Rapport du TP3

Introduction

Le projet a été fait sous Android Studio en utilisant mon Honor 9 comme appareil Android pour exécuter les projets.

Description du tp

Ce tp est composé de 4 exercices. Les 3 premiers se font dans la même application sous Android Studio avec, si besoin, l'ajout de fragment supplémentaire, et d'un dernier exercice servant à créer un service. Notre application est donc, au final, composée de 2 fragments ("UserData" qui est le fragment de saisie des informations de l'utilisateur, et "ResultData" qui est le fragment d'affichage de ces informations) et d'une classe ("DataForm) pour faciliter la gestion des données rentrées par l'utilisateur. De plus, comme nous avions le choix, le service a été créé avec node.

Application avec les fragments

Il s'agit donc des 3 premiers exercices. Ces exercices consiste à créer une application dans laquelle un utilisateur lambda pourra saisir des informations comme son nom, son prénom, sa date de naissance, etc..., et une fois qu'il aura validé la saisie, une autre page l'affichera pour qu'il puisse vérifier. Cette application devra utiliser les fragments et non pas les activités. Autrement dit, une seule activité sera créée et le reste sera gérer à l'aide des fragments.

Donc, dans notre application, une fois que l'utilisateur a saisie les informations et clique sur le button pour valider, le premier fragment passe en quelque sorte la main au deuxième fragment pour qu'il puisse les afficher. Ceci se fait grâce à la classe FragmentTransaction :

```
1 assert getFragmentManager() != null;
2 FragmentTransaction fragmentTransaction = getFragmentManager().
    beginTransaction();
3 ResultData fragment  = new ResultData();
4 fragment.setArguments(bundle);
5 fragmentTransaction.replace(R.id.fragmentContainerView , fragment);
6 fragmentTransaction.commit();
```

Nous pouvons voir ci-dessus que l'on crée un objet de type FragmentTransaction qui va remplacer le fragment visible actuel par le nouveau fragment grâce à la méthode replace(). En fait, dans l'activité, il y a une vue qui sert de container au fragment afficher. Et le principe de replace(), et simplement qu'elle remplace le fragment dans ce container par celui que l'on souhiate afficher à la place.

setArguments() permet d'envoyer, par exemple un bundle, à un autre fragment. C'est une manière d'envoyer des informations entre fragment. A la fin, commit() permet d'éxécuter en quelque sorte les actions au dessus, donc setArguments() et replace().

Par la suite, au lieu de juste envoyer les données avec, par exemple, une classe "Parcelable", nous devons les mettre dans un fichier Json et ensuite l'envoyer. Pour faire cela, nous utilisons ceci :

```
submit.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Bundle bundle = new Bundle();
        DataForm f = new DataForm(name.getText().toString(),
                    surname.getText().toString(),
                    date.getText().toString(),
                    email.getText().toString(),
                    sport.isChecked(),
                    music.isChecked(),
                    reading.isChecked(),
                    videogames.isChecked());

        bundle.putString("Form", new Gson().toJson(f));

        assert getFragmentManager() != null;
        FragmentTransaction fragmentTransaction = getFragmentManager().
    beginTransaction();
        ResultData fragment  = new ResultData();
        fragment.setArguments(bundle);
        fragmentTransaction.replace(R.id.fragmentContainerView , fragment);
        fragmentTransaction.commit();
    }
});
```

On peut remarquer la ligne 14. Dans celle ci, nous ajoutons dans un bundle sous le nom ”Form”, un fichier Json créé avec new Gson().toJson() à partir de ”f” qui est de type ”DataForm”. Autrement dit, on prend une instance de type ”DataForm” qui contient les informations saisies par l'utilisateur, et on en crée un fichier Json.

Ensuite, nous avons implémenter un bouton retour dans le fragment d'affichage qui permet de revenir au fragment de saisie, pour par exemple modifier la date de naissance. Sauf que pour pouvoir faire cela, il faut en quelque sorte sauvegarder les données déjà saisies ou les recevoir du second fragment. Donc, nous avons fait en sorte que le fragment lise sur un fichier à chaque fois. Si ce fichier est vide, il regarde s'il a des arguments, c'est à dire si un fragment lui a envoyé des données. Si oui, il complète les champs avec ces données, sinon, il les laisse vide. Si le fichier n'est pas vide, il remplit les champs à partir des données du fichier.

```
File file = new File(getActivity().getFilesDir() + "/" + MainActivity.
    fileName);
if(file.exists() && file.length() > 0){
    try {
        InputStream input= getActivity().openFileInput(MainActivity.
    fileName);
        byte[] buffer = new byte[input.available()];
        input.read(buffer);
        input.close();
        String text = new String(buffer);
        DataForm f = new Gson().fromJson(text, DataForm.class);

        name.setText(f.getName());
        surname.setText(f.getSurname());
        date.setText(f.getDate());
        email.setText(f.getEmail());
        sport.setChecked(f.isSport());
        music.setChecked(f.isMusic());
        reading.setChecked(f.isReading());
        videogames.setChecked(f.isVideogames());

        getActivity().openFileOutput(MainActivity.fileName,0).close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}else {
    if (this.getArguments() != null) {
        String form = this.getArguments().getString("Form");
        DataForm f = new Gson().fromJson(form, DataForm.class);

        name.setText(f.getName());
        surname.setText(f.getSurname());
        date.setText(f.getDate());
        email.setText(f.getEmail());
        sport.setChecked(f.isSport());
        music.setChecked(f.isMusic());
        reading.setChecked(f.isReading());
        videogames.setChecked(f.isVideogames());
    }
}
```

Service

Le service a été réalisé en node pour des raisons de simplicités car nous avions déjà un server en node pour un autre projet et c'était donc plus simple de le réutiliser et le réadapter plutôt que d'en créer un autre.

Pour commencer, il faut créer la base de données avec MongoDB qui va contenir le fichier form.json. Ensuite, il suffit de lancer le server node (serveur.js), et de s'y connecter avec la bonne url.

```
MongoClient.connect(url, {useNewUrlParser: true}, (err, client) => {
    let db = client.db("Android");

    console.log("Webservice REST for Android app");
    console.log("Tabs: form");

    app.get("/form/_id/:valeur", (req,res) => {
            let val = req.params.valeur;
            console.log("/form" + "/_id" + "/" + val);
            try {
                let s = {"_id": ObjectID(val)} ;
                returnvalue = [] ;

                db.collection("form").find(s).toArray((err, documents) => {
                    res.end(JSON.stringify(documents[0]));
        });
            } catch(e) {
                console.log("Erreur sur /form" + "/_id" +
                            "/" + val +" : "+ e);
                res.end(JSON.stringify([]));
            }
        });
});
```