web*. Mis à jour le 09/07/2018 [En ligne]. <https://openclassrooms.com/fr/courses/3449001-utilisez-des-api-rest-dans-vos-projets-web/3501901-pourquoi-rest> [Consulté le 29 mars 2019]

ROMEI, Christophe. *20 chiffres sur le marché mobile à connaître en 2018*. Mis à jour le lundi 8 janvier 2018. [En ligne]. <https://www.servicesmobiles.fr/20-chiffres-sur-le-marche-mobile-a-connaître-en-2018-38749/> [Consulté le 23 mars 2019].

SEGUY, Mathias. *Apprendre à utiliser les services Android*. Mis à jour le 21 novembre 2016 [En ligne] <https://mathias-seguy.developpez.com/tutoriels/android/utiliser-services/> [Consulté le 20 mars 2019]

TAFFOREAU, Jean. *Enquête de santé par interview, la pratique d'activités physiques*. Mis à jour 2008 [En ligne] https://www.wiv-isp.be/epidemiologie/epifr/crospfr/hisfr/his08fr/r2/3.la_pratique_dactivites_physiques_r2.pdf [Consulté le 23 mars 2019]

WASSON, Mike. *Authentication Filters in Asp.net Web API 2*. Mis à jour le 25/09/2014 [En ligne]. <https://docs.microsoft.com/en-us/aspnet/web-api/overview/security/authentication-filters> [Consulté le 3 avril 2019]

WISELEY, Matt. *What is Asp.Net and Why should I use It ?* Mis à jour le 27/11/2018 [En ligne]. <https://www.wakefly.com/blog/what-is-asp-net-and-why-should-i-use-it/> [Consulté le 2 avril 2019]

/ .Bienfaits généraux de l'activité physique Mis à jour le 20 avril 2015 [En ligne]. <http://sante.lefigaro.fr/mieux-etre/sports-activites-physiques/bienfaits-generaux-lactivite-physique/quels-sont-benefices> [Consulté le 23 mars 2019]

/ . Comparaison cross-platform frameworks. / [En ligne] https://www.altexsoft.com/media/2018/02/Xamarin-vs-Ionic-vs-React_main-1.png [Consulté le 25 mars 2019]

/ . Compilation à la volée. / [En ligne] https://fr.wikipedia.org/wiki/Compilation_%C3%A0_la_vol%C3%A9e [Consulté le 3 avril 2019]

/ . MD5 . [En ligne] <https://fr.wikipedia.org/wiki/MD5#Annexes> [Consulté le 2 avril 2019]

/ . Tutoriel Service Android [En ligne] <https://o7planning.org/fr/10421/le-tutoriel-de-android-service> [Consulté le 20 mars 2019]

Annexes

Annexe I : Documentation Swagger

Event			Show/Hide List Operations Expand Operations
GET	/api/Events	Return each events present in the table event, group by sport	
POST	/api/Events	Add New Event	
PUT	/api/Events	Update an existing event	
GET	/api/Events/Around	Return events around the user	
GET	/api/Events/{idEvent}/Participation	Get every participations in the Event	
GET	/api/Events/{idEvent}/Participation/{offset}	Load 10 participations from the events	
DELETE	/api/Events/{id}	Delete an Event by id	
GET	/api/Events/{id}	Get Event by id	
ForgotPassword			Show/Hide List Operations Expand Operations
POST	/api/ForgotPassword/{email}	Add new Guid in forgotPasswordTable	
GET	/api/ForgotPassword/{guid}	Get guid by id in forgotPasswordTable	
Image			Show/Hide List Operations Expand Operations
DELETE	/api/Image	Delete Profil Image	
GET	/api/Image	Get Profil Image	
PUT	/api/Image	Update profil Image	
GET	/api/Image/{id}	Get Profil Image for a givenUserId	
Invitation			Show/Hide List Operations Expand Operations
GET	/api/Invitations/{idEvent}	Get Invitation for an Event	
DELETE	/api/Invitations	Delete an Invitation for a given Event, userID	
GET	/api/Invitations	Get Invitation for the user that make the API call	
POST	/api/Invitations	Add a new Invitation for a given Event, userID	
PUT	/api/Invitations	Update the statut of an Invitation for a given Event, userID	
Login			Show/Hide List Operations Expand Operations
POST	/api/Login/CreateProfil	Add new user	
DELETE	/api/Login	Delete an existing user	
GET	/api/Login	Get user and return JWT	
PUT	/api/Login	Update an existing password	
PUT	/api/Login/CreateNewPassword	Update a existing password when the previous has been lost	
MessageGroupEvent			Show/Hide List Operations Expand Operations
POST	/api/MessagesGroup	Add new GroupMessage	
GET	/api/MessagesGroup/{idEvent}	Return each message for an Event	
GET	/api/MessagesGroup/Async/{idEvent}/{idMessage}	Return each message for an Event	
GET	/api/MessagesGroup/{idEvent}/{idMessage}	Return each message for an Event	

Messages

[Show/Hide](#) | [List Operations](#) | [Expand Operations](#)

POST	/api/Messages	Create new Message
------	---------------	--------------------

Notification

[Show/Hide](#) | [List Operations](#) | [Expand Operations](#)

GET	/api/Notification	Get User's Notifications
-----	-------------------	--------------------------

Participations

[Show/Hide](#) | [List Operations](#) | [Expand Operations](#)

GET	/api/MyParticipations	Get User's Events
-----	-----------------------	-------------------

POST	/api/MyParticipations	Add a new Participation
------	-----------------------	-------------------------

GET	/api/MyParticipations/{idEvent}	Get User's Role in Event by id
-----	---------------------------------	--------------------------------

PUT	/api/MyParticipations/{idEvent}	Update the statut of a participation
-----	---------------------------------	--------------------------------------

DELETE	/api/MyParticipations/{id}	Delete an existing participation
--------	----------------------------	----------------------------------

Profil

[Show/Hide](#) | [List Operations](#) | [Expand Operations](#)

DELETE	/api/Profil	Delete an existing profil
--------	-------------	---------------------------

GET	/api/Profil	Get User's Profil by JWT
-----	-------------	--------------------------

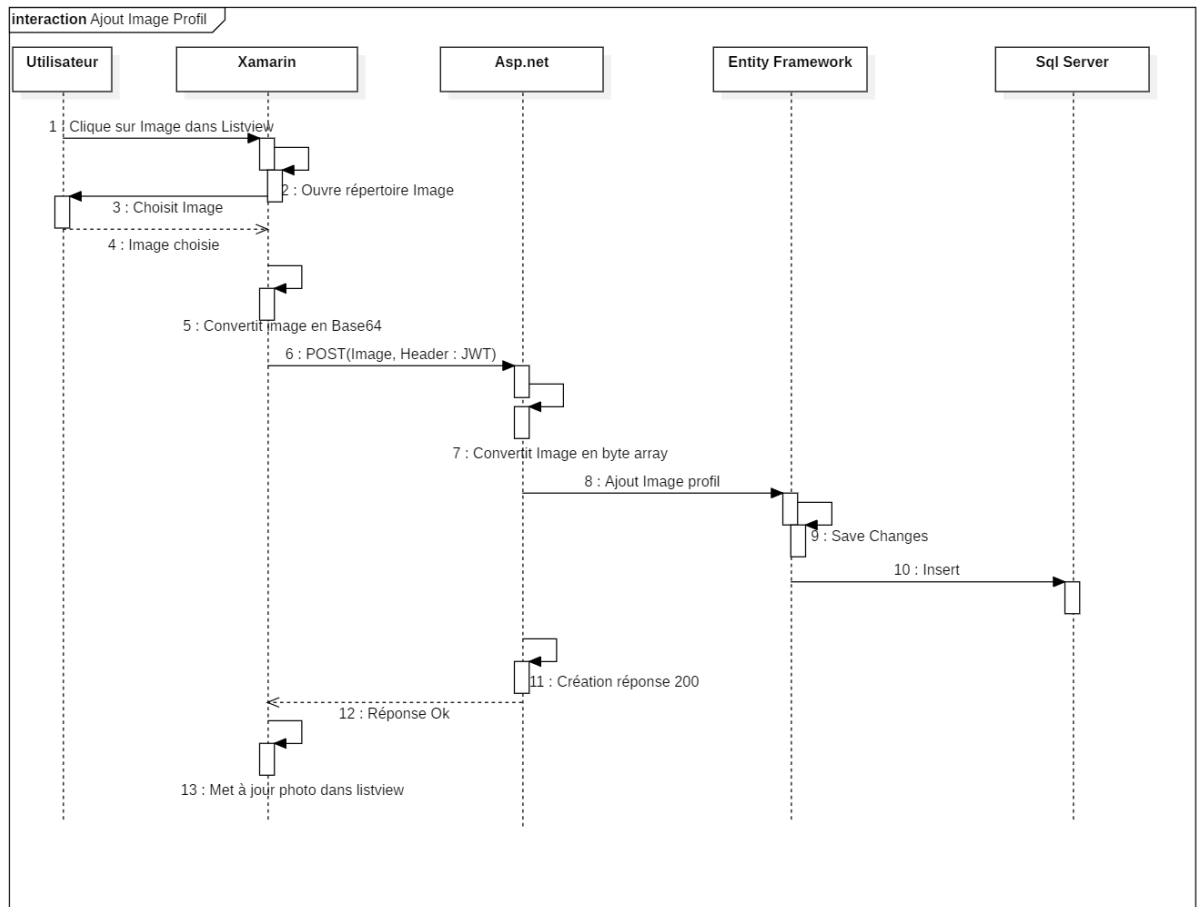
POST	/api/Profil	Add a new Profil
------	-------------	------------------

PUT	/api/Profil	Update an existing profil
-----	-------------	---------------------------

GET	/api/Profil/{username}	Get user's profil with a matching username
-----	------------------------	--

GET	/api/Profil/5/{username}	Get first user's profil with a matching username
-----	--------------------------	--

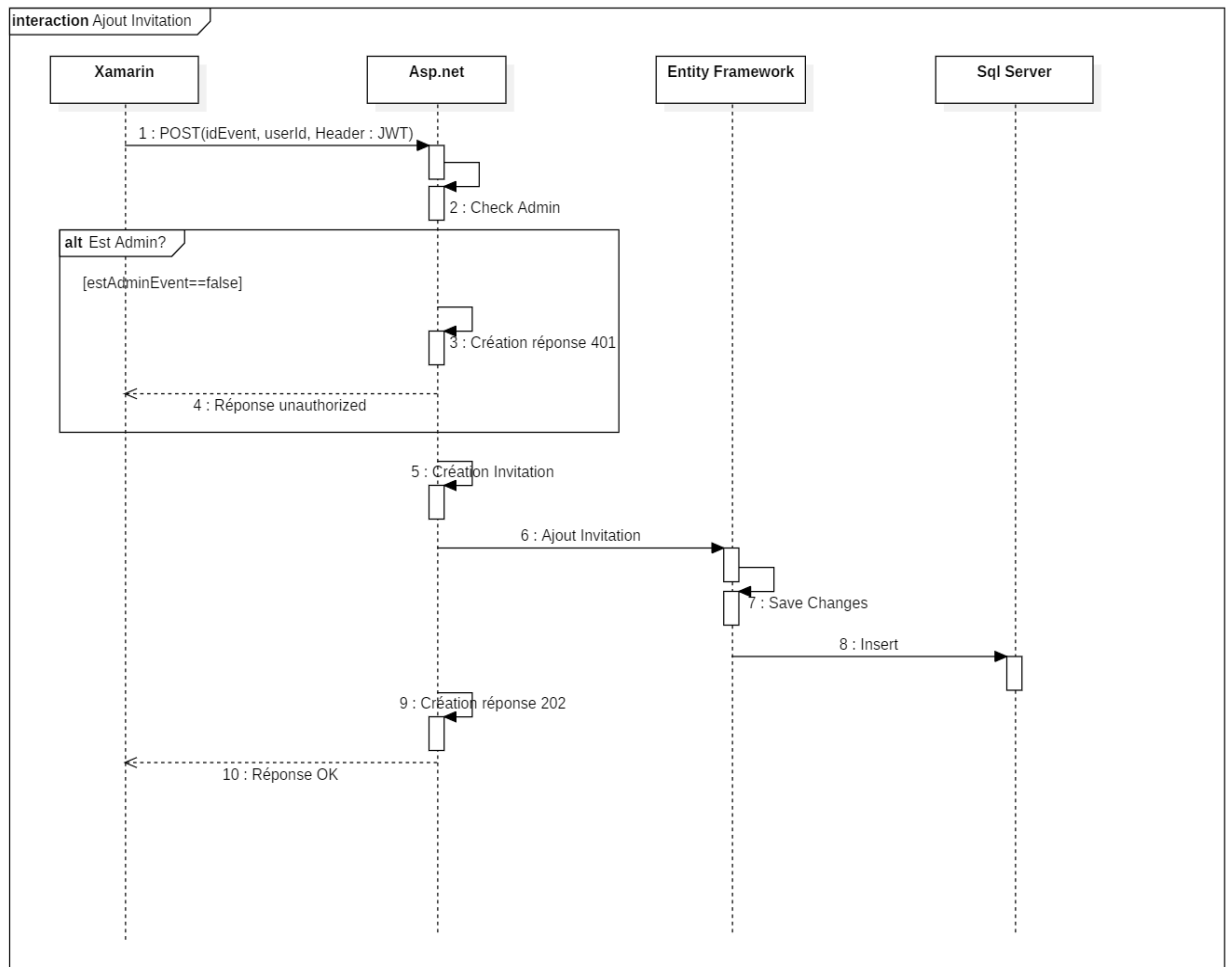
Annexe II : Diagramme de séquences : Image de profil



Le processus ci-dessus représente l'ajout d'une nouvelle image de profil pour un utilisateur :

- Lorsque l'utilisateur clique sur une l'image de profil présente dans la listview de navigation, Android ouvre donc le répertoire Images du téléphone de l'utilisateur.
- L'utilisateur choisit une image présente dans son répertoire.
- L'image choisie est convertie en une représentation base64.
- Une requête est envoyée.
- L'image envoyée dans la requête est convertie en byte array et insérée dans la base de données.

Annexe III : Invitation



Le processus ci-dessus décrit l'ajout d'une invitation à un évènement :

- Une requête est envoyée de la partie mobile avec le userId de la personne invitée et l'id de l'évènement auquel il est invité.
- On vérifie que celui qui a envoyé l'invitation est bien l'administrateur de l'évènement.
- Si ce n'est pas l'administrateur, une réponse unauthorized est renvoyée.
- L'invitation est créée.
- L'invitation est ajoutée dans la base de données.
- Une réponse 200 est renvoyée à l'utilisateur.