# Projet Android

Interface Homme-Machine : Android

8 février 2022

L'objectif principal est de réaliser une application produisant un bilan de santé à partir d'un questionnaire personnalisé. L'application consistera en une série d'activités (fenêtres) qui enregistreront des données au fur et à mesure, et produiront un résultat à partir de ces données.

Le site qui servira d'exemple se trouve à l'adresse Fédération française de cardiologie. Plus précisément, vous reprendrez le questionnaire de la page Comment va votre cœur ?. Des images d'exemples sont fournies dans ce document : vous pouvez vous en inspirer, mais faites en sorte de personnaliser votre application.

L'ensemble du projet se découpe en plusieurs parties, décrites dans le reste de ce document.

## 1 Préambule

Rendez-vous sur le site Comment va votre cœur ? et réalisez le test pour comprendre le déroulement de l'application.

On suppose que vous avez installé correctement `Android Studio` et que vous êtes capable d'exécuter le simulateur intégré ou de connecter le logiciel à votre smartphone. Créez un nouveau `Projet` avec une `Empty Activity` par défaut. Vous pouvez choisir un nom quelconque pour le projet.

**Attention** `Android Studio` est un animal capricieux, qui se comporte parfois de manière inattendue. Il est recommandé de sauver régulièrement l'ensemble de votre projet après chaque étape importante. Parfois, fermer puis ré-ouvrir un projet peut régler des problèmes d'affichage et de rendu.
Si l'erreur persiste, vérifier que le fichier `.xml` existe bien et éditer le code existant. Souvent, commenter des parties du code de ce fichier fait réapparaître les `widgets`... [1]

## 2 Widgets, agencement et traitement d'erreur

### 2.1 Widgets et contraintes

Revenez à la page de début du test. Dans la zone illustrant le contenu du fichier `.xml`, retirez la `widget` par défaut contenant le texte *"Hello World !"* et ajoutez au moins deux `widgets` :

— Une `ImageView` reprenant le logo du test *"Comment va votre cœur ?"*. Ce type e `widget` déclenche un utilitaire permettant de rechercher une image (vous pouvez choisir ce que vous souhaitez) et de la placer dans votre projet.

— Un champ texte pour écrire votre nom ou votre pseudo.

— Un `Button` contenant le texte *"Faites le test"*. Pour le moment, aucune action n'est associée à ce bouton.

Des *contraintes* doivent être posées sur ces `widgets`, pour éviter que ces dernières changent de position lorsque le smartphone change d'orientation. Chaque `widget` doit avoir au moins une contrainte verticale et une contrainte horizontale.

La figure1 est un exemple de réalisation de la première activité. Le logo (en français) est tiré du site de référence mais l'application est en anglais.

1. `autofillHints` et `android:tools` en particulier.

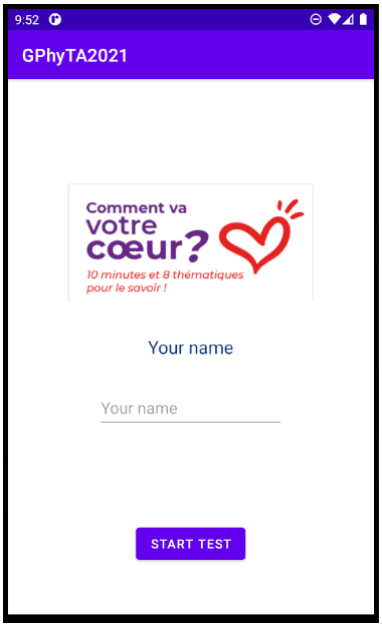


FIGURE 1 – Démarrage de l'application (version anglaise, orientation *Portrait*)

## 2.2 Orientation

Testez la pertinence des contraintes en faisant pivoter l'interface : passage du mode par défaut *Portrait*au mode *Paysage* (*Landscape*). Ce n'est pas grave si les widgets se déplacent entre les orientations mais il faut veiller à ce qu'elles restent toujours sur l'écran et qu'elles ne se chevauchent pas.

La figure 2 affiche la même activité que précédemment, en mode *Paysage*.

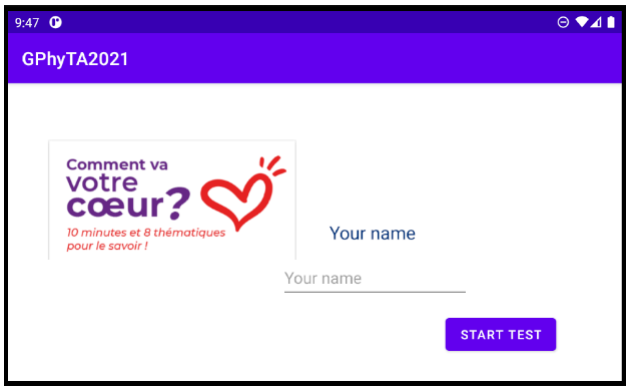


FIGURE 2 – Démarrage de l'application (orientation *Paysage*)

## 2.3 Internationalisation

Par défaut, `Android Studio` utilise la langue anglaise (plus précisément, le codage américain "*en-us*" de cette langue).

On souhaite pouvoir afficher les textes dans chaque langue que supportera l'application. On suppose que vous rajouterez au moins le français (de France) au codage par défaut, libre à vous d'ajouter d'autres langues. Utilisez le `Translation Editor` qui, pour chaque texte associé à une `widget`, détermine le contenu de ce texte dans chaque langue choisie. Par exemple, le texte en français *"Faites le test"* associé au bouton défini précédemment correspondra au texte *"Take the test"* en anglais.

La images de la figure 5 reprennent les activités précédentes, traduites en français.

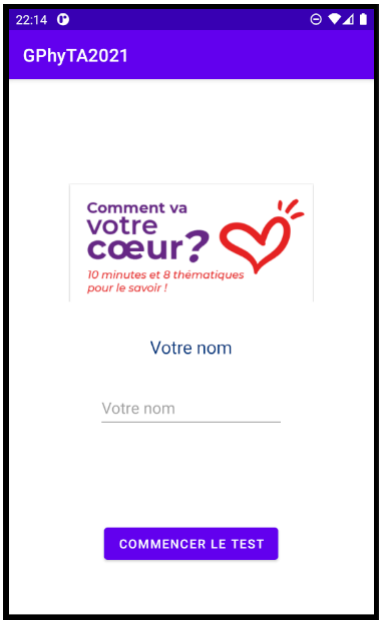


FIGURE 3 – Portrait

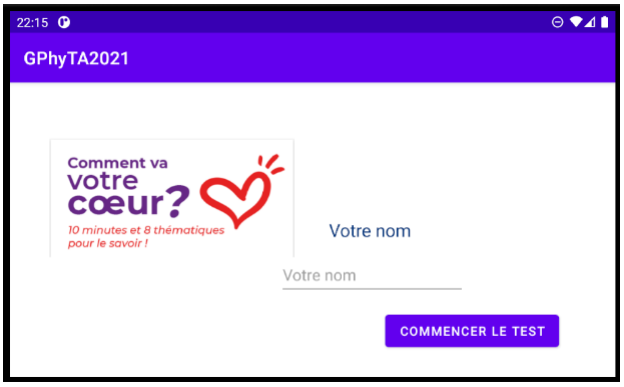


FIGURE 4 – Paysage

FIGURE 5 – Première activité, traduite en français

## 2.4 Correction des erreurs

Au fur et à mesure de la construction de l'activité, `Android Studio` détecte les erreurs à corriger (elles sont affichées dans la console ouverte en cliquant sur *"Problems"* dans la barre de menu en bas de l'interface). Ces dernières sont de gravités diverses : les triangles jaunes correspondent à des conseils de bonne construction, les cercles en rouge à des risques de blocage lors de l'exécution de l'application.

De manière générale, il faut corriger l'ensemble des erreurs, quelle que soit leur gravité. `Android Studio` propose la plupart du temps des conseils de résolution et peut "réparer" l'erreur tout seul ; c'est pratique mais il est conseillé de chercher la solution par vous-même, bien entendu.

## 2.5 Construire d'autres activités

Toujours dans le même projet, ajoutez de nouvelles activités qui s'inspirent du contenu des parties "*Je suis*", "*Mon cœur*", et "*Mon suivi cardiaque*". Vous noterez que les questionnaires ont toujours la même forme : une série de cases à cocher (des `RadioButtons` contenus dans un `RadioGroup`). Outre ce type de `widget`, vous devez utiliser (quitte à ajouter des champs qui n'existent pas dans le site web) la plus large gamme de `widgets` possibles.

**Remarque** Dans le test original du site Web, il est impossible de passer d'une séquence à la suivante tant que le questionnaire en cours n'est pas rempli : lorsque l'on clique sur *"étape suivante"*, le message *"Vous devez répondre à*

*toutes les questions"* est affiché. En ce qui vous concerne :

— vous devez créer les boutons *"Previous Step"* et *"Next Step"* mais ils ne provoquent pas le passage à l'activité suivante pour le moment ;

— si toutes les réponses n'ont pas été fournies, vous devez afficher un texte d'avertissement (par exemple à l'aide d'un `Toast`) ;

# 3 Activités supplémentaires

## 3.1 Activités supplémentaires

Si ce n'est fait, terminez la création des activités "*Je suis*", "*Mon cœur*", "*Mon suivi cardiaque*", etc...

La figure 9 affiche des exemples de telles activités. Plusieurs types de `widgets` ont été utilisés, notamment :

— `Button`

— `ImageView`

— `RadioButton` et `RadioGroup`

— `Text` pour récupérer des chaînes de caractères ou des nombres

— `Spinner` (listes déroulantes) symbolisés par des petits triangles dans la figure 7

— `ScrollView` pour faire défiler des éléments qui ne rentrent pas sur l'écran (également dans la figure 7)

— `Switch` (interrupteurs)

— `ImageButton`

— `CheckBox`

Pour chacune de ces activités, n'oubliez pas de gérer l'orientation, l'internationalisation et la correction des erreurs.

## 3.2 Vérification des données

Complétez le code des fichiers `.java` pour vérifier :

— que les données entrées sont cohérentes (par exemple, que les réponses correspondent aux valeurs et aux types attendus) ;

— si l'utilisateur entre une donnée erronée ou s'il manque des réponses quand on appuie sur le bouton "*Next Step*", afficher un message d'erreur (par exemple en utilisant un `Toast` ou des vibrations) ;

— de manière générale, n'hésitez pas à utiliser les `logs` pour faciliter le débogage de l'application et montrer une maitrise de l'environnement.

**À propos des logs** La console `Logcat` affiche les messages spécifiés lors des appels à la class `Log`. Mais parfois, ces messages ne s'affichent plus et le processus lié à l'application porte la mention `"DEAD"`. Pour y remédier, sélectionner dans le menu principal `File` → `Invalidate Caches / Restart...`.
Alternativement, la console `Run` peut capturer et afficher les `logs`, qui sont un peu noyés dans les messages système, vous avez vu quoi faire durant les séances de TP...

## 3.3 Préparation au stockage des données

Créez une classe `Person` dont les membres correspondront aux données entrées dans les différentes activités (nom, age, sexe, etc.). Pour le moment, ne pas instancier cette classe : se contenter de définir ses attributs et ses accesseurs.

# 4 Une classe pour stocker les données

L'objectif de cette section est de récolter les informations entrées dans chaque activité et de les conserver au cours du cycle de vie de l'application.

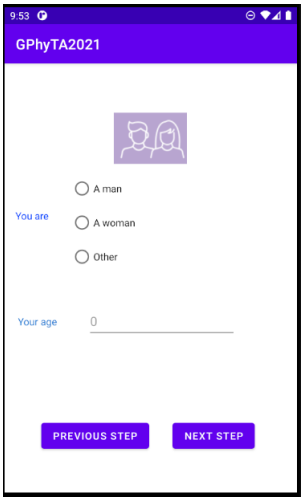


FIGURE 6 – Je suis

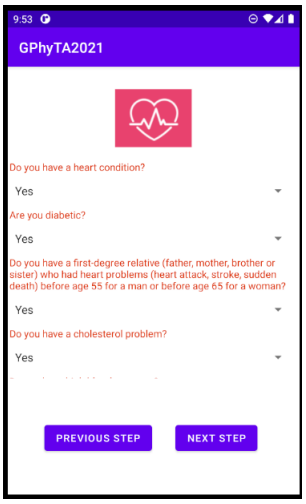


FIGURE 7 – Mon cœur

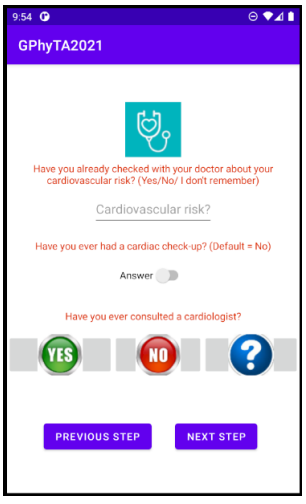


FIGURE 8 – Risque cardiaque

FIGURE 9 – Les autres activités

`Android` propose plusieurs manières de sauvegarder l'information, de manière temporaire ou permanente (avec les `SharedPreferences` par exemple), et de la récupérer en cas de fermeture inattendue de l'application (à l'aide des Bundles utilisés dans les méthodes onCreate(), onSaveInstanceState() et onRestoreInstanceState()).

Un autre moyen de procéder est d'utiliser les Intent permettant de passer d'une activité à l'autre, et de transférer (resp. récupérer) des données *via* les méthodes putExtra() et getExtra() de la classe Intent.
Généralement, ces méthodes sont utilisées pour transférer des données de types simple (String, int, ...). Mais nous souhaitons ici utiliser directement une instance de la classe Person définie dans le TA précédent.

### 4.1 Implémentation de la classe Person

À ce stade, nous supposons que cette classe contient des membres permettant de stocker toutes les valeurs qui seront récoltées dans les activités.

Pour que la classe Person soit "transférable" avec les méthodes de Intent, elle doit implémenter l'interface Parcelable et surcharger en particulier les méthodes describeContents(), writeToParcel(). Des explications et des exemples sont fournis sur cette page.

### 4.2 Transfert de Person

Utilisez les Intent étudiées en TP pour transférer une instance de Person d'une activité vers la suivante.

### 4.3 Contrôle des erreurs

Nous imposons une nouvelle contrainte : désormais, avant le passage d'une activité à une autre (quel que soit le sens de passage), il faut s'assurer que toutes les widgets de l'activité courante ont bien reçu une valeur. Cela revient, au moment de cliquer sur l'un des boutons "*Previous Step*" ou "*Next Step*", à passer en revue toutes les widgets de l'activité en cours et vérifier qu'elles ont reçu une valeur. Utilisez les logs, des Toasts et la méthode vibrate() vue en TP pour signaler les erreurs

Une fois cette vérification effectuée, vous pourrez mettre à jour les membres de l'instance de la classe Person en fonction des valeurs contenues dans les widgets. L'instance de Person pourra alors être transférée.

**Attention** Lors du retour vers une activité précédente, les valeurs des widgets de cette activité devraient être remises à jour en fonction des membres de la classe Person transférée. Dans l'activité suivante (figure 6), le genre et l'âge de l'utilisateur sont demandés. Si, après avoir rempli ces widgets (et donc mis à jour les attributs correspondants de Person), on presse le bouton "*Previous Step*", on revient dans la première activité. Dans celle-ci, il faut mettre à jour la valeur de la widget contenant le nom de l'utilisateur, en utilisant l'attribut name de l'instance de Person transférée.

## 5 Accès à un site externe et bilan

### 5.1 Accès au site de la fédération de cardiologie

Dans la dernière activité que vous avez conçue, ajoutez une intention permettant d'ouvrir la page Web du site Fédération française de cardiologie.

### 5.2 Bilan du questionnaire

#### 5.2.1 Bilan officiel

Après avoir parcouru réalisé le test sur le site de la Fédération, attardez-vous sur la dernière page du questionnaire (figure 10) :

Chaque étape est sélectionnable et affiche la liste des réponses au questionnaire associé à l'étape, et des conseils de médecin plus bas dans la page (figure 11) :

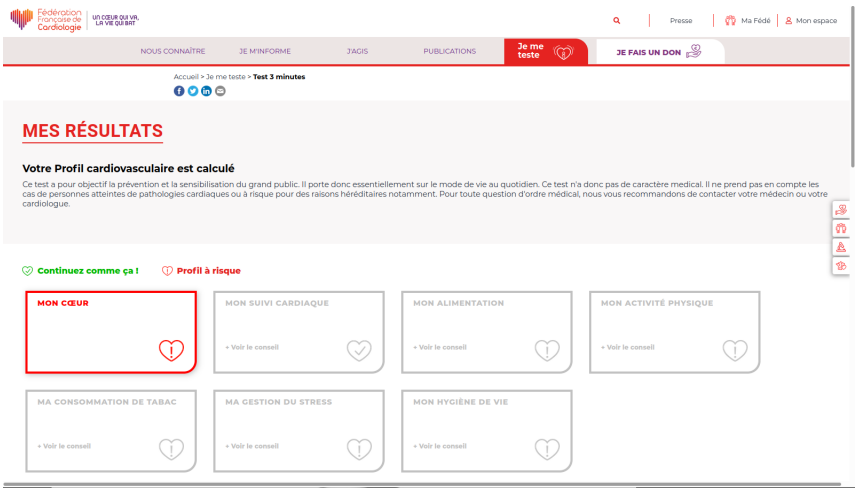


FIGURE 10 – Bilan (partie supérieure)

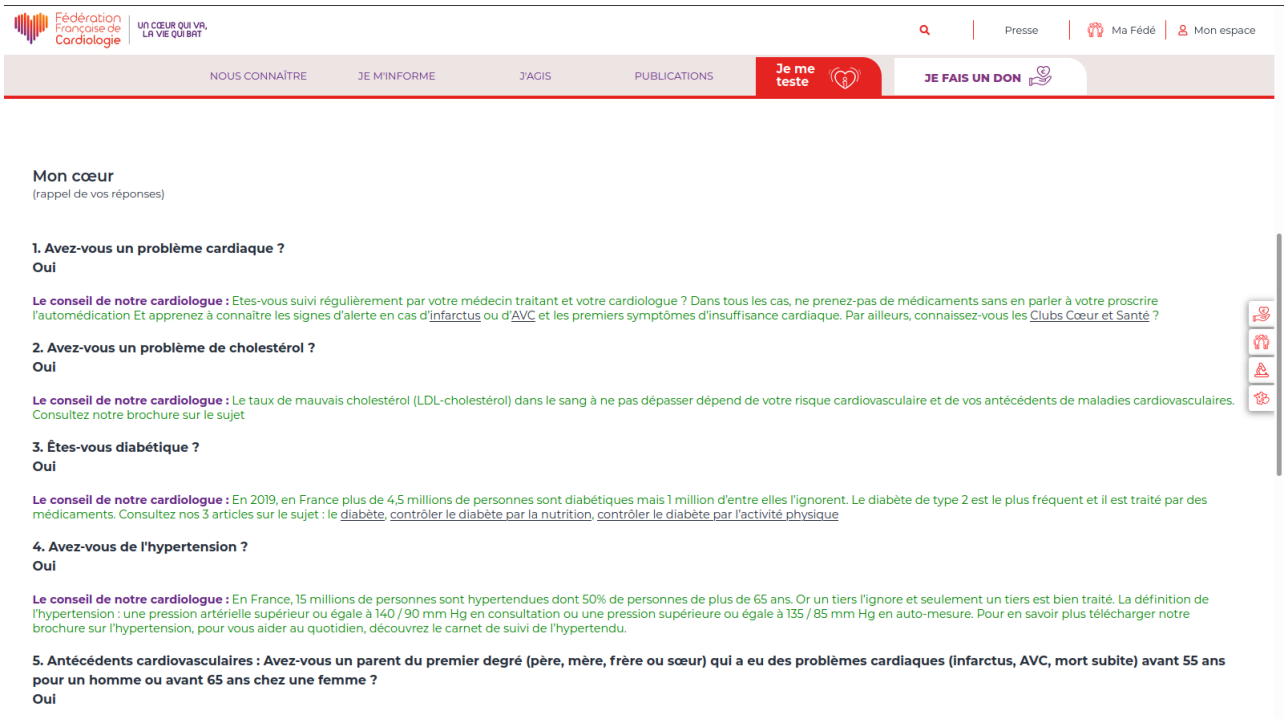


FIGURE 11 – Bilan (partie inférieure)

### 5.2.2 Votre bilan

Inspirez-vous du bilan officiel pour créer une activité :

— résumant les choix réalisés au cours de votre questionnaire ;

— réalisant un traitement à partir de ces choix (par exemple, calculer l'indice de masse corporelle, suggérer des conseils d'alimentation, ...) ;

— afficher le résultat de ce traitement à la manière du "conseil du cardiologue" dans la figure précédente.

# 6 Finaliser l'application

## 6.1 Un peu d'originalité

Jusqu'ici, même s'il a été demandé de vous inspirer du site de la Fédération de cardiologie, mais il est probable que votre projet ressemble beaucoup à ceux de vos camarades.

Créez au moins une nouvelle activité **personnalisée** sur un thème qui vous intéresse (par exemple : alimentation, activité sportive, consommation d'alcool/tabac/drogue, *etc.*) et placez-la juste avant l'activité du bilan décrite dans la section précédente. Le contenu de cette nouvelle activité est libre ; la contrainte est qu'elle contienne des informations qui pourront être prises en compte pour la construction de votre bilan final.

## 6.2 Quitter proprement l'application

Dans la dernière activité, ajoutez un bouton appelant la méthode finish () vue en TP. Comme toujours, vérifiez le bon déroulement de cette opération.

## 6.3 Sauvegarde persistante

`Android` propose plusieurs approches de sauvegardes persistantes. Parmi elles, vous avez expérimenté en TP :

— l'écriture/lecture dans un fichier spécifique, qui nécessitent des permissions particulières.

— (Optionnel) les "préférences partagées" qui stockent les informations (avec SharedPreferences.Editor) dans un endroit géré par le système, et accessibles par plusieurs applications (avec SharedPreferences) ;

— (Optionnel) La sauvegarde dans une base de donnée au sein du téléphone.