

RAPPORT D'AVANCEMENT

LAST NIGHT

GROUPE 4.I



ROBIN LOUIS LEELLO MICHAEL DANIEL LUCIE PAUL-ELIAN
BLANCHARD COMBALDIEU DADI DASSEVILLE MAYAU MOULINAS TABARANT

TUTEUR : JEAN LEFEUVRE

ENCADRANT GL : SOUMIA DERMOUCHE

TELECOM
ParisTech



Sommaire

1	<u>RESUME DU SUJET CHOISI EN FRANÇAIS</u>	3
2	<u>ENGLISH SUMMARY</u>	4
3	<u>ÉTUDE D'ANTERIORITE ET JUSTIFICATION DE LA PROPOSITION</u>	5
3.1	<u>DESCRIPTION DE LA PROPOSITION</u>	5
3.2	<u>DESCRIPTION DE L'ETAT DE L'ART</u>	5
4	<u>SCENARIOS D'USAGE</u>	8
5	<u>ARCHITECTURE DU PROJET</u>	10
5.1	<u>SCHEMA D'ARCHITECTURE</u>	10
5.2	<u>DESCRIPTION DES INTERFACES</u>	12
5.2	<u>DIAGRAMME D'ACTIVITE</u>	14
5.3	<u>INTERFACE UTILISATEUR GRAPHIQUE</u>	15
5.4	<u>TABLEAU DETAILLE DES TACHES</u>	17
6	<u>ORGANISATION DU PROJET</u>	18
6.1	<u>DIAGRAMME DE PLANIFICATION TEMPOREL DES TACHES</u>	18
6.2	<u>REPARTITION DES ELEVES PAR MODULE</u>	19
6.3	<u>PLANS DE TEST</u>	19
6.4	<u>DIAGRAMME D'AVANCEMENT DES TACHES</u>	27
7	<u>BIBLIOGRAPHIE</u>	29
ANNEXE A.	<u>FICHE D'IDENTITE DU GROUPE</u>	31
ANNEXE B.	<u>MODIFICATIONS</u>	32
B.1.	<u>MODIFICATIONS DE FOND</u>	32
B.2.	<u>MODIFICATIONS DU RAPPORT</u>	32
B.2.1.	<u>MODIFICATIONS DU RAPPORT AU PAN3</u>	32
B.2.2.	<u>MODIFICATIONS DU RAPPORT AU PAN4</u>	36
ANNEXE C.	<u>COMPTE RENDU DE REUNIONS</u>	37
ANNEXE D.	<u>SUIVIS DES MODULES</u>	41

1 Résumé du sujet choisi en français

Notre sujet prend source dans le contexte festif des soirées dansantes. L'idée est de concevoir une plateforme en ligne capable de générer un récapitulatif de chaque soirée à laquelle l'utilisateur a participé. LastNight utilise pour cela la source quasi-commune à tous les invités, les téléphones portables: capteurs de mouvement et appareils photo sont mis à contribution pour retracer au mieux le déroulement de la soirée au niveau d'un serveur. Parce que bonne soirée rime avec musique adaptée et bonne organisation, LastNight permet également au DJ et aux organisateurs d'interagir avec les participants via l'application par envoi de sondages et messages divers concernant la soirée. En associant les données récupérées sur l'ensemble des téléphones des invités aux musiques jouées par le DJ, le but est d'apporter des informations pertinentes à l'utilisateur dans son récapitulatif : la comparaison de ses mouvements au tempo de la musique servira à estimer à quels moments de la soirée il a dansé, avec quelles personnes et à lui conseiller des titres à écouter. Les photos partagées sur la plateforme par les invités sont regroupées sur un axe temporel résument la soirée et ajoutées à ce compte-rendu personnalisé.

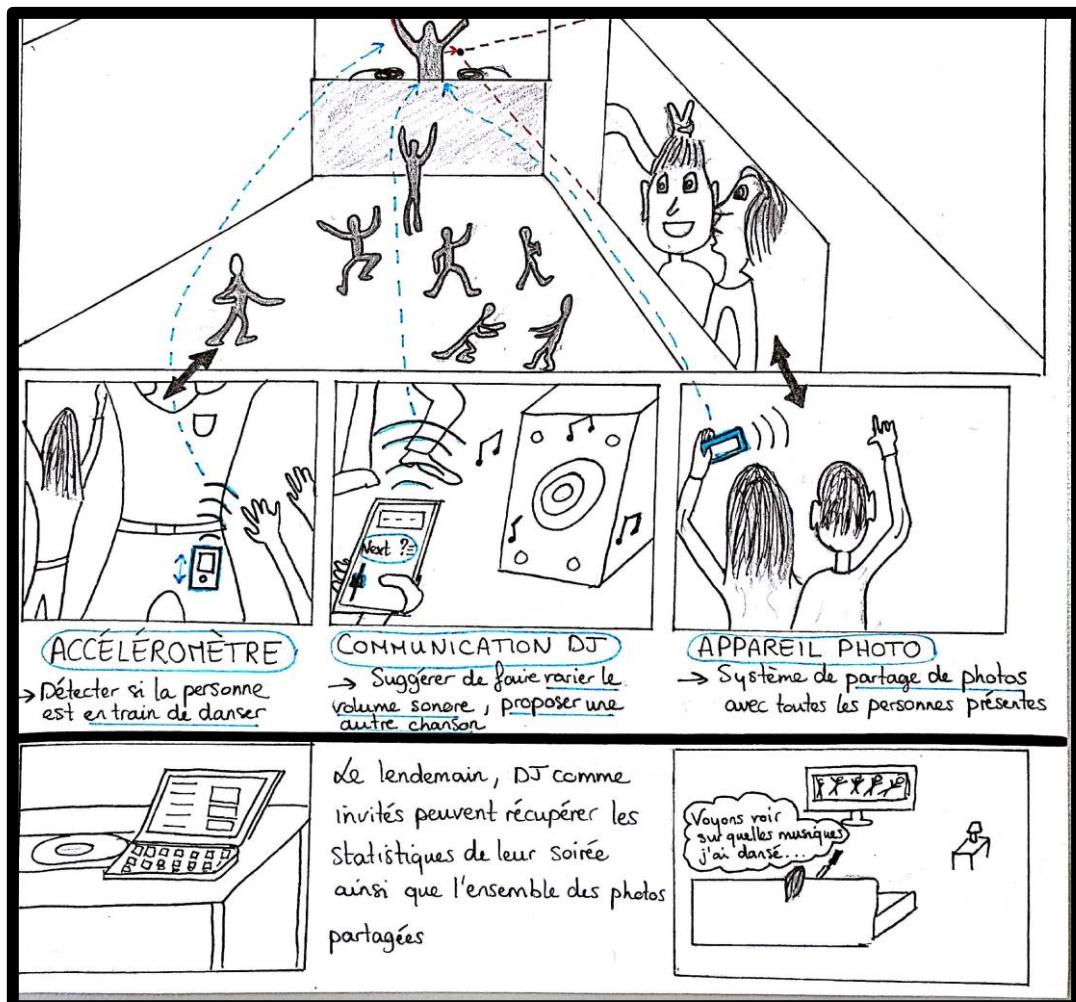


Figure 1 : Mise en scène de l'application

2 English Summary

Our project is an answer to the nostalgic and forgetful partygoer's call. We aim to provide a platform that can render personalized recaps of the parties our users have attended. To achieve this noble goal, LastNight takes advantage of the prevalence of smartphones in our society: we gather data from phone sensors on our servers, as well as submitted pictures, to generate a solid timeline for our users to better remember an eventful night. But for a party to be worth remembering, good organization and well-tailored music are essential, so we also facilitate communication between the DJ and his audience through features that allow him to poll his listeners and allow organizers to send push notifications to attendees. Our goal is to give our users as much useful information as possible in our recaps. By combining the smartphone data we gathered from a specific user with the DJ's playlist, we're able to determine, through rhythmic analysis, the times at which he danced the most, the people he danced along and suggest titles he might like. We add to that time-stamped pictures from fellow users at the same event to complete the night's timeline.

3 Étude d'antériorité et justification de la proposition

3.1 Description de la proposition

Les soirées musicales sont des moments particuliers où l'on partage et reçoit une grande quantité d'informations. Au final, nous n'en retenons qu'une petite partie dont le souvenir s'atténue avec le temps. LastNight est une plateforme en ligne visant à centraliser des informations relatives aux soirées de l'utilisateur afin qu'il en conserve un souvenir impérissable et exhaustif.

D'une manière générale, les traces numériques de nos soirées se limitent souvent à un ensemble de photos dispersées sur les téléphones ou ordinateurs de nos amis ou dans le meilleur des cas sur Facebook. Non seulement il est contraignant de consulter l'ensemble de ces photos mais il peut être intéressant de retenir d'autres informations marquantes de la soirée. C'est pourquoi nous avons pensé à un système capable de savoir à quels moments de la soirée chaque utilisateur a dansé, connaître les musiques jouées par le DJ et tirer des conclusions sur l'ambiance générale.

Il nous a semblé pertinent de nous pencher aussi sur les ingrédients d'une soirée réussie : LastNight cherche également à mettre en relation le DJ (ou organisateurs) et les invités durant la soirée par l'intermédiaire de sondages et messages informatifs afin d'éviter l'insatisfaction.

Lorsque l'on doit se rendre à une soirée, il est assez courant de se demander si nos amis sont déjà arrivés ou non. Notre plateforme en ligne permet, par un système de scan de QR code à l'entrée de la soirée, de se tenir informé en direct de la présence de nos amis à l'événement. Cette solution permet aussi de découvrir des soirées dont on ignorait l'existence et où l'on peut rejoindre nos amis à l'improviste.

3.2 Description de l'état de l'art

Articles scientifiques

A Study on Human Activity Recognition Using Accelerometer Data from Smartphones Akram Bayat , Marc Pomplun, Duc A. Tran : Avec des données récoltées de 4 individus qui ont fait 7 activités différentes pendant $\leq 280s$, les auteurs de l'article ont été capables de classifier les données de l'accéléromètre avec une précision de 91%.

Activity Recognition using Cell Phone Accelerometers Jennifer R. Kwapisz, Gary M. Weiss, Samuel A. Moore : Les auteurs de cet article font la même chose que les précédents mais avec des catégories d'activité différentes. Ils qualifient leurs résultats de prometteurs : ils ont réussi à classifier les activités avec plus de 90% de précision en n'utilisant que des échantillons de 10 secondes.

Towards Rhythmic Analysis of Human Motion using AccelerationOnset Times, Eric Lee, Urs Enke and Jan Borchers, Leo de Jong : En s'inspirant des algorithmes qui analysent le rythme de la musique, ils ont développé un algorithme permettant de déterminer le rythme d'un mouvement en temps réel à partir des données d'un accéléromètre.

Sensing Dance Engagement for Collaborative Music Control, Michael Kuhn, Roger Wattenhofer Distributed Computing Group, Martin Wirz, Matthias Fluckiger, Gerhard Troster Wearable Computing Lab, ETH Zurich : Les chercheurs ont voulu déterminer une façon d'obtenir un feedback sur la musique jouée de manière non intrusive. Ils utilisent les données de l'accéléromètre pour mesurer 'l'engagement' des gens. Ils mesurent d'une part l'intensité de l'activité et d'autre part l'écart du rythme de cet activité avec celui de la musique. Un algorithme de classification permet de classifier les personnes. En supposant que la satisfaction avec la musique jouée est synonyme d'engagement des gens dans la danse, le feedback est obtenu.

Applications similaires

partify.club : Plugin Chrome + Spotify qui permet aux invités de collaborer pour créer une playlist de soirée.

Jukebox for Spotify, Party DJ, Qcast, OutLoud : Ces applications permettent aux invités de voter pour les morceaux qui vont suivre. Les titres avec plus de votes se déplacent et se jouent plus tôt. Les invités peuvent agrandir la file d'attente en ajoutant des morceaux de leur bibliothèque Spotify/iTunes ou de celle de l'organisateur.

Spring - Music for Running Walking : iOS Spring détecte la cadence de notre pas et adapte le rythme de la musique jouée.

tracktl.com : Ce service web + application permet aux organisateurs de soirées de créer une 'trackparty' qui est une page web personnalisée. Avec l'URL de la trackparty, les invités peuvent voter sur la musique, prendre des photos et faire des tweet qui seront affichés sur la page. Ils peuvent également accéder à l'historique pour récupérer la playlist entière ou télécharger une infographie qui retrace les meilleurs moments de la soirée.

LastNight possède plusieurs spécificités qui la distinguent des applications et services listés. *LastNight* est la seule application qui suit les mouvements des utilisateurs pour savoir s'ils dansent. C'est également la seule qui offre des fonctionnalités pour les organisateurs et les invités. Le DJ peut mesurer le succès de ses choix de musique en suivant le nombre de personnes qui dansent et les invités peuvent voter et influer sur plusieurs choses dont les choix de musique.

4 Scénarios d'usage

Le Gala de Télécom approche à grands pas ! Le BDE veut faciliter l'accès aux informations concernant cet événement et le promouvoir au maximum. Pour cela, il crée un événement détaillé sur Facebook et définit la liste des invités. En parallèle, il crée l'événement sur la toute nouvelle plateforme LastNight grâce à l'application du même nom, en spécifiant un lien vers la page Facebook et les heures de début et de fin. Un QR code est automatiquement généré. Celui-ci est imprimé et placé à l'entrée du complexe où se déroule la soirée. Le DJ, qui possède son propre compte LastNight, est déclaré par l'organisateur comme « administrateur » sur l'événement.

La soirée commence ! Un invité muni de l'application peut scanner le QR code à l'entrée et LastNight reconnaît la soirée correspondante. Sa présence à l'événement est alors consultable par tous ses amis inscrits sur la plateforme, avec la possibilité de paramétrier une notification du type « Votre ami Pierre est à la soirée <Lien Facebook de l'événement>, rejoignez-le ! » au moment de son arrivée.

Pendant toute la durée de la soirée, une section spécifique à l'événement est créée sur la plateforme LastNight grâce à laquelle le DJ et les invités ayant scanné le QR code pourront interagir. Le DJ, qui possède un statut spécial sur l'événement (administrateur), relie son PC à son compte LastNight via un utilitaire préalablement installé pour que le serveur récupère les musiques jouées en temps réel. En plein dilemme entre trois musiques, il peut par exemple poster un sondage sur la page pour une durée d'une dizaine de minutes et les invités votent pour la musique de leur choix sur leur application. Cette fonctionnalité d'envoi de messages / sondages permet au DJ d'orienter ses choix afin d'obtenir une satisfaction optimale.

Pour les personnes souhaitant garder leur téléphone dans leur poche lorsqu'ils dansent, le serveur est capable d'estimer si l'invité est en train de danser à tout instant grâce aux capteurs de mouvement de son téléphone et au tempo des musiques jouées par le DJ.

Enfin, tout invité peut s'il le souhaite partager des photos de la soirée vers la plateforme avec son application. Chacune d'elles est associée à son heure exacte de prise.

Après la soirée, l'utilisateur ayant participé à l'événement a accès à une page récapitulative. Il visualise une timeline personnalisée où sont superposés l'historique des musiques jouées par le DJ et son activité de danse à tout moment de la soirée. Le classement des musiques sur lesquelles il a dansé vient compléter ces informations, lui permettant de conserver leur titre pour les retrouver s'il ne les connaît pas.

L'ensemble des photos récupérées durant la soirée, préalablement triées par l'organisateur, sont repérées sur la timeline par leur instant t auquel elles ont été prises. L'invité peut donc replacer précisément dans le temps chaque événement particulier qu'il a pu vivre mais aussi ceux qu'il a ratés du fait que les participants ont partagé des photos depuis des positions différentes. Ainsi, il voit par exemple qu'il était en train de danser en rythme avec ses amis sur Beat It de Michael Jackson à 00:30 sur la piste de danse pendant qu'un anniversaire était célébré au niveau du bar.

LastNight stocke sur le compte de l'utilisateur les récapitulatifs de chaque événement auquel il a participé. Il peut les consulter et ainsi « revivre » ses meilleurs souvenirs lorsqu'il le souhaite.

5 Architecture du projet

5.1 Schéma d'architecture

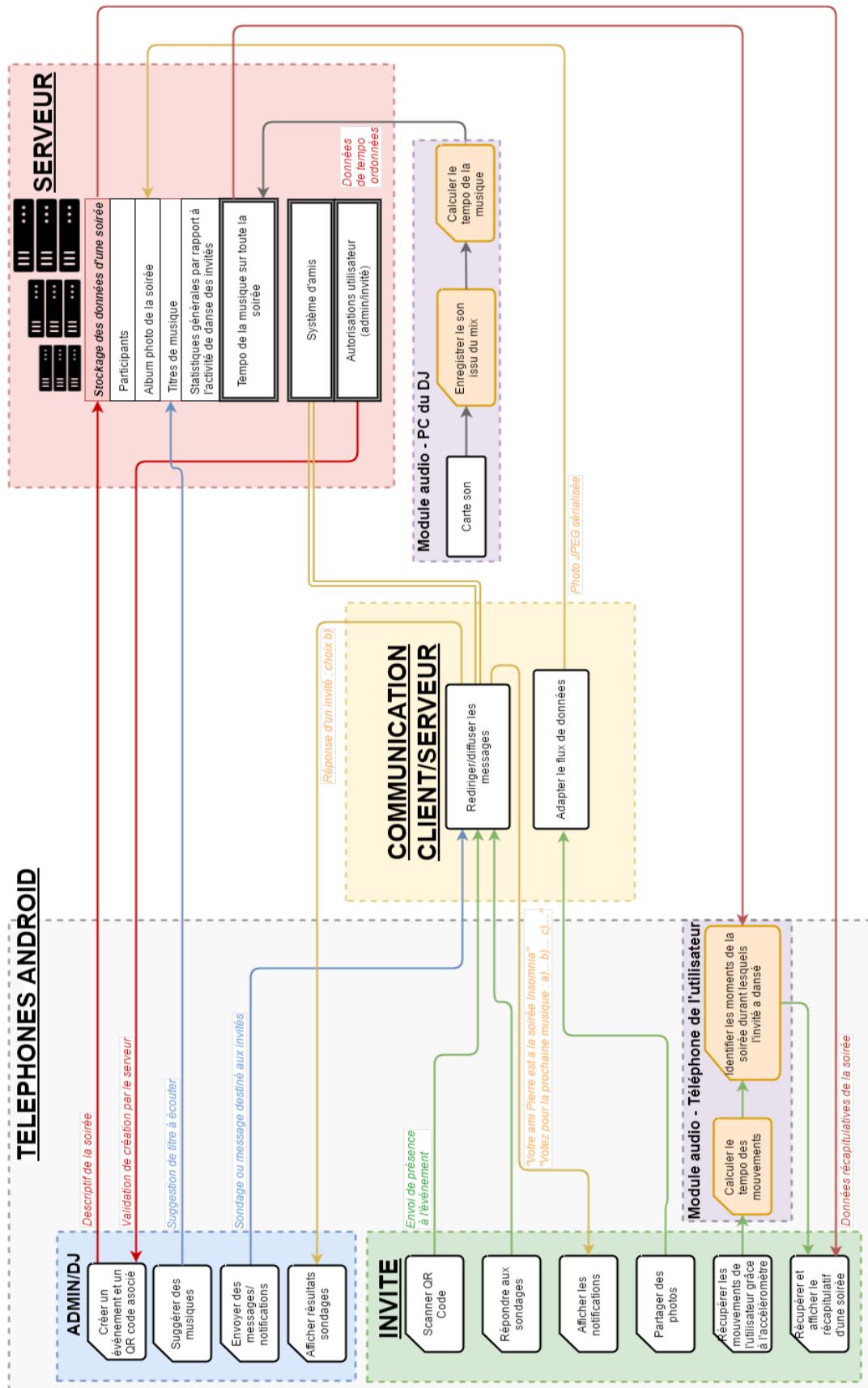


Figure 2 : Diagramme d'architecture

5.1.1 Android – Application globale et Timeline

Ce module se charge de toutes les fonctions utilisateur de LastNight. Son architecture se découpe en deux parties principales :

La première donne à un utilisateur les moyens d'organiser sa propre soirée sur la plateforme LastNight. L'application Android doit donc lui permettre d'entrer toutes les informations nécessaires à la présentation de sa soirée et sa création sur le serveur. A terme (PAN4), ce procédé doit lui permettre d'obtenir un QR Code identifiant l'événement, que les utilisateurs pourront scanner pour indiquer au serveur qu'ils ont rejoint la soirée. Pendant la soirée, des options sont réservées à l'organisateur et aux personnes déclarées administrateurs de la soirée. L'application doit leur donner la possibilité d'éditer des sondages et des messages afin de les proposer aux invités. Enfin, une fois la soirée terminée, une fonction de tri des photos constituant l'album de l'événement devrait être implémentée.

La deuxième partie concerne les utilisateurs invités à une soirée donnée. Pour rejoindre une soirée créée via l'application, ils bénéficieront (PAN4) d'une fonction de scan de QR code répondant à la fonction détaillée dans le paragraphe précédent. L'application doit alors proposer une interface permettant à l'utilisateur de visualiser le sondage en cours s'il y en a un, et visualiser les messages postés par les administrateurs de l'événement. Un service d'arrière-plan doit également suivre en permanence les mouvements de l'utilisateur à partir de l'accéléromètre de son téléphone, et stocker ces mesures dans un fichier, avec toutes les informations nécessaires pour estimer le tempo associé. A terme (PAN4), l'application devra proposer un outil de capture photo pour pouvoir les poster dans l'album de l'événement.

Enfin, lorsque l'utilisateur voudra consulter les informations relatives à une soirée passée, l'application présentera un récapitulatif comprenant au minimum les données de base renseignées par l'organisateur, complétées par un graphe d'activité mettant en forme les résultats du module audio, c'est-à-dire les moments durant lesquels on peut considérer qu'il a dansé sur la musique.

5.1.2 Communication client/serveur

Cette partie couvre toutes les fonctions nécessaires au transfert et à la redirection des données entre utilisateurs de la plateforme LastNight. Elle doit donc implémenter les programmes gérant la connexion des applications au serveur ainsi que l'adaptation, l'envoi et la réception des requêtes.

En termes de nature des données échangées, le bloc de communication doit assurer par l'intermédiaire de flux bufferisés la transmission correcte des données simples (formulaires, choix de sondages, messages) comme complexes (images, fichiers d'analyse audio).

Le serveur doit, de son côté, implémenter une fonction d'interprétation des requêtes reçues pour pouvoir rediriger les messages entre utilisateurs ou mettre à jour sa base de données suite aux différentes requêtes.

5.1.3 Serveur

Le bloc « serveur » est le cœur matériel de la plateforme en ligne LastNight. Grâce à une base de données MySQL, il assure d'abord un rôle de gestion des comptes utilisateurs en stockant les données qu'ils ont renseignées à leur inscription. Il doit également enregistrer toutes les données relatives aux

soirées organisées par les utilisateurs, c'est-à-dire les métadonnées (lieu, horaires, prix, etc) mais surtout les informations recueillies en « live » (sondages, messages, photos, tempo de la musique).

5.1.4. Audio – Corrélation rythme de danse / tempo de la musique

Le bloc Audio concerne toutes les méthodes nécessaires pour remplir l'objectif principal de LastNight, à savoir fournir à l'utilisateur un récapitulatif des moments d'une soirée pendant lesquels il a dansé. Il doit donc implémenter une fonction de calcul du tempo qui s'applique à la fois aux mouvements et l'utilisateur et à la musique jouée par le DJ. Il s'implémente donc à la fois au niveau des applications Android et du PC utilisé par le DJ pour mixer.

Une fois ces calculs effectués, la partie la plus importante réside dans l'établissement d'une corrélation entre le tempo de la musique et celui des mouvements de l'invité. Les personnes du module Audio doivent donc réfléchir à un critère mathématique de corrélation qui permette de rendre compte de la manière la plus fidèle possible à la question suivante : l'utilisateur danse(ou essaie !)-t-il de danser sur la musique jouée actuellement ?

La compatibilité des fonctions d'analyse et de calcul avec les applications Android impose leur implémentation en Java dans leur version définitive.

5.2 Description des interfaces

5.2.1 Interface Application Android – Serveur

L'interface entre les applis utilisateur et le serveur LastNight doit contenir toutes les méthodes « client » permettant aux élèves du module Android d'interagir avec le serveur de manière quasi-transparente.

On distingue donc deux types de méthodes :

- Les requêtes « actives » qui vont modifier les données enregistrées dans la base MySQL, par exemple lors d'une inscription, lorsqu'un utilisateur rejoint une soirée créée sur la plateforme ou lorsqu'il veut partager une photo dans l'album de l'événement
- Les requêtes « passives » qui vont se contenter d'aller chercher des informations dans la base de données, comme par exemple les données relatives à une soirée après avoir scanné le QR Code affiché par l'organisateur dans la salle où se déroule l'événement

5.2.2 Interface Audio – Serveur

L'interface Audio-Serveur intervient, dans le schéma d'architecture, entre le client installé sur le PC du DJ et le serveur LastNight. Elle fait le lien entre stockage du tempo de la musique au cours de la soirée et transfert de ces résultats vers le serveur.

Elle contient donc des méthodes d'authentification du DJ sur la plateforme depuis son PC pour vérifier qu'il est bien déclaré « administrateur » de la soirée, mais aussi une méthode permettant d'envoyer le fichier stockant les résultats de calcul du tempo vers le serveur.

5.2.3 Interface Android – Audio

Elle contient principalement deux types de méthodes, permettant au programme de calcul audio et à l'application Android de ne faire qu'un dans notre système, à savoir :

- Les méthodes nécessaires au stockage des données accélérométriques dans un fichier, et leur lecture pour effectuer ensuite les calculs de tempo à partir de ces données
- Les méthodes de lecture et d'interprétation des résultats de corrélation entre tempo des mouvements de l'utilisateur et celui de la musique, pour permettre à l'application de les afficher sur une interface lisible

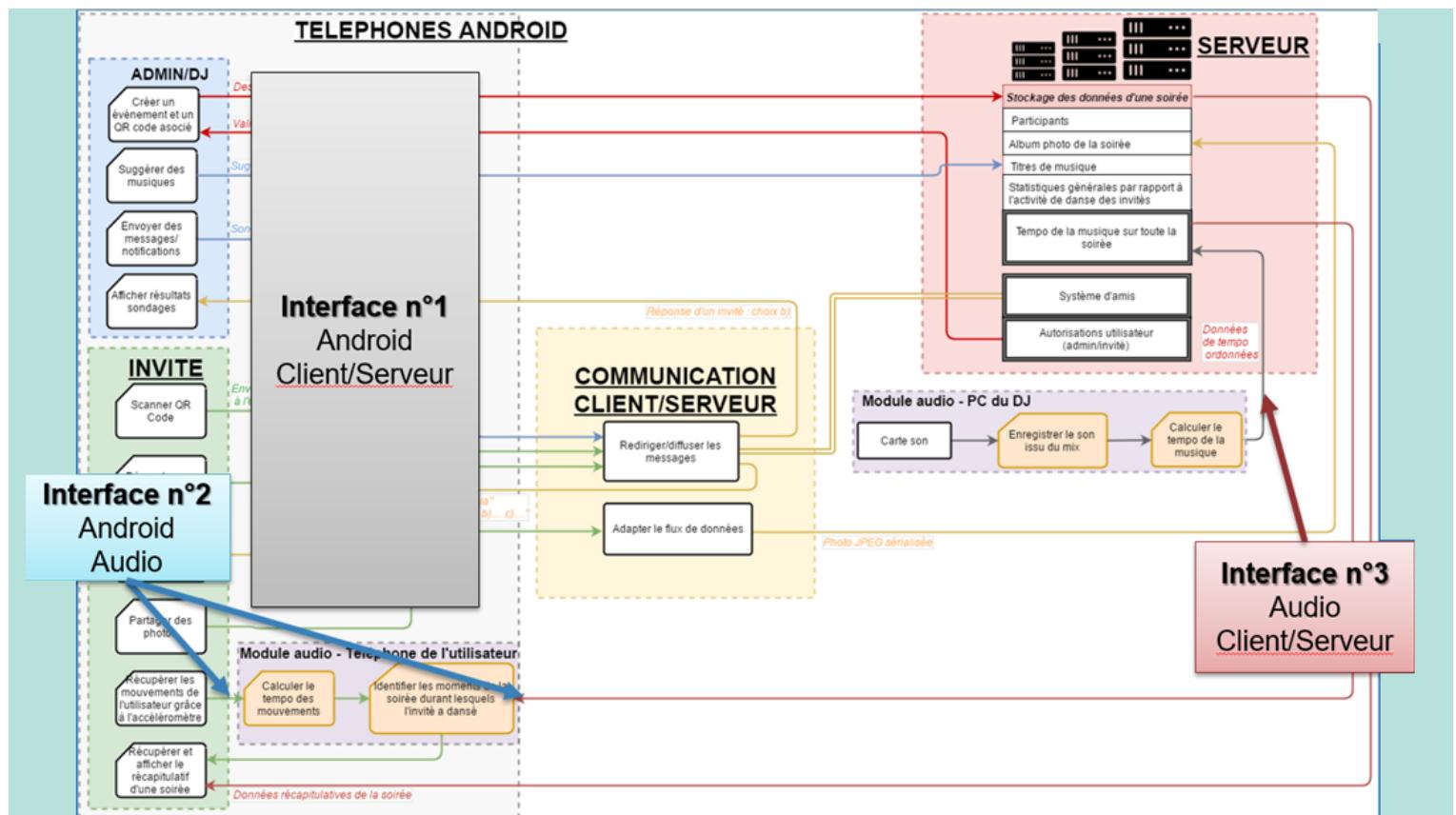


Figure 3 : Situation des interfaces au sein de l'architecture du projet

5.2 Diagramme d'activité

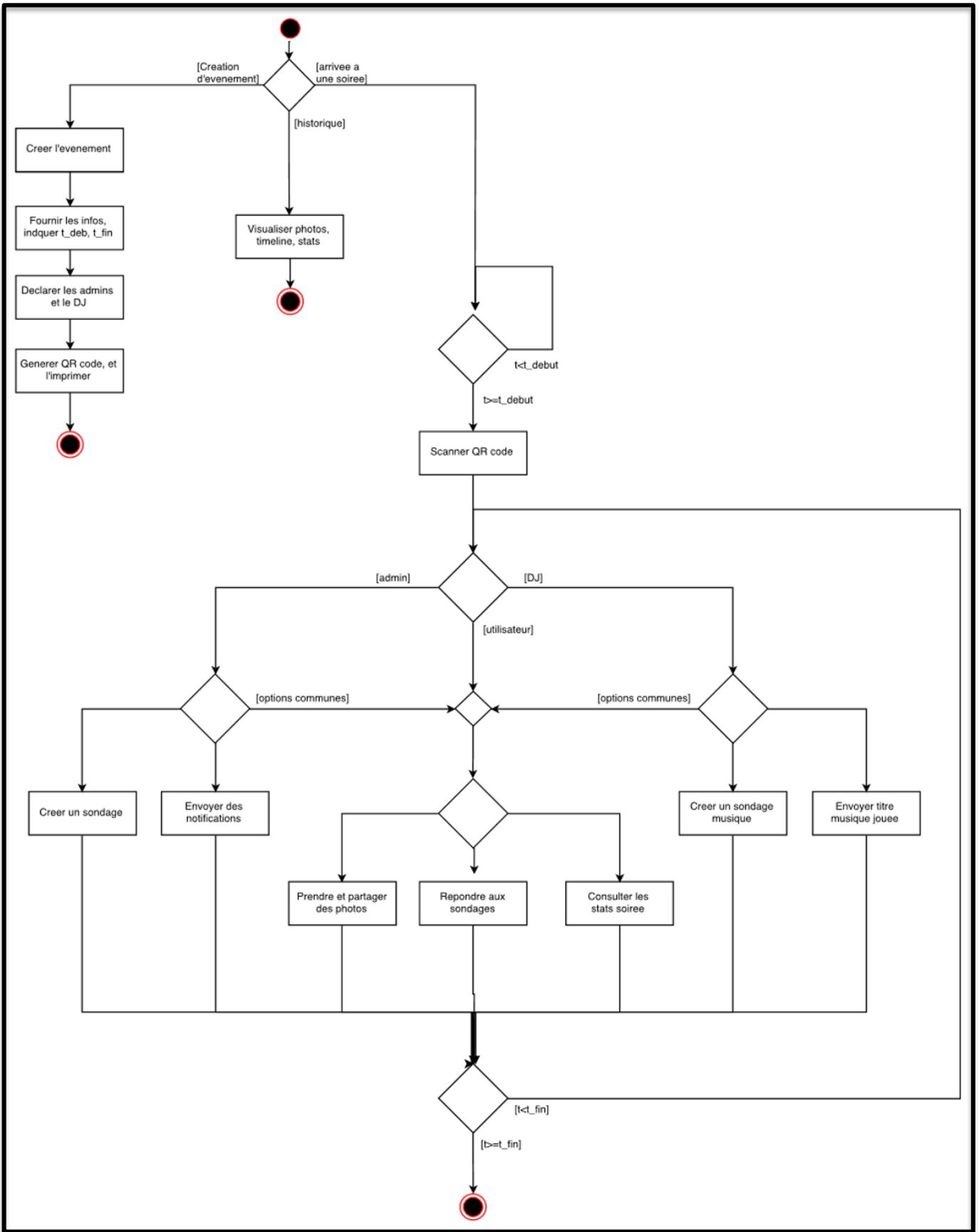


Figure 4 : Diagramme d'activité

5.3 Interface utilisateur graphique

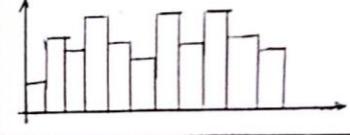
<p>NON:</p>  <p>Date : / /</p> <p>Horaires :</p> <p>Descriptif</p>  <p>Statut :</p> <p>Prix :</p> <p> Terminé! </p> <p>Page de création de Soirée</p>	<p>NOM DE LA SOIREE:</p>  <p>ACTIVITÉ</p> <p>Sondage :</p> <p>Message à tous :</p> <p>Promotion :</p> <p>En cours (sondage; message etc.)</p> <p>Page DJ</p>
<p>NOM DE LA SOIREE:</p> <p>En cours (Sondage ; Promotion ; photo très marquante ...)</p> <p>Musique en cours </p> <p>Où sont vos amis ?</p>    <p>Page Soirée</p>	<p>Vos dernières soirées :</p> <ul style="list-style-type: none"> - - - <p>Vos musiques favorites :</p> <ul style="list-style-type: none"> - - - <p>Comme organisateur :</p> <ul style="list-style-type: none"> - <p></p> <p>Créer votre événement</p> <p>Page d'accueil</p>

Figure 5 : Storyboard 1

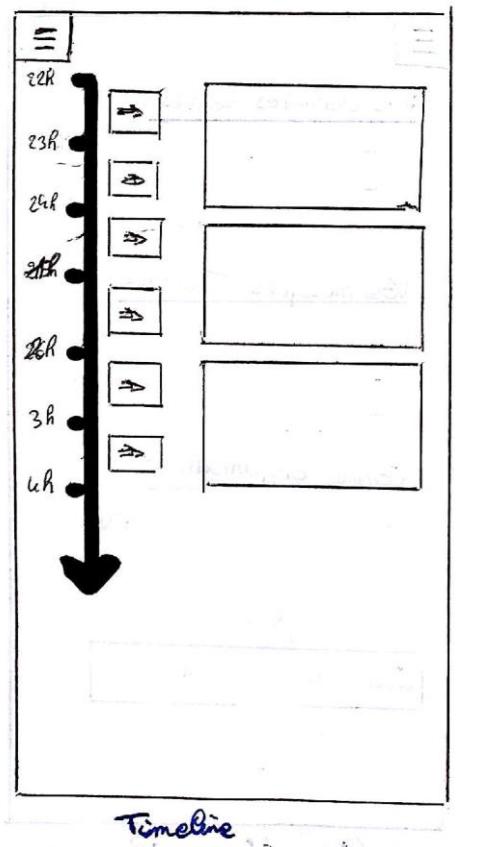


Figure 6 : Storyboard 2

5.4 Tableau détaillé des tâches

	PAN 3	PAN 4
Module Audio		
Récupérer des données accélérométriques	X	
Obtenir le tempo des signaux	X	
Déterminer un critère de corrélation	X	
Déterminer si un utilisateur danse ou pas	X	(affiné)
Récupérer du son depuis le lecteur du DJ	X	
Module Androïd		
Création et lecture de QR codes		X
Menu complet et fonctionnel	X	
Outil de création de comptes utilisateur	X	
Système de notification		X
Fonctionnalités de soirée en cours	X	(+Partage photos)
Mise en forme et envoi des données	X	
Visualisation esthétiquement plaisante des données		X
Page affichant en direct les données de la soirée		X
Module Client/Serveur		
Détermination de la nature des messages	X	
Stockage et envoi de données en temps réel	X	
Gestion des requêtes SQL	X	
Redirection des messages		X
Mise en place d'un système d'amis	X	
Module SES		
Etude d'impact sur la vie privée	X	
Mise en place	X	
Intégration		
Intégration des modules	X	
Présentation		
Montage de la vidéo		X

Tableau 1 - Description des tâches

6 Organisation du projet

6.1 Diagramme de planification temporel des tâches

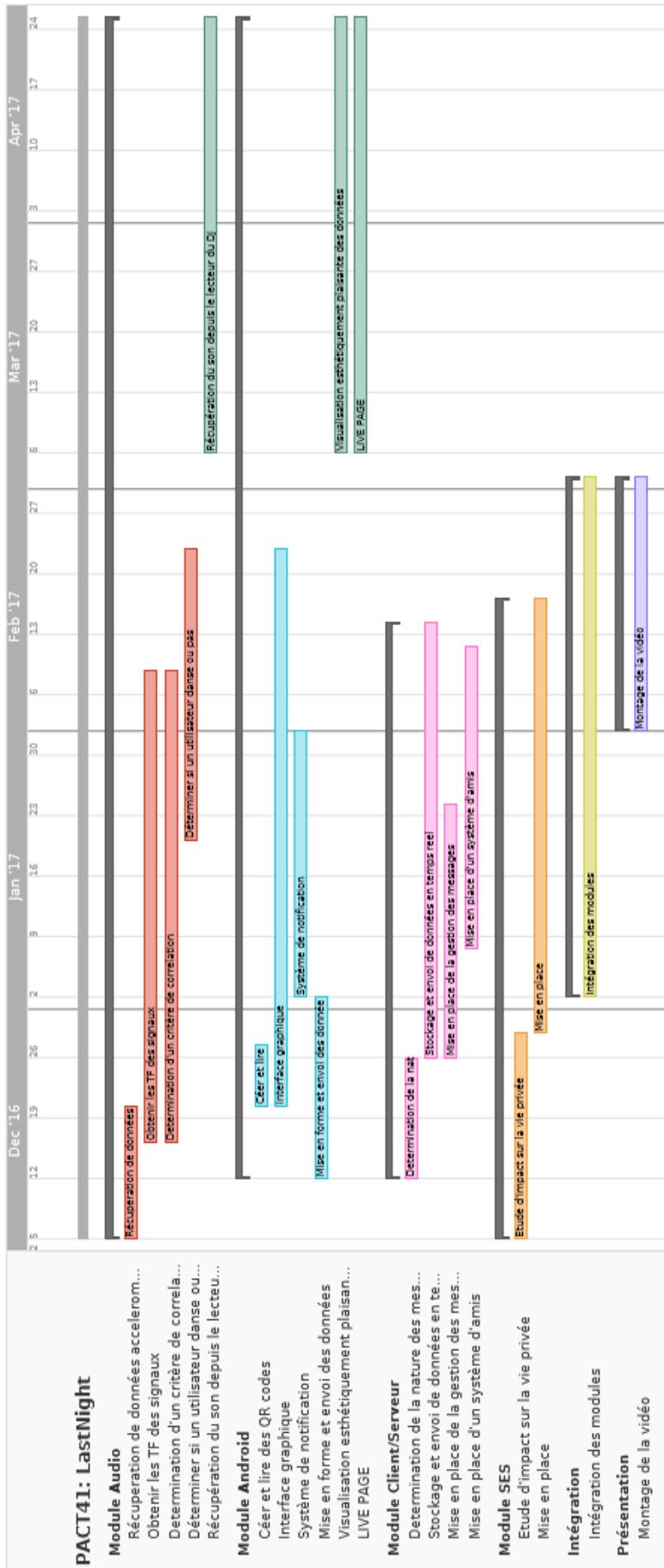


Figure 7 : Diagramme de Gant

6.2 Répartition des élèves par module

Modules	Audio	Communication client-serveur	Android	SES	Test et Intégration
Encadrant	Bertrand David	Jean-Claude Dufourd	Jean-Claude Dufourd	Caroline Rizza	Soumia Dermouche
Elève 1	Leello Tadesse Dadi	Louis Combaldieu	Daniel Mayau	Robin Blanchard	Paul-Elian Tabarant
Elève 2	Lucie Moulinas	Michael Dasseville	Robin Blanchard	Louis Combaldieu	Daniel Mayau
Elève 3			Paul-Elian Tabarant		Louis Combaldieu
Elève 4					
Elève 5					
Elève 6					
Elève 7					

Tableau 2 - Répartition élèves-modules

6.3 Plans de test

6.3.1 Plan de test - Module Android

Menu de l'application

Contexte du test : L'utilisateur se connecte sur l'application. Il dispose alors d'un menu lui donnant accès aux options de gestion de son compte, sa liste d'amis, l'historique des événements auxquels il a participé, un outil d'organisation de soirée et enfin une section pour rejoindre une soirée en cours et fournir ensuite toutes les fonctionnalités « live » d'une soirée.

Entrée : Ensemble de scénarios de navigation dans l'application

Test de fonctionnement : On sélectionne chacun des items du menu pour vérifier la bonne redirection.

Scénario particulier : l'utilisateur sélectionne l'item « soirée en cours » et rejoint ensuite une soirée à l'aide des options fournies dans l'activité lancée. Il navigue dans d'autres activités de l'appli ou ailleurs sur son téléphone puis sélectionne à nouveau « soirée en cours ».

Résultats attendus : Chaque item du menu redirige vers l'activité voulue. Deux clics successifs sur le même item ne relancent pas 2 fois la même activité. On pourra éventuellement exiger qu'un appui sur un item différent du précédent dans le menu ouvre la nouvelle activité en fermant la précédente, mais ce critère d'ergonomie reste subjectif.

Résultat du scénario particulier : Une fois la soirée rejointe, l'application enregistre cet état et l'item « soirée en cours » redirige directement vers l'activité de gestion de l'événement sans passer par l'outil permettant de rejoindre une soirée.

Saisie de formulaires

Contexte du test : Afin de s'inscrire sur la plateforme en ligne LastNight, se connecter, éditer son profil ou organiser des soirées, l'utilisateur a à sa disposition des activités de saisie de formulaire sur

l'application qui lui permettent de renseigner toutes les informations nécessaires au serveur pour enregistrer ses requêtes dans la base de données.

Entrée : Série de saisies comprenant toutes les possibilités de saisies incorrectes et un ensemble de saisies correctes

Test de fonctionnement : On remplit les formulaires avec la série de saisies prévues, on valide la saisie et on examine le résultat affiché par l'application.

Résultats attendus : Les saisies incorrectes sont communiquées explicitement à l'utilisateur au moyen de boîtes de dialogue ou de messages Toast tout en indiquant à quel endroit se trouve le problème. On pourra éventuellement enregistrer les champs remplis correctement. L'ensemble des saisies correctes doivent aboutir à l'affichage d'un message de confirmation.

Affichage d'informations

Contexte du texte : Lorsque l'utilisateur consulte un profil d'ami ou le sien ou qu'il consulte une soirée à laquelle il a participé ou va participer (scan d'un QR code), l'application doit mettre en forme les données associées et les lui présenter fidèlement.

Entrée : Jeux de données liées à un profil utilisateur ou à une soirée

Test de fonctionnement : On affecte une série de valeurs à des variables associées à chaque information à afficher (par débogage manuel par exemple), on lie ces variables aux widgets associés puis on examine le résultat visuel sur l'activité d'affichage.

Résultats attendus : Chaque changement de valeur d'une variable doit actualiser l'affichage des données en conséquence sur l'activité.

Suivi des mouvements

Contexte du test : Lorsque l'utilisateur rejoint un événement en cours, un service de suivi des mouvements est lancé et écrit les valeurs d'accélération dans un fichier pour qu'un traitement de tempo soit effectué par la suite. Celui-ci tourne en arrière-plan jusqu'à ce que l'utilisateur quitte la soirée ou que la soirée se termine.

Entrée : Différents scénarios de soirée

Test de fonctionnement : On affiche des logs dans les sections OnStart() et OnDestroy() du service pour délimiter son arrêt et son démarrage. On utilise également les logs pour afficher le contenu du fichier de suivi des mouvements une fois le service stoppé. On rejoint une soirée en cours via l'application d'une durée déterminée à l'avance et on attend la fin du décompte. On réessaie avec la même durée mais cette fois en quittant manuellement la soirée avant la fin du décompte.

Résultats attendus : Le message de log indiquant que le service s'est lancé est constaté à l'instant où l'utilisateur confirme qu'il souhaite rejoindre la soirée via l'application. Le message d'arrêt apparaît après le décompte de fin de soirée dans le premier cas, et après appui sur le bouton « quitter la soirée » dans le second cas. Dans les deux cas évoqués, le moniteur de débogage Android liste correctement le contenu du fichier avec les données accélérométriques enregistrées sous la forme « t= ... ms, {Ax = ..., Ay = ..., Az = ...} » pour chaque point enregistré.

Messages de l'organisateur

Contexte du test : Pendant une soirée, les invités peuvent utiliser l'application pour visualiser les messages postés par l'organisateur, dans la section consacrée à l'événement sur la plateforme LastNight. L'organisateur doit, lui, pouvoir utiliser l'application pour éditer des messages.

Entrée : Ensemble de chaînes de caractères décrivant des messages, statut de l'utilisateur par rapport à la soirée en question (invité ou organisateur)

Test de fonctionnement : On se connecte avec un compte invité et un compte organisateur sur la soirée. On visualise la page associée à la soirée sur l'application.

Résultats attendus : En tant qu'invité, l'application affiche l'ensemble des messages postés par l'organisateur. En tant qu'organisateur, l'activité propose également une option d'édition de messages.

Sondages

Contexte du test : Pendant une soirée, les personnes déclarées comme administrateurs sur l'événement en question et ayant rejoint la soirée auparavant via l'application peuvent envoyer des sondages aux invités à raison de 1 par intervalle d'une dizaine de minutes. Les invités peuvent consulter le sondage en cours sur leur application.

Entrée : Ensemble de données décrivant des sondages, statut de l'utilisateur sur la soirée en question (invité ou administrateur) et différents choix d'invités sur le sondage

Test de fonctionnement : On se connecte avec un compte invité et un compte organisateur sur la soirée. On visualise la page associée à la soirée sur l'application.

Résultats attendus : En tant qu'invité, l'application affiche le sondage en cours avec une option de réponse au sondage. L'activité de réponse récupère l'indice de la réponse de l'invité dans ses attributs. En tant qu'organisateur, l'activité propose également une option d'édition de sondages et stocke la liste des options sous la forme d'une liste de chaînes de caractères (String).

Capture de photos (Prototype final)

Contexte du test : Lorsque l'utilisateur souhaite associer une photo à son profil sur LastNight, à la soirée qu'il organise ou partager des photos dans l'album de la soirée à laquelle il se trouve, l'application doit récupérer la photo qu'il a choisie depuis la galerie ou l'appareil photo.

Entrée : Série de photos provenant d'activités différentes

Test de fonctionnement : On sélectionne l'option de capture photo dans l'application, on sélectionne la photo voulue depuis la galerie ou l'appareil photo grâce à une boîte de dialogue et on valide le choix

Résultats attendus : La photo est récupérée dans l'activité principale de capture photo dans un objet Bitmap et s'affiche correctement dans un widget de type ImageView. En cas d'annulation lors de la prise de photo, rien ne se passe et on revient à l'activité de capture.

Mise en forme du récapitulatif de soirée (Prototype final)

Contexte du test : L'utilisateur peut consulter le récapitulatif de toutes les soirées passées auxquelles il a participé via la partie « historique » de l'application. Il visualise alors l'évolution de son activité de danse pendant la soirée ainsi que des informations générales qui lui sont associées (lieu, date etc).

Entrée : Un ensemble de fichiers décrivant les résultats de corrélation entre tempo de la musique jouée pendant la soirée et les mouvements de l'utilisateur, sur une certaine durée

Test de fonctionnement : On connaît à l'avance les instants correspondant à un résultat « l'utilisateur est en train de danser ». On applique la génération du récapitulatif à partir des fichiers de corrélation et on visualise les graphiques correspondants.

Résultats attendus : Les instants durant lesquels l'utilisateur a dansé doivent être mis en évidence, soit par un signal binaire (1 correspond à de la danse), soit par un score plus ou moins élevé dépendant du critère de corrélation établi par le module Audio.

Génération et scan du QR code d'une soirée (Prototype final)

Contexte du test : Lorsqu'un utilisateur crée une soirée qu'il organise par l'application LastNight, un QR code est généré. Ce QR code peut ensuite être imprimé et affiché dans l'enceinte de la soirée pour que les invités le scannent et s'identifient comme présents à l'événement. Ils ont alors accès à toutes les fonctionnalités décrites précédemment.

Entrée : Une série d'identifiants d'événements (leur nom en 1^{ère} approximation)

Test de fonctionnement : On effectue la génération des QR codes à partir des identifiants. On scanne ensuite les QR obtenus grâce à l'application.

Résultats attendus : Chaque scan de QR code aboutit à la régénération de l'identifiant initial de la soirée avec lequel il avait été créé.

6.3.2 Plan de test - module intégration

Création d'un événement : modules Android + Client-Serveur

Contexte du test : Dans la section « Création d'événement » disponible sur l'appli Android, l'utilisateur saisit les informations de base relatives à une soirée qu'il souhaite organiser (nom de la soirée, date, horaires, prix, descriptif, lien Facebook, nom du DJ). Un appui sur le bouton « valider » doit envoyer toutes ces informations vers le serveur.

Entrée : Série de descriptifs d'événements, certains valides et d'autres invalides : soit information mal saisie (horaires ou date dans le mauvais format, lettres dans le champ « prix » au lieu de chiffres par exemple, ou absence de certaines informations non facultatives).

Test de fonctionnement : On saisit les différents exemples d'événements dans la section « Création d'événement » et on examine la réponse du serveur

Résultats attendus : Si les données sont erronées, un message d'erreur doit s'afficher sur le téléphone. Si les informations sont correctement saisies, un message doit s'afficher sur le téléphone pour attester de la création de l'événement sur le serveur, et un QR code associé à la soirée doit pouvoir être téléchargé (après le PAN3 uniquement).

Rejoindre un événement : modules Android + Client-Serveur (Prototype final)

Contexte du test : Lorsque l'utilisateur voudra attester qu'il est à une soirée pour pouvoir bénéficier des fonctionnalités associées (suivi de son activité, réponses aux sondages, réception de messages de l'organisateur etc), il scannera le QR code présent à l'entrée de la salle qui permettra au serveur d'identifier à quelle soirée se trouve l'invité.

Entrée : QR code de l'événement

Test de fonctionnement : On applique un ensemble de QR codes générés auparavant par le serveur pour différentes soirées, et on examine le descriptif de l'événement reconnu. On pourra éventuellement vérifier qu'il n'y a pas de fonctionnement imprévu dans le cas du scan d'un QR code erroné.

Résultats attendus : Pour chaque QR code, un scan doit aboutir à la reconnaissance de l'événement associé dont les informations seront chargées sur le téléphone de l'utilisateur.

Dans le cadre du prototype allégé, le scan du QR code ne sera pas encore implémenté : une liste des événements en cours sera chargée sur le téléphone de l'utilisateur qui se servira de cette liste pour rejoindre l'événement qu'il souhaite.

Estimer si l'invité a dansé ou non à chaque instant de la soirée – Modules Audio + Client-Serveur

Contexte du test : Au cours d'une soirée, le téléphone de l'invité va mesurer ses mouvements à l'aide de l'accéléromètre. Le PC du DJ sera connecté au serveur et lui enverra le son issu de la musique jouée pour qu'il puisse déterminer son tempo. Une fois ce calcul effectué sur toute la durée de la soirée, les utilisateurs pourront récupérer via leur téléphone les résultats de ce calcul de tempo et l'application Android devra établir une corrélation entre le tempo de la musique et les données de l'accéléromètre sur toute la durée de la soirée.

Entrées : Un signal audio et un signal d'accéléromètre de plusieurs heures

Test de fonctionnement : On dit à la personne qui génère le signal accélérométrique avec son téléphone de danser à certains instants sur la musique qui est diffusée et à d'autres instants, au contraire, de ne pas danser. Les instants à laquelle la personne danse ou non sont planifiés et donc connus à l'avance.

Résultats attendus : Les calculs de corrélation entre le tempo de la musique et les mouvements de l'utilisateur sur la durée de l'enregistrement doivent correspondre aux prévisions : le système a détecté que l'utilisateur a dansé aux instants où on lui a demandé de danser en rythme avec la musique et bien-sûr uniquement à ces instants-là. Le calcul de tempo sur le serveur suivi du calcul de corrélation mouvements/tempo sur les téléphones doivent être suffisamment optimisés pour que chaque invité puisse consulter ses résultats le lendemain d'une soirée (5-6h maximum).

Partage de musiques par le DJ – Modules Android – Client/Serveur (Prototype final)

Contexte du test : Pendant la soirée, le DJ peut se connecter sur l'application Android avec son compte utilisateur et suggérer à tout moment les titres des musiques qu'il est en train de jouer.

Entrées : Ensemble de suggestions sous la forme de chaînes de caractères (titres de musiques)

Test de fonctionnement : On envoie depuis un compte connecté en tant qu'admin (DJ) une série de suggestions au cours de la soirée

Résultats attendus : Chaque suggestion doit être récupérée et stockée dans la base de données de la soirée correspondante avec la date exacte d'envoi.

Partage de photos par les invités – Modules Android – Client/Serveur

Contexte du test : Au cours d'une soirée, chaque invité peut récupérer sur son téléphone des photos issues de la soirée et les partager sur le serveur pour qu'elles apparaissent dans l'album de soirée qui sera consultable par tous les utilisateurs présents.

Entrée : Images JPEG dans la mémoire du téléphone des invités, avec date de capture associée

Test de fonctionnement : On procède depuis l'application Android à l'envoi de chacune des photos utilisées pour le test vers le serveur

Résultats attendus : Chaque image doit être stockée dans la base de données de la soirée en lui associant sa date de capture, avec une détérioration de sa qualité raisonnable (taux de compression suffisamment faible).

Interaction DJ-admin/invités – Modules Android – Client/Serveur

Contexte du test : Au cours d'une soirée, l'organisateur et le DJ peuvent envoyer des messages globaux et des sondages sur les téléphones des invités présents via leur application Android. En ce qui concerne les sondages, les invités doivent pouvoir y répondre et les résultats sont consultables par tous les utilisateurs présents.

Entrées :

- Message de l'organisateur/DJ saisi sur l'application Android
- Sondage créé par l'organisateur/DJ sur l'application et ensemble de réponses des invités

Test de fonctionnement :

- Pour l'envoi de messages : On utilise plusieurs comptes utilisateurs sur différents téléphones pour rejoindre une soirée en cours en tant qu'invités. On quitte la soirée sur l'un de ces comptes (option de l'application Android). L'organisateur ou le DJ envoie ensuite un message global à destination des invités depuis son téléphone via l'application Android.
- Pour les sondages : On crée plusieurs sondages sur un compte loggé en tant qu'organisateur ou DJ sur une soirée, avec un nombre d'options différentes. Avec des comptes « invités », on fournit des réponses connues à l'avance à ces sondages.

Résultats attendus :

Pour l'envoi de messages : L'ensemble des invités présents à la soirée doivent recevoir le message de l'organisateur sur leur téléphone. Il faudra aussi vérifier que l'invité qui a déjà quitté la soirée au moment de l'envoi ne reçoive pas le message.

Faire valider les photos par l'organisateur – Modules Android – Client/Serveur

Contexte du test : Une fois la soirée terminée, l'organisateur a accès à l'ensemble des photos qui ont été partagées sur l'événement. Il pourra les trier en spécifiant lesquelles doivent apparaître dans l'album de la soirée et lesquelles sont indésirables. Une fois cela fait, une validation du tri rend l'album photo consultable par les invités.

Entrées : Un ensemble de photos de soirée stockées sur le serveur, associé à une série de tris différents pour ce même album.

Test de fonctionnement : On utilise un compte déclaré comme organisateur d'une soirée pour trier un ensemble de photos mises sur le serveur à l'emplacement réservé à l'album de la soirée, puis on valide le tri sur l'application.

Résultat attendu : Pour chaque essai de tri effectué depuis le téléphone de l'organisateur, le serveur doit conserver uniquement la collection de photos triées dans sa base de données et supprimer les photos indésirables.

Générer un récapitulatif de soirée sur le téléphone de l'invité – Modules Android – Client/Serveur

Contexte du test : Lorsque la soirée est terminée et que les calculs de corrélation tempo/mouvements ont été effectués, l'utilisateur reçoit une notification sur son téléphone lui indiquant que son récapitulatif de soirée est disponible. L'application Android envoie alors une requête vers le serveur pour récupérer les informations relatives à la soirée et génère un récapitulatif détaillé avec une mise en forme agréable et intuitive.

Entrées :

- Données générales de la soirée stockées sur le serveur (nombre de participants, photos, titres de musiques suggérés par le DJ)
- Résultats des calculs de corrélation activité physique/tempo de l'invité stockés dans un fichier sur son téléphone

Test de fonctionnement : On génère un récapitulatif de soirée depuis un compte utilisateur à partir d'une soirée dont on connaît exactement chaque donnée enregistrée sur le serveur ainsi que dans le fichier de corrélation activité physique/tempo.

Résultats attendus : Le récapitulatif doit contenir le détail de l'activité de danse de l'invité sur une timeline fidèle au fichier stocké sur son téléphone. Les photos disponibles doivent correspondre au tri effectué préalablement par l'organisateur. Les titres de musiques suggérés par le DJ doivent apparaître au bon endroit sur l'échelle de temps de la timeline.

6.3.3 Plan de test - Module Audio

Détermination du BPM d'un extrait musical

Contexte du test : Nous avons besoin de pouvoir connaître le BPM d'un extrait musical, qui pourra être comparé à celui de l'utilisateur.

Entrées : Un fichier audio au format wav, dont la durée n'excède pas 30 secondes pour que les calculs ne soient pas trop longs.

Test de fonctionnement : Une fois l'algorithme mis en place, on le fait travailler sur des extraits dont le BPM est connu.

Résultats attendus : L'algorithme doit nous renvoyer le BPM du fichier, qui doit correspondre à celui du morceau dont il est extrait.

Détermination du BPM d'un signal accélérométrique

Contexte du test : Nous devons être capable de déterminer le BPM d'un signal accélérométrique, par 2 méthodes différentes : autocorrélation et banc de filtre

Entrées : Un tableau contenant les données accélérométriques enregistrées par un smartphone.

Test de fonctionnement : Utilisation de l'algorithme sur des fichiers collectées lorsque l'utilisateur danse sur une musique dont on connaît le BPM, ainsi qu'en tapant régulièrement la mesure.

Résultats attendus : Le BPM obtenu par les 2 méthodes : pour l'autocorrélation on retourne une valeur de BPM, et pour le banc de filtre on a un graphe énergie=f(BPM) et une valeur de BPM dominant. Ces valeurs doivent être cohérente avec le BPM de l'extrait audio utilisé.

Définition d'un critère précis de danse pour l'utilisateur

Contexte du test : Après avoir collecter les données sur le BPM de l'utilisateur et de la musique, il faut savoir comment les interpréter pour déterminer les instants où l'utilisateur danse.

Entrées : Un tableau contenant les données accélérométriques pour chaque axe, enregistrées par un smartphone.

Test de fonctionnement : Utilisation de l'algorithme sur des fichiers accélérométriques enregistrés dans des conditions connues. Pour chaque extrait, on sait si l'utilisateur dansait ou pas, et quelle chanson était diffusée.

Résultats attendus : Un score calculé en fonction des 2 valeurs de BPM obtenues et exprimant la « propreté » de la danse de l'utilisateur. Le score doit dépasser 8 lorsque l'extrait correspond à une danse.

6.3.4 Plan de test – Module communication client-serveur

Création de la base de données :

Contexte : Après avoir eu le scripte Postgre fourni par le logiciel Jmerise, nous l'envoyons sur le serveur.

Entrée : Scripte Postgre

Test de fonctionnement : Nous exécutons le scripte Postgre avec les commandes postgresql.

Résultats attendus : Avec les commandes postgresql nous devons voir s'afficher les tables de la base de données avec toutes leurs colonnes.

Requêtes SQL :

Contexte : Une fois la base de données mise en place, nous pouvons ajouter des éléments manuellement, et les voir manuellement. Alors après rédaction des requêtes SQL sur java, dès que la classe qui contient ses méthodes est mise sur le serveur nous créons une classe TestMySQL.java qui appelle les méthodes

SQL et affiche le résultat des requêtes.

Entrée : TestMySQL.java

Test de fonctionnement : Nous exécutons le programme TestMySQL.java.

Résultats attendus : Suite à une requête qui lit des éléments de la base, nous affichons le résultat avec un « println ». Pour les requêtes qui ajoutent ou modifient un élément de la base nous regardons manuellement si les ajouts ou modifications ont bien été faits.

Création d'un sondage :

Contexte : Lors d'un événement un administrateur veut envoyer un sondage aux utilisateurs présents. Alors il crée son sondage en mettant un titre et une liste d'éléments pour lesquels les utilisateurs pourront voter.

Entrée : Titre du sondage, liste d'éléments.

Test de fonctionnement : une fois le sondage créé nous affichons son contenu qui n'est composé que de String avec un « println ».

Résultats attendus : l'objet sondage correspond bien à ce que l'administrateur a créé.

Réponse à un sondage :

Contexte : Une fois que l'administrateur a créé et envoyé son sondage, les utilisateurs présents à l'événement peuvent le voir et voter.

Entrée : Sondage créé par l'administrateur.

Test de fonctionnement : Les utilisateurs présents à l'événement peuvent voter. On doit renvoyer l'indice dans la liste de l'élément pour lequel ils ont voté.

Résultats attendus : l'entier renvoyé est bien inférieur au nombre d'élément dans la liste, et coïncide avec le choix de l'utilisateur.

Affichage du sondage :

Contexte : Une fois que l'administrateur a créé et envoyé son sondage, toutes les personnes présentes à l'événement voient le sondage avec le pourcentage de vote pour chaque élément du sondage.

Entrée : Sondage avec des votes.

Test de fonctionnement : Les personnes présentes à l'événement peuvent voir s'afficher le sondage. Dès qu'un utilisateur vote, l'affichage est mis à jour.

Résultats attendus : L'affichage du sondage est bien visible pour toutes les personnes présentes à l'événement. L'affichage est bien mis à jour et le temps de mise à jour n'est pas trop long.

Connexion Client-Serveur avec Multi-threading :

Contexte : Plusieurs clients vont se connecter au serveur et voudront accéder aux données en même temps

Entrée : Plusieurs requêtes de connexion simultanées

Test de fonctionnement : Plusieurs comptes s'identifient et gardent leur session ouverte en même temps

Résultats attendus : Les clients sont bien connectés en parallèle.

Envoi d'Image :

Contexte : Les photos doivent être stockées dans une base de données séparée car il est difficile de les mettre sur la base SQL. La solution adoptée est un système de fichier.

Entrée : Une photo

Test de fonctionnement : Envoi d'une photo sur un profil test et sur une soirée test.

Résultats attendus : Côté serveur, les fichiers sont bien créés et contiennent bien les photos.

Envoi d'un message :

Contexte : Lors d'un événement un administrateur veut envoyer un message aux utilisateurs présents.

Alors il écrit son message et l'envoi.

Entrée : Texte du message.

Test de fonctionnement : une fois le message écrit et envoyé nous affichons son contenu qui n'est composé que de String avec un « `println` ».

Résultats attendus : l'objet message correspond bien à ce que l'administrateur a écrit.

6.4 Diagramme d'avancement des tâches

Module	Avancement Prévu (%)	Avancement Réel (%)	Temps passé (h)	Description brève du travail effectué, analyse des écarts constatés
Intégration et test	100	100	50	<ul style="list-style-type: none">Architecture du projet fixée et approuvée par l'ensemble des membres du groupe ainsi que notre tuteurInterfaces Audio-Android-Serveur définies avec les modules concernésCode de calcul du tempo des mouvement intégré sur les téléphones, premiers tests convaincantsClient de connexion intégré sur les téléphones avec classe de test. Les méthodes d'échange d'informations entre le client et le serveur ont toutes été testées grâce à la classe implémentée et les résultats des logs de test enregistrés dans un document Word.Implémentation d'une application spécifique pour tester l'envoi et la réception des photos de soirée et de profil utilisateur
Audio	100	100	160	<ul style="list-style-type: none">Détermination du BPM de données accélérométriques, et d'un fichier sonore.Nous voulions réaliser nos calculs en direct au cours de la soirée, ce qui n'est pas encore fait et explique les faibles pourcentages. Sans cela, le travail est presque terminé, le retard est lié à la détermination d'un critère universel indiquant si l'utilisateur danse.Détermination du BPM de données accélérométriques, et d'un fichier sonore. A partir du BPM de l'utilisateur, attribution d'un score reflétant la qualité de sa danseLes calculs sont réalisés en direct au cours de la soirée, sur le smartphone de chaque utilisateur, et envoyés sur le serveur. Le lendemain l'utilisateur peut consulter une timeline résumant sa soirée (score en fonction du temps).Le DJ peut accéder en direct au classement des meilleurs danseurs, et au score moyen des personnes présentes, ce qui lui donne une idée de la qualité de sa playlist.
Client-Serveur	100	100	180	<ul style="list-style-type: none">- Création d'une base de données pour stocker des informations relatives aux utilisateurs, événements, et administrateurs.- Conception des requêtes SQL en Java pour gérer la base de données- Administration du serveur distant dédié à notre projet, que

				<p>nous avons loué.</p> <ul style="list-style-type: none"> - Conception d'une méthode de multi-threading pour connecter les utilisateurs au serveur. - Conception d'une méthode d'envoi d'image entre les clients et le serveur. - Conception d'un système de messagerie. - Conception d'un système de sondage.
Android	100	100	240	<ul style="list-style-type: none"> • Menu de l'application fonctionnel et complet, avec modèle MVC sommaire, en accord avec le story-board conçu • Formulaires de création de comptes utilisateurs, événements, modifications à apporter et visualisation de données • Service de capture des mouvements calculant le tempo des mouvements de l'utilisateur ainsi que le score de danse. Envoie les résultats périodiquement au serveur et les stocke dans un fichier pour le récapitulatif individuel de l'utilisateur • Outils de visualisation, édition et envoi de sondages/messages • Capture de photos depuis la galerie ou le capteur photo du téléphone, transfert vers le serveur • Communication avec le serveur fonctionnelle au sein de l'application • Récapitulatif de soirée complet avec évolution du tempo et du score au cours de la soirée, affichage de l'album de soirée
Etude d'impact sur la vie privée	100	100	15	<ul style="list-style-type: none"> • Le PIA (privacy impact assessment) est terminé, il faut maintenant s'occuper de l'étude de risque. • Le PIA a été corrigé et complété. L'étude des risques a aussi été réalisée et corrigée.

Tableau 3 - Avancement des modules

7 Bibliographie

Etat de l'art :

Akram Bayat, Marc Pomplun, Duc A. Tran, "A study on human activity recognition using accelerometer data from smartphones", 2014

Jennifer R. Kwapisz, Gary M. Weiss, Samuel A. Moore, "Activity Recognition using Cell Phone Accelerometers", 2010, http://www.cis.fordham.edu/wisdm/public_files/sensorKDD-2010.pdf

Eric Lee, Urs Enke, Jan Borchers, Leo de Jong, "Towards Rhythmic Analysis of Human Motion using Acceleration-Onset Times", 2007, <https://pdfs.semanticscholar.org/4625/0c1a611254cbb206215da0d5d909cfed8400.pdf>

Michael Kuhn, Martin Wirz, Matthias Flückiger, Roger Wattenhofer, Gerhard Tröster, "Sensing dance engagement for collaborative music control", 2011, <http://disco.ethz.ch/publications/ISWC2011 -1.pdf>

Module Audio :

Eric D. Scheirer, "Tempo and beat analysis of acoustic musical signals", 1997, <http://www.ee.columbia.edu/~dpwe/papers/Schei98-beats.pdf>

Miguel A. Alonso Arevalo, "Extraction d'information rythmique à partir d'enregistrements musicaux", 2007, <https://pastel.archives-ouvertes.fr/file/index/docid/500407/filename/these.pdf>

M. Alonso, B. David, G. Richard, "A study of tempo tracking algorithms from polyphonic music signals", 2003, <https://pdfs.semanticscholar.org/7e7a/03cd318678e914349cf23732b7d9a377843f.pdf>

https://www.clear.rice.edu/elec301/Projects01/beat_sync/beatalgo.html

Module Android

Tutoriel de base ayant servi de fil directeur dans notre formation : Frédéric ESPIAU - Créez des applications pour Android. Open Classrooms [en ligne]. Disponible sur : <https://openclassrooms.com/courses/creez-des-applications-pour-android>

Réponse à la plupart de nos problèmes spécifiques pendant la programmation : Stack Overflow [en ligne]. Disponible sur : <http://stackoverflow.com/>

Exemple d'utilisation pour stocker l'état courant de l'application :

<http://stackoverflow.com/questions/708012/how-to-declare-global-variables-in-android>

Documentation sur les classes Java liées à l'environnement Android : Android Developers [en ligne]. Disponible sur : <https://developer.android.com/index.html>

Exemple d'utilisation pour la gestion du cycle de vie d'un service :

<https://developer.android.com/guide/components/services.html>

Documentation sur les classes Java plus générales : Oracle - Java Platform SE 7 [en ligne]. Disponible sur : <http://docs.oracle.com/javase/7/docs/api/overview-summary.html>

Problème de l'accéléromètre qui se désactive sur certains téléphones lorsqu'ils se verrouillent

Stack Overflow - Android accelerometer not working when screen is turned off [en ligne]. Disponible sur : <http://stackoverflow.com/questions/9982433/android-accelerometer-not-working-when-screen-is->

turned-off

Sleep Tracking when screen off – Sleep As Android [en ligne]. Disponible sur :
<http://sleep.urbandroid.org/cs/documentation/core/sleep-tracking/>

Librairie open source de tracé de graphes sur Android

Jonas Gehringer – Android Graph View plotting library [en ligne]. Disponible sur : <http://www.android-graphview.org/>

Annexe A. Fiche d'identité du groupe

La séance de brainstorming a permis de dégager 3 grandes catégories de hobbies et passions pratiquées par les membres du groupe. Cette cartographie n'est pas exhaustive mais on constate :

- La place primordiale du sport

Le groupe est riche de pratiquants de sports divers et variés, avec des oppositions individuelles (boxe, tennis ou tennis de table) et collectives (handball, football, hockey...). Cette diversité pourra nous être utile pour mettre en commun nos différentes expériences sportives dans un éventuel sujet de projet.

- Un aspect social

Plusieurs personnes du groupe privilégiennent les activités visant à se sociabiliser. Ainsi on retrouve un goût pour le voyage ou encore les soirées festives. Cela passe aussi par le sport collectif évoqué précédemment. Cette direction est à exploiter.

- Un goût pour différents types d'art

Ce groupe se compose à la fois de consommateurs réguliers de musique et de personnes jouant d'un instrument, ce qui est assez complémentaire. Cette même complémentarité se retrouve dans le milieu plus général de l'audiovisuel avec des consommateurs réguliers de séries télévisées d'une part, et des passionnés de création (vidéo, photo, pilotage de drones, simulation d'environnement virtuel) d'autre part. La lecture (littérature fantastique notamment) vient se rajouter aux hobbies artistiques mentionnés.

Annexe B. Modifications

B.1. Modifications de fond

Tableau des modifications de fond apportées au projet avec validation des experts et encadrant informatique

Libellé / date	Description brève	Validé par :
Début janvier : report de certaines fonctions Androïd	Décalage au PAN4 du QR code et du partage de photos car non réalisables dans un premier temps et non indispensables au fonctionnement.	Jean Le Feuvre
Décembre : Modification de l'architecture du projet	Les calculs sont effectués localement, le serveur sert juste de stockage et gestion de base de données. (cf schéma d'architecture)	Jean Le Feuvre

Tableau 4 : Modifications de fond

B.2. Modifications du rapport

B.2.1. Modifications du rapport au PAN3

La répartition des élèves par module a évolué, le tableau présent dans le rapport au PAN 1 était celui-ci :

Modules	Audio	Communication client-serveur	Androïd	SES	Test et Intégration
<i>Encadrant</i>	Bertrand David	Jean-Claude Dufourd	Jean-Claude Dufourd	Caroline Rizza	Soumia Dermouche
<i>Elève 1</i>	Leello Tadesse Dadi	Louis Combaldieu	Daniel Mayau	Robin Blanchard	P.E Tabarant
<i>Elève 2</i>	Lucie Moulinas	Michael Dasseville	Robin Blanchard	Louis Combaldieu	Daniel Mayau
<i>Elève 3</i>			Paul-Elian Tabarant	Lucie Moulinas	Louis Combaldieu
<i>Elève 4</i>				Leello Tadesse Dadi	
<i>Elève 5</i>				Paul-Elian Tabarant	
<i>Elève 6</i>				Michael Dasseville	
<i>Elève 7</i>				Daniel Mayau	

Tableau 5 : Ancien tableau de répartition par module

Le tableau détaillé des tâches a été modifié, voici la version présente dans le rapport au PAN 1

	PAN 3	PAN 4
Module Audio		
Récupérer des données accélérométriques	X	
Obtenir les transformées de Fourier des signaux	X	
Déterminer un critère de corrélation	X	
Déterminer si un utilisateur danse ou pas	X	
Récupérer du son depuis le lecteur du DJ		X
Module Androïd		
Création et lecture de QR codes	X	
Interface graphique	X	
Système de notification	X	
Mise en forme et envoi des données	X	
Visualisation esthétiquement plaisante des données		X
Page affichant en direct les données de la soirée		X
Module Client/Serveur		
Détermination de la nature des messages	X	
Stockage et envoi de données en temps réel	X	
Mise en place de la gestion des messages	X	
Mise en place d'un système d'amis	X	
Module SES		
Etude d'impact sur la vie privée	X	
Mise en place	X	
Intégration		
Intégration des modules	X	
Présentation		
Montage de la vidéo		

Tableau 6 : Ancien tableau détaillé des tâches

Les parties 5.1 et 5.2 ont évolué, voici leur version initiale :

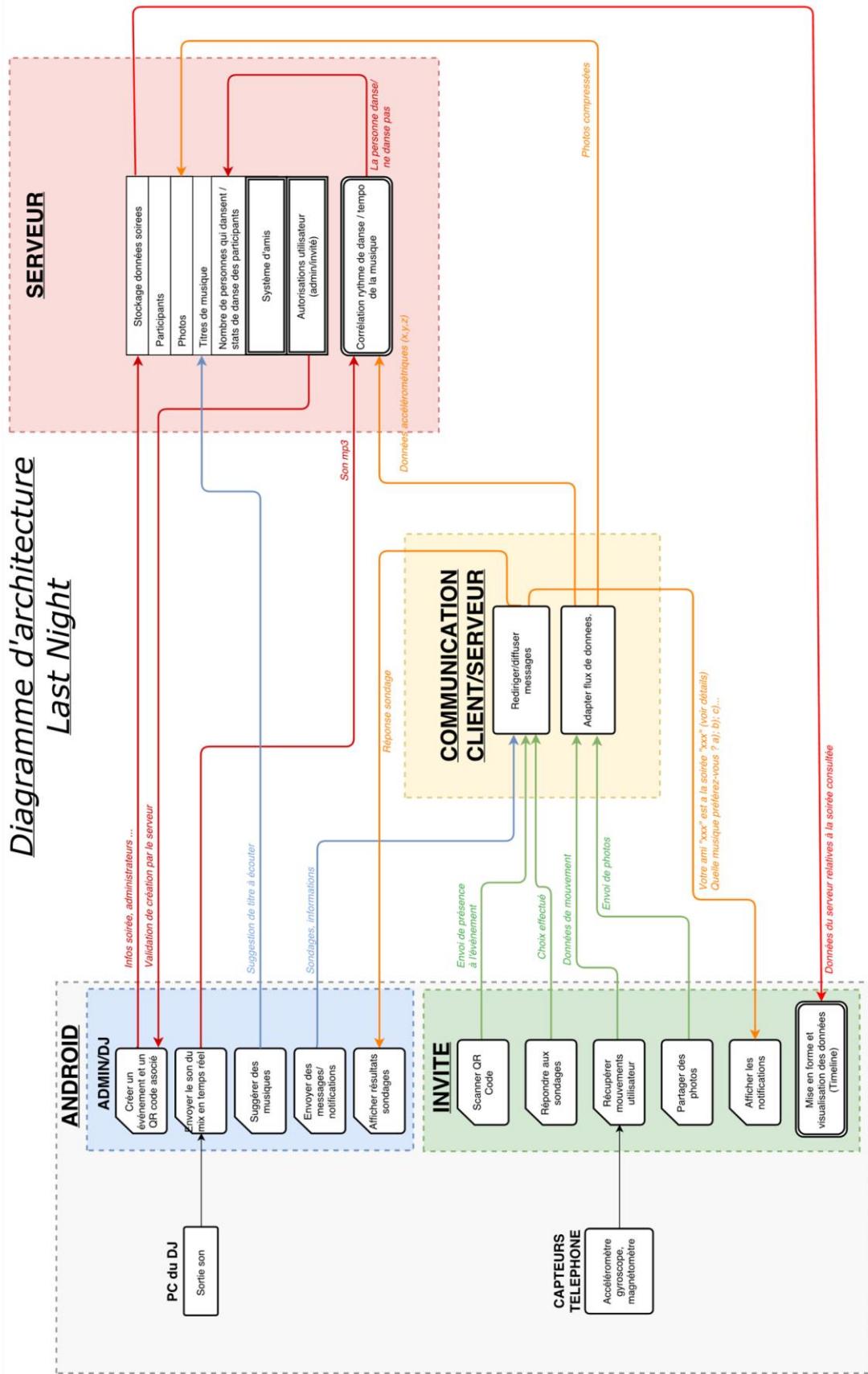


Figure 8 : Ancien diagramme d'architecture

Android - Application globale et Timeline

Menus utilisateur, écrans de connexion et envoi/réception de messages (sondages, notifications). Outil de génération et de lecture de QR Codes pour créer / rejoindre des événements. Suivi de l'activité de la personne avec les capteurs de mouvement.

L'ensemble des photos, des évènements et des activités est disposé sur une timeline que l'on pourra faire défiler pour visionner les moments forts de la soirée.

Communication client/serveur

Mettre en relation les utilisateurs de la soirée et le DJ. Ce bloc permet de formater et d'envoyer les données de la soirée vers le serveur, et de rediriger correctement les messages entre invités et DJ.

Serveur

Stocker les photos, les activités des utilisateurs et les musiques envoyées par le DJ sur une soirée. Système de comptes utilisateurs avec ajout d'amis.

Serveur (Audio) - Corrélation rythme de danse / tempo de la musique

Vérifier la corrélation entre les données accélérométriques d'un utilisateur et le tempo de la musique pour savoir si la personne est en train de danser.

Description des interfaces

Interface Android - Serveur de stockage

Requêtes et réponses

Le bloc Android pourra envoyer au serveur les informations relatives à la soirée ainsi que les noms des administrateurs. Le bloc serveur pourra devra confirmer les créations d'évènements.

Interface Android - Communication Client/Serveur

Requêtes et réponses

→Admin/DJ :

Le bloc Android pourra envoyer des informations et des sondages au bloc CCS afin qu'il les diffuse aux invités.

→Invité :

Le bloc Android pourra envoyer les indicateurs de présence à la soirée (obtenus grâce au QR Code), les réponses (individuelles) ainsi que les photos et les données du mouvement de l'invité au bloc CCS. Le serveur lui renverra en retour, une fois la soirée terminée, les données nécessaires à la génération d'une timeline.

Interface Communication Client/Serveur - Serveur de stockage

Le bloc CCS adaptera le flux sérialisé des photos vers l'espace de stockage du serveur. On utilisera un format de photo compressé.

Interface Communication Client/Serveur - Corrélation

Le bloc CCS pourra envoyer les données relatives au mouvement au serveur sous la forme de coordonnées accélérométriques (x,y,z). Le son joué par le DJ sera dans un premier temps (PAN3) stocké sur le serveur avec des analyses de tempo pré-calculées. Il sera dans un deuxième temps joué depuis le PC du DJ et envoyé avec une certaine période d'échantillonnage pour effectuer la corrélation audio/rythme en direct sur le serveur (PAN4)

Interface Serveur - Corrélation

Le bloc Corrélation pourra informer le bloc Serveur si la personne danse ou non par le moyen d'un message binaire.

B.2.2. Modifications du rapport au PAN4

Les plans de test du module Client/Serveur a été mis à jour.

Les comptes-rendus de réunions ont également été complétés avec les dates suivant le PAN3.

Chaque module a mis à jour son avancement et le rapport de suivi du travail effectué avec les derniers avancements post-PAN3, ainsi que les compléments bibliographiques associés.

Voici la version du tableau d'avancement du PAN3 :

Module	Avancement Prévu (%)	Avancement Réel (%)	Temps passé (h)	Description brève du travail effectué, analyse des écarts constatés
Intégration et test	70	40	32	<ul style="list-style-type: none">• Architecture du projet fixée et approuvée par l'ensemble des membres du groupe ainsi que notre tuteur• Interfaces Audio-Android-Serveur définies avec les modules concernés• Code de calcul du tempo des mouvements intégré sur les téléphones, premiers tests convaincants• Client de connexion intégré sur les téléphones avec classe de test préparée, en attente de la disponibilité du serveur pour les tests : prise de retard à ce niveau• Préparation du projet à intégrer sur le client PC du DJ

Audio	60 (90)	50 (70)	140	<ul style="list-style-type: none"> Détermination du BPM de données accélérométriques, et d'un fichier sonore. Nous voulions réaliser nos calculs en direct au cours de la soirée, ce qui n'est pas encore fait et explique les faibles pourcentages. Sans cela, le travail est presque terminé, le retard est lié à la détermination d'un critère universel indiquant si l'utilisateur danse.
Client-Serveur	75	70	130	<ul style="list-style-type: none"> Création d'une base de données pour stocker des informations relatives aux utilisateurs, événements, et administrateurs. Conception des requêtes SQL en Java pour gérer la base de données Administration du serveur distant dédié à notre projet, que nous avons loué. Conception d'une méthode de multi-threading pour connecter les utilisateurs au serveur. Conception d'une méthode d'envoi d'image entre les clients et le serveur. Conception d'un système de messagerie (pas fini). Conception d'un système de sondage (pas fini).
Android	80	70	120	<ul style="list-style-type: none"> Menu de l'application fonctionnel et complet, avec modèle MVC sommaire, en accord avec le story-board conçu Formulaires de création de comptes utilisateurs, événements, modifications à apporter et visualisation de données Service de capture des mouvements à durée et sûreté d'exécution contrôlée, écriture des résultats dans un fichier et méthode de lecture du fichier Outils de visualisation, édition et envoi de sondages/messages Capture de photos depuis la galerie ou le capteur photo du téléphone Décalage de la lecture de QR code (fonctionnalité secondaire et longue à implémenter) et de la génération de récapitulatif (corrélation pas encore disponible) au PAN4
Etude d'impact sur la vie privée	60	50	5	<ul style="list-style-type: none"> Le PIA (privacy impact assesment) est terminé, il faut maintenant s'occuper de l'étude de risque.

Annexe C. Comptes Rendus de réunions

Note: - *en orange sont indiquées les réunions en dehors des séances de PACT encadrées.*
- *Seules les reunions de groupe sont indiquées, pas les travaux individuels sur les modules*

- **Lundi 26 septembre** : Première rencontre, journée animée par un comédien pour apprendre à travailler en groupe. Mise en évidence de l'importance de l'écoute, de la tolérance, et de la clarté d'expression.
- **Mardi 27 septembre** : Première séance avec notre tuteur et début de recherche de sujet. Confection de la carte représentant les goûts et hobbies des membres du groupe. Recherche de sujets à partir de 3 capteurs.

- **Jeudi 29 septembre** : A partir des cartes de capteurs et des centres d'intérêts communs relevés, discussion pour élaborer une dizaine de sujets. Rédaction d'une fiche de présentation du groupe.
- **Vendredi 30 septembre** : Présentation au tuteur des sujets préparés individuellement et collectivement. Test d'inclusion de capteurs pour se conformer au sujet. En fin de séance, 5 sujets retenus.
- **Jeudi 6 octobre** : Nouvelle discussion autour des sujets envisagés pour n'en conserver que 3. Définition plus précise de sujets et début d'élaboration des affiches pour la foire aux experts du mercredi 12.
- **Vendredi 7 octobre** : Travail sur les 3 affiches pour la rencontre avec les experts et préparation des questions à leur poser.
- **Mardi 11 octobre** : Finalisation des affiches et mise en forme des questions.
- **Mercredi 12 octobre** : Foire aux experts. Avis des experts sur nos sujets et réponses à nos questions portant sur la faisabilité technique. Discussion avec d'autres groupes pour obtenir des avis extérieurs. A l'issue de la foire, le sujet "LastNight" séduit la majorité des membres du groupe. Il est retenu, après discussion avec notre tuteur.
- **Lundi 17 octobre** : 2 rencontres dans la journée pour élaborer un scénario d'utilisation de notre application. Les discussions pour le construire nous permettent de clarifier le projet.
- **Vendredi 21 octobre** : Suite au retour sur notre scénario, nouvelle discussion pour le clarifier et le préciser. Réécriture.
- **Jeudi 26 octobre** : Deuxième retour sur le scénario d'utilisation : rapide mise au clair du sujet entre nous.
- **Lundi 7 novembre** : Premier cours de génie logiciel, définition des diagrammes à rendre. Demi-journée de rencontre avec les experts. Séance de mise au point collective pour savoir quels experts nous devons rencontrer, puis prise de rendez-vous. Rencontre avec :
 - *Caroline Rizza* pour le module SES concernant la protection de la vie privée. Ce module sera suivi par 2 responsables dans le groupe.
 - *Mastane Achab* pour le module de Machine Learning. Notre intention était d'arriver à déterminer le type de danse de l'utilisateur grâce aux données issues des capteurs de son téléphone, comparés à une base de données. Cela est réalisable dans le cadre de ce module.
 - *Yves Grenier* pour le module d'analyse sonore. Il sera compliqué de récupérer depuis l'ordinateur du DJ la musique jouée, une solution de type Shazam serait plus simple car elle ne dépend de la plateforme utilisée par le DJ. Elle serait en revanche difficile à programmer dans le cadre de PACT.
 - *Cyril Concolato* pour le module Smartphones et tablettes Androïd. Ce module nous permettra de construire les fonctions essentielles de notre application, ce que nous envisageons de mettre en place est tout à fait réalisable.
 - *Jacob Montiel* pour le module Serveurs et stockage de données. Ce module nous fournira la structure à adapter pour envoyer des requêtes au serveur, il nous appartiendra ensuite d'élaborer le code correspondant.

- *Jean-Claude Dufourd* pour les modules Communication Client Serveur, Visualisation de donnée et Andoïd. Il y aura du travail pour nous dans ces 3 modules, afin de créer l'application et de fournir à l'utilisateur une timeline de sa soirée.
- **Jeudi 10 novembre** : Retour sur les rendez-vous avec les experts afin de synthétiser les informations. Clarification de notre projet. La répartition en module commence à prendre forme. Début de construction des diagrammes demandés. Rendez-vous avec Bertrand David que nous n'avions pas pu rencontrer lundi : le module Audio qu'il propose devrait nous permettre de déterminer le rythme de la musique diffusée et celui des mouvements de l'utilisateur, afin de déterminer si celui-ci danse.
- **Lundi 14 novembre** : Construction collective des diagrammes d'activité et d'architecture à partir des instructions données par les responsables génie logiciel.
- **Jeudi 17 novembre** : Fin de l'élaboration des diagrammes demandés et discussion sur le projet : le machine learning est mis de côté dans un premier temps car il n'est pas indispensable à notre application et prendrait beaucoup de temps à développer.
- **Lundi 21 novembre** : 2^{ème} séance de génie logiciel, retour des autres groupes sur le diagramme d'architecture et explication des attentes pour le PAN1. Mini cours dans l'après-midi sur les modules afin d'en définir les objectifs. Création du dépôt git.
- **Mercredi 23 novembre** : Retour sur les mini-cours et les discussions de chaque binôme avec l'expert responsable. Les informations varient selon les sources, clarification du projet en conséquence. Passage de notre tuteur, discussion avec lui sur la répartition en modules, sur d'éventuelles nouvelles fonctionnalités pour l'application, et sur le PAN1.
- **Lundi 28 novembre** : Réunion de crise suite à l'annonce de la date du rendu pour le PAN1. Mise au clair du travail restant et début de répartition pour s'y mettre le soir même. La tâche est finalement moins compliquée qu'il nous semblait.
- **Mardi 29 novembre** : En présence de tous le groupes, relecture du modèle de rapport pour répartir efficacement les rôles. Début de travail en groupe sur différentes parties.
- **Mercredi 30 novembre** : Clarification des tâches pour le tableau à remplir et le diagramme de Gantt.
- **Jeudi 1 décembre** : Finalisation collective du rapport pour le PAN1.
- **Lundi 5 décembre** : Passage à l'oral pour le PAN1
- **Mercredi 7 décembre** : Début du travail sur les modules
- **Lundi 12 décembre** : Avancement des modules
- **Jeudi 15 décembre** : Avancement des modules
- **Lundi 9 janvier** : Avancement des modules
- **Mercredi 11 janvier** : Avancement des modules
- **Jeudi 26 janvier** : Mise au point collective sur l'avancement des modules et les modifications apportées au projet

- **Lundi 30 janvier** : Passage des présentations individuelles pour le PAN 2
- **Lundi 6 février** : Travail sur les modules et passage des derniers oraux PAN 2
- **Lundi 20 février** : Travail sur les modules
- **Jeudi 23 février** : Début de préparation du PAN 3, répartition des tâches pour remplir le rapport d'avancement.
- **Jeudi 2 mars** : Avancement des modules et finalisation du rapport pour le PAN 3
- **Lundi 6 mars** : Présentation collective du PAN3, puis débriefing et rappel global des tâches prioritaires pour le PAN4
- **Lundi 13 mars** : Avancement des modules
- **Lundi 20 mars** : Bilan des apports personnels et collectifs du projet avec notre tuteur, recommandations importantes concernant le PAN4
- **Lundi 27 mars** : Avancement des modules et planification du PAN4 :
 - Réflexion sur les scènes, personnages et la forme de la vidéo
 - Planification de dates de tournage pendant la semaine de campagne BDE
- **Jeudi 30 mars** : Réunion de préparation du PAN4
 - Ecriture du script définitif de la vidéo
 - Brainstorming pour le poster et accord de l'ensemble du groupe sur sa forme
 - Partage des tâches (vidéo, poster, rapport et présentation orale, préparation du stand)
- **Mardi 4 avril** : Premier tournage des scènes de la vidéo (1^{ère} et 5^{ème} scènes)
- **Jeudi 6 avril** : Tournage des scènes restantes lors de la soirée de la liste VBV
- **Mercredi 19 avril** : Finalisation collective de la vidéo et du rapport
- **Jeudi 20 avril** : Arrangements sur le diaporama de présentation orale, puis entraînement des personnes concernées. Organisation de la journée de démos dans le Hall Barrault Lundi 24 avril

Annexe D. Suivis des modules

D.1 Module Android

Module Android

Groupe 4.1

Elèves responsables : Robin Blanchard, Paul-Elian Tabarant et Daniel Mayau

Encadrant : Jean-Claude Dufour

Descriptif

L'utilisation de smartphones ou de tablettes est devenue depuis quelques années presque incontournable. Parmi les différents systèmes d'exploitation permettant de contrôler ces smartphones et tablettes, le système Android représente près de la moitié des périphériques. La programmation des smartphones ou tablettes Android fait appel au langage Java, vu lors du cours INF103. Elle ne devrait pas poser de difficulté majeure, néanmoins une des difficultés souvent rencontrées est la prise en main de l'environnement de développement et des spécificités des applications Android. Ce module a pour vocation d'aider les élèves à surmonter cette difficulté.

Objectifs d'apprentissage

Prendre en main l'environnement de développement Android, comprendre le fonctionnement d'une application Android, savoir lire et utiliser la documentation des API Android, savoir créer une application Android de base et utiliser certaines fonctionnalités avancées (affichage graphique, réseau, multimédia ...), savoir debugger une application

Résultats attendus

→ PAN 1:

Démontrer que l'environnement de développement est en place et que des applications simples fonctionnent sur un émulateur

Expliquer le cycle de vie d'une application et la notion d'activité

→ PAN 2:

Réaliser une application Android sur émulateur mettant en oeuvre le cycle de vie, l'enchaînement de plusieurs activités, y compris en tâche de fond, et le lancement d'une autre application.

→ PAN 3:

Réaliser une application spécifique (à définir en fonction des besoins du projet). Cette application fonctionnera sur un smartphone ou une tablette et pourra mettre en oeuvre: la communication via le réseau, par exemple selon le protocole HTTP, un affichage graphique simple avec quelques éléments interactifs, la lecture ou la capture de données multimédia (son, image, vidéo) Décrire le fonctionnement et l'intégration du module dans le prototype allégé

→ PAN 4:

Analyser comment le module est intégré dans le prototype, quelles pistes d'améliorations seraient à envisager (performance, simplicité)

Menu et navigation dans l'application

Cette partie s'est décomposée en 2 sous-parties :

- Solution technique du menu
- Stockage et gestion de l'état actuel de l'application à l'aide d'un modèle vue-contrôleur basique

Solution technique du menu

Pour une première version de notre projet, nous avons choisi de gérer notre menu unique à l'aide de la Toolbar Android. Il se présente de cette manière :

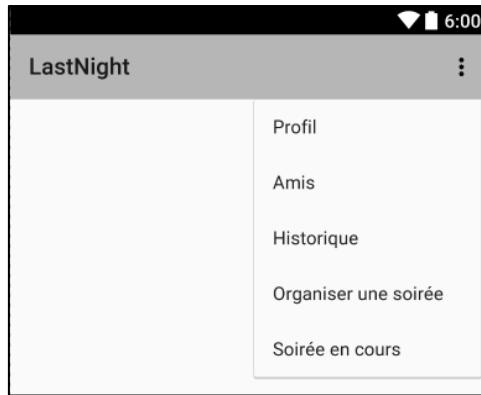


Figure 9 : Menu de l'application

Cette Toolbar est visible sur toutes les activités racines du menu. Afin de générer automatiquement ce menu sur toutes les activités concernées, nous avons choisi de respecter l’architecture suivante :

Ainsi pour générer le menu présenté ci-dessus, il suffit de déclarer que l’activité concernée hérite de `MenuActivity`.

Stockage et gestion de l’état courant de l’application

Lorsque l’utilisateur n’est pas à une soirée, un appui sur l’item « soirée en cours » du menu doit le rediriger vers l’outil de scan du QR code afin qu’il puisse, s’il le souhaite, rejoindre une soirée. Dans le cas contraire (l’utilisateur a rejoint un événement en cours), un appui sur « soirée en cours » doit rediriger directement sur la page de la soirée avec les options d’édition de sondages et de messages.

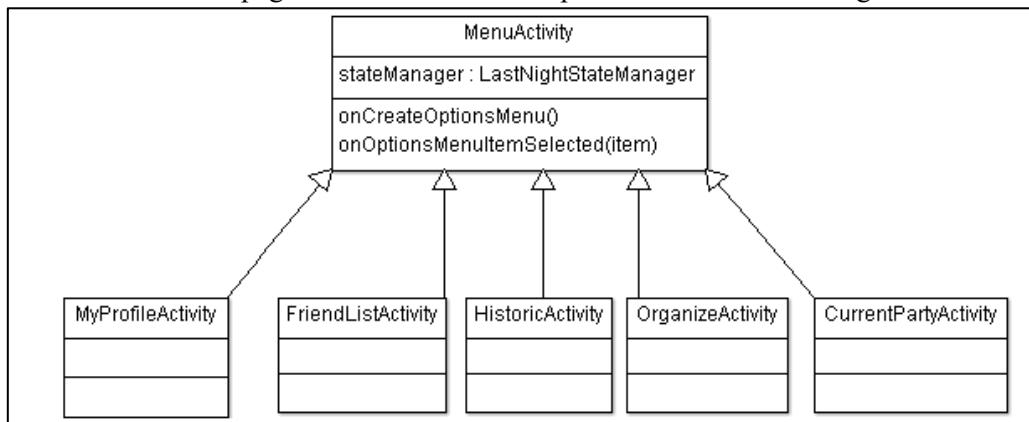


Figure 10 : Modélisation des activités du menu de l’application

C’est ce type d’exemple qui nous a incité à stocker l’état courant de l’application dans un objet qui joue un rôle de « modèle » dans l’architecture MVC, mis à jour avec les actions de l’utilisateur dans l’appli et les données du serveur. Nous avons donc créé une classe `LastNightStateManager` dont une instance est créée à chaque fois que l’application est lancée sur un téléphone, et sa référence est stockée en attribut de l’instance de l’application elle-même (cf. figure ci-dessous).

```

public class LastNight extends Application {
    private LastNightStateManager appStateManager;
    public LastNight() {
    }
    super();
    appStateManager = new LastNightStateManager();
}
public LastNightStateManager getAppStateManager() {
{
    return appStateManager;
}
}
  
```

Figure 11 : Stockage de l'état de l'application

Ainsi, on utilise notamment cet objet pour stocker l'état « soirée en cours » associé à l'utilisateur. Cet objet est consultable en récupérant le contexte d'exécution de l'application via `getApplicationContext()`. Il est modifiable grâce à des méthodes utilisables par toutes les activités et les services lorsque c'est nécessaire (ex. l'utilisateur vient de rejoindre une soirée). Avant d'ouvrir un volet du menu, l'activité vérifie l'état de l'application grâce à cet objet pour pouvoir lancer la bonne activité.

Un problème de fonctionnement nuisant au fonctionnement de l'application a été constaté après le PAN3 à ce niveau : lorsque l'écran se verrouille, le système Android passe en mode économie de batterie et libère de la mémoire en supprimant le contexte d'exécution de l'application. Afin de pouvoir le restaurer une fois l'application relancée, il était nécessaire de le stocker sous forme statique.

Nous avons pour cela utilisé l'outil « Préférences partagées » d'Android permettant de stocker des valeurs de variables dans un fichier lié à l'application. Ce fichier est accessible pour le programmeur sous la forme d'un objet `SharedPreferences` fonctionnant avec un système « clé/valeur ». Il suffisait donc d'associer à chaque variable d'état une clé spécifique pour pouvoir récupérer sa valeur dans le fichier de préférences, et la sauvegarder en écrivant dans ce fichier. Ainsi l'objet `LastNightStateManager` est initialisé grâce aux valeurs qui y sont stockées et l'application n'est pas impactée par l'optimisation mémoire du système Android.

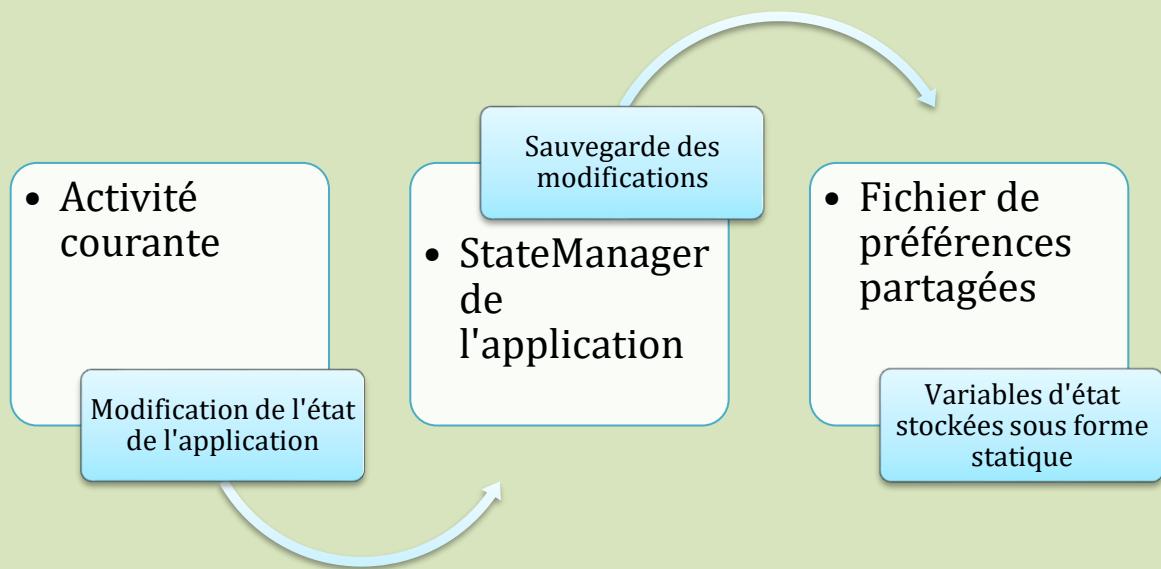


Figure 12 : Mécanisme opéré dans l'application lors d'un changement d'état
(ex : l'utilisateur rejoint une soirée)

Avec ce système, l'objet `LastNightStateManager` fait office d'intermédiaire entre les activités de l'application et le fichier de préférences partagées : il encapsule toutes les procédures d'écriture et de lecture dans ce fichier, qui sont totalement transparentes du point de vue des activités. De plus, il permet de limiter les accès en lecture en stockant les valeurs de ce fichier dans ses attributs.

L'exemple de récupération des attributs `partyInProgress` (indique si l'utilisateur est actuellement à une soirée) et `partyName` (le nom de cette soirée si elle existe) est visible ci-dessous.

```

partyInProgress = lastNightSharedPref.getBoolean(PARTY_IN_PROGRESS_KEY,
false);
partyName = lastNightSharedPref.getString(CURRENT_PARTY_NAME_KEY, null);

```

Figure 12 : Initialisation de deux attributs du StateManager de l'application à partir du fichier de préférences partagées

Formulaires et visualisation de données

L'application doit implémenter un certain nombre de formulaires de saisie permettant à l'utilisateur de compléter son profil, créer un compte ou organiser une soirée. Par exemple, la page d'accueil de l'application comporte un bouton « Créer un nouveau compte » qui le redirige vers un formulaire d'inscription.

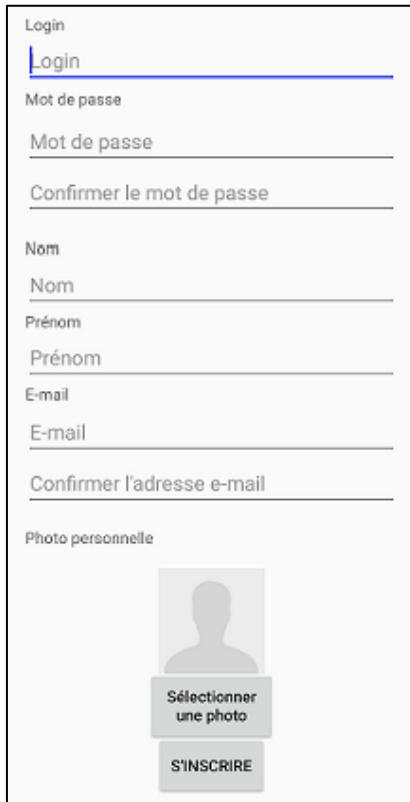


Figure 13 : Activité d'inscription

Il s'agit d'un exemple d'activité qui n'hérite pas de `MenuActivity` car, l'utilisateur n'étant pas connecté, il n'a pas encore accès aux options du menu. Le layout d'inscription a été fait au moyen d'un widget `ScrollView` permettant le défilement vertical des informations pour éviter les dépassements.

Cliquer sur le bouton « S'inscrire » finalise l'inscription. Il est vérifié au préalable que le e-mail est bien valide, que les e-mails inscrits sont bien les mêmes, et que les deux mots de passe sont bien les mêmes. Les messages `Toast` permettent de communiquer la source de l'erreur à l'utilisateur ou de confirmer la prise en compte des informations saisies.



Figure 14 : Exemple de message `Toast` obtenus après une saisie

L'utilisateur a également accès, via l'application, à sa liste d'amis et à l'historique des soirées auxquelles il a participé.

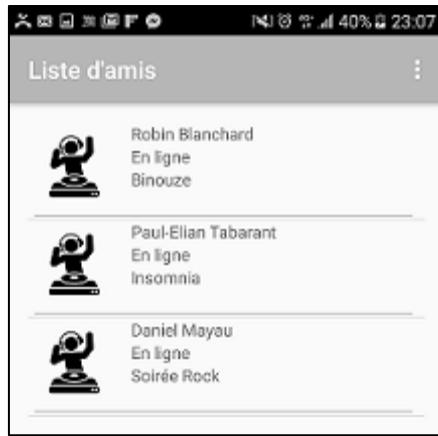


Figure 15 : Liste d'amis

Cette tâche a demandé un peu plus de compétences que pour les widgets classiques, à savoir un développement d'interface à deux niveaux.

Le premier est l'élément liste de base, la `ListView`, qui occupe l'écran de l'activité. Le deuxième correspond à un élément de la `ListView`, décrit dans un fichier de ressource à part entière et nous donnant la possibilité de répéter un motif complexe sur toute la liste à l'aide de widgets de base. Dans le cas de la liste d'amis, il est constitué d'une photo de profil de l'ami, de son nom d'utilisateur, ainsi que la soirée à laquelle il est présent s'il y en a une.

L'affectation des éléments individuellement à chaque item de la liste a nécessité l'utilisation d'objets `Adapter` faisant office de modèle pour la liste, en identifiant individuellement la valeur à affecter à chaque élément par une clé unique à l'aide d'une table de hachage (cf. figure ci-dessous).

```
map=new HashMap<String, String>();
map.put("friend_name", "Robin Blanchard");
map.put("friend_online_unavailable", "En ligne");
map.put("friend_current_party", "Binouze");
map.put("friend_pp", String.valueOf(R.drawable.app_logo));
list_of_friends.add(map);
```

Figure 16 : Remplissage de la table de hachage avant affectation des valeurs à un élément de la liste

Capture de photos depuis la galerie ou le capteur photo du téléphone

Afin de compléter son profil, les informations concernant une soirée qu'il organise ou partager une photo dans l'album de l'événement, l'application doit implémenter une fonction de capture photo permettant à l'utilisateur de sélectionner une photo depuis sa galerie locale ou en prendre une directement avec son appareil.

Pour revenir à l'exemple de l'inscription, appuyer sur le bouton « Sélectionner une photo » ouvre une `AlertDialog` permettant de choisir comment la photo sera récupérée.



Figure 17 : fenêtre de dialogue du choix de la source photo

Cliquer sur l'une ou l'autre des deux options crée un nouvel Intent et le démarre grâce à la fonction `startActivityForResult`, permettant de lancer l'application utilisée par défaut par l'utilisateur pour prendre une photo ou sélectionner une photo depuis la galerie et contrôler le résultat des actions de l'utilisateur dans l'activité de l'application externe.

```
public void onClick(DialogInterface dialog, int which) {

    if(picture_choice[which]=="Camera") {
        Intent cameraIntent = new Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
        startActivityForResult(cameraIntent, PHOTO_RESULT);
        dialog.dismiss();
    }

    else if (picture_choice[which]=="Galerie") {
        Intent galleryIntent = new Intent(Intent.ACTION_PICK,
        android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
        startActivityForResult(galleryIntent, GALLERY_RESULT);
        dialog.dismiss();
    }
}
```

Figure 18 : Programme permettant de sélectionner une source de photo

En fonction de la nature de la requête (galerie ou appareil photo), on implémente une manière de récupérer l'objet Bitmap dans l'activité d'inscription. Dans le cas de la caméra, il suffit de récupérer la photo en tant que Bitmap grâce à la fonction `Intent.getExtras()` et d'utiliser la fonction `ImageView.setImageBitmap(Bitmap)` pour charger la photo dans l'`ImageView` composant l'activité.

Dans le cas de la galerie, nous avons remarqué qu'en fonction de comment la photo avait été prise auparavant (caméra avant, arrière, téléchargement ou screenshot), la photo était tournée de +- 90°. L'orientation de la photo est récupérée grâce à `MediaStore.Images.Media.ORIENTATION`. La photo est remise à l'endroit grâce à un objet de type Matrix décrivant à l'objet Bitmap la rotation à effectuer.

```

Matrix matrix = new Matrix();

matrix.postRotate(orientation);

try {

    Bitmap bitmap =
    MediaStore.Images.Media.getBitmap(this.getContentResolver(), imageUri);

    Bitmap rotatedBitmap = Bitmap.createBitmap(bitmap, 0, 0,
    bitmap.getWidth(), bitmap.getHeight(), matrix, true);

    selected_picture.setImageBitmap(rotatedBitmap);

}

```

Figure 19 : Principe d'ajustement de l'orientation sur une photo Bitmap

Service de capture des mouvements

Quand l'utilisateur rejoint une soirée en cours, le suivi de ses mouvements doit démarrer et être enregistré dans un fichier tout au long de la soirée quoi qu'il fasse sur l'application. Ce besoin a demandé la création d'un service qui utilise l'accéléromètre du téléphone. L'utilisateur est d'abord averti du lancement du service de suivi par un message **Toast**.

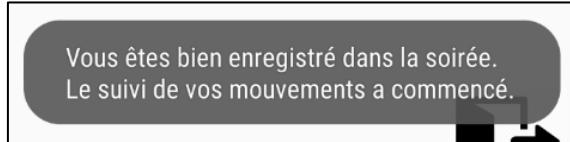


Figure 20 : Message affiché lorsque l'utilisateur vient de rejoindre une soirée via l'application

Une fois le service en route, afin d'éviter que le système Android tue le service pour une raison ou une autre, nous avons utilisé la méthode `startForeground()` qui permet de forcer son exécution et d'afficher une notification permanente sur l'écran de l'utilisateur pour lui indiquer que le service est en route.

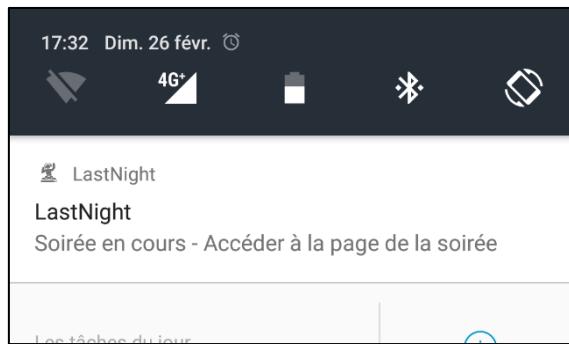


Figure 21 : Notification permanente affichée lorsque le service fonctionne

A partir de cet instant, seuls deux événements peuvent arrêter l'exécution du service :

- L'utilisateur décide de quitter la soirée
- L'horaire de fin de la soirée a été atteint

Dans le premier cas, il suffit d'implémenter un `Listener` sur le bouton « quitter » de l'activité gérant la soirée en cours, qui réagit en arrêtant le service.

Le second cas a demandé d'utiliser le service d'alarme du système Android, accessible depuis l'objet `AlarmManager`, qui permet d'effectuer une action particulière à une heure donnée grâce à la méthode `setExact()`. Ainsi en précisant au service d'alarme la date exacte de fin de la soirée et l'action à effectuer, on peut arrêter le service de suivi des mouvements à l'instant voulu.

L'implémentation du service contient les fonctionnalités suivantes :

- Accès à l'accéléromètre et acquisition des valeurs à chaque échantillonnage
 - Ecriture des coordonnées (x , y , z) d'accélération dans un fichier, pour les tests du module audio dans la mémoire externe mais à terme dans la mémoire interne (les 2 solutions sont implémentées), avec la date exacte d'acquisition en millisecondes.
 - Suivi des exceptions à l'aide de logs personnalisés
 - Calcul de la période moyenne d'échantillonnage sur la durée de l'acquisition et écriture du résultat en tête du fichier de sauvegarde
 - Lecture du contenu du fichier à la fin de l'acquisition et affichage dans la fenêtre de logs
- Afin de limiter la taille du fichier au maximum, étant donné que l'on manipule des données simples (entiers, flottants), nous avons utilisé un `DataOutputStream` afin d'écrire les valeurs d'accélération directement par leur représentation binaire (`writeFloat()` et `writeInt()`).

Avec cette solution, une capture de 5 minutes des mouvements avec la date associée donne un fichier d'environ 87 kB, ce qui donne un ordre de grandeur d'environ 1 MB pour 1h de suivi.

La solution exposée ci-dessus enregistre les données accélémétriques de l'utilisateur dans un fichier pour pouvoir ensuite effectuer des calculs de tempo et de score sur celles-ci. Dans la version finale de l'application, on souhaite suivre l'activité de l'utilisateur au cours de la soirée en calculant régulièrement un score de danse (module Audio) à partir de ces données. Il a donc fallu intégrer les fonctions de calcul audio au sein du service pour effectuer un calcul de score à intervalle de temps régulier.

Attributs de SensorService :

```
timeArray<long>
liste contenant les instants de chaque mesure en ms

aXArray<long>, aYArray<long>, aZArray<long>
listes contenant les valeurs d'accélération associées

long tDernierCalcul
variable stockant l'instant auquel le dernier calcul de tempo a été effectué

onSensorChanged (aXMesurée, aYMesurée, aZMesurée) :
    - long aX, aY, aZ = aXMesurée, aYMesurée, aZMesurée
    - long t = heure courante de L'horloge Android
    - Si (t - tDernierCalcul < 10 secondes)
        o Ajouter t à timeArray
        o Ajouter aX à aXArray, aY à aYArray, aZ à aZArray
    - Sinon
        o long score = Audio.score(timeArray, aXArray, aYArray, aZArray)
        o tDernierCalcul = t
        o Vider timeArray, aXArray, aYArray, aZArray
        o Ecrire score dans fichier_récapitulatif.txt
```

Figure 22 : pseudo-code du principe de calcul du score (onSensorChanged appelée à chaque fois que l'accéléromètre effectue une nouvelle mesure)

Cette solution fonctionne correctement, mais d'un point de vue pratique le système Android met en pause les services d'arrière-plan lorsque le téléphone est verrouillé pendant une certaine durée, de manière à économiser sa batterie. Il a donc fallu déclarer explicitement que nous utilisions le processeur

du téléphone lorsque le téléphone est en mode veille. Pour cela, le système dispose d'un objet `WakeLock` instanciable depuis la classe statique `PowerManager` qui regroupe toutes les fonctions permettant de gérer l'utilisation de la batterie au sein de notre application.

Une fois l'objet `WakeLock` créé et référencé au sein du service, il suffit d'appeler la méthode `acquire()` de cet objet pour indiquer que toutes les instructions qui vont suivre nécessiteront que le service fonctionne en arrière-plan même lorsque le téléphone est verrouillé. Une fois le suivi des mouvements terminé (l'utilisateur quitte la soirée), un appel à `release()` permet d'indiquer que l'application peut à nouveau passer en mode sommeil après verrouillage.

Il est cependant important de souligner que sur certains appareils, le service de mesure des données accélérométriques est lui-même mis en sommeil lorsque le téléphone passe en verrouillage. Ceci est un problème dont semblent souffrir toutes les applications Android basées sur un suivi permanent des données fournies par l'accéléromètre (cf. bibliographie).

La grande majorité des téléphones actuels du marché n'ont pas ce problème, mais dans une optique d'utilisation plus globale de l'application il conviendrait de penser à une solution alternative. La plus évidente semble être de conserver l'activation de l'écran pendant que le suivi est actif, avec une luminosité affaiblie afin d'éviter une surconsommation de batterie. Ceci peut être fait assez facilement sous Android en fournissant la constante `SCREEN_DIM_WAKE_LOCK` au constructeur de la classe `WakeLock`, au lieu de `PARTIAL_WAKE_LOCK` utilisé dans notre cas pour garder uniquement le processeur actif.

Le service de suivi des mouvements implémente également le partage du score vers le serveur toutes les 10 secondes, ce qui permet au DJ d'afficher le classement des invités en fonction de celui-ci et le score moyen en live. La communication avec le serveur sera mise en lumière ci-après.

Options utilisateur pendant une soirée – sondages et messages

Après avoir rejoint un événement en cours, l'utilisateur dispose d'une interface personnalisée qui remplit toutes les fonctions lui permettant d'interagir avec l'organisateur, le DJ et poster des photos dans l'album de l'événement.

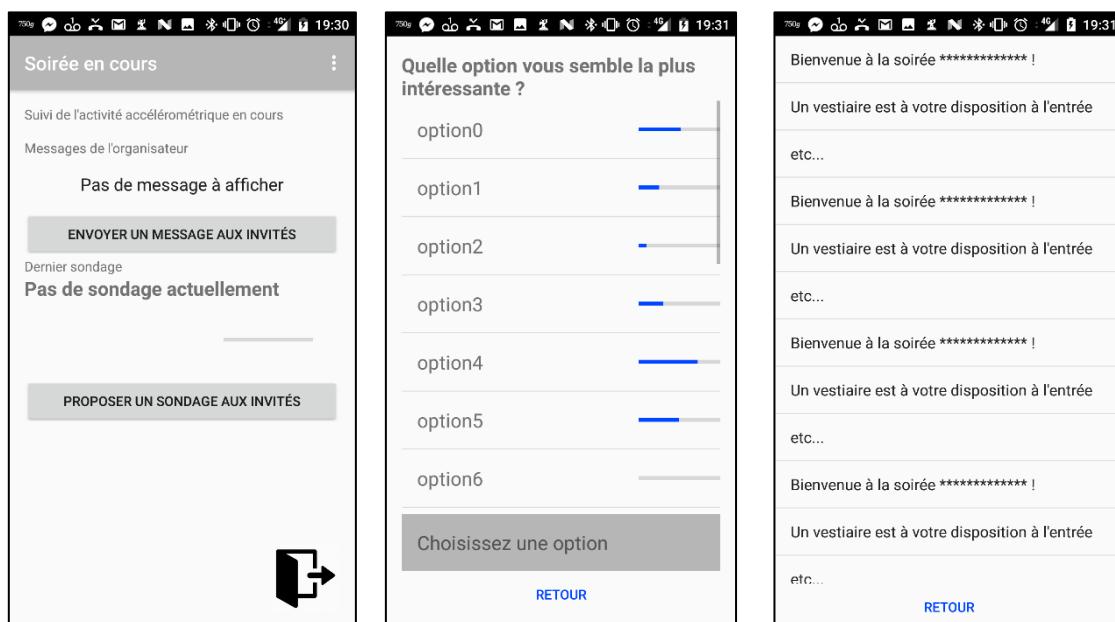


Figure 23 : De gauche à droite : fenêtre principale, dernier sondage, liste des messages

Notre application contient pour le moment les fonctionnalités de visualisation et d'édition de sondages et de messages, avec les interfaces associées.

La fenêtre de base de la soirée, une fois les méthodes de connexion à la base de données opérationnelles, affichera uniquement le dernier message de l'organisateur et l'option gagnante du sondage en cours. Il sera ensuite possible de développer la liste des messages et ainsi visualiser tous les messages postés par l'utilisateur depuis le début de la soirée. On pourra également afficher toutes les options du sondage en cours pour donner sa réponse.

Les sondages ont été modélisés par une liste (ListView), chaque élément étant composé d'un titre et d'une barre de progression indiquant le pourcentage de personnes ayant voté pour l'option considérée. La liste des messages utilise le même composant avec simplement une chaîne de caractères.

Appuyer sur « Proposer un sondage aux invités » ouvre une nouvelle activité, celle d'édition de sondages. Elle est constituée à la base d'un bouton « + » situé en haut de la page et un bouton « valider » en bas de page. Elle comporte aussi un ListView pour l'instant vide (cf. figure ci-dessous). Appuyer sur le bouton « + » ouvre une fenêtre de dialogue permettant à l'utilisateur de saisir une option de sondage et de valider l'ajout de l'option. Appuyer sur « créer une option » permet d'ajouter une option à la ListView du sondage. Ajoutons trois options à la ListView.

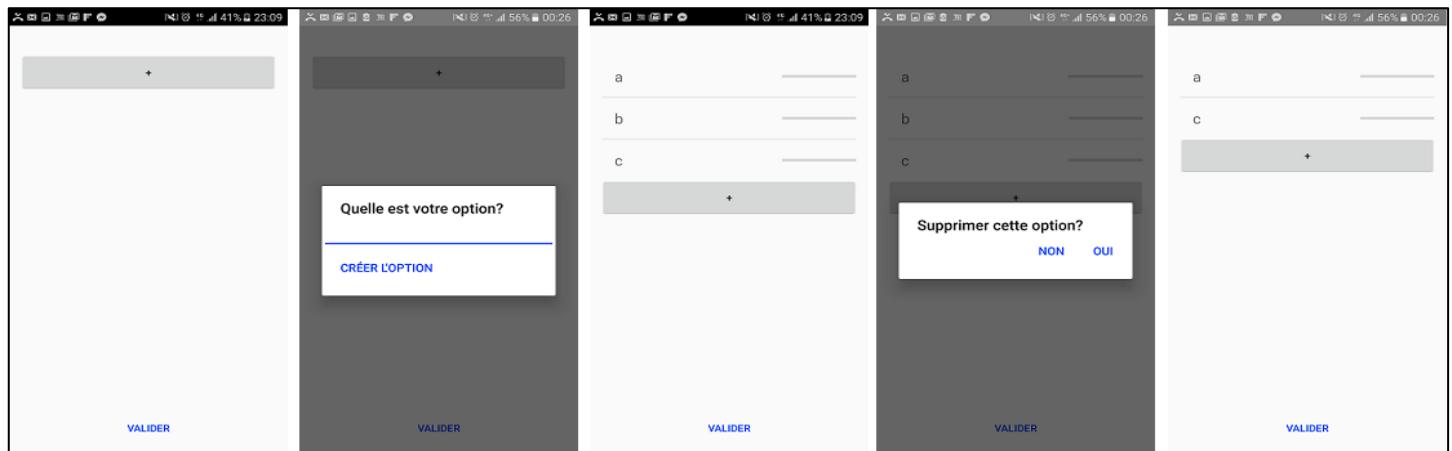


Figure 24 : Sondages

Cliquer sur une des options ouvre une nouvelle fenêtre de dialogue proposant de supprimer l'option sur laquelle on a cliqué, avec une fenêtre de validation permettant d'annuler le choix si nécessaire. L'exemple ci-dessus illustre la suppression de l'option « b » du sondage.

En appuyant sur le bouton « Envoyer un message aux invités », une boîte de dialogue s'ouvre et propose un outil sommaire d'édition de messages avec validation.



Figure 25 : Edition de message

Communication avec le serveur – intégration du client de connexion

Afin d'implémenter la communication avec le serveur LastNight, le binôme Client/Serveur nous a fourni une classe MyClient servant d'intermédiaire de connexion. Chaque requête à envoyer au serveur se fait en appelant la méthode qui lui est dédiée dans la classe MyClient et la valeur de retour de cette méthode correspond à la réponse du serveur à la requête. Chaque objet de type MyClient possède une référence vers un Socket de connexion. Le travail a donc principalement consisté à trouver une solution pour intégrer cette classe dans l'architecture de notre application. Les principales contraintes à respecter sont les suivantes :

- Connexion accessible depuis **n'importe quelle activité**
- **Un seul client de connexion** pour toutes les activités (seule l'activité active à l'écran utilise le réseau à un instant donné)
- Méthodes d'échanges réseau exécutées en **arrière plan** (en-dehors du thread UI) : contrainte du système Android (`NetworkOnMainThreadException`)

Afin que chaque activité puisse envoyer les requêtes qui la concernent vers le serveur, nous avons choisi d'instancier un objet MyClient dans le `LastNightStateManager` (objet qui modélise le contexte général d'exécution de l'application). Cela permet d'avoir un seul client de connexion pour toute notre application, accessible depuis chacune des activités via une méthode `StateManager.getClient()`.

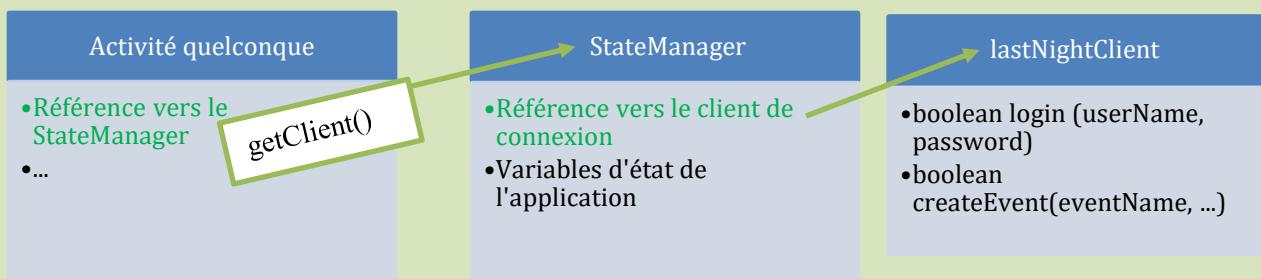


Figure 26 : Accès au client de connexion depuis une activité de l'application

Enfin, pour garantir l'exécution des opérations réseau en arrière-plan, nous avons utilisé la classe `AsyncTask`. Ce système est très simple de mise en œuvre et permet, en créant une classe héritant d'`AsyncTask`, d'à la fois isoler en arrière-plan les communications réseau en redéfinissant la méthode `doInBackground()` qui s'exécute dans un thread indépendant, et de communiquer les résultats au thread UI en redéfinissant `onPostExecute()` qui stockera automatiquement ce résultat en paramètre. Une fois ce résultat récupéré, on peut actualiser les widgets de l'activité en fonction de celui-ci.

```
private class LoginTask extends AsyncTask<String, Integer, Boolean>
{
    public Boolean doInBackground(String... params) {
        String login = params[0];
        String password = params[1];
        return getConnection().login(login, password);
    }

    public void onPostExecute(Boolean loggedIn) {
        if(loggedIn) {
            stateManager.notifyConnectionOf(login.getText().toString());
            goToMainMenu();
        }
        else {
            Toast.makeText(R.string.start_login_failed, Toast.LENGTH_SHORT).show();
        }
    }
}
```

```

        }
    }
}

```

Figure 27 : exemple d'une classe implémentant l'opération de login utilisateur en arrière-plan dans l'application

Récapitulatif de soirée

A partir de la liste des soirées affichées dans l'historique de l'utilisateur, il est nécessaire de charger le récapitulatif de la soirée sélectionnée. Celui-ci doit contenir l'évolution du tempo de ses mouvements et surtout de son score au cours de la soirée. Un système de fichiers sommaire a été mis en place pour retrouver facilement le récapitulatif : le service de suivi des mouvements l'enregistre sous la forme *<Nom d'utilisateur><Identifiant de la soirée>.txt*. Le nom de l'utilisateur connecté est récupéré depuis le **StateManager** de l'application et l'identifiant est communiqué par l'activité d'historique à l'activité de récapitulatif via son **Intent** de lancement.

Les graphiques d'évolution du tempo des mouvements (bpm) et du score sont ensuite chargés séparément, à l'aide de la librairie open source *Android Graph View Plotting Library* qui fournit des widgets de type **GraphView** voués au tracé de graphes.

La première valeur stockée dans le fichier de récapitulatif est la durée entre chaque calcul, écrite par le service de suivi à son démarrage. L'activité **PartyRecapActivity** se base directement sur cette valeur pour positionner les abscisses des points tracés. Ainsi, si l'on modifie cette durée dans le service, le récapitulatif s'adapte automatiquement. Les valeurs de tempo et de score suivent cette première valeur, directement stockés sous forme binaire.

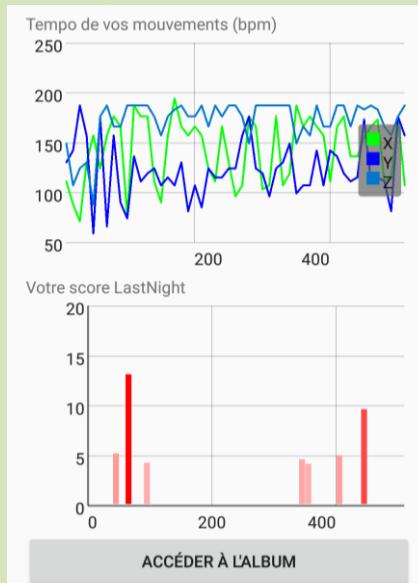


Figure 28 : exemple de récapitulatif d'une soirée peu mouvementée

Une dernière activité permet, depuis le bouton « Accéder à l'album » ci-dessus, de consulter l'album de la soirée considérée stocké sur le serveur. Avec l'activité de récapitulatif, il s'agit des deux seules activités rajoutées à l'architecture de l'application après le PAN3.

Compétences techniques acquises par au moins un membre du groupe

Compétences générales

- Edition de layouts pour présenter les vues des activités

- Utilisation de ressources partagées
- Gestion du cycle de vie d'une activité à l'aide des méthodes de callback `onCreate()`, `onResume()`, `onStop()`, `onDestroy()`
- Manipulation des widgets et affichage sur une activité donnée, réaction aux interactions de l'utilisateur avec ces widgets (**Listeners**)
- Articulation de l'application entre plusieurs activités ou autres composants (**Service**, **Receiver**...) grâce aux **Intent**, échange de données entre composants et contrôle du résultat d'exécution d'une activité (`.startActivityForResult()`)
- Génération et mise en forme d'un menu dans la **Toolbar** d'une activité

Compétences spécifiques

- Capture photo à l'aide d'**Intent** implicites (lancement de l'application appareil photo ou galerie)
- Manipulation de widgets plus complexes : **Dialog** et **ListView**
- Gestion du cycle de vie d'un service d'arrière-plan
- Gestion de l'accéléromètre du téléphone
- Lecture/écriture dans un fichier à l'aide de flux de données, en mémoire interne et externe
- Enregistrement de l'état de l'application dans un objet lié à l'instance de cette même application

Bibliographie

Tutoriel de base ayant servi de fil directeur dans notre formation

Frédéric ESPIAU - Créez des applications pour Android. Open Classrooms [en ligne]. Disponible sur : <https://openclassrooms.com/courses/creez-des-applications-pour-android>

Réponse à la plupart de nos problèmes spécifiques pendant la programmation

Stack Overflow [en ligne]. Disponible sur : <http://stackoverflow.com/>

Exemple d'utilisation pour stocker l'état courant de l'application :

<http://stackoverflow.com/questions/708012/how-to-declare-global-variables-in-android>

Documentation sur les classes Java liées à l'environnement Android

Android Developers [en ligne]. Disponible sur : <https://developer.android.com/index.html>

Exemple d'utilisation pour la gestion du cycle de vie d'un service :

<https://developer.android.com/guide/components/services.html>

Documentation sur les classes Java plus générales

Oracle - Java Platform SE 7 [en ligne]. Disponible sur :

<http://docs.oracle.com/javase/7/docs/api/overview-summary.html>

Problème de l'accéléromètre qui se désactive sur certains téléphones lorsqu'ils se verrouillent

Stack Overflow - Android accelerometer not working when screen is turned off [en ligne]. Disponible sur : <http://stackoverflow.com/questions/9982433/android-accelerometer-not-working-when-screen-is-turned-off>

Sleep Tracking when screen off – Sleep As Android [en ligne]. Disponible sur : <http://sleep.urbandroid.org/cs/documentation/core/sleep-tracking/>

Librairie open source de tracé de graphes sur Android

Jonas Gehring – Android Graph View plotting library [en ligne]. Disponible sur : <http://www.android-graphview.org/>

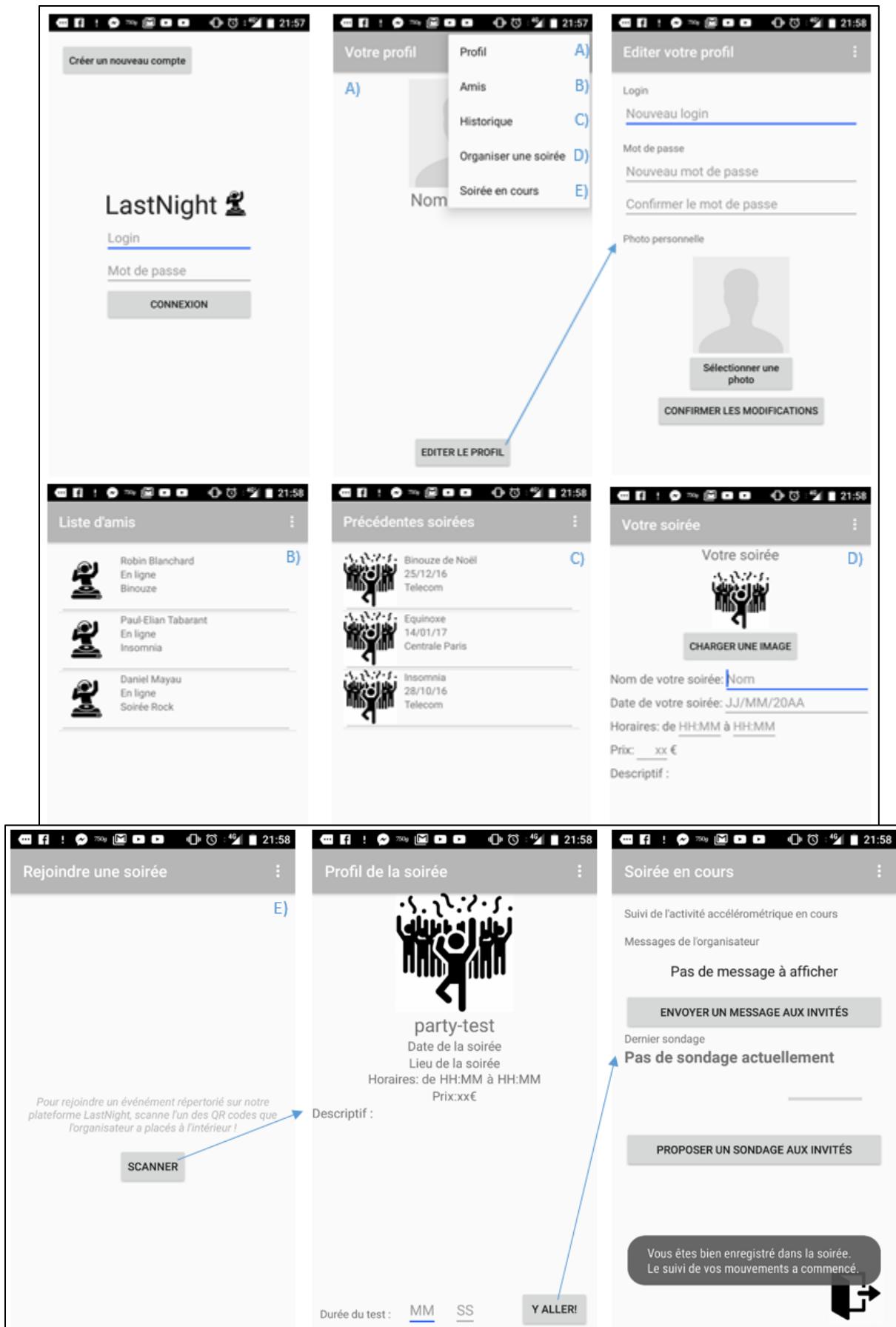


Figure 29 : Ensemble des vues de l'application

D.2 Module Audio

Module Audio : Détection/estimation de rythme et corrélation

Groupe 4.1

Elèves responsables : Leello Tadesse Dadi et Lucie Moulinas

Encadrant : Bertrand David

Descriptif

Le mot "attaque" désigne le début d'un évènement (sonore), ou l'instant où une note est jouée. Détecer ces attaques est un aspect fondamental nécessaire dans les applications de détection et d'analyse de rythme. On peut également l'étendre à toute forme de signal dont on possède une représentation temporelle. Si celle-ci présente des pics saillants mettant en évidence les attaques, on peut mettre en place une démarche de détection de rythme en 2 étapes : détection de l'enveloppe puis dérivation de celle-ci. Dans un cas plus général, la méthode d'autocovariance permet d'estimer la période d'un signal périodique ou quasi-périodique.

Objectifs d'apprentissage

A l'issu de ce module, comprendre la démarche de détection de rythme d'un signal (principes liés à la Transformée de Fourier, élaboration de filtres) pour la mettre en place dans notre application. Une fois déterminés les profils rythmiques de la musique diffusée et des mouvements d'un utilisateur, nous pourrons chercher s'ils sont corrélés afin de déterminer si l'utilisateur danse ou ne danse pas.

Résultats attendus

→ PAN 1:

Avoir lu les slides et visionné les vidéos OASIS Séries (<https://pact.wp.mines-telecom.fr/detection-de-tempo/>) (<https://bedavid.wp.mines-telecom.fr/enseignement/atiam/fpa-signal/>) . Savoir définir l'autocovariance.

→ PAN 2:

Choix de la méthode la plus efficace pour la détection de période. Description précise du pseudo code à implémenter dans Matlab pour réaliser une analyse de rythme, et étude des données récupérables par l'accéléromètre d'un smartphone.

→ PAN 3:

Code Java clair et commenté pour réaliser la détection de rythme, et test sur des données musicales ainsi que des données d'accéléromètre. Mise en œuvre du test de corrélation entre ces 2 types de données. Pouvoir déterminer ainsi les instants où une personne dansait.

→ PAN 4:

Enrichissement envisagé : comparer les périodes de mouvement de 2 utilisateurs, pour déterminer s'ils dansent ensemble. Déetecter en direct la période de la musique diffusée et des mouvements des utilisateurs, pour afficher le nombre de personnes en train de danser.

Fonctions sur Python

- Détection d'attaque : Dans un premier temps nous avons dû mettre en place un processus de détection d'attaque d'un signal. Pour cela il faut d'abord calculer son enveloppe en utilisant des fonctions de FFT de python, puis la dériver

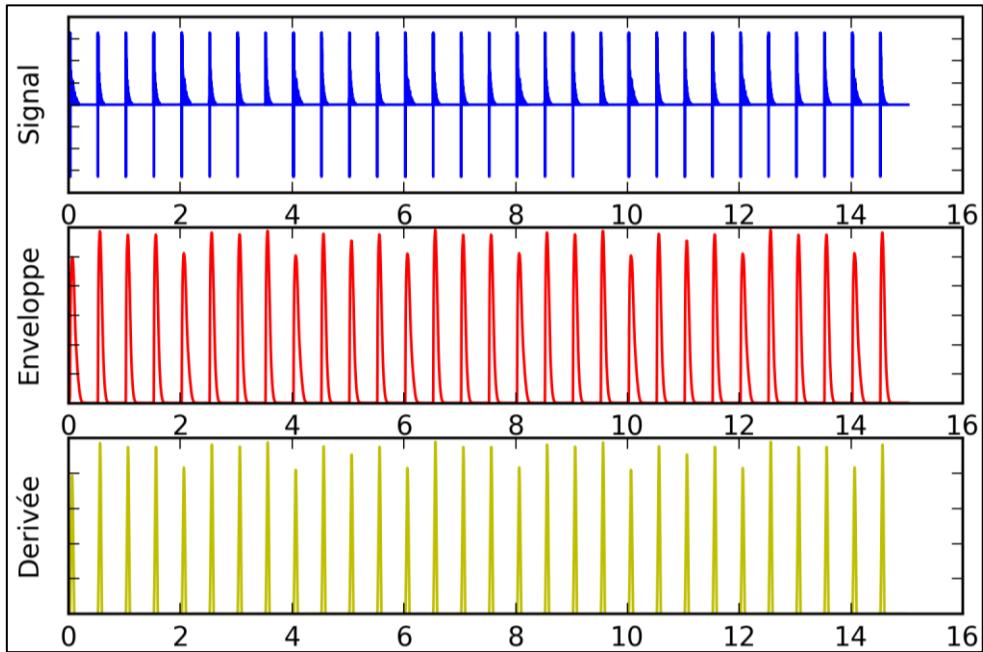


Figure 30 : Résultat de la détection d'attaque

- Détermination de BPM : Dans le cas d'un fichier audio au format wav, il est d'abord découpé en 6 plages de fréquences pour plus de précision. Les signaux obtenus sont ensuite passés dans un banc de filtre en peigne pour déterminer leur BPM. On somme alors les résultats pour chaque plage de fréquence. On calcule aussi le BPM d'une autre manière, par autocorrélation. Pour les données accélérométrique, la procédure est la même mais il n'est pas nécessaire de commencer par découper le signal car les fréquences présentes ne sont pas très étalées.

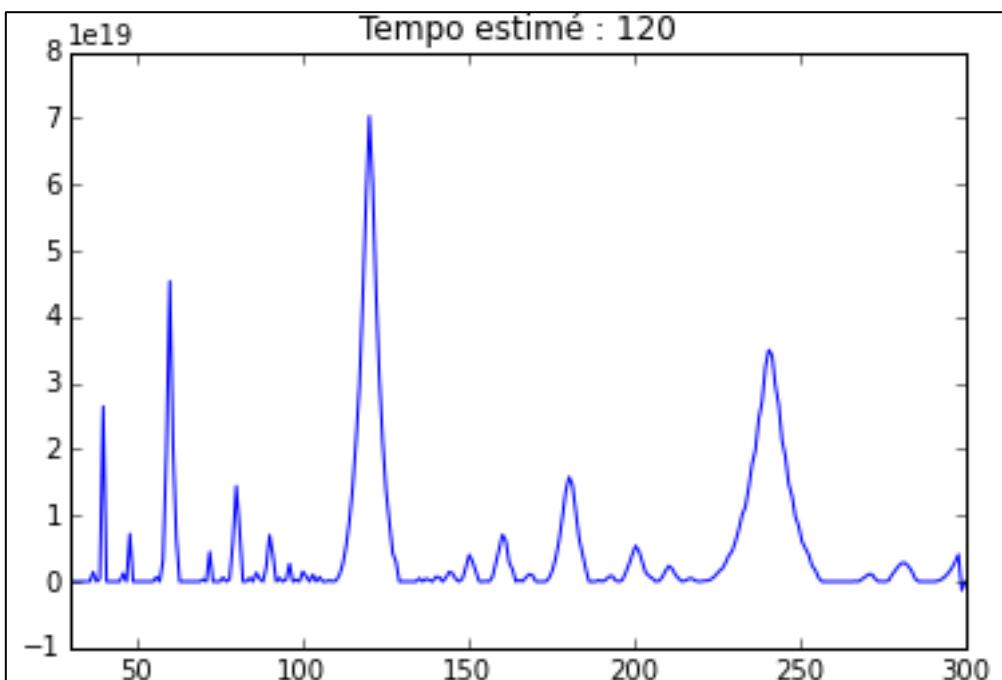


Figure 31 : Résultat du passage par banc de filtre

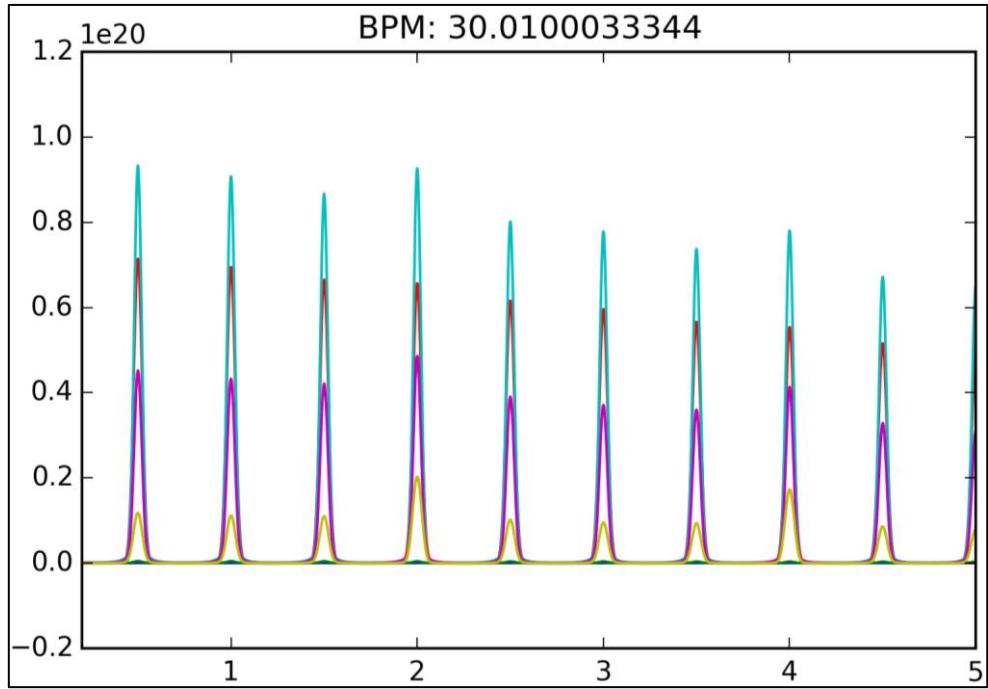


Figure 32 : Pour le même fichier, résultat de l'autocorrélation

- Calcul d'un score :** Le tracé du BPM par la fonction d'autocorrélation est plus ou moins propre. En notant cette propreté et en comparant avec le résultat de l'autocorrélation, on évalue la qualité de la danse de l'utilisateur par l'attribution d'une note. Sa valeur permet de déterminer si l'utilisateur danse.

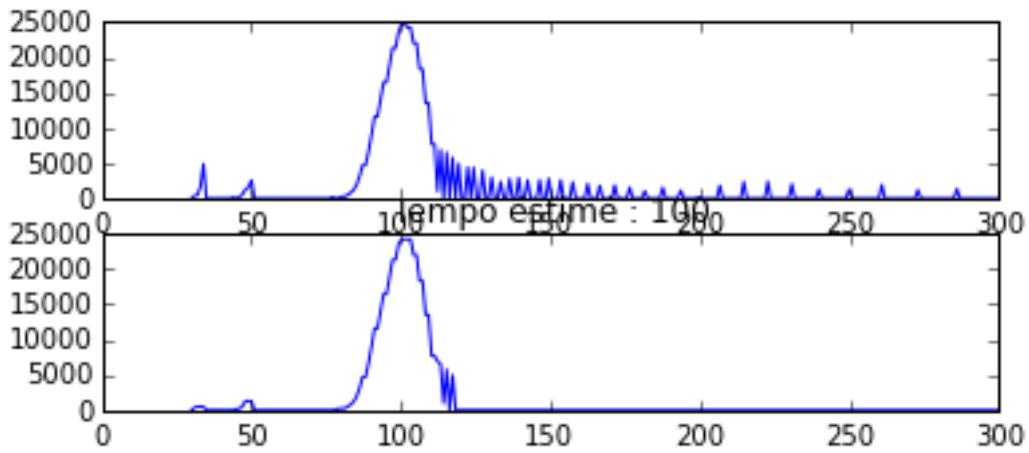


Figure 33 : Un fichier propre : pic très marqué et peu de bruit

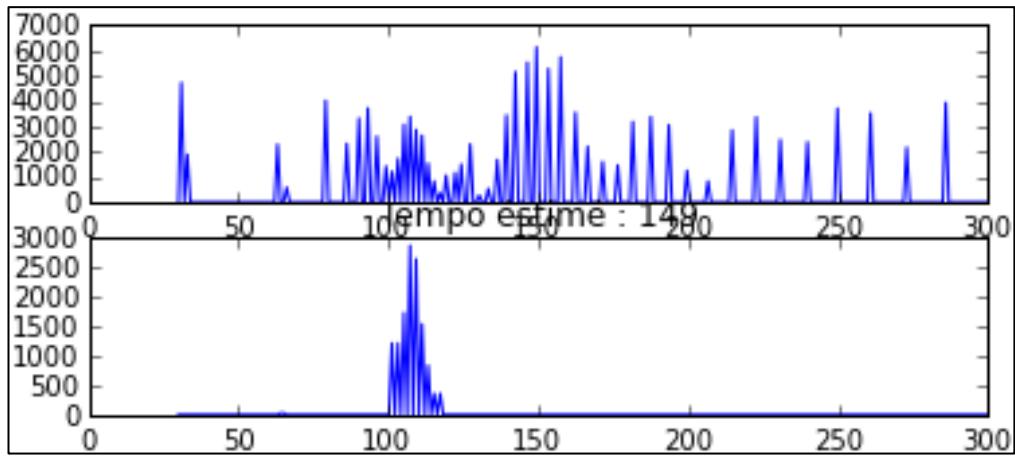


Figure 34 : Fichier moins propre : beaucoup de pic très fins

Développement d'un client pour le PC du DJ : sur l'ordinateur du DJ, affichage en direct du classement des danseurs présents, ainsi que de la moyenne des scores de tous les utilisateurs. Permet au DJ de connaître l'ambiance en direct, pour adapter la musique diffusée.

Fonctions sur JAVA

- Traduction en java de la détermination du BPM de données accélérométriques
- Développement d'un outil de récupération directe du son diffusé par le DJ

Bibliographie

Eric D. Scheirer, "Tempo and beat analysis of acoustic musical signals", 1997, <http://www.ee.columbia.edu/~dpwe/papers/Schei98-beats.pdf>

Miguel A. Alonso Arevalo, "Extraction d'information rythmique à partir d'enregistrements musicaux ", 2007, <https://pastel.archives-ouvertes.fr/file/index/docid/500407/filename/these.pdf>

M. Alonso, B. David, G. Richard, "A study of tempo tracking algorithms from polyphonic music signals", 2003, <https://pdfs.semanticscholar.org/7e7a/03cd318678e914349cf23732b7d9a377843f.pdf>

https://www.clear.rice.edu/elec301/Projects01/beat_sync/beatalgo.html

D.3 Module communication client serveur

Module communication client serveur

Groupe 4.1

Elèves responsables : Louis Combaldieu et Michael Dasseville

Encadrant : Jean-Claude Dufour

Descriptif

Dans tous les cas où plusieurs entités informatiques discutent entre elles au sein d'un projet, une communication « client-serveur » est mise en oeuvre (même quand les deux entités sont des pairs).

Ce module permet d'explorer les aspects simples de la communication entre deux programmes et/ou entre deux machines par le réseau, avec pas mal d'options:

- communication entre deux programmes Java tournant sur la même machine
- les options ci-dessus entre deux PC différents, ou entre un appareil Androïd et un PC, ou entre deux appareils Androïd
- les options ci-dessus avec plus de 2 entités communicantes
- les options ci-dessus utilisant Internet/Wifi ou bluetooth
- une communication simple commande/réponse en texte
- une communication plus complexe incluant aussi des transferts de ressources
- une communication à débit plus élevé nécessitant l'utilisation d'un format binaire...

Objectifs d'apprentissage

- Savoir communiquer entre deux programmes tournant sur la même machine ou sur des machines différentes par le réseau.
- Définir le bon niveau de communication et les éléments du dialogue entre les deux (ou plus) entités.

Résultats attendus

→ PAN 1:

Ecrire la description de toutes les commandes et de toutes les réponses possibles entre le client et le serveur.

→ PAN 2:

Livrable: identification des interfaces utilisées et des bibliothèques offrant le motif socket + socket server; premier squelette de code réalisant la connexion entre nœuds du réseau.

→ PAN 3:

Livrable : code Java / Android réalisant une communication de messages entre les deux nœuds (initialisation de l'interface, formatage et décodage de messages simples).

Description de toutes les commandes et de toutes les réponses possibles entre le client et le serveur.

Base de données:

Base de profils	Nom	Prénom	e-mail	Mot de passe	Photo	Liste d'amis	Liste de soirées
Base d'événements	Titre	Date	Photo	Admins	QR code	Pointeur de Base d'une soirée	Participants
Base d'une soirée commencée	Temps	Photos	Musique	Activité			
Base d'un sondage	Choix 1	Choix 2	...				

En rouge les éléments dépendants du temps échantillonné toutes les 30s.

L'historique personnel d'activité à une soirée est stocké sur le téléphone. A l'ouverture de l'application, il est supprimé s'il est trop ancien.

Liste des requêtes et réponses possibles :

- L'utilisateur crée un profil : envoie une photo de profil, un nom, un prénom, un e-mail valide, un mot de passe.
- L'utilisateur modifie son profil : modifie une des données ci-dessus.
- L'utilisateur crée un événement : envoie un titre, une date, une photo, les admins.
- L'utilisateur modifie son événement : modifie une des données ci-dessus.
- L'utilisateur accède à un événement : le serveur envoie les données ci-dessus + si l'utilisateur a participé à l'événement.
- l'utilisateur accède à la timeline de la soirée : le serveur renvoie la base de la soirée
- L'utilisateur envoie un QR code scanné : l'appli lui envoie la base d'événement correspondante, l'ajoute à la liste des participants et ajoute la soirée à sa liste de soirée.
- L'utilisateur ajoute/supprime un ami : cet ami est ajouté/retiré à la liste personnelle d'amis.
- L'utilisateur accède à un profil :
- S'il n'est pas ami : le serveur envoie Nom, Prénom, Photo
- S'il est ami : le serveur renvoie en plus la liste de soirées
- Un admin lance un sondage : le serveur envoie des notifications aux utilisateurs et renvoie les résultats à l'admin toutes les 10s pendant 2min.
- Un admin rajoute une musique : elle est ajoutée dans la base de la soirée.
- Un utilisateur rajoute une photo : elle est ajoutée dans la base de la soirée.

Connexion entre clients et serveur via multi-threading

Sur le serveur : Une classe Start lance le premier thread, qui se met en écoute du port 2001 de la machine.

Une fois qu'un client se connecte, il est accepté et on lance immédiatement un autre thread, à l'écoute du port suivant. Les requêtes se font par envoi de strings à travers le socket.

Sur le client : On essaie de se connecter à tous les ports entre 2000 et 2100 jusqu'à trouver celui que le serveur écoute. Les méthodes renvoient toutes des booléens pour faciliter le travail de débogage et vérifier la bonne exécution du code.

Envoi d'images

Les images ne pouvant être stockées dans une base SQL, un système de fichiers est mis en place :

`/home/michael/lastnight/users/` fichier d'utilisateurs

`/home/michael/lastnight/parties/` fichier de soirées

Dans le fichier user, on trouve La photo de profil et éventuellement des albums photo.

Un fichier de soirées contient les photos relatives à une soirée.

Création de la Base de données

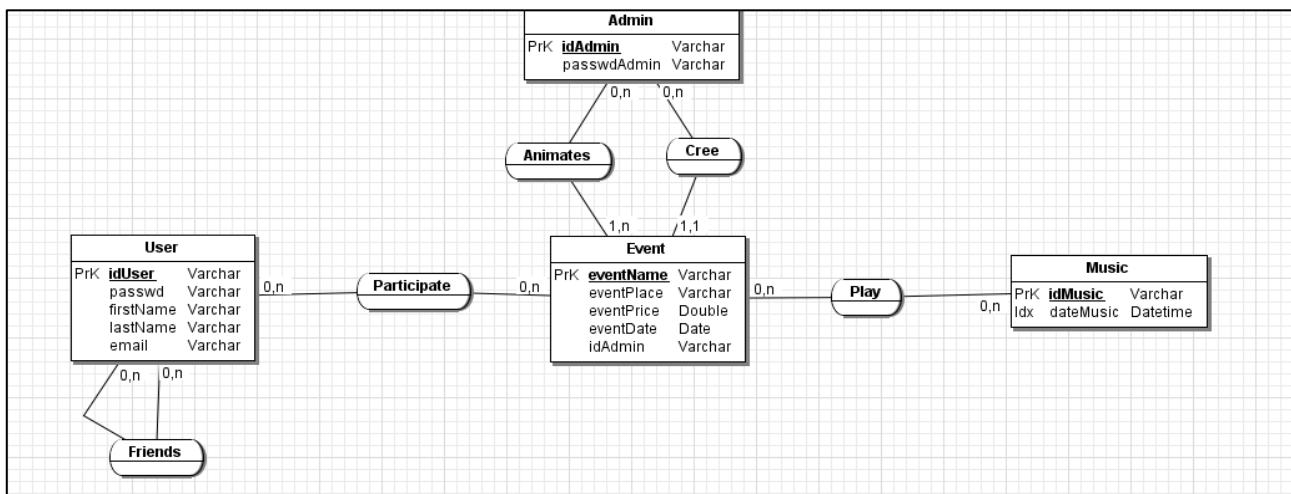


Figure 35 : Création de la base de données grâce au logiciel Jmerise

```

-- Script Postgre

-- Table: Admin
CREATE TABLE public.Admin(
    idAdmin VARCHAR (80) NOT NULL ,
    passwdAdmin VARCHAR (80) NOT NULL ,
    CONSTRAINT prk_constraint_Admin PRIMARY KEY (idAdmin)
)WITHOUT OIDS;

-- Table: Event
CREATE TABLE public.Event(
    eventName VARCHAR (80) NOT NULL ,
    eventPlace VARCHAR (80) NOT NULL ,
    eventPrice FLOAT8 NOT NULL ,
    eventDate DATE NOT NULL ,
    idAdmin VARCHAR (80) NOT NULL ,
    idAdmin_1 VARCHAR (80) NOT NULL ,
    CONSTRAINT prk_constraint_Event PRIMARY KEY (eventName)
)WITHOUT OIDS;

-- Table: Music
CREATE TABLE public.Music(
    idMusic VARCHAR (80) NOT NULL ,
    dateMusic DATE NOT NULL ,
    CONSTRAINT prk_constraint_Music PRIMARY KEY (idMusic)
)WITHOUT OIDS;
  
```

Figure 36 : Script Postgre

Conception des requêtes SQL

Les requêtes peuvent être divisées en deux parties, l'ajout d'éléments à la base (Image 4) et la lecture d'éléments de la base (Image 3).

```
public String getEventPlace(String eventName) {
    String place=null;
    try{
        Connection conn = DriverManager.getConnection(url, user, passwd);
        Statement state = conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCUR_UPDATABLE);

        ResultSet result = state.executeQuery("SELECT 'lieuevent' FROM evenement WHERE nomevent='"+eventName+"'");
        ResultSetMetaData resultMeta = result.getMetaData();
        while(result.next()){
            place=result.getObject(1).toString();
        }
        result.close();
        state.close();
        return place;
    }catch(Exception e){
        e.printStackTrace();
        return null;
    }
}
```

Figure 37 : Affichage du lieu de l'évènement

```
public void addUtilisateur(String myId, String firstName, String lastName, String email, String myPassword){
    @SuppressWarnings("unused")
    boolean reg=isRegistered(myId,"Utilisateur");
    if (reg=false){
        try{

            Connection conn = DriverManager.getConnection(url, user, passwd);
            //System.out.println("connection effective");

            Statement state = conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCUR_UPDATABLE);

            String qry="INSERT INTO utilisateur"+ " VALUES ('"+myId+"','"+myPassWord+"','"+firstName+"','"+lastName+"','"+email+"')";
            state.executeUpdate(qry);

            state.close();
        }catch(Exception e){
            e.printStackTrace();
        }
    }
}
```

Figure 38 : Ajout d'un utilisateur à la base de données

Administration du Serveur distant

Nous avons loué un serveur distant (VPS) sur le site OVH. Une fois le serveur entre nos mains nous avons ajouté un utilisateur pour travailler l'administration du serveur. Nous avons dû y ajouter les programmes et logiciels utiles à notre projet (comme postgresql) via la commande «apt-get», et y envoyer nos programmes java via le logiciel FileZilla qui utilise une connexion FTP.

Conception d'un système de messagerie

Grâce à la base de données, nous avons pu concevoir un système de messagerie entre les administrateurs d'une soirée et les utilisateurs. Ces messages s'affichent en clair sur l'application à tous les utilisateurs.

Conception d'un système de sondage

Grâce à la base de données, nous avons pu concevoir un système de sondage qui permet aux utilisateurs de voter pour leurs musiques préférées, ou pour toutes suggestions des administrateurs lors de la soirée.

D.4 Module Intégration et tests

Module Intégration et tests

Groupe 4.1

Elèves responsables : Daniel MAYAU, Louis COMBALDIEU et Paul-Elian TABARANT en responsable principal

Encadrant : Soumia DERMOUCHE

Objectifs d'apprentissage

- Vérifier que chaque bloc implémenté prend les bons paramètres en entrée et délivre le résultat sous la forme attendue. Faire des retours aux responsables du module en cas de problème
- Contrôler dès que possible le fonctionnement de chaque bloc grâce aux tests fournis par les responsables de chaque module
- Rassembler dès que possible les travaux des différents modules pour construire le système global, puis rajouter progressivement les nouvelles fonctionnalités
- Une fois les modules rassemblés, faire des tests de l'ensemble du système pour vérifier qu'il fonctionne de A à Z comme prévu, puis qu'il répond aux contraintes d'ergonomie et de temps de réponse (performances). Résoudre les problèmes survenus sur le plan de test global qui ne dépendent pas d'un module particulier

Résultats attendus

→ PAN 1:

Avoir établi un schéma d'architecture du système et un diagramme d'activité. Décrire textuellement chaque bloc du schéma et les interfaces entre ces blocs. Avoir créé un dépôt Git pour le groupe de PACT avec les droits accordés à chaque membre et à l'encadrant GL. Demander un commit de chaque membre du groupe dans un fichier readme.txt pour montrer que tout le monde y a accès correctement

→ PAN 2:

Définir le plan de test du projet :

- Liste des tests avec leur nature (fonctionnels, performance, ergonomie..)
- Contrat à remplir pour chaque test : type des entrées et résultat attendu en sortie (avec plus ou moins d'exigence de précision selon les cas)

Etablir un squelette logiciel de simulation, sans implémentation réelle des méthodes (retour pour montrer qu'elles ont été appelées, retour d'un résultat fixe ou implémentation sommaire)

→ PAN 3:

Faire un rapport d'intégration des modules dans le prototype allégé. Pouvoir fournir des résultats de test des modules intégrés.

Avoir fait le nécessaire pour remplir les objectifs du prototype allégé : système capable de gérer comme prévu les événements (création et récupération des données d'une soirée). Stockage des musiques « offline » sur le serveur avec une analyse du tempo effectuée préalablement, et envoi du flux audio vers le DJ. Bonne redirection des messages entre le DJ et les invités. Génération d'un récapitulatif complet de la soirée, sans contrainte de temps.

→ PAN 4:

Nouveau rapport d'intégration, cette fois pour le prototype final, avec les tests.

Avoir fait le nécessaire pour remplir les objectifs du prototype complet : flux audio envoyé par le PC du DJ vers le serveur, corrélation tempo/mouvements effectuée en temps réel. Informations (photos déjà partagées, nombre de personnes qui dansent, etc) visibles par l'utilisateur pendant la soirée contrairement au prototype allégé.

Architecture globale de l'appli

Cette tâche a été nécessaire pour assurer la cohérence de notre plateforme en ligne. Celle-ci a abouti à un changement de répartition du code dans les différentes entités communicantes.

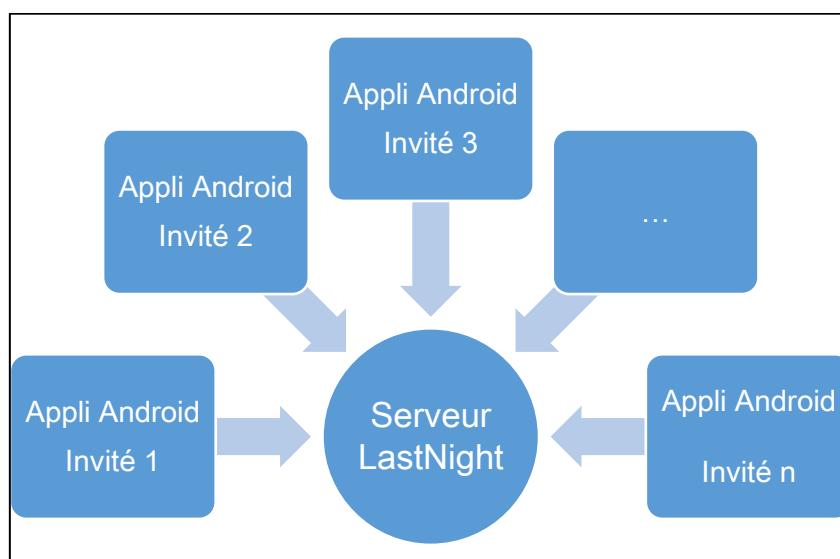
Ce choix a été motivé par deux critères principaux :

- Limitation des calculs au niveau du serveur, peu puissant
- Affranchissement d'un maximum de contraintes pour l'utilisateur de l'application et le DJ

L'ancienne solution répartissait les calculs de cette manière :

- Capture des mouvements d'accélération sur les téléphones, envoi vers le serveur
- Capture du son joué par le PC du DJ, envoi vers le serveur
- Récupération du son et des données de mouvement des utilisateurs sur le serveur, puis calcul du tempo de la musique de la soirée, et des mouvements pour chaque utilisateur afin d'établir la corrélation individuelle
- Envoi des résultats individuels à chaque utilisateur, avec éventuellement des stats globales comparées avec les autres invités

Le problème principal de cette solution est qu'elle est difficilement applicable à des conditions réelles, c'est-à-dire à une soirée comprenant N utilisateurs, N potentiellement grand.



La nouvelle solution profite plutôt de la puissance de calcul individuelle des téléphones pour l'analyse audio. Elle se décompose de cette manière :

- Capture des mouvements d'accélération sur les téléphones, calcul du tempo associé et stockage dans un fichier en local
- Capture du son joué par le PC du DJ, calcul du tempo sur son PC et envoi vers le serveur
- Récupération du tempo de la musique par les utilisateurs et calcul de corrélation avec le tempo des mouvements stocké en local sur les téléphones

Les avantages critiques qui ont motivé ce choix sont :

- Le calcul déporté sur les appareils mobiles et le PC du DJ, qui réduit presque à zéro les tâches de calcul du serveur
- La limitation du trafic réseau en supprimant l'envoi des mouvements vers le serveur
- L'utilisation d'un serveur distant qui rend l'implémentation de code Java moins confortable

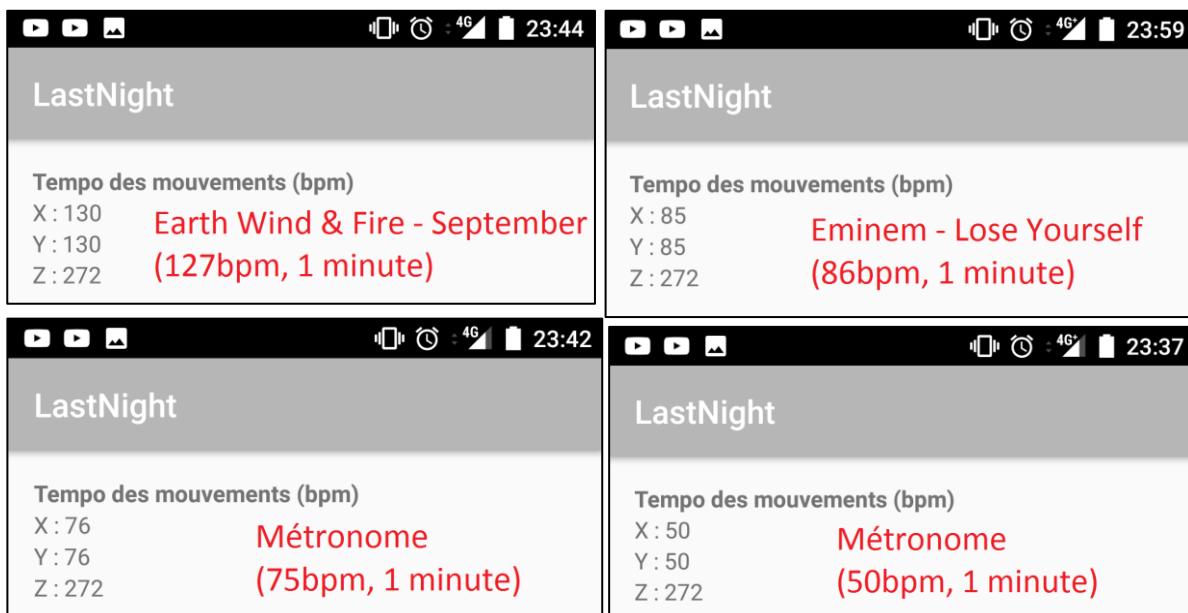
Le seul problème auquel nous allons devoir faire face est la génération de statistiques globales de la soirée, étant donné que la corrélation s'effectue individuellement sur les téléphones. Pour en bénéficier, il faudra penser à un envoi des résultats de corrélation vers le serveur pour les mettre en commun.

Fonctions Audio dans l'application Android

Le travail d'intégration doit permettre de faire communiquer la capture des mouvements de l'utilisateur, le calcul du tempo et la comparaison avec celui de la musique, puis l'affichage des résultats.

Le code Java de calcul du tempo des mouvements a été testé et donne des résultats satisfaisants. Il est désormais intégré à l'application et détermine le tempo à partir d'un fichier généré dans la mémoire externe ou interne du téléphone via l'application. Une séance d'intégration organisée avec le module Audio a été nécessaire.

L'intégration a demandé de rajouter des fonctionnalités côté Android dans le service d'écriture des mouvements dans un fichier : côté Audio, le tempo se base sur une fréquence d'échantillonnage supposée régulière. Il a donc fallu implémenter une nouvelle fonction de calcul de la précision moyenne du capteur sur la durée de la mesure. Après quelques ajustements sur le module Android pour fournir la fréquence correcte, vous trouverez ci-dessous quelques résultats



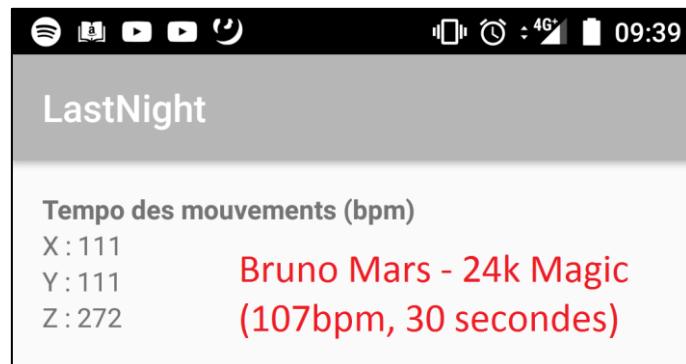


Figure 40 : tests d'intégration du calcul du tempo sur les téléphones Android

La fonction de calcul du score de l'invité faisait également partie de l'intégration de la partie Audio dans l'application.

Une fois le calcul audio attesté comme acceptable au vu des résultats ci-dessus, il devait être utilisé dans le cadre d'une soirée, c'est-à-dire répété à un intervalle de temps régulier pour pouvoir suivre l'évolution de la danse au cours du temps. Pour cela, il fallait fixer cet intervalle de temps : une durée trop faible conduirait à un manque de données accélérométriques pour que le calcul soit fiable. A l'inverse, trop augmenter l'intervalle de temps entre chaque calcul ferait perdre l'intérêt de statistiques « live » pendant la soirée puisque le rafraîchissement des résultats serait trop lent. De plus le récapitulatif n'aurait pas suffisamment de points de mesure. Après discussion avec le module audio et des tests effectués avec différentes durées de calcul, un bon compromis entre ces deux contraintes nous a semblé être un intervalle de 10 secondes. Cette durée nous permettait de détecter le tempo avec une précision estimée à 5% tout en ayant 6 points de mesure par minute, soit par exemple 18 sur une musique de 3 minutes.

L'intégration finale des fonctions Audio a pu être testée sur différents extraits de mouvements d'une personne, dont l'un d'eux est visible ci-dessous.

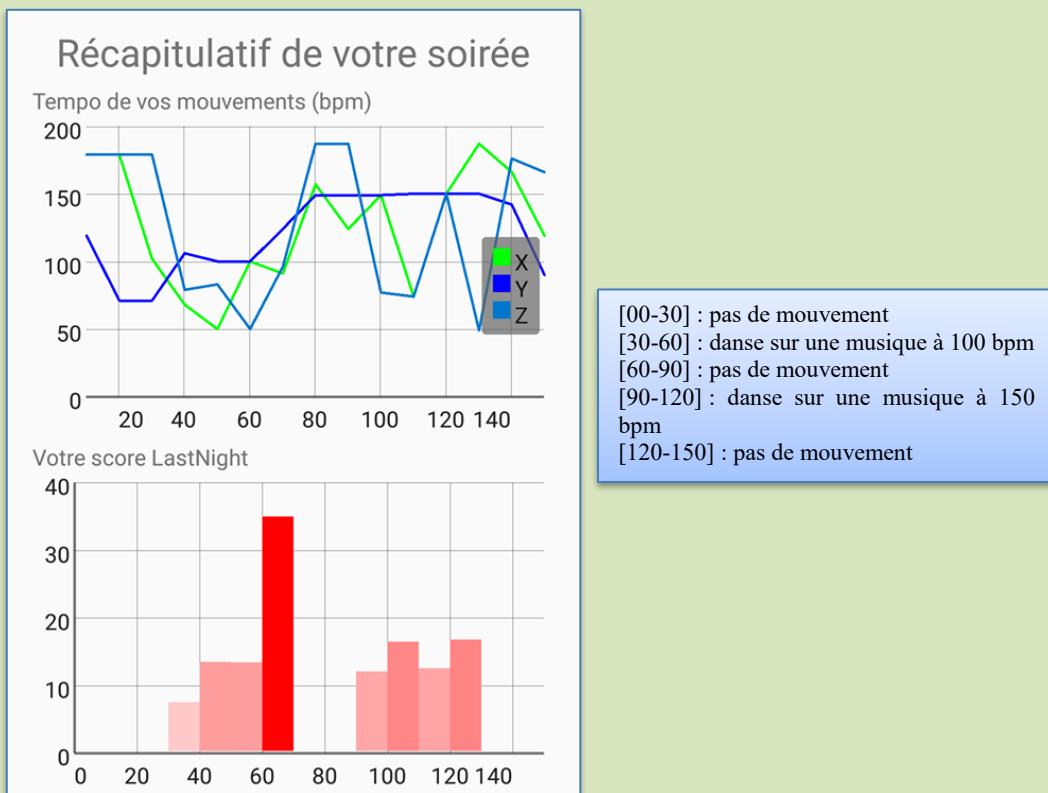


Figure 41 : tests d'intégration des fonctions Audio sur un extrait de 2min30s

Les périodes où l'invité danse et où il ne danse pas sont bien délimitées par l'évolution du score. Le bruit du capteur peut expliquer la détection d'un certain tempo même lorsque l'invité ne danse pas, mais ceci est compensé par un calcul de score égal à 0 dans ce cas : dans une version finale de l'appli il conviendra d'afficher uniquement le graphe d'évolution du score qui sera le seul à intéresser l'utilisateur. La conformité du score en fonction de la qualité de la danse est un critère subjectif difficile à appréhender. Néanmoins, le calcul du score est correctement intégré et toute amélioration peut être faite sur celui-ci sans pour autant modifier le comportement de l'application.

Nous avons donc une première version d'évolution du score au cours d'une soirée, calculée toutes les 10 secondes sans problème majeur.

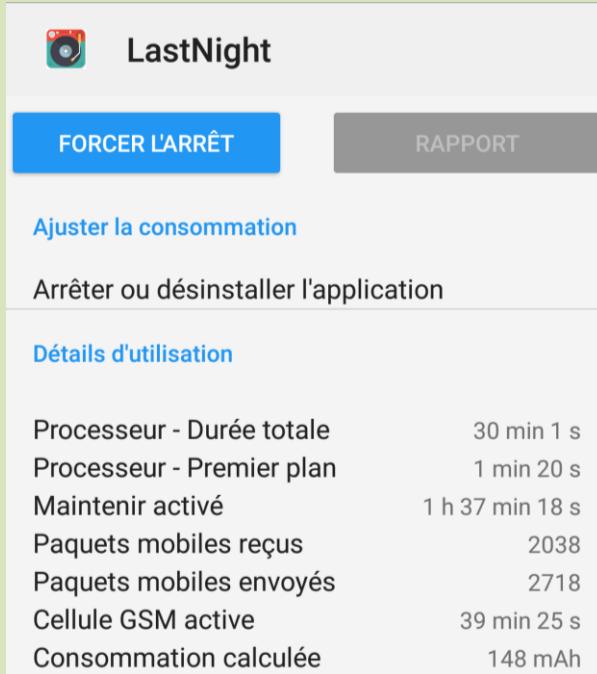


Figure 42 : test de consommation de l'application en mode « suivi des mouvements » sur une durée de 1h37min

Les batteries de téléphone du commerce actuel ont une capacité comprise entre 2000 et 3000 mAh en moyenne, ce qui indique que notre application est particulièrement gourmande en énergie lorsque le suivi est en route. Cependant, ceci est à nuancer par le fait que ce suivi est uniquement activé pendant la durée d'une soirée, où l'utilisateur n'utilise que très peu son téléphone pour d'autres applications. Cette consommation reste acceptable pour une première version. Dans une version optimisée, il serait intéressant de réduire cette consommation. Plusieurs pistes d'amélioration sont possibles en jouant sur les paramètres suivants :

- Optimisation du **calcul du score**
- Augmentation de la **durée entre chaque calcul** de score (> 10 secondes)
- Réduction de la **fréquence d'échantillonnage** de l'accéléromètre
- Ajout d'une option « **activer/désactiver le suivi** » pour économiser la batterie de l'utilisateur si besoin

Fonctions de communication Client dans l'application Android

L'intégration doit ici permettre de faire le lien entre le fonctionnement interne de l'appli Android et les données échangées avec le serveur LastNight.

Une classe de test des requêtes client Android avec le serveur et l'interface de communication a été définie avec les responsables Client/Serveur. Elle contient les méthodes nécessaires au test de gestion des comptes utilisateurs et d'organisation d'une soirée via l'application. Il reste à préparer des

tests pour les fonctions « live » au cours d'une soirée, le transfert des fichiers de tempo et le système d'amis. Le code du client de connexion est intégré à l'application Android et fonctionne correctement.

Le programme gérant les requêtes côté serveur est fonctionnel. Les options suivantes sont maintenant disponibles sur l'application et communiquent correctement avec la base de données :

- **Création de comptes et connexion** d'un utilisateur, **modification** des détails de son profil
- **Création d'une soirée** dans la base de données, déclaration des **administrateurs** de la soirée
- **Authentification** d'un utilisateur dans une soirée en cours
- Création, envoi de **sondages** et de **messages** si l'invité est déclaré administrateur de la soirée, visualisation et réponse aux sondages pour un invité ordinaire
- **Partage du score** vers la base de données pour chaque invité
- **Transfert de photos** : photo de profil de l'utilisateur et partage de photos pendant une soirée

Afin de faciliter le test des fonctions de transfert de photos entre l'application et le serveur, nous avons conçu une application de test qui permettait au binôme Client/Serveur de bénéficier de conditions réelles de l'environnement Android tout en ayant directement les résultats du transfert affichés à l'écran, basés directement sur la réponse du serveur.

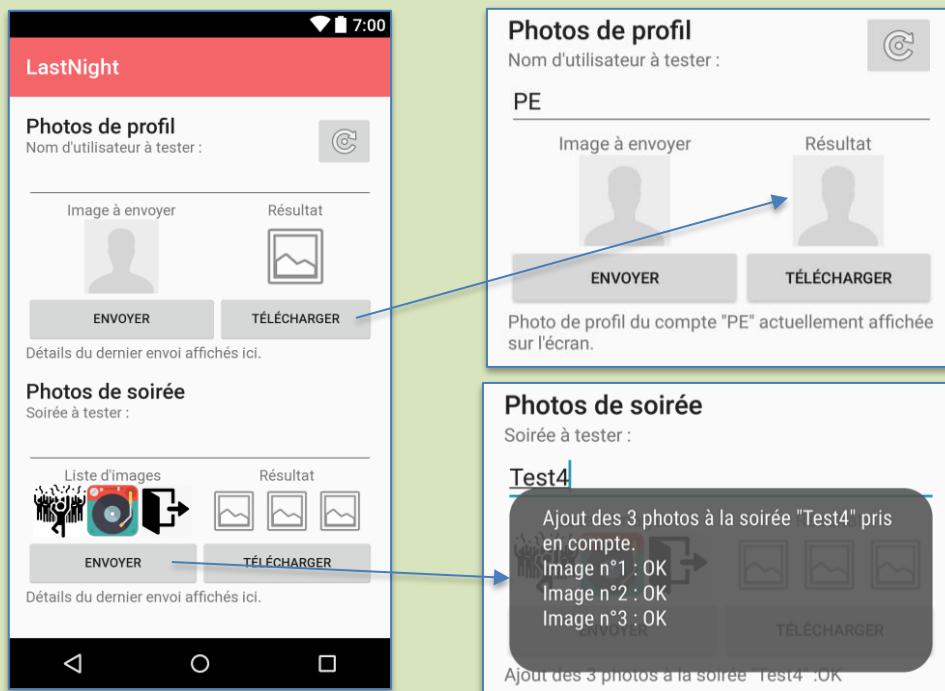


Figure 43 : Application de test de transfert des photos
1^{ère} flèche : test de téléchargement d'une photo de profil depuis le serveur
2^{ème} flèche : test d'envoi de 3 photos à l'album d'une soirée

Fonctions Audio et Client dans le programme du PC du DJ

Cette partie concerne uniquement le code intégré sur le PC du DJ, à savoir l'authentification, l'analyse audio du tempo de la musique et l'envoi des résultats vers le serveur LastNight.

L'architecture logicielle du programme client a été créée et est accessible à tous les membres des modules Audio et Client/Serveur via le dépôt Git. Elle contient un programme de capture Audio à partir de la carte son du PC.

Au vu de la qualité du calcul de danse sans corrélation avec le tempo de la musique, le tempo n'est pas envoyé au serveur dans le client PC. Cependant, il affiche des statistiques « live » intéressantes pour le DJ et les invités.

- **Evolution temporelle du score moyen, rafraîchie toutes les 10 secondes** : le score moyen est un bon indicateur pour savoir sur quelles musiques les invités ont le plus dansé
- **Classement des invités en fonction de leur dernier score enregistré** : cette option accroît l'interactivité entre les invités au cours de la soirée

D.5 Module SES

Module « Etude d'impact sur la vie privée »

Proposé par : Caroline RIZZA (caroline.rizza@telecom-paristech.fr -)

Réalisé par : groupe 4.1, élèves COMBALDIEU/BLANCHARD,

Descriptif

- Mener une étude d'impact sur la vie privée relative à votre application / produit en deux temps :
 - 1) votre application répond-elle aux obligations légales telles que définies par la loi Informatique et Liberté en France ;
 - 2) quels sont les risques associés à votre application et comment les « mitiger ».
- Montrer comment une telle étude vient impacter le design d'une application / d'un objet

Objectifs d'apprentissage

- Etre en mesure de comprendre les enjeux relatifs à la protection des données personnelles des futurs utilisateurs de votre application et protéger ces données ;
- Conduire une étude d'impact sur la vie privée
- Etre en mesure, d'une manière « light » de mettre en œuvre une méthodologie de privacy-by-design en anticipant les enjeux relatifs au respect de la vie privée des futurs utilisateurs dès le design d'une application ou d'un objet.

Résultats attendus

PAN 1 :

- Rien / spécification du module

PAN 2 :

- Document de synthèse de l'étude d'impact sur la vie privée comprenant 1) les obligations légales et 2) Comment le module a impacté sur la design de l'application / l'objet ; 3) l'étude des risques

PAN 3 :

- Rien, (si retard, document de synthèse de l'étude d'impact sur la vie privée – Etude des risques).

Privacy impact Assessment

Une version corrigée du PIA a été rédigée pour mieux rendre compte des enjeux de l'application et les spécifications de la CNIL. En particulier ont été réalisées une étude du contexte, une étude des données à caractère personnel (DCP), une étude des supports et une étude de conformité pour respecter les exigences légales. L'étude des risques a été commencée.