Example

Projet PA

Physio Data Visualization

Auteurs :
Gander Laurent

Responsables :
Rizzotti Aïcha

25 janvier 2018

This document is the base config file you have to update

Table des matières

Table des figures

1 Introduction

Ce projet a été réalisé de le cadre du cours de projet P3 et du cours d'android du cursus informatique de troisième année. L'objectif était de concevoir un programme fonctionnel et réalisable en un semestre. Le but principal est de présenter la méthode de réalisation ainsi que le logiciel terminé. Le langage utilisé pour ce projet est le Java sur Android. Le but de mon application est de trouver des solutions de visualisation sur mobile pour traduire à l'utilisateur des émotions comme le taux de stress par le biais d'un bracelet avec des capteurs physiologique telque l'EDA l'electrodermal activity, la variabilité cardiaque... etc
Le travail consiste à faire la conception et l'implémentation de la chaine, des capteurs, le serveur et l'application sur mobile. La réalisation a commencé par une phase d'analyse. Certaines informations qu'il a fallu compléter par des recherches personnelles ont été transmises. Vint ensuite une phase de préconception durant laquelle il a fallu établir un graphique UML définissant l'implémentation ainsi qu'un use case et un diagramme de séquence. Dans la phase suivante, le projet a été codé.
Le rapport se construit de la manière suivante. L'explication des différents termes et fondements du projet. Il y est ensuite expliqué le résultat obtenu, les difficultés encourues et finalement une comparaison avec les spécifications détaillées de départ.

2 Déroulement

Durant la phase de conception, un cahier des charges, des spécifications détaillées, un diagramme UML, un diagramme de cas d'utilisation et un diagramme de séquence ont été réalisés. Une phase de réalisation a suivi cette phase de conception. Durant celle-ci, l'application a été implémentée de façon à respecter au mieux les objectifs décidés durant la première phase, mais certains ajustements ont été nécessaires. La phase finale est la phase de test durant laquelle le bon fonctionnement du programme a été examiné et les éventuels problèmes corrigés.

2.1 Phase de conception

Pour la phase de conception, les points importants des spécifications détaillées ont été réfléchis. J'ai commencé par faire une recherche et comparer des applications qui permettait de faire ce qui m'était demandé afin de trouver ce qui serait bien de faire et ce qu'il ne ferait pas faire.
Après avoir fais ce travail de recherche et en ayant discuté avec Madame Rizzotti, nous avons décidés de réaliser une application qui - permette de recevoir des données provenant d'un bracelet de type Empatica. - permette d'introduire des données utilisateurs - permette de calculer les battements de coeur avec la caméra. - permette d'enregister les données utilisateurs dans une base de données - permette d'enregistrer les logs dans une base de données - sois userfriendly
Pour parfaire ces objectifs, et comme décrit dans le diagramme UML, nous avons choisi de créer la hiérarchie suivante :

- Activity
  - MeasuresDetailActivity
- Fragments
  - AnalyseFragment
  - HomeFragment
  - MeasuresDetailFragment
  - MeasuresFragment
  - PhysioDataVisualisationFragment
  - VisualisationFragment
- HeartBeat
  - HeartBeatView
  - ImageProcessing
  - PhysioDataActivity

- Logs
  - CallReceiver
  - DatabaseLog
  - Log
  - LogRepository
  - LogSchema
  - PhoneCallReceiver
  - PhoneSMSReceiver
  - SMSReceiver
- Measures
  - DatabaseMeasures
  - MeasuresRepository
  - Measures
  - MeasuresAdapter
  - MeasureSchema
  - MeasuresConnection
  - MeasureService
- MainActivity

2.2 Phase de réalisation

Durant la phase de réalisation, nos objectifs ont été suivis et atteints. Cependant, l'objectif, qui consistait à recevoir les données du bracelet, n'a pas été réalisé. M. Noguera n'avait pas de données à envoyer. Concernant le déroulement de cette phase, j'ai commencé par m'occuper du design. Puis j'ai codé les différentes classes et intégré les fonctionnalités correspondantes.

2.2.1 MainActivity

Mon activity principale permet contient un fragment qui sera changé suivant les actions de l'utilisateur. Le fragment qui est chargé de base est le HomeFragment. Un utilisateur peut changer de fragment en ouvrant le menu qui se trouve en haut à gauche, ce menu glisse pour prendre part en partie sur l'écran. L'utilisateur à la choix entre plusieurs options : - Home - Analyse - Measure - Visualisation
Chaque fragment est expliqué plus loin dans le rapport.
Pour pouvoir changer de fragment, j'ai overrider une fonction de la classe suivante

```
NavigationView.OnNavigationItemSelectedListener
```

La fonction est la suivante

```
public boolean onNavigationItemSelected(MenuItem item) {
    int id = item.getItemId();


    if (id == R.id.nav_home){
        HomeFragment homeFragment = new HomeFragment();
        getSupportFragmentManager().
        beginTransaction().
        replace(R.id.main_fragment ,homeFragment).commit();
        homeFragment.onMeasureServiceConnected();
    }
    if (id == R.id.nav_visualisation) {
        VisualisationFragment visualisationFragment =
        new VisualisationFragment();

        getSupportFragmentManager().
        beginTransaction().
        replace(R.id.main_fragment ,visualisationFragment).commit();
```

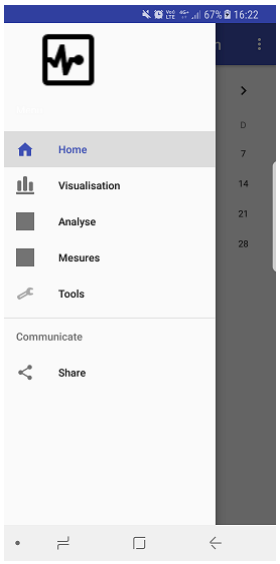

FIG. 1 : Menu gauche

```
            visualisationFragment.onMeasureServiceConnected();

        } else if (id == R.id.nav_analyse) {
            AnalyseFragment analyseFragment = new AnalyseFragment();
            getSupportFragmentManager().
            beginTransaction().
            replace(R.id.main_fragment, analyseFragment).commit();
            analyseFragment.onMeasureServiceConnected();

        } else if (id == R.id.nav_mesures) {
            MeasuresFragment mesuresFragment = new MeasuresFragment();
            getSupportFragmentManager().
            beginTransaction().
            replace(R.id.main_fragment ,mesuresFragment).commit();
            mesuresFragment.onMeasureServiceConnected();
        } else if (id == R.id.nav_share) {

        }

        DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
        drawer.closeDrawer(GravityCompat.START);
        return true;
    }
```

Mon activity commence déjà par vérifier si j'ai l'autorisation d'utiliser les capteurs qui seront utiles à mon application - Camera - Appel entrant, sortant et manqué (READ_PHONE_STATE) - SMS reçu (SMS_RECEIVE)
Ces vérifications se font de la manière suivante

```
if (ContextCompat.checkSelfPermission(this, Manifest.permission.CAMERA)
                    == PackageManager.PERMISSION_DENIED)
        {
            ActivityCompat.requestPermissions( this, new String[]
            {Manifest.permission.CAMERA}, 1);
        }

if (ContextCompat.checkSelfPermission(this,Manifest.permission.READ_PHONE_STATE)
== PackageManager.PERMISSION_DENIED)
        {
            ActivityCompat.requestPermissions( this, new String[]
            {Manifest.permission.READ_PHONE_STATE}, 1);
        }

if (ContextCompat.checkSelfPermission(this, Manifest.permission.RECEIVE_SMS)
== PackageManager.PERMISSION_DENIED)
      {
```

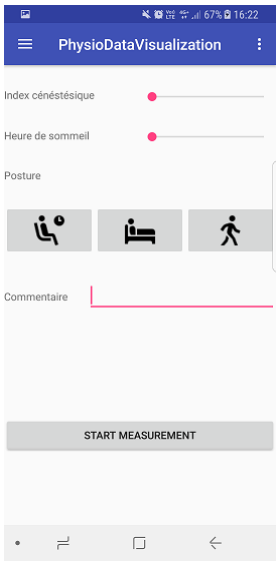

FIG. 2 : Mesures

```
            ActivityCompat.requestPermissions( this, new String[]
19          {Manifest.permission.RECEIVE_SMS}, 1);
        }
```

Il faut aussi ajouter ces autorisation dans le manifest.xml
Que contient un manifest ?
Le fichier manifest permet de décrire votre application. On y retrouve :

- le nom du package de l'application. Il servira d'identifiant unique
- tous les composants de l'application (Activity, Services, BroadCast Receivers, Content providers).
- on détermine dans quels processus les composants de l'application seront contenus
- les permissions nécessaires pour le bon fonctionnement de l'application
- les informations contenant les versions de l'Android API requis pour exécuter votre application
- les bibliothèques utilisées par votre application.

Ensuite je crée un service qui me permettra d'interagir avec ma base de données DatabaseMeasures MeasureService recupère une Instance de ma base de données.Je crée un objet MeasuresRepository. Je passe une Instance de ma base à MeasuresRepository. Comme je l'expliquerai plus loin dans le rapport, MeasuresRepository me permet d'interagir avec ma base de données.

2.2.2 HomeFragment

Ce fragment apparaît à l'ouverture de l'application. Il contient un calendrier.

2.2.3 MeasuresFragment

Ce fragment permet l'ajout de données utilisateurs et de faire une nouvelle mesure. On peut y entrer plusieurs paramètres : - L'index cénesthésique, c'est à dire la façon dont on se sent - les heures de sommeil - la position dans laquelle on fait la mesure - un commentaire
Dès que toute les données sont rentrées, on peut appuyer sur le bouton 'start measurement', en cliquant dessus, on ouvre PhysioDataActivity.

```
final Intent intent = new Intent( getActivity(), PhysioDataActivity.class);
startActivityForResult( intent, 1 );
```

on recupère le resultat avec la fonction

FIG. 3 : Heart

```
public void onActivityResult(int requestCode, int resulCode, Intent data)
{
    super.onActivityResult( requestCode, resulCode,data );
    Date date = new Date();
    if(1 == requestCode)
    {
        avgHeartBeat = data.getIntExtra( "Heart_Beat", 0 );
        int stressIndex = calculStressIndex(index, sommeil, avgHeartBeat);
        ((MainActivity)getActivity()).getService().createMeasure
        (index, sommeil, position, comment, avgHeartBeat, stressIndex, date);
    }
}
```

2.2.4 PhysioDataActivity

Cette activité nous permet de mesurer notre rythme cardiaque en utilisant la caméra. L'utilisateur va posé son doigt sur la caméra, celle ci va capturer, en boucle, des images de notre doigt et ainsi comparer le taux de rouge dans l'image. A chaque battement de coeur, il y aura un afflux de sang de sang dans notre doigt, il deviendra ainsi plus rouge. En incrementant une variable à chaque changement de couleur, on trouve notre rythme cardiaque.
on renvoie le résultat à notre fragment avec

```
public void onBackPressed()
{
    int avg = 0;

    for(int i = 0; i< listMeasures.size(); i++)
    {
        avg += listMeasures.get( i );
    }

    Intent intent = new Intent( );

    intent.putExtra( "Heart_Beat", avg/listMeasures.size() );
    setResult(0, intent);
    finish();
}
```

2.2.5 HeartBeatView

Permet d'afficher la view qu'on a l'écran. - canvas - image rouge si il y a un battement de coeur - image verte si il n'y a pas de de battement de coeur

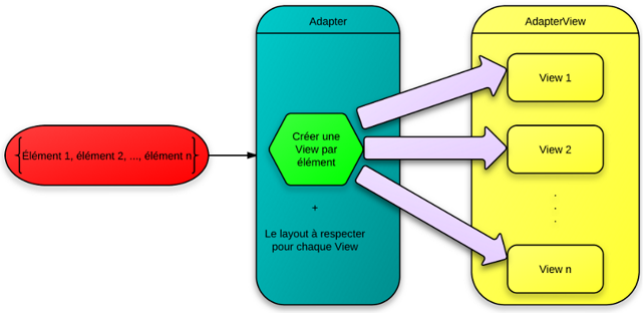

FIG. 4 : Adapter

FIG. 5 : Analyse

2.2.6 ImageProcessing

C'est dans cette classe que tout le calcul sur la couleur des pixels ce fait.

2.2.7 AnalyseFragment

Ce fragment contient - une SearchView qui permet à l'utilisateur de rechercher une mesures par rapport à une date. On entre une date dans la barre de recherche et on va chercher le filter dans la classe MeasuresAdapter et qui nous retourne les valeurs correspondates à la date. - une ListView de mesures triées par date.
Quand on clique sur un élément de la ListView, on ouvre une nouvelle Activity (MeasuresDetailActivity) grâce à un AdapterView. La ListView a un adapter qui lui est setter. Un Adapteur est un objet qui gère les données, c'est comme un intermédiaire entre les données et et la vue qui les représente.

2.2.8 MeasuresAdapter

Comme expliqué précédemment, un adapter permet de gérer les données. La classe MeasuresAdapter permet de récolter les mesures effectués et de les retourner au AnalyseFragment pour qu'elles soient affichées. Au préalable, on aura instancier un objet Filter qui nous permet de filtrer les mesures par dates et de retourner la liste filtrée.
Dans ce fichier nous avons une autre classe, MeasuresHolder, qui nous permet de choisir ce qui sera affiché dans la liste. Personnellement, j'ai choisi d'afficher la position dans laquelle la mesure a été faites, la date ainsi que la l'indice de stress.

2.2.9 VisualisationFragment

Ce fragment permet à l'utilisateur de visualiser son stress. Il lui est possible de choisir la date et il verra sur un graphe le taux de stress qu'il a eu à un moment de la journée.

Pour réaliser ce fragment, j'ai récupéré les mesures contenu dans ma base de données grâce à mon

3 Résultat obtenus

A l'ouverture de l'application, on a un fragment qui s'ouvre avec un calendrier. On peut naviguer entre les fragments en utilisant le menu qui se trouve en haut à gauche. On peut ajouter une nouvelle mesures en cliquant sur 'Measures', l'ajout d'une mesures ce passe de la manière suivante : - Entrer l'index cénéstésique - Entrer les heures de sommeil de la nuit précédente - Entrer la position dans laquelle on est pendant la mesures - Assis - Couché - Debout - Entrer un commentaire - Cliquer sur le bouton 'start measurement'
Une nouvelle activity s'ouvre et on pose notre doigt sur la camera et on attend sans bouger que l'application calcul notre rythme cardiaque. Quand ceci est fais, on peut cliquer sur le bouton retour de notre application et la mesure est enregistré dans la base de donnnées.
On peut aller visualiser les mesures prises en allant sur 'Visualisation' dans notre menu. On peut choisir la date dans une liste déroulante qui se trouve en haut à gauche. Un graphe représente le niveau de stress à l'heure où la mesure a été faites.
Pour voir toute les mesures, on peut aller sur 'Analyse' qui se trouve dans le menu gauche. Le fragment contient la liste de toute les mesures et qui affiche - La position - La date - L'indice de stress
En cliquant sur une mesure, on ouvrira une nouvelle activity qui affichera la mesure plus en détail.

4 Tests

La phase de tests s'est déroulée depuis la création de la première version fonctionnelle jusqu'à la rédaction du rapport, soit un suivi constant de l'état de fonctionnement du programme. Le code n'a pas été testé via des tests unitaires ou d'implémentation, mais un test de fonctionnalité a été réalisé. Mon application a été utilisé par de tierce personnes qui m'ont données certains retour - Design pas excellent

5 Amélioration possibles

Les différentes étapes de conception ont été réalisées correctement. La problématique a rapidement été cernée et les spécifications détaillées réalisées. Il a été possible de définir ces dernières après avoir déterminé les points chauds. La phase de réalisation a suivi les spécifications détaillées mais des changements ont dû être appliqués.
La reception du résultat de Monsieur Noguera n'a pas pu être réalisé car il n'y avait pas de résultat, donc la comparaison entre les deux résultats n'a pas pu être faites.

6 Conclusion

7 Annexes