Preuves pour le projet Android : FUT Market réalisé par Yoann PERIQUOI et Maxim Pozdnyakov

Documentation

Je sais décrire le contexte de mon application, pour que n'importe qui soit capable de comprendre à quoi elle sert :

Vous pouvez retrouver notre description du contexte et un diagramme de classe ainsi qu'une explication de notre architecture dans le fichier Contexte et description architecture dans le dossier documentation.

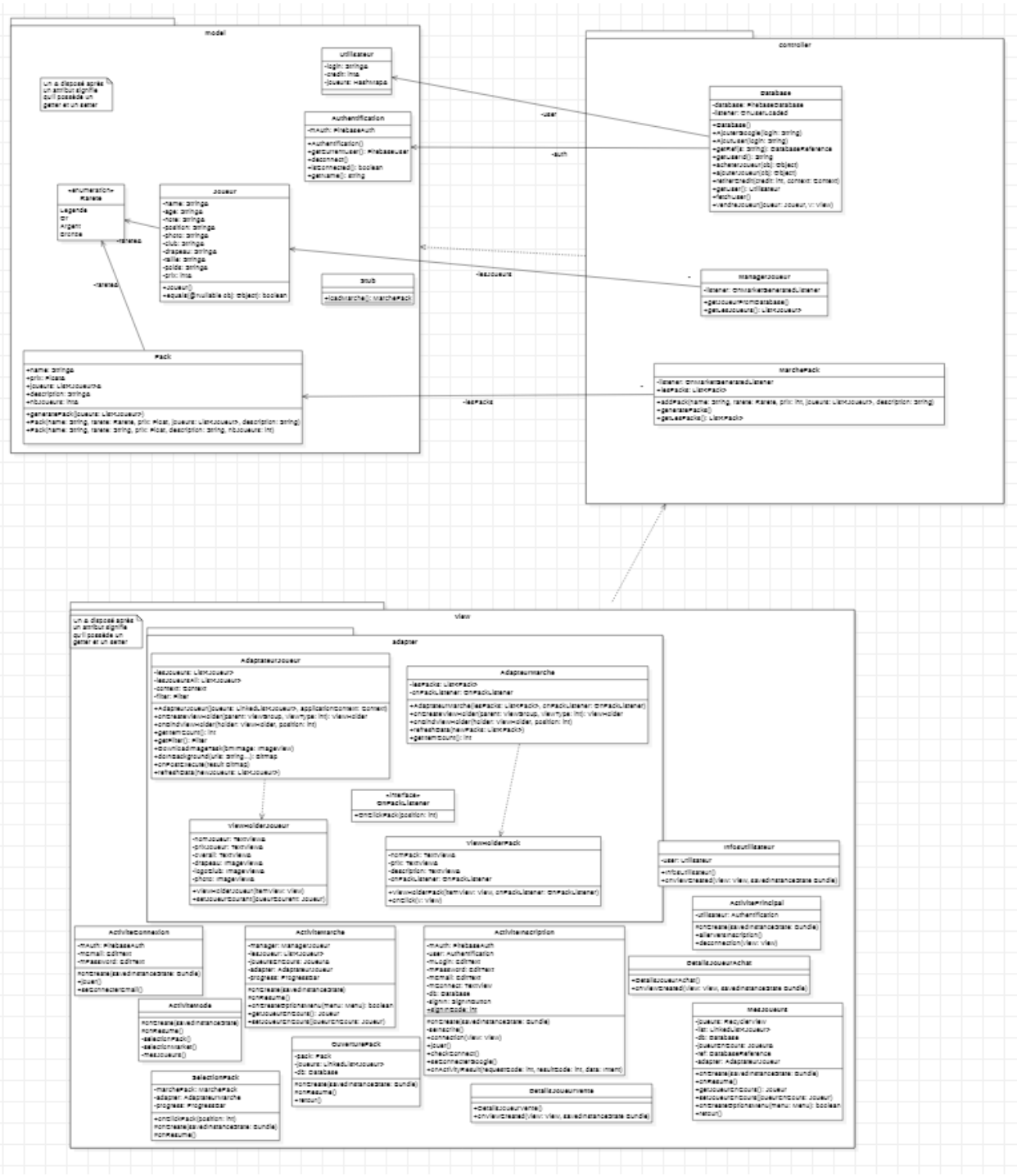
Voici le contexte :

Dans le cadre de notre matière de conception d'application mobile nous avons eu à réaliser une application mobile en duo en 7 semaines. Nous avons choisi comme sujet, le football, et plus particulièrement les joueurs de football. Nous avons voulu créer une application du style « Ligue fantasy » ou du mode Fifa Ultimate Team de Fifa qui est un jeu où les participants endossent le rôle de propriétaires d'équipes sportives. Le joueur a donc l'objectif de collectionner tous les footballeurs qui sont représenté par des cartes de jeu. Pour cela, la principale méthode pour acquérir des joueurs est d'ouvrir des packs ou paquet de cartes. Evidemment, les paquets de cartes sont donnés en échange de crédits. Ces crédits peuvent être ensuite obtenu en revendant les joueurs possédés. De plus, si vous recherchez un joueur en particulier, il peut être retrouvé dans la boutique. Il vous sera donné en échange de crédits. Il vous faudra donc ouvrir des paquets, pour ensuite vendre les joueurs obtenus pour acheter les joueurs de vos rêves et collectionner tous les joueurs en ouvrant encore une fois des paquets. Tout cela pour un seul objectif, être le plus grand collectionneur de FUT Market.

Je sais concevoir et décrire un diagramme de cas d'utilisation pour mettre en avant les différentes fonctionnalités de mon application :

Je sais concevoir un diagramme UML de qualité représentant mon application :

Voici le diagramme de classe qui peut être retrouvé dans Diagrammes, une description de celui-ci est aussi disponible dans le document Contexte et description architecture dans le dossier documentation.



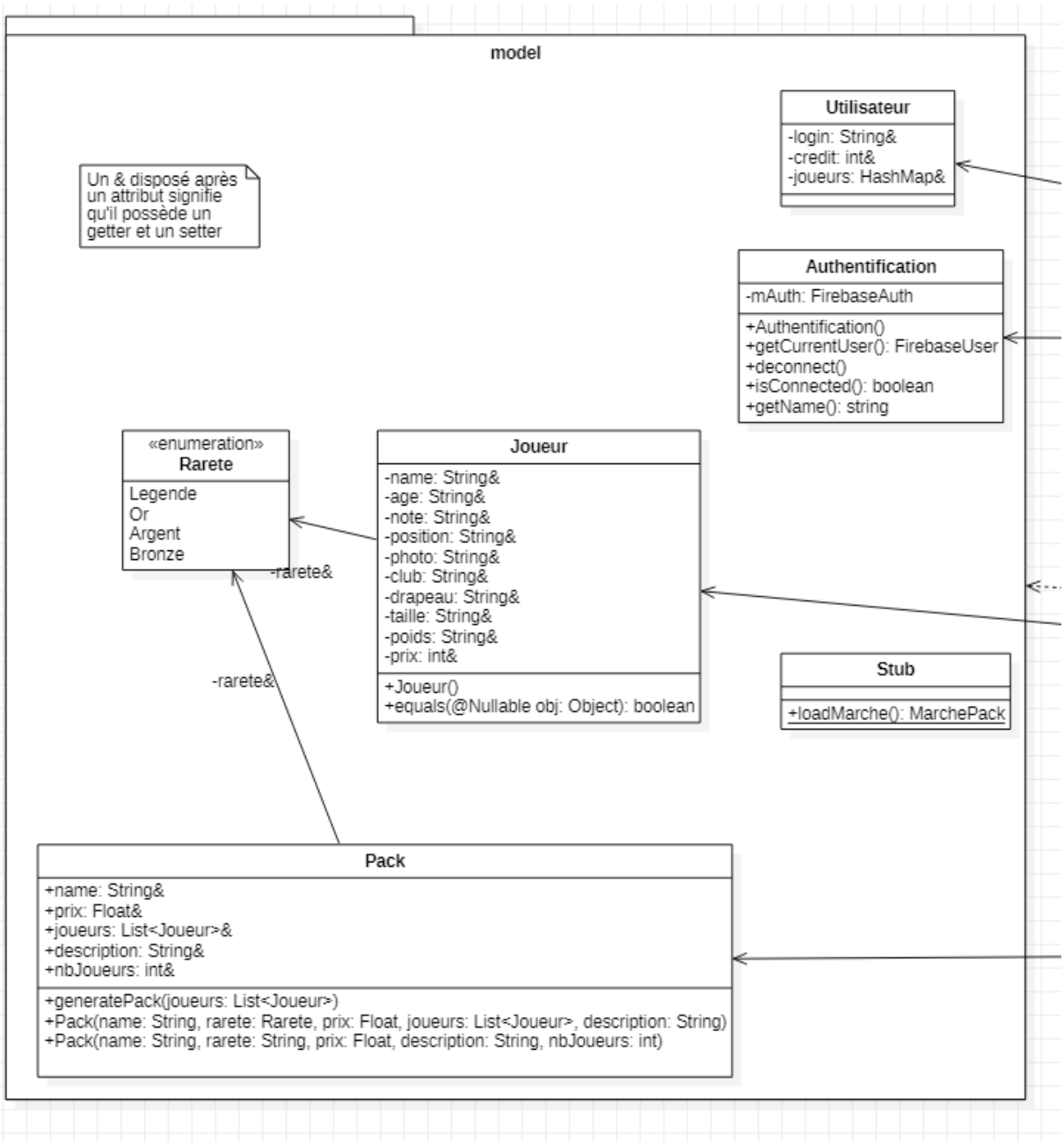
Je sais décrire mon diagramme UML en mettant en valeur et en justifiant les éléments essentiels :

La description du diagramme UML est disponible dans Contexte et description architecture dans le dossier documentation.

Voici un extrait de celui-ci :

Nous allons donc décrire à un à un l'architecture de notre application :

Nous retrouvons ici le métier, celui-ci est responsable de déterminer les classes nécessaires à l'implémentation de l'application. On retrouve donc la définition du Pack, d'un Utilisateur et d'un Joueur. Ces classes sont alors réutiliser partout dans l'application. Nous avons aussi la présence d'un Stub simulant une persistance et nous permettant de tester notre application en générant des données fictives. Pour finir nous avons la classe d'Authentification qui permet à l'utilisateur de se connecter aussi bien avec son compte google qu'avec une connexion anonyme avec son propre mail.



CODE

Je sais utiliser les Intent pour faire communiquer deux activités :

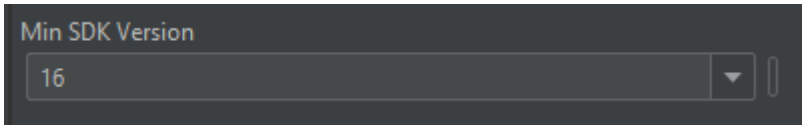
Nous avons eu a faire communiquer plusieurs fois les activités dans notre application en voici un exemple dans le fichier ActiviteInscription ligne 109 :

```
/**
 * le passage a l'activite de choix des modes
 */
private void jouer(){
    startActivity(new Intent(getApplicationContext(), ActiviteMode.class));
    finish();
}
```

Je sais développer en utilisant le SDK le plus bas possible :

Extrait du gradle.build :

```
defaultConfig {
    applicationId "com.example.futmarket"
    minSdkVersion 16
    targetSdkVersion 30
    versionCode 1
    versionName "1.0"
```



Je sais distinguer mes ressources en utilisant les qualifier :

Exemple dans le code :

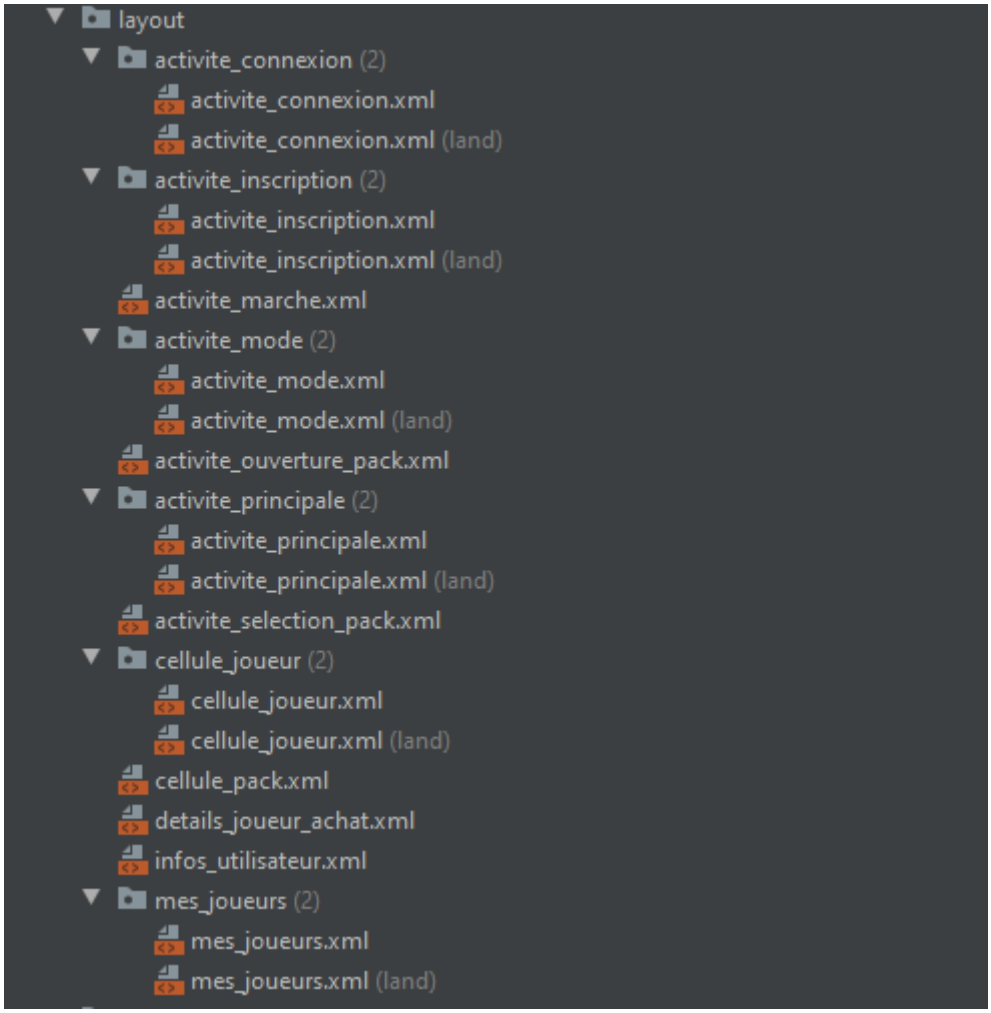
```
if(login.isEmpty() && mdp.isEmpty() && email.isEmpty()){ //on met l'erreur si login ou mot de passe sont vides
    mLogin.setError(getString(R.string.LoginVide));
    mPassword.setError(getString(R.string.MdpVide));
    mEmail.setError(getString(R.string.EmailVide));
    return;
```

Exemple dans la vue :

```
<Button
    android:id="@+id/retour"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Retour" />
<TextView
    android:id="@+id/ValeurPack"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="TextView"
    android:layout_marginTop="9dp"
    style="@style/CustomText"/>
```

Je sais faire des vues xml en utilisant layouts et composants adéquats :

Voici toutes les vues en XML :



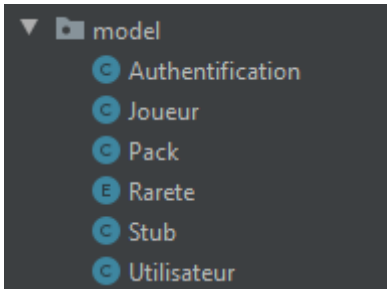
Exemple du détail de l'une d'entre elle :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".view.OuverturePack"
    android:background="@drawable/background_stade">

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/container"
        android:layout_width="match_parent"
        android:layout_height="20dp">


    </androidx.fragment.app.FragmentContainerView>
    <Button
        android:id="@+id/retour"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Retour" />
    <TextView
        android:id="@+id/ValeurPack"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="TextView"
        android:layout_marginTop="9dp"
        style="@style/CustomText"/>

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/listView2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
</LinearLayout>
```

Je sais coder une application en ayant un véritable métier :

Notre architecture est fabriquée ce telle sorte qu'il existe un métier permettant de définir les objets qui seront utilisé dans l'application.

Voici le détail du fichier représentant le métier :



Je sais coder proprement mes activités, en m'assurant qu'elles ne font que relayer les évènements :

On remarque sur ce bout de code que la vue ne fait que relayer les informations qu'elle obtient pour réaliser le traitement logique du côté du métier (fichier ActiviteInscription) :

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState); //la creation de l'activite
    setContentView(R.layout.activite_connexion); //recuperer le design de l'activite
    mEmail = findViewById(R.id.email);// affectation du text
    mPassword = findViewById(R.id.motDePasse); // le mot de passe
    mAuth = FirebaseAuth.getInstance();// recuperation de l'instance d'authentification
    seConnecterEmail(); // la connection via le mail



}

/**
 * le passage a l'activite de choix des modes
 */
private void jouer(){
    startActivity(new Intent(getApplicationContext(), ActiviteMode.class));
    finish();
}

/**
 * pour se connecter avec l'email
 */
private void seConnecterEmail(){
    Button connect = findViewById(R.id.Connecter); // on recupere le bouton pour valider la connexion
    connect.setOnClickListener(v-> { // activation on clique sur le bouton
        String email= mEmail.getText().toString(); //recuperation du texte
        String mdp= mPassword.getText().toString(); //recuperation du texte
        if(email.isEmpty() && mdp.isEmpty()){ // si l'email ou le mot de passe sont vides on afiiche qu'il doivent etre non vides
            mPassword.setError("Le mot de passe ne doit être pas être vide vide");
            mEmail.setError("Le mail ne doit être pas être vide vide");
            return;
        }
        mAuth.signInWithEmailAndPassword(email,mdp).addOnCompleteListener(task -> { //la connexion via l'email
            if(task.isSuccessful()){
               jouer();//on passe au choix des modes
            }
            else {
                Toast.makeText( context: ActiviteConnexion.this, "L'Email ou le Mot de passe ne sont pas corrects",Toast.LENGTH_SHORT).show();
            }
        });
    });
}
}
```

Je maîtrise le cycle de vie de mon application :

Aucune vue ne nécessite vraiment de faire attention au cycle de vie de l'application mais nous avons fais en sorte par exemple de recharger les packs dès lors que le téléphone se charge pour n'avoir aucun souci :

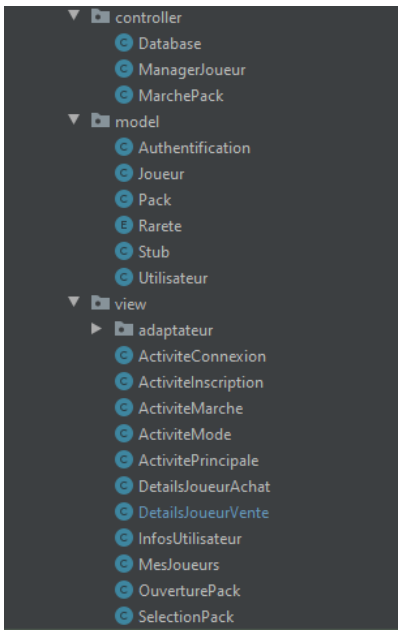
```
@Override
protected void onResume() {
    super.onResume();
    getSupportFragmentManager().beginTransaction()
            .add(R.id.container,new InfosUtilisateur(), tag: null).commit();

    lesJoueurs = new LinkedList<>();

    manager= new ManagerJoueur();
    adapter.refreshData(new LinkedList<>());

    progress.setVisibility(View.VISIBLE);

    manager.listener = new MarchePack.OnMarketGeneratedListener() {
        @Override
        public void onPackGenerated() {
            lesJoueurs = manager.getLesJoueurs();
            adapter.refreshData(lesJoueurs);
            progress.setVisibility(View.GONE);
        }
    };
    manager.getJoueurFromDatabase();
}
```

Méthode OnResume de ActiviteMarche qui permet de s'assurer que l'on recharge les packs lors que l'on revient sur l'application ou que l'on tourne l'écran.

Je sais parfaitement séparer vue et modèle :

Comme expliqué dans la description de l'architecture disponible dans le fichier Contexte et description architecture dans le dossier documentation. Nous avons mis en place un MVC ou modèle/vue/contrôleur qui nous permet de totalement extraire la logique de la vue pour la mettre dans le modèle.

Vue d'ensemble de l'architecture :



Je sais utiliser le findViewById à bon escient :

```
signIn= findViewById(R.id.signGoogle);
mLogin = findViewById(R.id.login);
mPassword = findViewById(R.id.motDePasse);
mConnect = findViewById(R.id.connect);
mEmail = findViewById(R.id.email);
```

Je sais gérer les permissions dynamiques de mon application :

Notre application ne nécessite pas de permission dynamique en effet elle utilise seulement internet afin de faire la liaison avec la base de données.

Je sais gérer la persistance légère de mon application :

La persistance légère est assurée dans la page de connexion de l'application mais avons utiliser une sauvegarde en fichiers afin de faire passer des données d'une activité à une autre :

Voici l'exemple du passage du contenu du pack à SelectionPack et OuverturePack :

```
File file = new File( pathname: getFilesDir()+"/OuverturePack");
try {
    file.createNewFile();
} catch (IOException e) {
    e.printStackTrace();
}
try(ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream( name: getFilesDir()+"/OuverturePack"))) {
    oos.writeObject(marchePack.getLesPacks().get(position));
} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
};
```

```
try{
    ObjectInputStream objectIn = new ObjectInputStream(new FileInputStream( name: getFilesDir()+"/OuverturePack"));
    pack = (Pack) objectIn.readObject();
} catch (FileNotFoundException fileNotFoundException) {
    fileNotFoundException.printStackTrace();
} catch (IOException ioException) {
    ioException.printStackTrace();
} catch (ClassNotFoundException classNotFoundException) {
    classNotFoundException.printStackTrace();
}
```

Je sais gérer la persistance profonde de mon application :

La persistance profonde de l'application est gérée grâce à l'utilisation de Firebase :

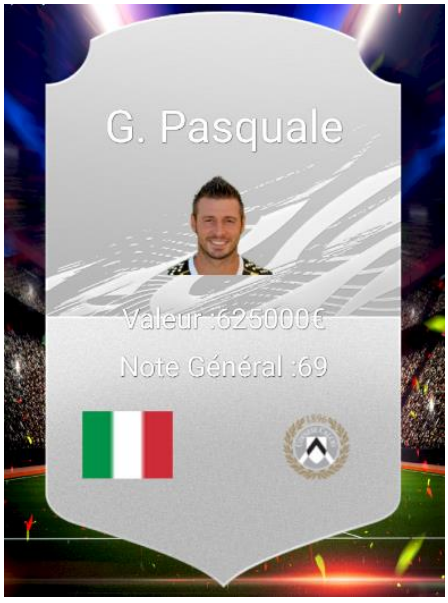
Exemple d'ajout d'un utilisateur dans la base de données :

```
public void AjoutUser(String login){
    DatabaseReference userId=database.getReference( path: "Users").child(auth.getCurrentUser().getUid());
    userId.child("credit").setValue(1000000);
    userId.child("login").setValue(login);
}
```

Je sais afficher une collection de données :

Chargement et affichage d'une collection :

```
@Override
protected void onResume() {
    super.onResume();
    getSupportFragmentManager().beginTransaction()
            .add(R.id.container,new InfosUtilisateur(), tag: null).commit();

    lesJoueurs = new LinkedList<>();

    manager= new ManagerJoueur();
    adapter.refreshData(new LinkedList<>());

    progress.setVisibility(View.VISIBLE);

    manager.listener = new MarchePack.OnMarketGeneratedListener() {
        @Override
        public void onPackGenerated() {
            lesJoueurs = manager.getLesJoueurs();
            adapter.refreshData(lesJoueurs);
            progress.setVisibility(View.GONE);
        }
    };
    manager.getJoueurFromDatabase();
}
```



Affichage d'un joueur de la collection stocké en base de données.

Code retrouvable dans l'activité ActiviteMarche.

Je sais coder mon propre adaptateur :

```java
/**
 * Classe permettant l'affichage des packs
 */
public class AdaptateurMarche extends RecyclerView.Adapter {
    private List<Pack> lesPacks;
    private OnPackListener onPackListener;

    public AdaptateurMarche(List<Pack> lesPacks,OnPackListener onPackListener) {
        this.lesPacks=lesPacks;
        this.onPackListener=onPackListener;
    }

    @NonNull
    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        LinearLayout leLayout =(LinearLayout)LayoutInflater.from(parent.getContext()).inflate(R.layout.cellule_pack,parent, attachToRoot: false);
        return new ViewHolderPack(leLayout,onPackListener);
    }

    @Override
    public void onBindViewHolder(@NonNull RecyclerView.ViewHolder holder, int position) {
        ((ViewHolderPack)holder).getNomPack().setText(lesPacks.get(position).getName());
        ((ViewHolderPack)holder).getPrix().setText(lesPacks.get(position).getPrix());
        ((ViewHolderPack)holder).getDescription().setText(lesPacks.get(position).getDescription());
    }

    /**
     * Permet de notifier la liste d'un changement de données
     * @param newPacks nouvelle liste
     */
    public void refreshData(List<Pack> newPacks){
        lesPacks = newPacks;
        notifyDataSetChanged();
    }

    @Override
    public int getItemCount() { return lesPacks.size(); }
}
```

Je maîtrise l'usage des fragments :

Voici l'exemple de l'utilisation d'un fragment :

```java
@Override
protected void onResume() {
    super.onResume();
    getSupportFragmentManager().beginTransaction()
            .replace(R.id.container,new InfosUtilisateur(), tag: null).commit();


}
```

Je maîtrise l'utilisation de Git :

Nous avons décider de diviser notre git en branches personnelles afin de travailler tranquillement :

root @ master

URL : https://forge.clermont-universite.fr/git/android_projet2021_yoannperiquoi_maximpozdnyakov | Statistiques | Branche: master | Révision :

devYo
main
master

Nom	Taille
Code	
Documentation	
app	
README.txt	2,288 ko
local.properties	330 octet

Dernières révisions

#	Date	Auteur	Commentaire
2c761135	23/03/2021 15:30	Maxim POZDNYAKOV	diagramme des classes
20284e43	22/03/2021 17:56	Maxim POZDNYAKOV	les commentaires des fonctions
e43e8ed3	22/03/2021 03:06	Maxim POZDNYAKOV	Merge remote-tracking branch 'origin/devYo' 1. Conflicts: 2. Code/app/src/main/java/com/example/futmarket/view/ActiviteMarche.java 3. Code/app/src/main/java/com/example/futmarket/view/ActiviteMode.java 4. Code/app/src/main/java/com/example/futmarket/view/MesJoueurs.java...
14027865	19/03/2021 23:16	Yoann PERIQUOI	Ajout d'une notice d'utilisation dans le README
7120261a	19/03/2021 23:01	Yoann PERIQUOI	Réorganisation du projet avec un dossier Code
a1fb691d	19/03/2021 00:14	Yoann PERIQUOI	Finalisation code, ajout de commentaire, réalisation documentation
d3aa0152	18/03/2021 12:53	Yoann PERIQUOI	Récupération de code
c3b263ed	18/03/2021 00:03	Yoann PERIQUOI	Génération de la java doc, ajout de la méthode d'achat
97537d81	17/03/2021 00:36	Yoann PERIQUOI	Essaie de mise en place du master/detail pour l'achat/vente des joueurs. Mise en place d'un style global pour éviter la redondance dans les fichier xml.
42a7a182	16/03/2021 22:10	Maxim POZDNYAKOV	petits modifs, pas grand chose

Voir les différences

Voir toutes les révisions | Voir les révisions

APPLICATION

Je sais développer une application sans utiliser de librairies externes :

Aucune libraire externe n'a été utiliser pour développer l'application.

Je sais utiliser Firebase :

Exemple d'application du gradle et d'utilisation de la base de données Firebase :

```
implementation platform('com.google.firebase:firebase-bom:26.6.0')
implementation 'com.google.firebase:firebase-analytics'
implementation 'com.google.firebase:firebase-auth'
implementation 'com.google.firebase:firebase-database'
```

```
public void acheterJoueur(Object obj,View view){
    ajouterJoueur(obj);
    listener = new OnUserLoaded() {
        @Override
        public void Userloaded() {
            if (getUser().getCredit() - ((Joueur)obj).getPrix() < 0){
                Toast.makeText(view.getContext(), "Vous n'avez pas les crédits pour ce joueur :"+ Integer.toString(((Joueur)obj).getPrix()) +"Crédits" ,Toast.LENGTH_SHORT).show();
                return;
            }
            int credits = getUser().getCredit() - ((Joueur)obj).getPrix() ;
            Utilisateur user = getUser();
            user.setCredit(credits);
            Map<String,Object> map = new HashMap<>();
            map.put(getUserId(),user);
            database.getReference( path: "Users").updateChildren(map);
            Toast.makeText(view.getContext(), "Joueur acheté pour :"+ Integer.toString(((Joueur)obj).getPrix()) +"Crédits" ,Toast.LENGTH_SHORT).show();
        }
    };
    fetchUser();
}
```