Université De Montpellier

Rapport Final

Programmation Mobile

Ingo DIAB
Chamy KACI

Mai 2023

Table des matières

1 Liens

Nos vidéos sont disponible sur YouTube (`https://youtu.be/rurJqKfhuD0` pour l'application et `https://youtu.be/IRjFnJpdqnk` pour l'ajout des groupes après réouverture du dépot) et sur notre Github dans Rendu Final (avec nos diapos). Voici le lien GitHub de notre application : `https://github.com/IngoDiab/HAI811_Dev_Mobile/tree/main/Projet`. Ce rapport explicite ce qui a été montré dans la vidéo (la partie après la conclusion porte sur le travail effectué après réouverture du dépôt).

2 Présentation du projet

Notre application s'appelle Reverleaf et a été testée sur un virtual device Pixel 6 API 33. Cette application gère la création d'évènements sportifs et culturels. L'application est codée en Java/XML. Nous avons utilisé plusieurs API :

1. Firebase Realtime Database pour gérer les datas de l'application (les évènements, les utilisateurs, les catégories,...).
2. Firebase Authentication pour les inscriptions/connexions (que ce soit avec un email et mot de passe ou avec un compte Google).
3. Firebase Storage pour stocker des fichiers (pour nous ce sont des images).
4. Picasso pour permettre de charger des images stockées sur Firebase Storage.
5. Paypal Checkout pour permettre à l'utilisateur les paiements via Paypal (nous avons utilisé des comptes Sandbox pour effectuer nos tests).
6. Google Places pour permettre l'autocomplétion des lieux lorsque un utilisateur veut créer/chercher un évènement et doit rentre le lieu de l'évènement.

3 Architecture

Nous avons plusieurs classes représentant des data (User, Events, Abonnements, ...) ainsi que des Managers permettant de condenser le code à un unique endroit et éviter la réplication de code. Notre code utilise énormément les Consumer<T> que nous utilisons comme callbacks (bind pour les boutons, exécution lorsque nous récupérons des data depuis Firebase,...).

3.1 ViewManager

Ce ViewManager contient toutes les méthodes relatives aux Views (récupération d'élément avec findViewById, affichage de popup, bind sur le OnClick, création de View en runtime, ...). Par exemple, une méthode que nous avons beaucoup utilisé est BindOnClick(View _view, Consumer<View> _callback).

Cette méthode permet de bind le _callback sur le OnClick de _view. Cette méthode est très utile car elle nous permet d'éviter d'utiliser setOnClickListener partout où nous voulons bind le OnClick d'une View. Nous avons donc moins de réplications de code et un code beaucoup plus propre.

```
1 usage   IngoDiab
protected void BindButtons()
{
    ViewHelper.BindOnClick(mFavoris, _view->ManageFavorisEvent(!mUserHasInFavoris));

    ViewHelper.BindOnClick(mCategorie, _view->ToggleSearch(mCategorie));
    ViewHelper.BindOnClick(mDate, _view->ToggleSearch(mDate));
    ViewHelper.BindOnClick(mLieu, _view->ToggleSearch(mLieu));
    ViewHelper.BindOnClick(mPrice, _view->ToggleSearch(mPrice));

    ViewHelper.BindOnClick(mInscription, _view->ManageInscriptionEvent(!mUserIsRegisteredToThisEvent));
    ViewHelper.BindOnClick(mSearchSame, _view->SearchSame());

    ViewHelper.BindOnClick(mBackButton, _view->finish());
}
```

FIGURE 1 – Exemples de BindOnClick

3.2 FirebaseManager

Ce FirebaseManager contient toutes les méthodes relatives à Firebase (Firebase Realtime Database, Firebase Authentication, Firebase Storage) et nous permet de lier notre application à Firebase facilement et sans duplication de code. Par exemple, dans le hub, nous chargeons tous les évènements nécessaires grâce à FirebaseManager. Nous l'utilisons aussi pour load des images et créer les catégories/abonnements de l'application en allant les chercher dans Firebase Realtime Database mais nous en parlerons plus précisément dans le sous chapitre 4.9 Catégories.

```
1 usage   IngoDiab
public void LoadEvents()
{
    mInscriptions.removeAllViews();
    mFavoris.removeAllViews();
    mTendances.removeAllViews();
    Activity _activity = getActivity();
    FragmentManager _fragManager = getParentFragmentManager();
    //Events Inscrit
    FirebaseManager.LoadEventsFromUserList(CARD_MOD.EVENTS_INSCRIPTIONS_CARD, _activity, _fragManager,  listID: "mIDInscritEvents", _cards->ViewHelper.DisplayCardsLoadedEvents(mInscriptions, _cards));
    //Events Favoris
    FirebaseManager.LoadEventsFromUserList(CARD_MOD.EVENTS_FAVORIS_CARD, _activity, _fragManager,  listID: "mIDFavorisEvents", _cards->ViewHelper.DisplayCardsLoadedEvents(mFavoris, _cards));
    //Events Tendances
    FirebaseManager.LoadMostInscriptionEvents(CARD_MOD.EVENTS_TENDANCES_CARD, _activity, _fragManager,  nbItemsByCategory: 5, _cards->ViewHelper.DisplayCardsLoadedEvents(mTendances, _cards));
}
```

FIGURE 2 – LoadEvents

3.3 PaypalManager

Enfin nous possédons aussi un PaypalManager, ce manager nous sert à initialiser les boutons Paypal ainsi qu'à effectuer les paiements.

FIGURE 3 – Exemple d'utilisation du PaypalManager

4 Features

4.1 Inscription/Connexion avec Google

L'utilisateur peut s'inscrire avec un compte Google grâce à Firebase Authentication. Une fois que l'utilisateur s'est inscrit avec Google, un compte Reverleaf est créé sur Firebase Realtime Database avec le maximum d'informations disponible (l'Uid, le nom, le mail et le numéro de téléphone). Par défaut, le type d'abonnement de l'utilisateur est None. Une fois que la création du compte est terminée, l'utilisateur est redirigé vers une activité lui demandant de payer un abonnement.

4.2 Inscription

L'utilisateur peut s'inscrire manuellement grâce à Firebase Authentication en remplissant certaines données obligatoires (email, téléphone, mot de passe et confirmation du mot de passe). L'utilisateur peut aussi rentrer des informations complémentaires telles que le nom, prénom et l'adresse. L'utilisateur doit impérativement :

1. Remplir tous les champs obligatoires.
2. Avoir un email de type abc@def.xyz et un mot de passe de minimum 6 caractères.
3. Confirmer son mot de passe.
4. Accepter les conditions.

Une fois que l'utilisateur s'est inscrit, un compte Reverleaf est créé sur Firebase Realtime Database avec toutes les informations saisies + un Uid généré. Par défaut, le type d'abonnement de l'utilisateur est None. Une fois que la création du compte est terminée, l'utilisateur est redirigé vers une activité lui demandant de payer un abonnement.

4.3 Connexion

L'utilisateur peut se connecter manuellement grâce à Firebase Authentication en utilisant un email et un mot de passe. L'utilisateur peut se connecter si le duo email/mot de passe existe dans la Firebase Authentication. Si l'utilisateur connecté possède un abonnement alors il est redirigé vers l'activité Home

qui représente le hub de l'application. Sinon il est redirigé vers une activité lui demandant de payer un abonnement.

4.4 Abonnement

L'utilisateur doit avoir un abonnement pour pouvoir utilisé l'application. Ces abonnements ne sont pas hardcodés mais ils sont chargés depuis Firebase Realtime Database (les datas de l'abonnement) et Firebase Storage (l'image de l'abonnement). Ainsi il est très facile pour le développeur de créer un nouvel abonnement sans toucher au code. Pour créer un nouvel abonnement, il suffit de :

1. Ajouter une nouvelle entrée à la Firebase Realtime Database avec un nom, une description, un prix et une Uri pour l'image de l'abonnement.
2. Stocker une image à l'Uri dans Firebase Storage.

De cette manière, l'abonnement sera chargé par l'application pour chaque utilisateur voulant acheter un abonnement.

Pour acheter un abonnement, l'utilisateur doit cliquer sur l'abonnement voulu. Un pop-up s'ouvrira avec les informations de l'abonnement choisi. L'utilisateur pourra cliquer sur "Pay with PayPal" afin de payer avec un compte paypal. Une fois le paiement effectué, l'utilisateur recevra l'abonnement choisi (que l'on stock dans Firebase Realtime Database) et sera redirigé vers l'activité Home qui représente le hub de l'application. A noter que les abonnements en dessous de Premium ne nous permettent pas de créer d'évènements.

4.5 Hub

Une fois arrivé sur le hub, l'utilisateur peut se déconnecter en appuyant sur le symbole Off en haut à droite de l'écran. En bas de l'écran, il y a une barre de navigation. Cette barre possède 4 icônes : le fragment principal (homepage), le fragment de recherche d'évènements, le fragment pour créer/consulter ses évènements créés et le fragment groupe (nous n'avons malheureusement pas eu le temps d'implémenter les groupes donc nous avons laissé les placeholders).

4.6 Homepage

Lorsque nous arrivons sur le hub, par défaut nous avons le fragment Homepage. L'utilisateur possède 4 listes :

1. Les évènements où il est inscrit
2. Les évènements qu'il a ajouté à ses favoris
3. Les évènements qui sont à proximité de la géolocalisation de l'utilisateur
4. Les 5 évènements les plus tendances (les évènements qui possèdent le plus d'utilisateurs inscrits).

Pour la liste d'évènements à proximité, cette liste est disponible seulement si la géolocalisation est activée (un popup demandant l'accès à la géolocalisation est affiché lorsque l'on arrive sur le hub).

L'utilisateur peut cliquer sur n'importe quel évènement affiché afin de le consulter.

4.7 Recherche d'évènement

Le 2nd fragment disponible permet de chercher un évènement. Comme critère de recherche, nous pouvons choisir un certain titre, une ou plusieurs catégories, un lieu avec Google Places ainsi qu'une fourchette de dates et de prix. Une fois tous les critères rentrés, nous pouvons appuyé sur "Rechercher" afin de trouver tous les évènements qui correspondent à notre recherche. Les catégories ne sont hardcodées et correspondent aux catégories disponibles dans Firebase Realtime Database mais nous en parlerons plus précisément dans le sous chapitre 4.9 Catégories.

4.8 Mes évènements

Ce 3e fragment regroupe tous les évènements créés et permet de créer un nouvel évènement. Si l'utilisateur a un abonnement au moins Premium, il peut créer un évènement en cliquant sur le bouton "Créer un évènement". Ce bouton nous amène sur un fragment regroupant toutes les catégories disponibles pour pouvoir créer un évènement. Lorsque l'utilisateur clique sur une catégorie, il est renvoyé sur une activité correspondant à la création d'un évènement de la catégorie choisie (par exemple si nous cliquons sur la catégorie Sport, nous sommes renvoyés sur la création d'un évènement sportif).

4.9 Catégories

Le système de catégories disponibles est, comme pour les abonnements disponibles, entièrement relié à Firebase Realtime Database. Nous utilisons le système de reflection de Java pour trouver les méthodes à exécuter selon le nom de la catégorie. Nous avons aussi une énumération nous permettant de créer des "cartes" représentants les catégories d'évènements, le tout avec différentes dimensions et pré-sets.

FIGURE 4 – Création des cartes représentant des évènements sportifs

Grâce à ce système, nous pouvons plutôt facilement créer une nouvelle catégorie utilisable par les utilisateurs. Pour cela nous devons :

1. Ajouter une nouvelle entrée à la Firebase Realtime Database avec un nom, une Uri pour l'image carrée de la catégorie et une Uri pour l'image de base de la catégorie.
2. Stocker une image pour les deux Uri dans Firebase Storage.
3. Dans le code, ajouter une activité/view "ConsultEvent_NomDeLaCategorie" (doit hériter de ConsultEvent) et "CreateEvent_NomDeLaCategorie" (doit hériter de CreateEvent) pour consulter/créer un évènement de cette catégorie.
4. Dans le code, ajouter une class "EventData_NomDeLaCategorie" (doit hériter de EventData) et ajouter les informations spécifiques à cet évent.

Ce qui est pratique c'est que nous n'avons pas de "if" ou de "switch" à changer lorsque nous voulons ajouter/supprimer une catégorie. Pour facilement rendre indisponible une catégorie, nous pouvons simplement l'enlever de la Firebase Realtime Database, ainsi l'application n'exécutera pas les méthodes liées à cette catégorie.

4.10 Créer un évènement

Lorsque l'utilisateur possède l'abonnement Premium, il peut cliquer sur "Créer un event" dans le fragment "Mes évènements". Après avoir choisi la catégorie de l'évènement, il peut rentrer les informations de l'évènement. Les premières informations sont communes à chaque catégorie et sont obligatoires : le titre, une description, un lieu (l'utilisateur utilise Google Places pour choisir le lieu), une date (l'utilisateur utilise un popup calendrier) et un prix en €. D'autres informations complémentaires sont possible selon la catégorie comme le nom d'un chanteur et le type de musique pour les évènement musicaux ou le sport pratiqué, le championnat et la durée du match pour les évènements sportifs. En appuyant sur "Créer", l'évènement se créé dans la Firebase Realtime Database. Il est maintenant disponible pour les recherches, les inscriptions, ...

4.11 Supprimer un évènement

L'utilisateur ayant créé un évènement peut à tout moment le supprimer en cliquant sur la croix blanche sur fond rouge dans le fragment "Mes Évènements". Après suppression, l'évent est supprimé de la Firebase Realtime Database et n'est plus accessible pour aucun utilisateur.

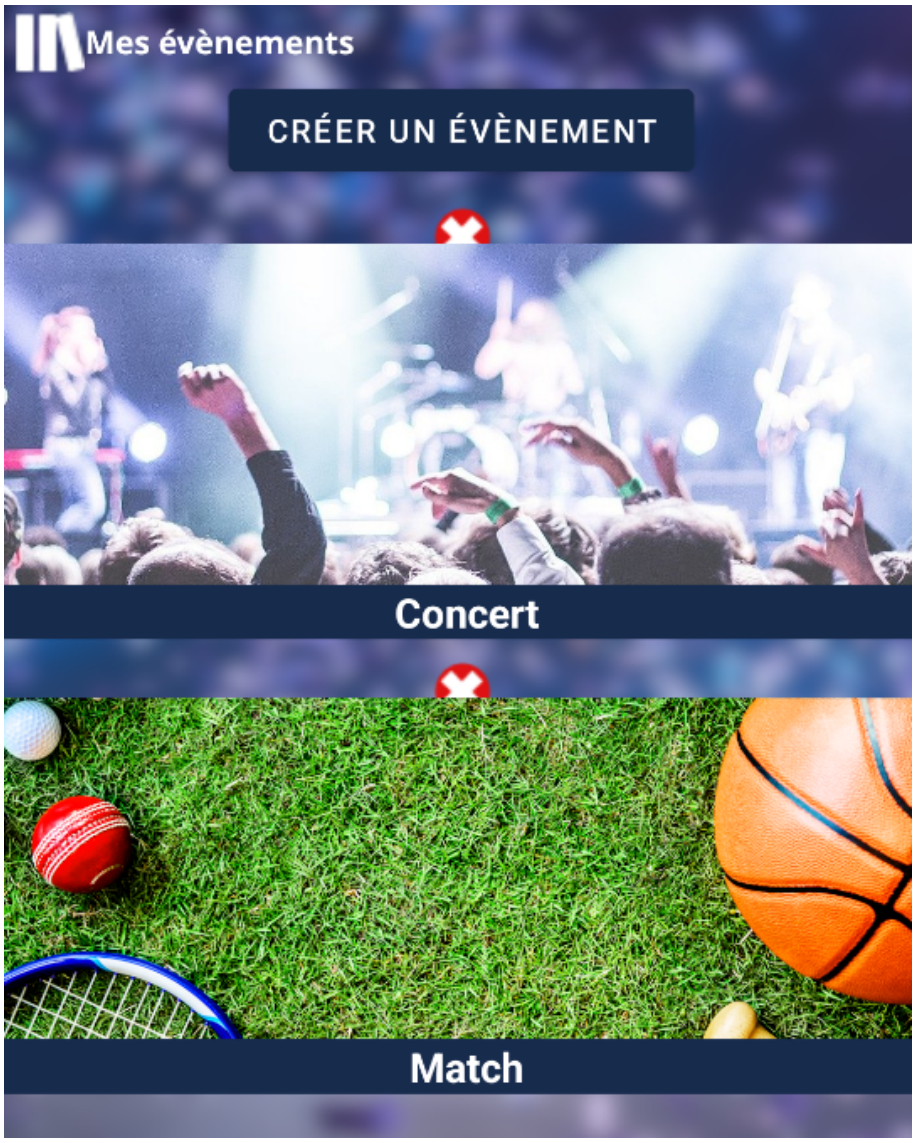

FIGURE 5 – Mes évènements

4.12 Consulter un évènement

Lorsque l'utilisateur clique sur un évènement, une activité spécifique au type de catégorie de l'évènement est lancée. L'utilisateur a accès aux informations de l'évent et à ses statistiques : le titre, le nombre d'utilisateur qui ont cet évènement en favoris, le nombre d'utilisateur inscrit à cet évent, le catégorie de l'évent, la description, le lieu, la date et le prix. Il dispose aussi des informations liées à la catégorie (chanteur, type de musique,...). A partir d'un évènement consulté, nous pouvons effectuer 3 actions : s'inscrire, l'ajouter aux favoris ou rechercher un évènement équivalent.

4.13 Inscription à un évènement

Tous les utilisateurs peuvent s'inscrire aux évènements sauf l'utilisateur qui a créé l'évènement. Pour tous les autres utilisateurs, un bouton "Inscription" est situé en bas de l'activité de l'évènement. Lorsque nous cliquons sur le bouton, il se transforme en "Désincription" afin que l'utilisateur puisse annuler son inscripition. Lors d'une inscription, le compteur d'utilisateurs inscrit à cet évent est incrémenté de 1 (si c'est un désinscription, il est décrémenté de 1).

4.14 Ajouter un évènement aux favoris

Tous les utilisateurs peuvent ajouter un évènement aux favoris sauf l'utilisateur qui a créé l'évènement et les utilisateurs qui sont déjà inscrits. Pour tous les autres utilisateurs, un bouton en forme de coeur est situé en haut à droite de l'activité de l'évènement. Le coeur est gris de base (l'évent n'est pas en favoris) Lorsque nous cliquons sur le coeur, il se transforme en coeur rouge (l'évent est en favoris). L'utilisateur peut recliquer sur le coeur pour enlever l'événèment de ses favoris. Lors qu'un utilisateur ajoute un évènement en favoris, le compteur d'utilisateurs qui ont mit cet évent en favoris est incrémenté de 1 (si c'est un utilisateur l'enlève de ses favoris, il est décrémenté de 1). Si un utilisateur s'inscrit à une évènement qu'il a ajouté à ses favoris, l'évènement est automatiquement retiré de ses favoris.

4.15 Chercher un évènement équivalent

Lorsque nous consultons un évènement, nous pouvons cliquer sur certaines informations comme la catégorie, le lieu, la date et le prix. En cliquant sur une informations le bouton devient bleu clair et l'information est sélectionnée. Lorsque nous cliquons sur le bouton "Rechercher" en bas de la page, nous recherchons tous les évènements qui correspondent aux informations sélectionnées (par exemple si nous consultons un évènement sportif qui a lieu à Montpellier le 28/05/2023 et que le prix est 10€, si nous cliquons sur la catégorie et le lieu puis sur "Rechercher", nous aurons comme résultat tous les évènements Sportif à Montpellier, peu importe la date et le prix).

5 Blocages

Nous n'avons pas eu de blocages mais nous avons rencontré un problème. Nous utilisons Picasso afin de charger des images depuis Firebase Storage. Cependant il arrive que l'image ne se charge pas (voir dans la vidéo à 1 :30, l'image de la catégorie Sport n'est pas chargée mais le bouton existe bien). Après des recherches, il semblerait que ce problème soit commun : le garbage collector de Java supprimerait la Target utilisée par Picasso car Picasso aurait une weak reference sur la Target. Nous avons essayé d'encapsuler cette Target dans un TargetWrapper et d'en faire un attribut de classe pour éviter que la Target ne soit supprimée par le garabage collector mais le problème n'est pas réglé.

6 Conclusion et perspectives

Pour conclure, ce rapport, le problème principal que nous avons rencontré durant ce projet était le temps. Pour les rendus intermédiaires, nous avons fait tout notre possible pour essayer d'avancer le projet le plus possible. Malheureusement, avec les autres matières et la surcharge de projet que nous avons eu (4/5 projets en plus de celui-ci), nous n'avons pas pu suffisamment avancé afin de pouvoir terminer ce projet dans les temps. Nous nous sommes concentrés sur le développement des features (inscriptions, connexions, évènements, favoris, etc.) en essayant d'avoir une architecture la plus propre/réutilisable et la plus facile d'utilisation possible (managers, consumers, etc.). Notre application n'exploite pas toutes les features (aucune différence entre l'abonnement Classique et l'abonnement Plus) mais nous avons préféré développer le plus de features possible étant donné que le projet à pour but l'apprentissage de la programmation Android et non la commercialisation de cette application.

Concernant les perspectives d'améliorations, nous aimerions implémenter le système de groupes (messageries, partages d'évènements, ...) que nous aurions fait si nous avions eu plus de temps. Nous aimerions aussi régler le problème de l'image qui ne s'affiche pas tous le temps et créer un système de cache pour limiter la bande passante utilisée lorsque nous chargeons des images.

7 Ajout suite à la réouverture du dépôt

Étant donné que le dépôt a été réouvert, nous en avons profiter pour créer le système de groupe avec les partage d'évènements et le chat textuel que nous voulions faire.

L'utilisateur peut désormais cliquer sur la 4e icône de la barre du menu pour aller dans le fragment Groups. Ce fragment comporte la liste des groupes auquel le joueur appartient ainsi qu'un bouton pour créer un groupe.

L'utilisateur peut cliquer sur le bouton pour créer un groupe avec le nom souhaité. Par défaut l'utilisateur est seul dans le groupe mais il peut ajouter d'autres utilisateurs avec le bouton en haut à gauche du groupe.

Pour ajouter un utilisateur, nous devons connaître son adresse mail (c'est "l'identifiant" d'un utilisateur sur notre application). Une fois l'utilisateur ajouté, une notification est envoyée dans la conversation pour montrer que l'utilisateur a été ajouté. Un utilisateur déjà présent dans le groupe ne peut pas être ajouté une seconde fois. Les utilisateurs peuvent s'envoyer des messages et seront identifié par leur prénom (s'il a été fournit à l'inscription) ou par son email (s'il n'y a pas de prénom fournit).

Enfin, les utilisateurs peuvent partager un event sur les groupes de leur choix en cliquant sur l'icône qui ressemble à "un avion en papier" sur le page de consultation de l'évent (à coté de l'icône pour ajouter en favoris). Lorsqu'un évent est envoyé dans un groupe, tous le monde peut cliquer dessus comme on pourrait le faire depuis la page Home (cliquer sur l'évent affiche les informations de cet évent, on peut s'inscrire, l'ajouter aux favoris, rechercher des événts équivalents ou même le repartager à d'autres groupes).