université angers

FACULTÉ
DES SCIENCES
Unité de formation
et de recherche
DÉPARTEMENT
INFORMATIQUE

M2-ACDI

Rapport de projet

« Application de visualisation et d'annotation de vidéos sportives »

Auteurs

- CISSÉ Sékou Aboubacar
- TRAORÉ Hamidou

Année 2018/2019

Table des matières

Introduction

Le module de **management de projet** a pour objectif d'initier les étudiants du Master 2 ACDI de l'université d'Angers à la conduite de projets informatiques. Cette unité consiste à mettre en œuvre les savoirs acquis lors des unités disciplinaires dans le cadre d'un groupe de travail comme en situation d'entreprise. Il s'agit pour les étudiants d'encadrer un groupe d'au moins deux étudiants du Master 1 informatique, d'établir le cahier des charges, de participer et d'animer les réunions hebdomadaires. Ils participent également aux soutenances et délibérations qui s'en suivront. C'est dans ce contexte que s'inscrit ce projet de ***« Application de visualisation et d'annotation de vidéos sportives »***.

Le sujet a été proposé par le département informatique de l'université et répond à un besoin réel. Son encadrement a été assuré par M. Vincent BARICHARD et Mme Rachel COUDRAY.

Le présent document fait le rapport de la conduite de ce projet. Il est décomposé en quatre parties dont la première porte sur le contexte général du projet. Dans la deuxième partie nous définissons de façon claire les spécifications fonctionnelles et techniques à suivre pour la réalisation de ce projet. Nous évoquons notre conduite du projet et les différentes phases de programmation des fonctionnalités de l'application dans la troisième partie. La quatrième partie fait état de l'architecture application mise en œuvre pour la gestion des annotations (édition et lecture/affichage). La cinquième et dernière partie est quant à elle consacrée à un bilan critique et des différentes orientations pour une continuité du projet.

I. Présentation générale

I.1. Contexte

L’idée de la réalisation d’une application de visualisation et d’annotation de vidéos sportives est née d’un réel besoin au sein d’un ensemble de club de gym de la ville d’Angers. Ces clubs de gym disposent de vidéos sportives sur lesquelles les coachs (entraîneurs) désirent ajouter des indications à destination de leurs athlètes pour alerter ces derniers sur des points à travailler. Dans un premier temps, l'entraîneur visualise et annote la vidéo. Celle-ci est ensuite mise à disposition de l'athlète qui peut travailler son enchaînement avec toutes les indications nécessaires.

Une tablette est facilement transportable sur le terrain d'entraînement et permet à la fois la visualisation des vidéos par l'athlète et l'édition par l'entraîneur.

I.2. Objectifs et fonctionnalités attendues

A partir du contexte ci-dessus décrit, l’objectif est de toute évidence de développer une application Android orientée tablette permettant :

- de visualiser des vidéos sportives
- de faire des annotations sur ces vidéos

Les types d’annotations attendues sont :

- *textuelle* : commentaire textuel pour indiquer des détails sur un point particulier
- *graphique* : ajout de forme géométrique sur la vidéo
- *audio* : enregistrement vocal pour expliquer des détails sur un point particulier

L’application doit également intégrer le mode ralenti et le zoom qui sont des fonctionnalités standards d’un lecteur classique de vidéos.

Elle doit enfin permettre le partage des vidéos annotées entre ses différents utilisateurs.

II. Spécifications et analyse du sujet

II.1. Spécifications fonctionnelles

Après avoir pris connaissance du sujet, nous avons effectué des rencontres clients pour mieux cerner les attentes et les contraintes à prendre en compte pour la réalisation de l'application. Nous avons alors adopté le formalisme *UML* des cas d'utilisations pour formaliser ces besoins. Ainsi, nous avons pu identifier deux types d'acteurs qui interagiront avec l'application :

- **Coach** : visualise les vidéos et fait des annotations
- **Athlète** : visualise les vidéos avec les annotations ajoutées par le coach

Suivant les fonctionnalités attendues du système, nous avons défini les cas d'utilisations que nous présentons dans le diagramme *UML* pour mettre en évidence les liens entre acteurs (utilisateurs) et les possibles interactions avec l'application.

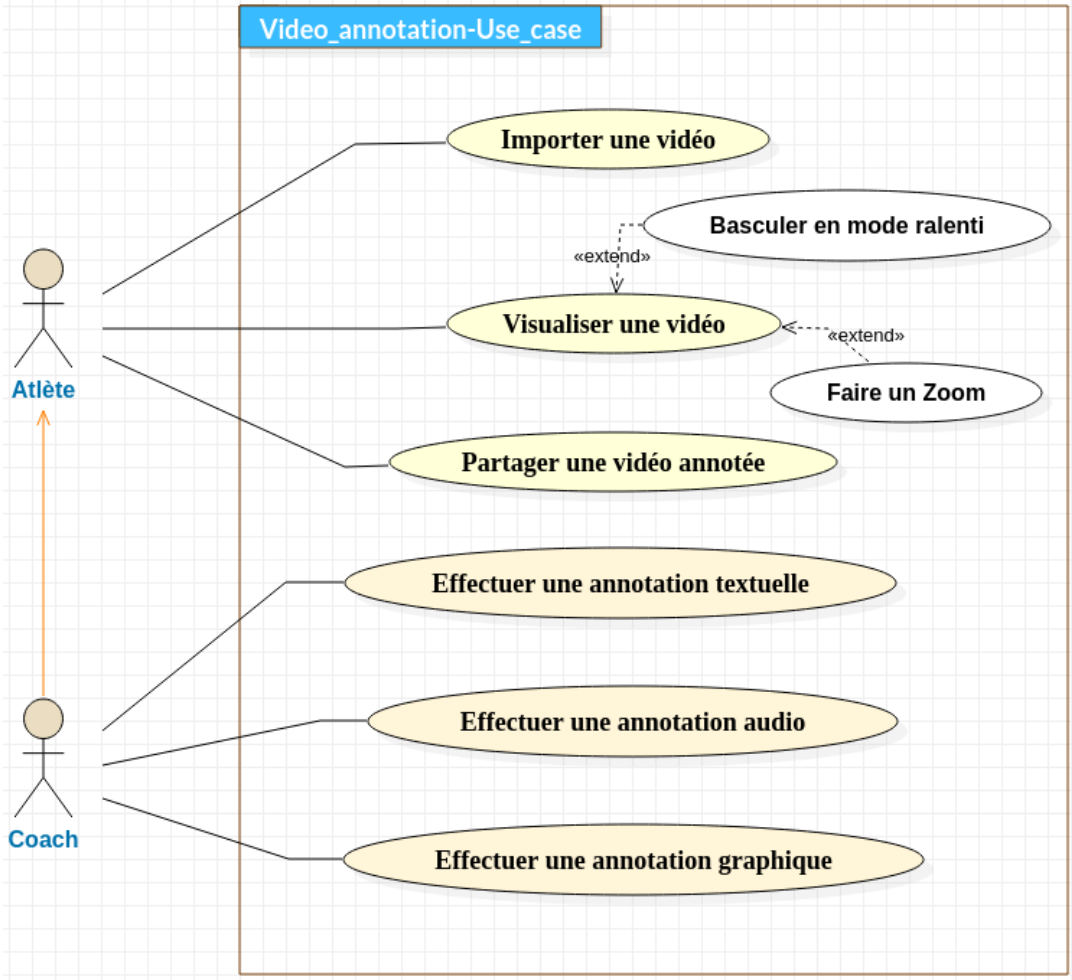


Illustration 1: diagramme de cas d'utilisation

Le formalisme *ULM* de diagramme de classe a également été utilisé pour modéliser la structure des données qui seront traitées par l'application.

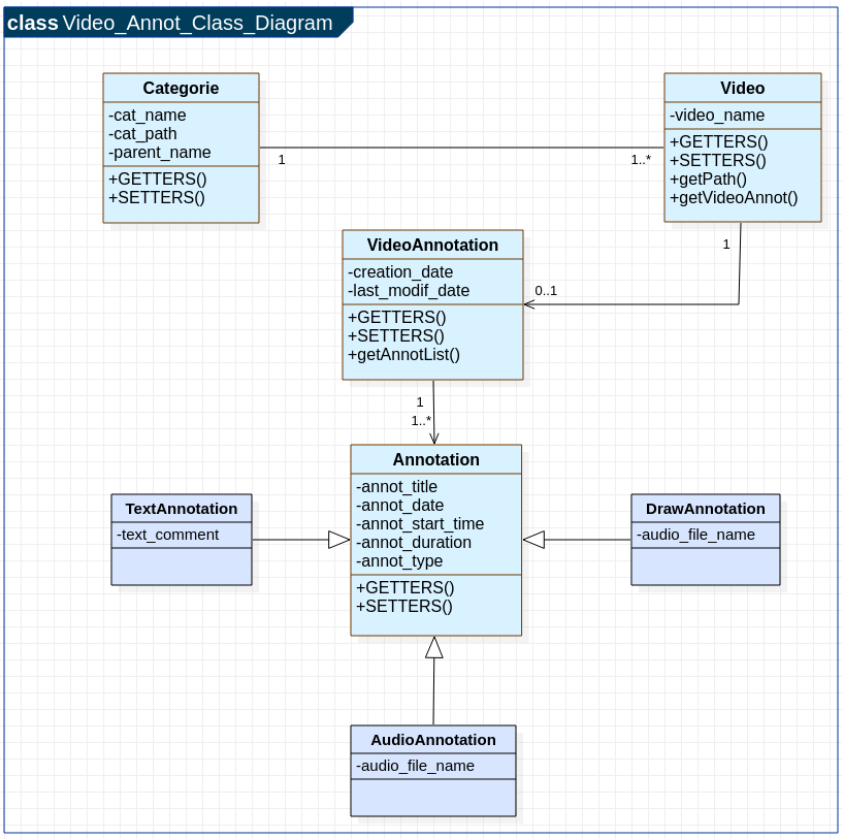


Illustration 2: diagramme de classe du système

Une maquette de la vue globale de l'application attendue se présente comme suit :

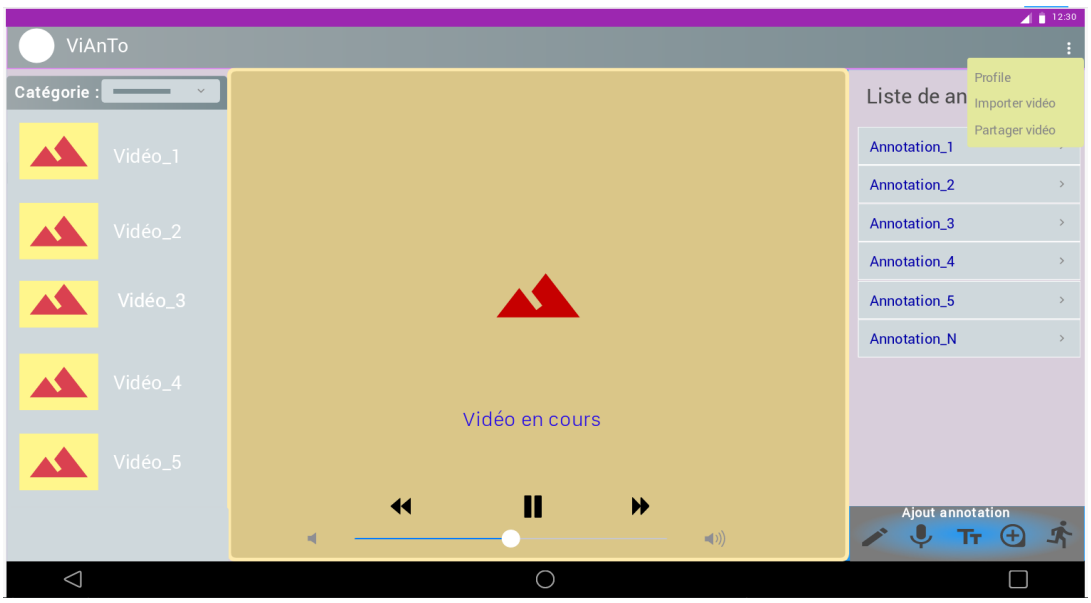


Illustration 3: maquette d'interface générale de l'application

II.2. Spécifications techniques

II.2.1 Système de stockage des données

Les vidéos sont classées en 4 grandes catégories, et chaque catégorie est subdivisée en 4 sous-catégories. Nous avons alors défini un système de fichier assez simple qui facilitera l'identification de chaque ressource dans l'application :

- le dossier nommé « ***videos*** » : c'est le répertoire de base de l'application. Il contient 4 répertoires correspondants aux *4* grandes *catégories* définies pour le classement des vidéos.
- Chaque catégorie contient à son tour *4 sous-répertoires* contenant les fichiers vidéos de la sous-catégorie avec les fichiers d'annotations associées. Ces fichiers vidéos et leurs annotations sont regroupés par dossier et chaque dossier porte le nom du fichier vidéo contenu. En fonction du nombre et des types d'annotations faites sur les vidéos correspondantes, ce dossier contiendra les types de fichiers suivants:
  - un fichier «`nom_video.json` » décrivant les annotations audios de la vidéo.
  - des fichiers « `nom_video_dateTime.mp3` » qui sont des audios pour les annotations audios enregistrées.
  - des fichiers « `nom_video_dateTime.png` » qui sont des images pour des annotations graphiques faites sur la vidéo correspondante.

Les vidéos traitées par l'application sont au format *mp4* et nous utilisons pour les fichiers audio le format *mp3*. Pour le stockage des annotations, nous utilisons la structure et le format *JSON*.

II.2.2 Composants techniques et fonctionnels

Il s'agit ici des principaux composants techniques et fonctionnels qui interagiront pour donner le rendu graphique de l'application. Cette structuration est illustrée par la figure suivante :

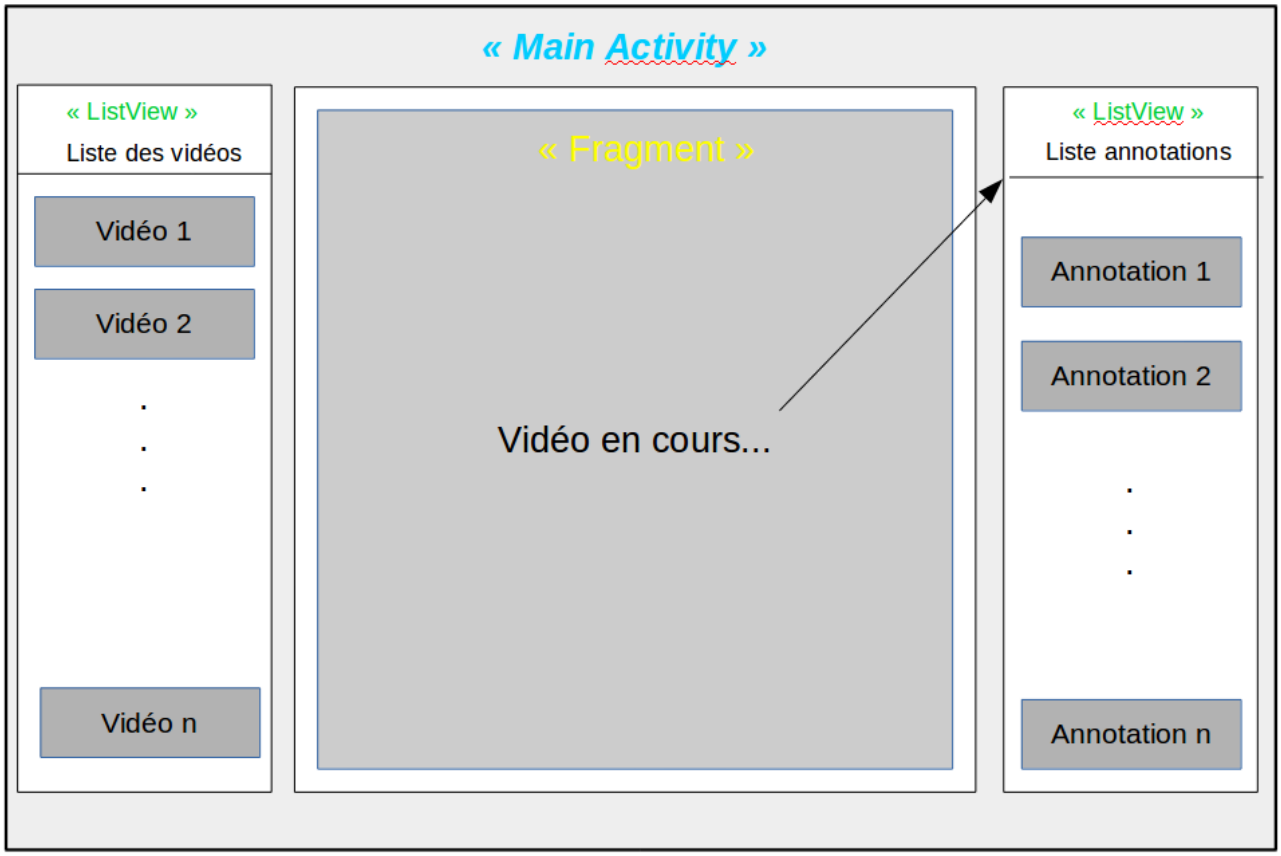


Illustration 4: architecture applicative

On distingue bien trois grandes zones :

- ***Liste des vidéos*** : permettant de lister les vidéos de la catégorie en cours de lecture
- ***Lecteur vidéo*** : permettant de lire une vidéo sélectionnée dans la *listView* précédente et d'appliquer les différentes fonctionnalités liées à l'annotation
- ***Liste des annotions*** : permettant de visualiser les différentes annotations liées à la vidéo en cours de lecture

II.2.3 Technologies de développement

(1) Environnement et plate-forme de développement

L'objectif du projet étant la réalisation d'une application Android, nous utilisons les outils et langages recommandés par Google pour la phase développement.

- ***La plate-forme Android*** : elle offre un framework riche, suffisamment rigide et définie une architecture simple à respecter pour la construction des différents composants de l'application
- **Android Studio** : basé sur Intellij IDEA, c'est l'environnement de développement idéal pour développer des applications Android
  - Création de projets
  - Édition du code
  - Gestion des émulateurs
  - Compilation, débogage et exécution dans un émulateur

(2) API – Librairies

- **Exoplayer** : librairie open source créée par Google distribuée séparément du SDK Android. Elle repose sur des APIs de gestion de média dans un plus bas niveau et donne accès à la logique de lecture de vidéos. Nous utilisons cette librairie pour implémenter le lecteur et ses fonctionnalités. L'implémentation du mode ralenti et zoom sera fait sur la base des possibilités d'extension offertes par cette librairie.
- **MediaRecorder et MediaPlayer** : ce sont des classes prédéfinies dans le SDK Android. *MediaRecorder* est utilisé pour l'enregistrement audio et *Mediaplayer* pour la lecture audio.
- **Librairie Gson** : pour la sérialisation et la dé-sérialisation des objets.

  Il faut noter que nous utilisons des fichiers au format *JSON* pour la sauvegarde des annotations. Les objets *JSON* seront convertis en objets *Java* pour les traitements. Ces objets *Java* sont à leur tour reconvertis en objets *JSON* pour la sauvegarde. La librairie Gson est utilisée pour faciliter la réalisation de ces opérations. C'est une librairie Java proposée par Google permettant de convertir des objets *Java* en *JSON* et inversement.

III. Conduite du projet et implémentation

III.1. Équipe projet

L'encadrement pédagogique du projet a été assuré par M. Vincent BARICHARD (aspects techniques) et Mme. Rachel COUDRAY (aspects management de projet).

– ***Les chefs de projets*** : constitués de nous-mêmes deux étudiants de master 2 ACDI

- CISSÉ Sékou Aboubacar
- TRAORÉ Hamidou

  Nous avons assuré la conduite et le management du projet. Nous étions chargés notamment de l'élaboration du cahier des charges, de la définition et de l'attribution des tâches, de l'animation des réunions. Nous avions sous notre direction le groupe de développeurs et nous rendions régulièrement compte aux encadreurs pédagogiques qui devaient valider nos choix.

– ***Le groupe de développeurs*** : il est principalement assigné aux tâches de programmation des fonctionnalités selon les spécifications techniques définies de commun accord avec les chefs de projet. Il est constitué par quatre étudiants de master 1 informatique :

- LEMOINE Théo
- ROUSSEAU Nans
- SAOUT Thomas
- VAUX Charline

Il faut ajouter à ces acteurs Mme. Christèle DUBLE, avec qui nous avons tenu des rencontres pour dans un premier temps afin de recueillir les besoins et les contraintes à prendre en compte dans cette application et dans un second temps faire valider les fonctionnalités que nous avons définies.

III.2. Définition et répartition des tâches

La liste des tâches, leur délai de réalisation et leur répartition est donnée dans le tableau suivant :

Nom	Date de début	Date de fin	Responsable
App Visualisation et Annotation de vidéo	10/09/18	08/02/19	
Récueil et spécification du besoin	10/09/18	21/09/18	CISSE, TRAORE
Analyse et conception	10/09/18	27/09/18	
Cahier de spécification fonctionnel	10/09/18	21/09/18	CISSE, TRAORE
Cahier des spécifications techniques	12/09/18	27/09/18	CISSE, TRAORE
Montée en compétence sur les technologies	12/09/18	09/10/18	Groupe développeurs
Prémière phase de Développement	10/10/18	07/12/18	
Interface générale de l'application	10/10/18	30/10/18	Théo
Gestion de la liste de vidéos et des annotations	01/11/18	21/11/18	Théo
Implémentation du lecteur vidéo et ses fonctionnalités	10/10/18	21/11/18	Thomas
Gestion du lancement de la pile d'annotation	21/11/18	05/12/18	Thomas
Lecture et sauvegarde des annotations	01/11/18	05/12/18	Théo
Ajout et enregistrement d'annotation audio et textuelle	10/10/18	05/12/18	Nans
Ajout et enregistrement d'annotation graphique	10/10/18	05/12/18	Charline
Tests et correctifs	12/11/18	07/12/18	CISSE, TRAORE
Rédaction de rapport M1	01/10/18	11/12/18	Groupe développeurs
Deuxième phase de développement	07/01/19	08/02/19	
Révue de code	07/01/19	24/01/19	Chefs de projet
Re-structuration du code du projet	07/01/19	11/01/19	CISSE, TRAORE
Gestion de l'affichage automatique des annotation	14/01/19	24/01/19	TRAORE
Gestion de l'import et du renommage de vidéos	16/01/19	31/01/19	CISSE, TRAORE
Implémentation du mode zoom	07/01/19	06/02/19	CISSE
Tests et correction de bugs	16/01/19	08/02/19	CISSE, TRAORE
Rédaction de rapport M2	16/01/19	08/02/19	CISSE, TRAORE

Illustration 5: liste des tâches et répartition

III.3. Implémentation des fonctionnalités

Comme on peut le voir sur la liste des tâches et répartition, il y a eu deux phases dans le développement :

(1) Première phase : le groupe des master 1

Pendant cette première phase de développement, chaque membre du groupe de développeurs (master1) s'est investi à développer les fonctionnalités qui lui avaient été confiées en fonction des spécifications définies. Les chefs de projet s'assuraient pendant ce temps du bon déroulement de l'exécution des tâches par leur appuis technique. Elle s'est déroulée entre le 10/10/2019 et 05/12/2018. A l'issue de cette phase, nous avions un premier prototype de l'application avec les fonctionnalités suivantes :

- ***L'interface globale de l'application*** : repartit en trois zones avec un menu dans la barre de titre. (cf figure suivante)
- ***Gestion du chargement de la liste des vidéos*** : dans la première zone s'affiche la liste des vidéos de la catégorie sélectionnée. Le clic sur vidéo de cette liste lance la lecture de celle-ci dans le lecteur et affiche les annotations associées dans la troisième zone.
- **Lecteur et gestion de la lecture des vidéos** : élément centrale de l'application, le lecteur de vidéo charge et permet la lecture vidéo. Il intègre le contrôleur de lecture vidéo telle que la pause, la reprise de lecture et le mode ralenti. Le mode zoom n'était pas encore pris en charge dans cette première version.
- ***L'ajout et la sauvegarde d'annotation*** : on pouvait dans cette première version éditer et sauvegarder les trois types d'annotations (textuelle, audio, graphique) définies dans les spécifications.
- ***Liste des annotations et affichage*** : dans la troisième zone de l'interface de l'application s'affiche la liste de annotations de la vidéo en cours de lecture. La fonctionnalité de lancement n'étant pas encore disponible dans cette version, on pouvait lire (afficher) une annotation en cliquant sur celle-ci. La vidéo se met alors en pause pendant la durée de la lecture de l'annotation.

L'implémentation d'autres fonctionnalités étaient déjà cours mais n'étaient pas assez avancées pour être exploitables. D'autres bien que fonctionnelles, nécessitaient des améliorations plus ou moins majeures. Ces fonctionnalités sont évoquées dans la partie consacrée à la deuxième phase de développement.

(2) Deuxième phase : le groupe des master 2

Cette deuxième phase s'est déroulée après la soutenance des master 1 entre le 14/12/2018 et 10/02/2019. Le groupe de développeurs n'étant plus de la partie et les fonctionnalités restant à développer étaient de notre responsabilité en tant que chefs de projet.

Nous avons d'abord commencer par restructurer le code existant et corrigé les petits bugs relevés à l'issue de tests. Les tâches majeurs ont été :

- ***La gestion du lancement (lecture) automatique des annotations*** : cette fonctionnalité bien qu'ébauchée lors de la première phase n'était pas bien avancée dans la réalisation. Nous avons gardé le principe mis en place et mener à bout son implémentation.

  Il s'agit essentiellement d'une classe Java qui étend la classe *Runnable* pour gérer la file (liste) d'annotation. Cette classe est activée au lancement d'une vidéo et la file d'annotation est initialisée avec les annotations de la vidéo. Elle utilise un *Handler* pour envoyer des messages à l'activité (*MainActivity*) principale qui se charge de l'affichage (lecture) des annotations. Elle gère également un *écouteur d'événement* qui permet de suivre le *time-line* de la vidéo pour pouvoir envoyer les messages de lancement des annotations.

- ***L'implémentation de la prise en charge du mode zoom*** : elle s'est basée sur les possibilités d'extension offertes par la librairie *Exoplayer*. Pour ce faire, nous avons redéfinie toute la logique de la classe du lecteur (interface + contrôle) :

  - **la classe *ZoomableTextureView*** : elle étend la classe prédéfinie *TextureView* et défini les propriétés d'affichage de l'interface du lecteur : dimensions d'affichage, détection d'évènements de clic (appuis) et de drag (essentiel pour le zoom).

  - **la classe *ZoomableExoPlayerView*** : elle hérite de *FrameLayout* et utilise des classes prédéfinies de la librairie *ExoPlayer* et la classe nommée ci-dessus pour redéfinir toutes les propriétés de notre lecteur et les différentes fonctionnalités de contrôle de celui-ci.

- ***Autres fonctionnalités et améliorations*** : entre autres fonctionnalités dont nous avons dû achever l'implémentation, on peut citer :

  - l'import et le renommage de vidéo de vidéo,

  - la gestion des profils (passage du mode consultation au mode édition et vice-versa)

  - amélioration de l'interface graphique de façon générale

A l'issue de cette deuxième phase de développement, nous avons une version exploitable de l'application en terme de fonctionnalités avec une interface plus ou moins intuitive. Certaines fonctionnalités reste à améliorer et pour un outil vraiment aboutit d'autres fonctionnalités sont à ajouter. Nous évoquerons ces fonctionnalités dans la suite du document.

Un aperçu de l'interface de l'application est illustré dans la figure ci-dessous :

Illustration 6: interface générale du lecteur

III.4. Sources du projet

Git – GitHub : standards pour le développement collaboratif, ces outils offrent un service en ligne pour héberger les codes des projets et pour suivre et bien gérer les évolutions. Nous avons de ce fait utilisé ces outils pour faciliter le suivi et l'intégration des codes de chaque membre du projet. Les sources sont disponibles à l'URL suivante : https://github.com/VideoAnnot/VideoAnnotTool.git.

III.5. Planification et tenu de rencontres client

Le sujet du projet bien que proposé par le département informatique de l'université d'Angers, est né d'un réel besoin notamment au sein d'un ensemble de club gym. Il était donc logique et nécessaire d'avoir le point de vue des acteurs de ce domaine pour comprendre avec plus de précision le besoin et les attentes d'une telle application. C'est alors que nous avons eu pour référente client Mme DUBLE avec qui nous avons tenue régulièrement des rencontres :

(1) 03/10/2018 : cette première rencontre a eu pour objectif de définir le contexte et de recueillir les attentes des futurs utilisateurs de l'application.

(2) 26/11/2018 : nous avons présenter lors de cette rencontre un premier prototype de l'application au client. L'objectif était d'avoir son point de vu pour ce qui était de l'interface globale et des fonctionnalités prévues. Elle a permis de confirmer certains choix et d'ouvrir la porte à d'autres idées.

(3) 01/02/2019 : cette troisième rencontre a également eu pour objectif de présenter une deuxième version de l'application qui comportait plus de fonctionnalités. C'était aussi le lieu de tester l'application sur une tablette du client. Nous avons pour l'occasion noter d'autres idées d'amélioration de la part du client.

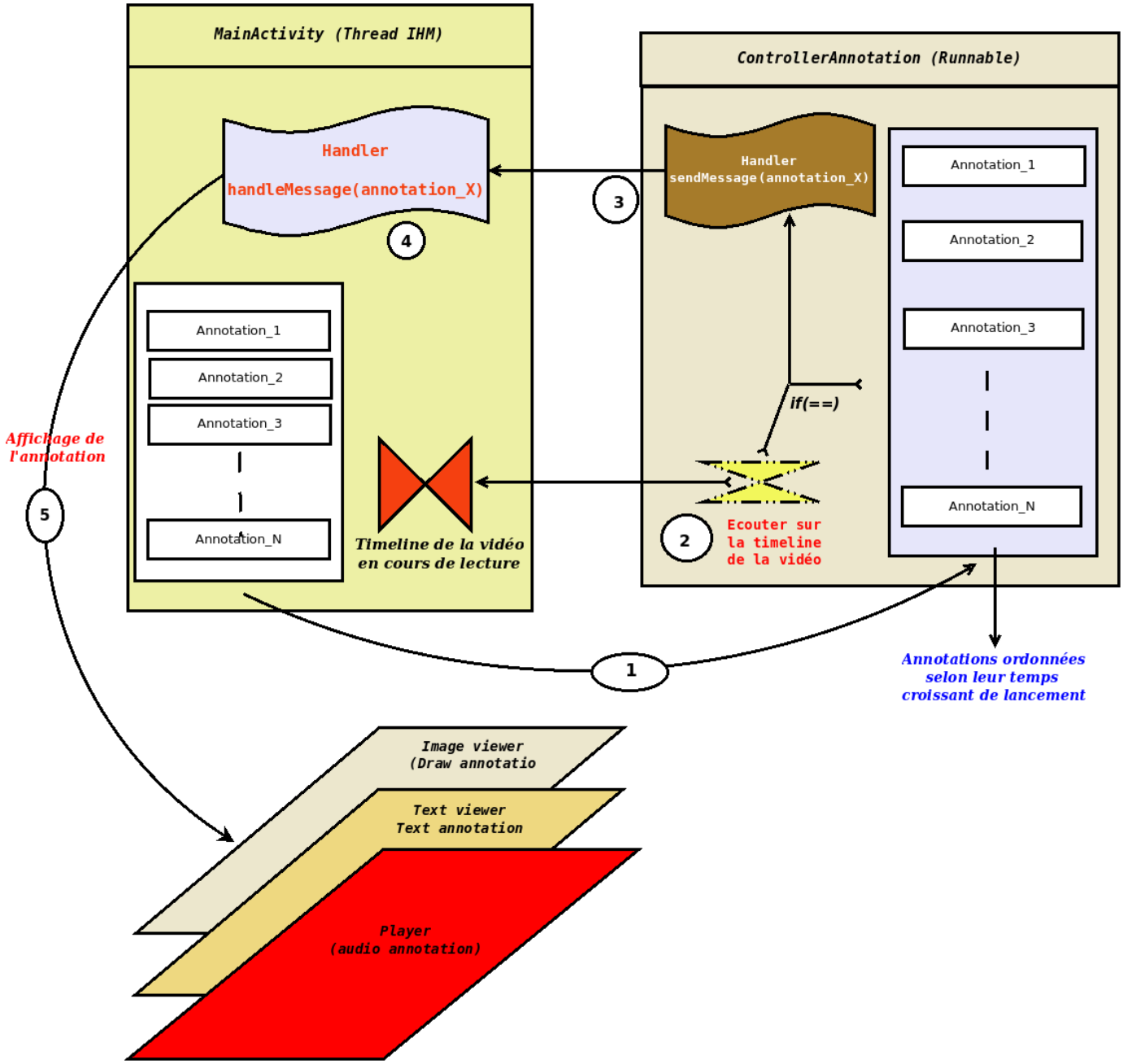
M. Vincent BARICHARD a pris part a toutes ces rencontres en tant qu'enseignant référent.

En plus de ces rencontres, Mme DUBLE a assisté à la soutenance des étudiants du master 1 et sera également présente à celle de master 2.

IV. Architecture applicative et lecture des annotations

IV.1. Système de lecture/affichage des annotations

La gestion de la lecture des annotations est illustrée par la figure suivante :



Le fonctionnement de mécanisme peut être détaillé comme suit :

(1) Initialisation du Handler et de la liste des annotations

A chaque chargement d'une vidéo dans le lecteur, une nouvelle instance de *ControllerAnnotation* est créée. Cette instance est initialisée avec la liste des annotations de la vidéo et le *Handler* du Thread principal (*MainActivity*). Cet *Handler* sera ensuite utilisé pour l'envoie de message au même Thread principal.

L'instance de *ControllerAnnotation* dispose également d'une interface d'écoute pour accéder aux méthodes du lecteur (*ExoPlayer*) : méthodes d'accès au temps courant sur la vidéo, à la durée de la vidéo ...

(2) Écouteur sur la timeline de la vidéo

Il faut noter que la classe *ControllerAnnotation* implémente l'interface *Runnable* pour lancer ses instances à travers des threads.

L'interface d'écoute sur les méthodes du lecteur lit continuellement la progression de la vidéo dans le thread de l'objet auquel il est associé.

(3) Envoi de messages au Thread principal : sendMessage()

Le *ControllerAnnotation* dispose les annotations dans une file selon leur temps croissant de lancement. Le temps de lancement de l'annotation en tête de liste est en permanence comparé avec le temps courant de lecture de la vidéo. Dès que ces deux valeurs sont égales, le *Handler* envoie un message au Thread d'IHM avec l'annotation en question grâce la méthode *sendMessage()*. L'annotation envoyée est supprimée de la file d'annotation à lancer et la suivante est immédiatement placée en tête de file d'attente.

(4) Gestion des messages reçus : handleMessage()

Le *Handler* au niveau du Thread d'interface surcharge sa méthode *handleMessage()* pour répondre aux messages qui lui sont envoyés. Dans notre cas, ces messages encapsulent pour chacun un objet annotation, ce dernier disposant des propriétés tel que le type de l'annotation, le temps de lancement sur la vidéo, la durée, le chemin éventuel vers les fichiers associés (audio, image). Le message est traité par le *Handler* à partir de ces propriétés et l'interface est alors mise à jour [5] (affichage pour les annotations graphique et textuelle, lecture audio pour audio). Pendant la durée de lecture de l'annotation, la vidéo est mise en pause. A la fin du traitement de l'annotation, la lecture vidéo continue automatiquement.

Nous illustrons dans les captures de code suivantes, les cas d'utilisation des méthodes du Handler citées :

sendMessage()

```
// methode pour lancer les annotation en foncton de leur type
public void launch(Annotation annotation) {

    Message message = mainHandler.obtainMessage();
    Bundle bundle = new Bundle();
    bundle.putSerializable("annotation", annotation);
    message.setData(bundle);
    mainHandler.sendMessage(message);

}
```

sendMessage()

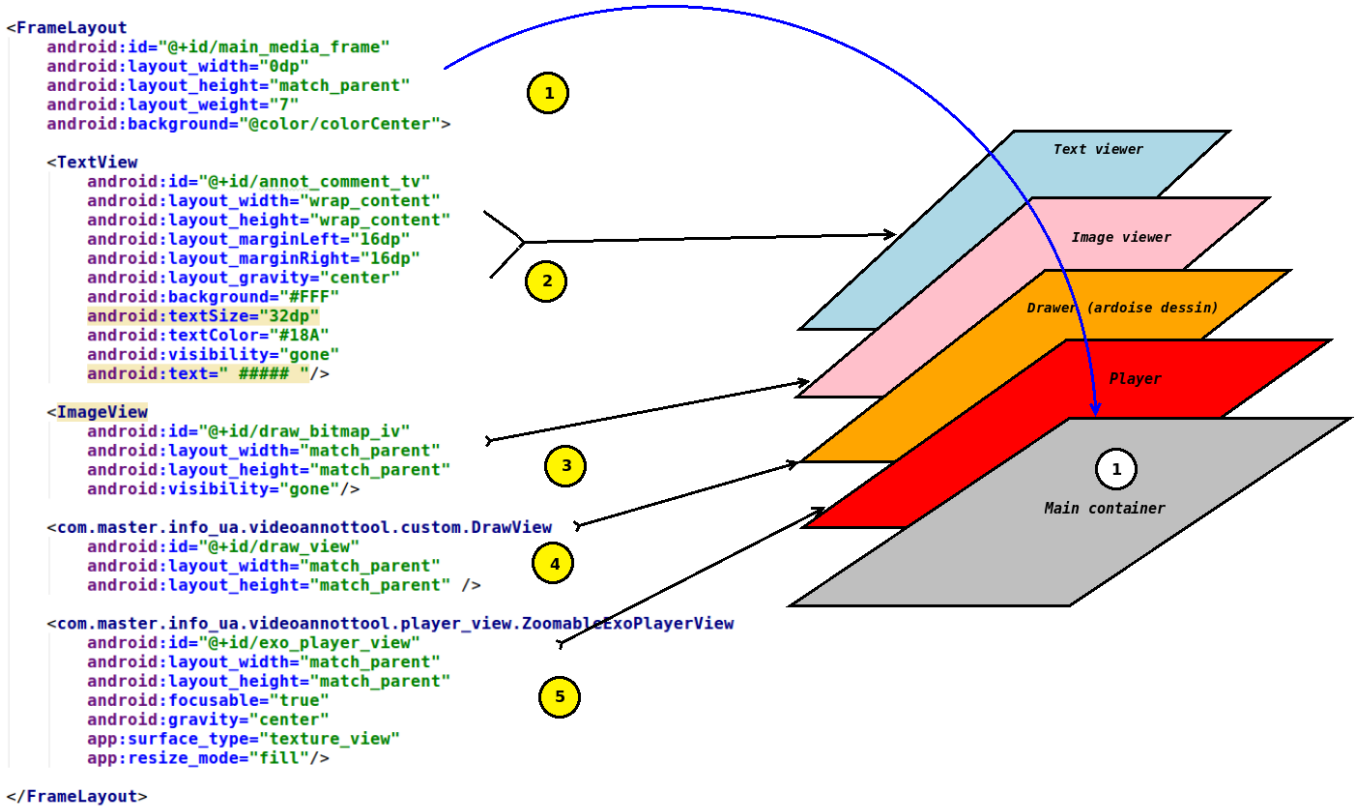
```
mainHandler = new Handler(new Handler.Callback() {
    @Override
    public boolean handleMessage(Message msg) {

        Bundle bundle = msg.getData();
        Annotation annotation = (Annotation) bundle.getSerializable( key: "annotation");
        //Lancement de l'annotation
        onAnnotationLauched(annotation);

        return true;
    }
});
```

IV.2. Agencement des composants de l'interface du lecteur

L'interface du lecteur est composé d'une combinaison des plusieurs composant d'interface Android pour alternativement ou simultanément afficher un calque de dessin, afficher du texte ou de l'image pendant la lecture vidéo. L'ensemble est disposé dans un *FrameLayout* qui est un composant adapté pour ce genre de tâche. Cet agencement est illustré par la figure suivante :



On peut remarquer que le *FrameLayout* (Main Container) contient les autres composant. Chacun de ces composant est affiché ou masqué au besoin. Ainsi au lancement de la vidéo, c'est le *player* seulement qui est visible avec la vidéo en lecture. Pour enregistrer une annotation dessin, le Drawer est mis en avant. Après sauvegarde du dessin, il est immédiatement masqué. De même l'*Image Viewer* et le *Text viewer* sont respectivement mis en avant lorsqu'on veut afficher une annotation graphique et textuelle. A l'issue de la lecture de l'annotation, le composé sollicité est immédiatement masqué.

A cela s'ajoute les boite de dialogues pour l'édition des annotations : audio, graphiques et textuelles.

Il faut noter que le composant *ZoomableExoPlayerView* intègre les événements pour le zoom. Le ralenti est plus coté code Java en se basant sur les fonctions du moteur offert par la librairie *ExoPlayer*.

V. Post-mortem – bilan et perspectives

V.1. De l'atteinte des objectifs

Il faut rappeler que le déroulement du projet s'est tenu en deux étapes :

(1) Objectifs master 1

Dans la première étape, nous étions dans le rôle de managers du projet avec une équipe de 4 étudiants de master 1 ayant le rôle de développeurs. Pour cette première étape l'objectif principal était d'aboutir à une version basique de l'application avec les fonctionnalités suivantes :

- une interface assez ergonomique et intuitive
- un moteur de lecteur capable de prendre en charge la lecture d'une vidéo avec les contrôles de lecture nécessaire (pause, retour arrière, avance, ralenti, plein écran)
- l'édition et la sauvegarde des trois types d'annotations : textuelle, graphique et audio
- l'intégration de la lecture de ces annotations pendant la lecture vidéo (chaque annotation ayant un temps de lancement sur la vidéo et une dureé d'affichage ou de lecture)
- l'import d'une vidéo avec éventuellement ses annotations

Après que ces fonctionnalités à développer aient été définies, des délais avaient été fixés pour chacune. Les étudiants de master 1 ont débuté la programmation la semaine du 10/10/2018 et ont continué dessus selon leur planning jusqu'à la semaine 10/12/2018. A cette date, l'application résultante comportait la majeure partie des fonctionnalités ci-dessus citées. Les fonctionnalités manquantes ou non-abouties étaient :

- ✗ l'intégration de la lecture de ces annotations pendant la lecture vidéo : cette fonctionnalité n'était pas aboutit. Elle pouvait juste alerter sur le fait qu'un point de lancement d'une annotation a été atteint.

  A la place, on pouvait lire (afficher) une annotation en cliquant sur celle-ci dans la liste des annotations liées à la vidéo en cours de lecture.
- ✗ l'import d'une vidéo avec éventuellement ses annotations : cette fonctionnalité était absente et son implémentation n'avait pas encore débuté.

En somme l'objectif affiché pour cette première étape n'était pas atteint et cela pour un certain nombre de raisons :

- le niveau moyen des étudiants dans les technologies utilisées
- la mauvaise estimation de la difficulté de certaines tâches
- le rythme des séances de travail sur le projet : souvent assez décalé d'une séance à l'autre

(2) Objectifs masters 2

La deuxième étape sur le développement des fonctionnalités de l'application intervient après la soutenance des master 1 et particulièrement de *janvier à la première semaine de février*. Pour cette étape nous nous étions fixés pour objectifs :

- restructurer le code produit par les étudiants de master 1
- finir l'implémentation des fonctionnalités qui étaient prévues pour la phase d'avant
- définir des interfaces et fonctions d'ajouts pour des annotations prédéfinies
- implémenter et intégrer le zoom
- et si le temps le permet, implémenter la fonctionnalité de partage de vidéos annotées à travers l'application.

La priorité à cette étape était mise sur la finition des fonctionnalités qui étaient prévues pour une version de base exploitable : il s'agit de l'import de vidéos et de la lecture automatique des annotations. Nous avons ensuite fait quelques tests, implémenté la fonctionnalité du zoom qui fut assez complexe à réaliser. Nous avons également revu l'interface de l'application et fait des tests et corrections pour rendre l'application assez robuste.

A l'issue de ces deux étapes, nous avons à ce jour une version exploitable de l'application bien qu'elle nécessite encore bon nombre d'améliorations notamment en terme d'ergonomie de l'interface et aussi en terme de fonctionnalités :

- ✗ import de vidéos déjà annotées
- ✗ partage de vidéos annotées
- ✗ ajout d'annotations prédéfinies
- ✗ gestion de la lecture des annotations en mode plein écran

V.2. Du management du projet

Pour ce qui est de notre conduite du projet, nous avons dès le départ fait le choix d'aller dans une dynamique d'autonomie supervisée avec notre équipe de développeur. Ainsi nos choix effectués ne leurs étaient pas imposés mais soumis au préalable à leurs amendements. Puis des décisions finales étaient adoptées par tout le groupe avant de passer à leurs applications. Pour preuve, chaque membre de l'équipe des développeurs s'est lancé sur la fonctionnalité sur laquelle il se sentait le plus à l'aise avec des objectifs plus ou moins personnels à atteindre à l'issue du projet. Des délais étaient fixés mais chacun était libre d'avancer à son rythme.

Pour la communication autour du projet, en plus des séances communes de management de projet avec les étudiants de master 1, nous leur demandions souvent des comptes rendus (individuels) sur leur état d'avancement.

Nous animions les réunions et leur servions d'appuis (technique) au besoin. Nous faisions régulièrement la fusion des fonctionnalités développées au fur et à mesure que celle-ci étaient envoyées sur le dépôt central avec éventuellement les corrections et ajustements nécessaires.

Un aspect négatif dans notre démarche a été le trop de confiance qui faisait qu'on s'en tenait à de compte rendu verbaux sans vérifier le résultat. Cela a eu un impact sur le respect des délais dans la réalisation de certaines fonctionnalités tel que l'édition et l'enregistrement d'annotation graphique, la gestion de la liste des annotations et l'import de vidéos. Il faut encore pointé ici du doigt le rythme imposé par le département pour travailler sur le sujet.

On n'oublie pas de mentionner la gestion de la documentation liée à la conduite d'un tel projet. Nous avons au début du projet élaboré un cahier des charges fonctionnel et un cahier des charges technique. Des maquettes d'interfaces ont aussi été conçues pour faciliter les échanges avec le client quant au rendu final attendu. Pour enfin revenir sur cet aspect très important dans la conduite du projet, nous avons tout au long du projet eu des échanges avec le référent client. Elles ont l'intérêt de mieux cerner les besoins métiers, d'informer le client sur l'avancement du projet et de réajuster les objectifs au fil du projet et dès l'arrivée de nouvelles contraintes. Ces échanges étaient principalement du mailing et des rencontres physiques.

Les comptes rendus de réunions et de séances ont été régulièrement transmis aux différents acteurs.

V.3. Perspectives

Nous mentionnons ici des points à travailler pour améliorer l'application :

- Recherche de vidéos par mot-clés
- Création de catégories et sous-catégories de façon dynamique
- Définition d'annotations prédéfinies (cercle, rectangle, triangle,…) pour effectuer une annotation graphique
- Possibilité de réédition (modification) d'une annotation sans avoir à la supprimer pour la rééditer
- Pouvoir renommer les vidéos, et de même les dossiers et sous-dossiers qui les contiennent.
- En mode plein écran avoir des marqueurs sur la bande en bas pour se positionner directement sur les annotations
- Prise en charge de la lecture des annotations texte et dessin en plein écran
- Pouvoir noter des vidéos en favoris pour les retrouver plus facilement
- Flouter les visages en mode annotation
- Revoir la taille du bandeau du bas pour une utilisation sur téléphone, la vidéo apparente est très réduite pour une bonne exploitation dans la version actuelle
- Pouvoir renommer les catégories et sous-catégories pour utilisation dans une autre discipline
- Rajouter une vignette pour chaque vidéo dans le menu déroulant
- Rajouter des formes pour les annotations, on peut choisir des couleur différentes, on pourrait choisir une ligne droite, un cercle, un carré ou une ligne courbe
- Insérer un chrono sur la vidéo qui respecte le temps réel même quand on est en mode ralenti.
- Dans le menu déroulant des sous-catégories, voir un filtre « *difficulté* » pour faire apparaître que certains éléments suivant leur difficulté pré-établie lors de l'import. Peut-être au même titre que ajouter une vignette issue de la vidéo, on pourrait ajouter un niveau de difficulté.

Certaines fonctionnalités bien qu'utilisables dans la version actuelle de l'application peuvent être améliorées. Cela nécessite de revoir le code déjà en place. C'est notamment le cas du *zoom* et de la prise en charge de la file d'annotation : lecture et affichage à la volée. Nous avons mise en place une structuration modulaire du code avec assez de commentaires ce qui facilitera sa reprise par un acteur qui n'a pas eu à travailler avec nous sur ce projet.

En tant qu'outil abouti et facile d'utilisation, il peut être envisager une interface web pour la gestion des vidéos propre à l'ensemble de club ou à un club en particulier. Ceci faciliterait la gestion des sources de données et aussi le partage sur une plus large échelle.

Conclusion

Au terme de ce projet, il faut noter que l'implémentation des fonctionnalités les plus importantes permettant une utilisation de l'application ont été menées à bout. Des améliorations restent à faire et aussi elle nécessiterait de nouvelles fonctionnalités pour être plus efficace et plus facile à utiliser.

En plus de parvenir à livrer une application fonctionnelle, on peut noter d'autres aspects qui nous ont été beaucoup bénéfiques à l'issue de ce projet. Nous avons pu avoir un aperçu des réalités liées au développement d'une application tel qu'il se déroule dans le milieu professionnel. Il a été pour nous une occasion de mettre en pratique les connaissances acquises dans les différentes unités disciplinaires et particulièrement l'aspect management d'un projet informatique.

Enfin le projet s'est déroulé en deux temps. Dans un premier temps, nous avons travaillé avec quatre étudiants du master 1 informatique. Nous avions le rôle de chefs de projet, et les étudiants de master 1 dans le rôle de développeurs sous notre direction. Dans un second temps, les chefs de projets ont pris les choses en main afin de livrer une version fonctionnelle avec tous les engagements tenus depuis le départ. Arriver à manager une équipe de développeur sans expérience préalable était pour nous un challenge. Nous pouvons affirmer sans prétention que nous y somme parvenu au-délà des points d'amélioration évoqués plus haut.

Bibliographie

[1] - https://developer.android.com/guide/topics/media/mediaplayer

[2] - https://academy.realm.io/posts/360andev-effie-barak-switching-exoplayer-better-video-android/

[3] - https://github.com/google/gson

[4] - https://github.com/Manuiq/ZoomableTextureView

[4] – Cours de développement mobile en Master de Mr Vincent BARICHARD