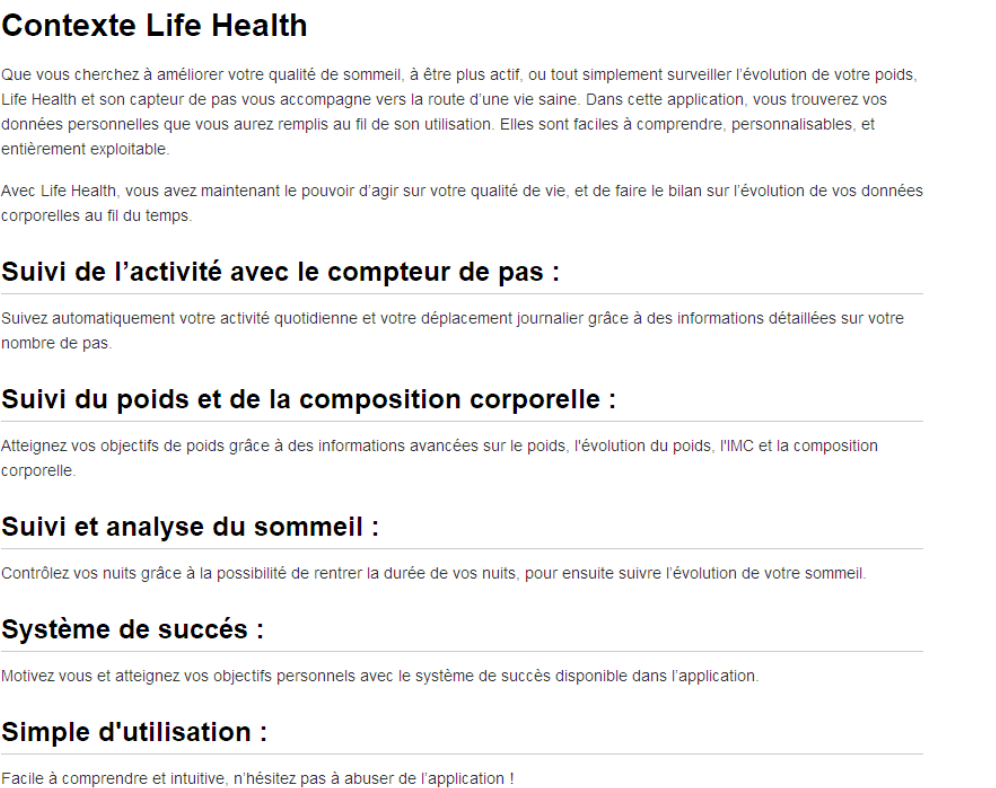
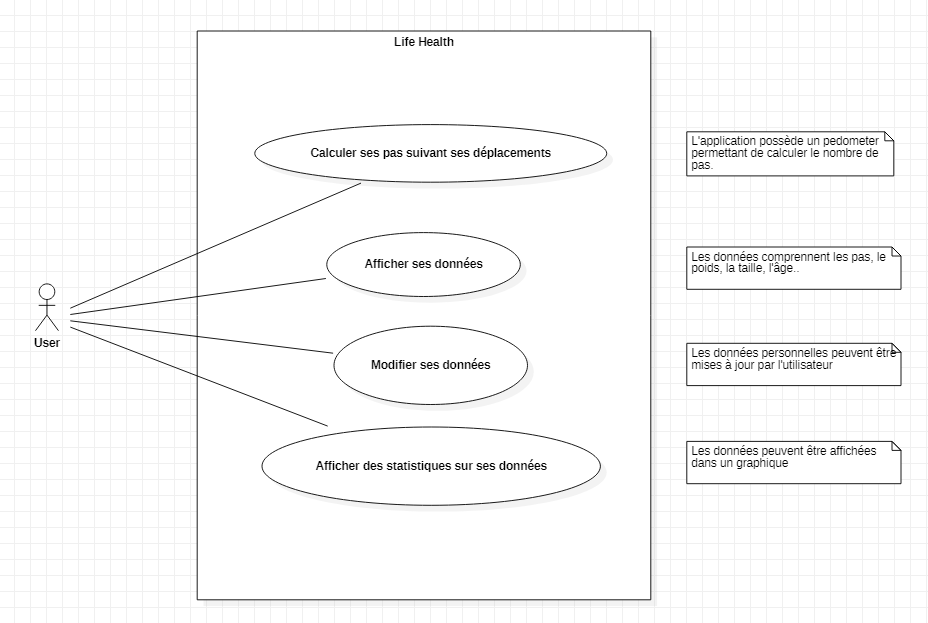
Preuves *Life Health*

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

Le contexte est présent dans le dossier Documentation.

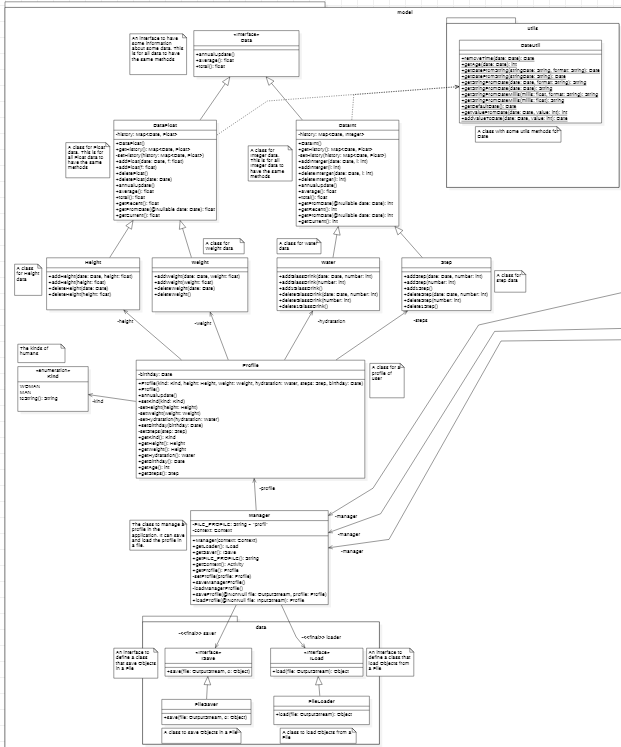
****

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



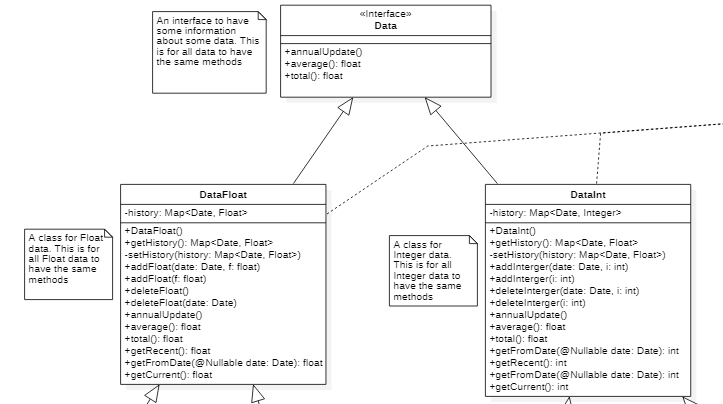
Le diagramme de cas d’utilisation se trouve dans le dossier Documentation.

*Je sais concevoir un diagramme UML de qualité représentant mon application.*



Le diagramme de classe au complet se trouve dans le dossier Documentation.

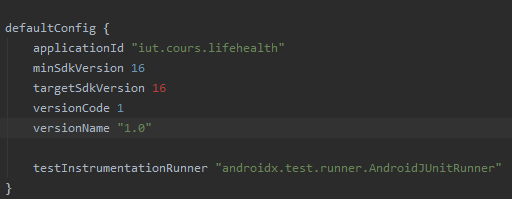
*Je sais décrire mon diagramme UML en mettant en valeur et en justifiant les éléments essentiels.*



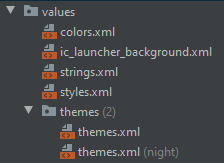
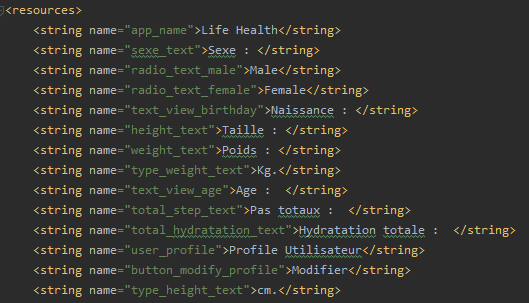
Différents commentaires sont présents sur celui-ci. Il est au complet dans le dossier Documentation.

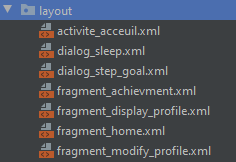
Code :

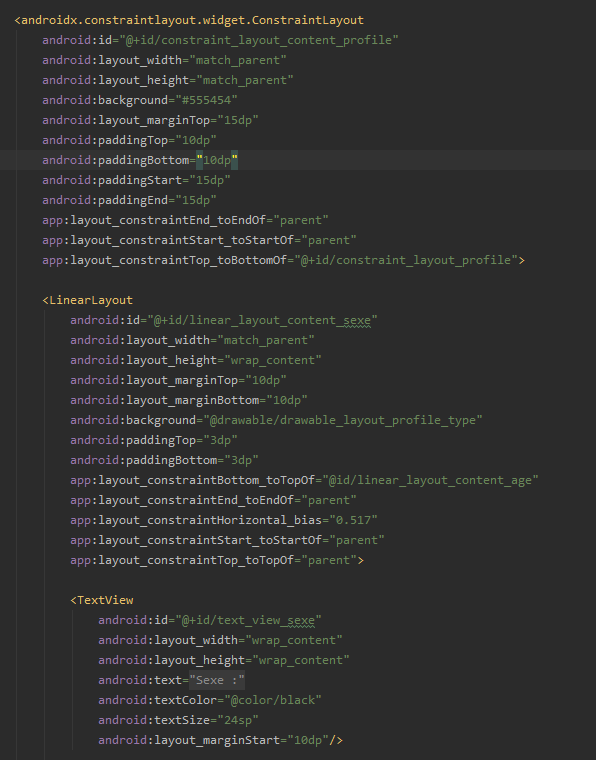
*Je sais utiliser les Intent pour faire communiquer deux activités.*  
  
*Je sais développer en utilisant le SDK le plus bas possible.*

Fichier Gradle  


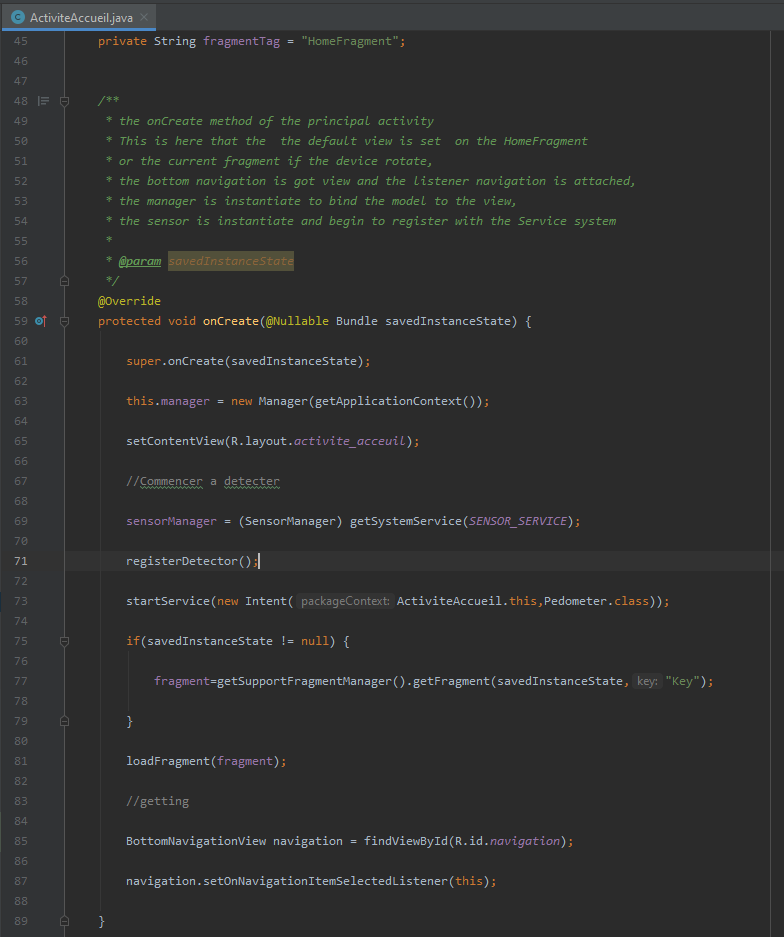
*Je sais distinguer mes ressources en utilisant les qualifier.*

Fichier strings.xml **

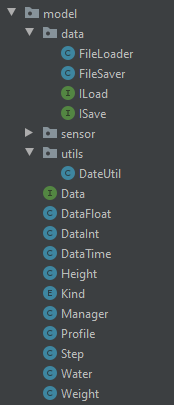
*Je sais faire des vues xml en utilisant layouts et composants adéquats.*

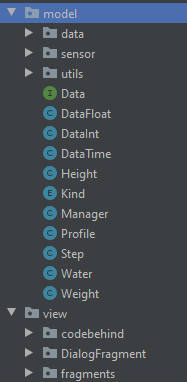
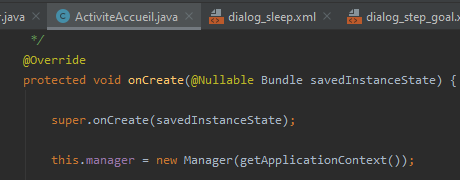
Fichier fragment\_display\_profile.xml dans res/drawable/layout

Nous avons séparé différents layout, et utilisons des ConstraintLayout pour avoir des vues responsives.

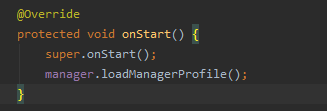
*Je sais coder proprement mes activités, en m’assurant qu’elles ne font que relayer les évènements*

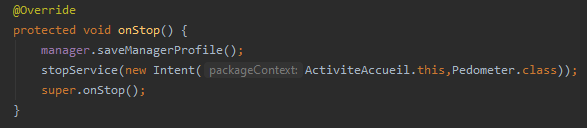
Classe ActiviteAccueil dans view/codebehind

*Je sais coder une application en ayant un véritable métier*  
La partie modèle est le métier, avec par exemple le manager, qui gère un profil pour l’utilisateur.

*Je sais parfaitement séparer vue et modèle*  
  
Les vues utilisent le manager, mais ne font pas les tâches du manager.

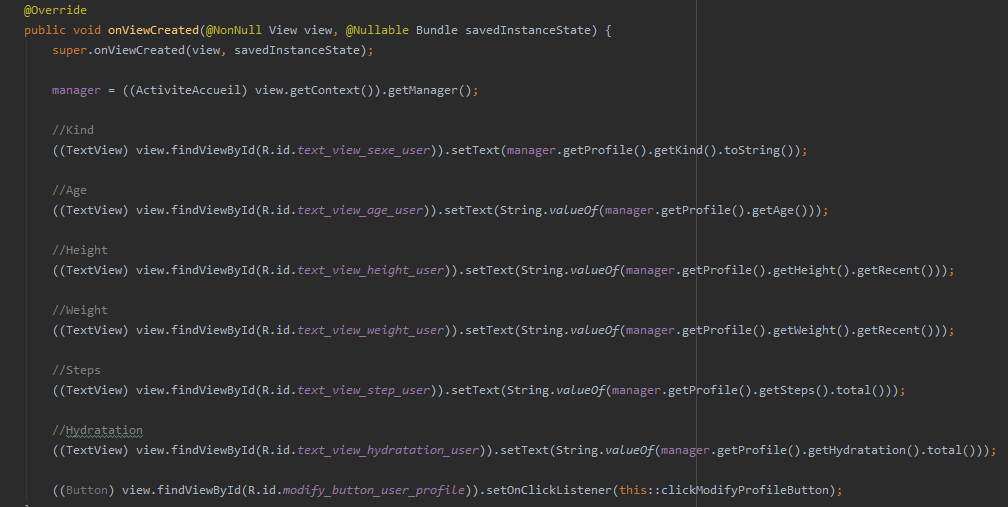
*Je maîtrise le cycle de vie de mon application*

Classe ActiviteAcceuil dans view/codebehind



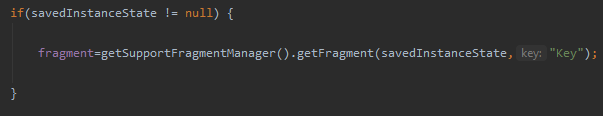
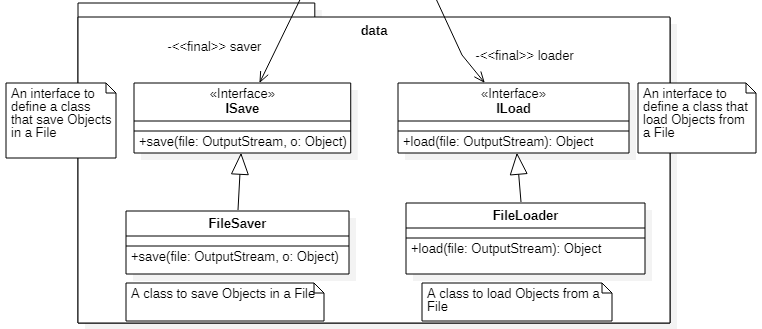
Lorsque nous avons un onStop, nous mettons également un onStart, pour, par exemple ici, sauvegarder et charger le manager.

*Je sais utiliser le findViewById à bon escient*Classe DisplayProfileFragment view/fragments

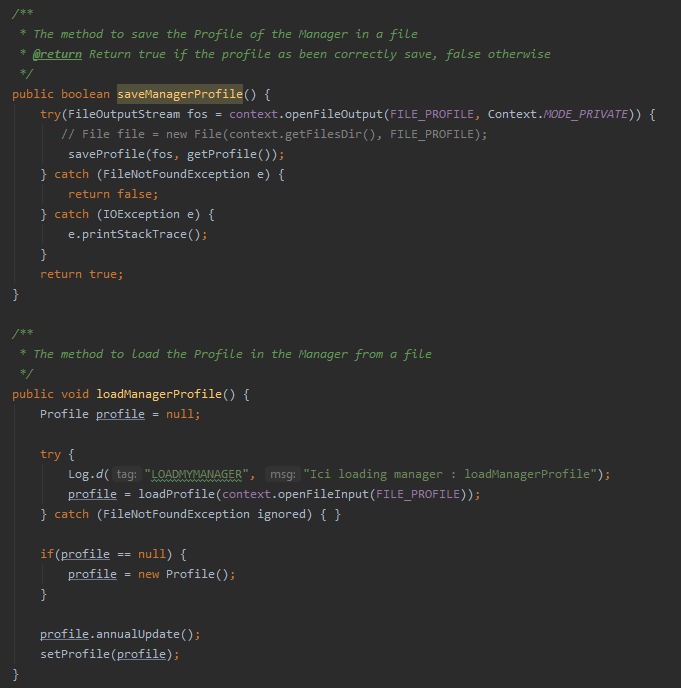
  
Ceux-ci nous permettent d’insérer les données souhaitées où nous le voulons.

*Je sais gérer les permissions dynamiques de mon application*

*Je sais gérer la persistance légère de mon application*

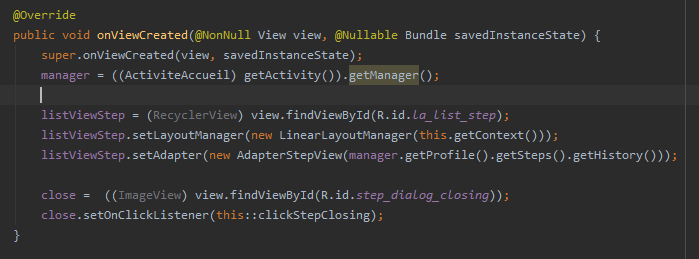
Ici, on fait la persistance légère du fragment qui était en cours d’affichage  
  
*Je sais gérer la persistance profonde de mon application*

Classe Manager dans model :



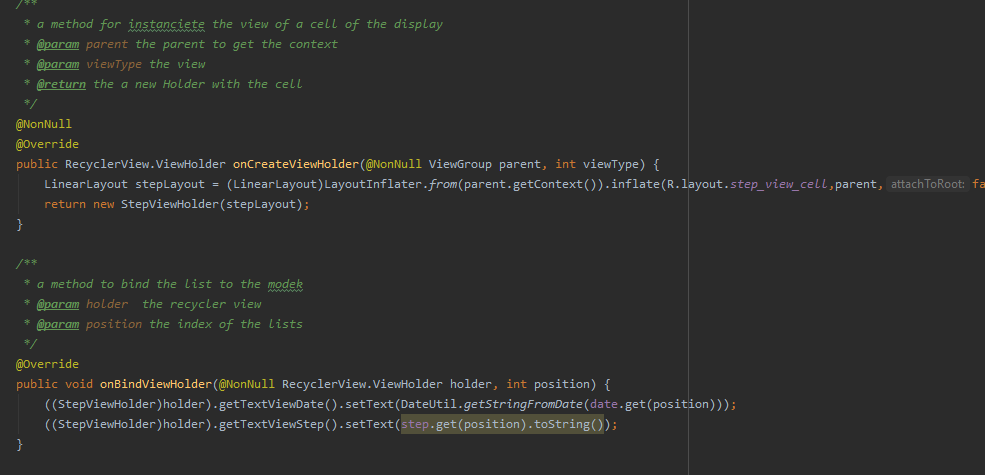
Nous sauvegarder le profil dans un fichier du téléphone.

*Je sais afficher une collection de données*



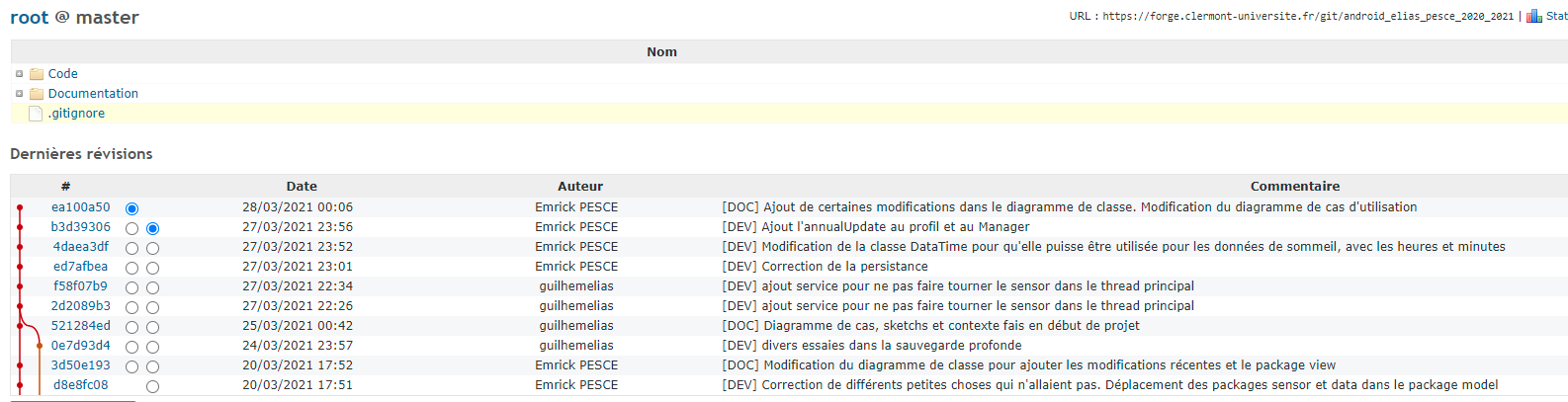
Extrait de la classe StepGoalDialog pour instancier la RecyclerView et l’adapter à l’affichage de la Map.

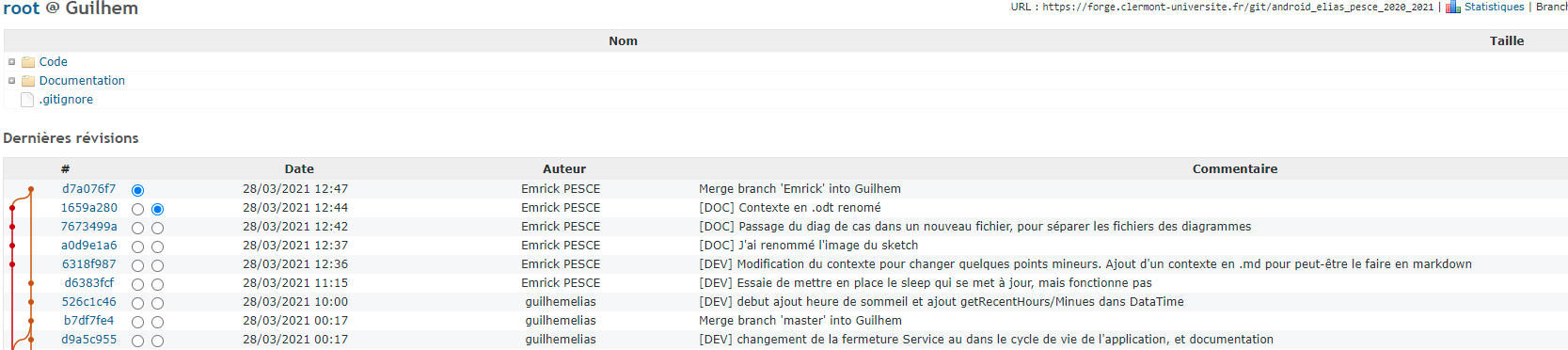
Voir aussi les classes AdapterStepView/StepViesHolder

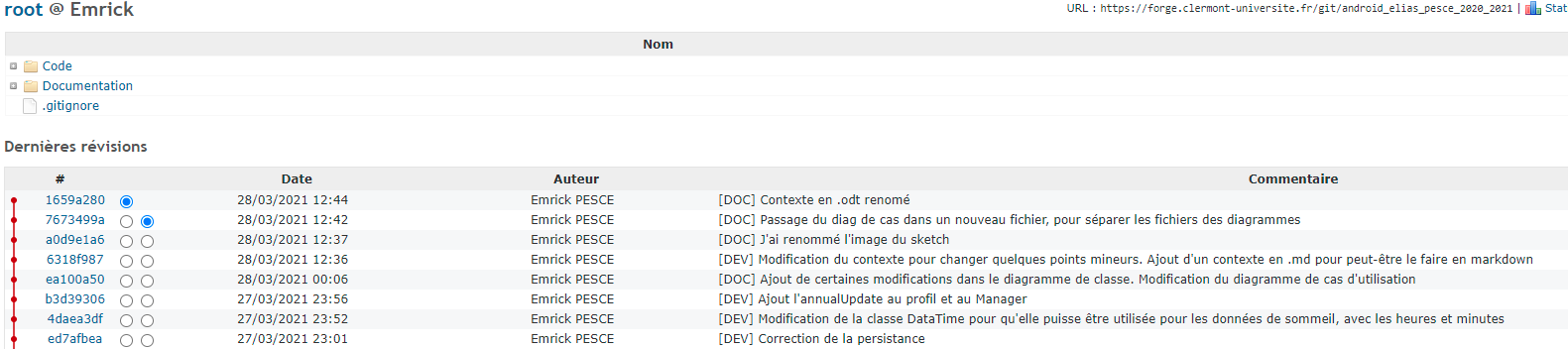
*Je sais coder mon propre adaptateur* Extrait de la classe AdapterStepView

*Je maîtrise l’usage des fragments*  
Ici, on a le gestionnaire de navigation utilisant les fragments

*Je maîtrise l’utilisation de Git*







On peut voir que l’on a chacun une branche personnelle où on code, et on merge le tout sur la branche principale master, ou sur nos branches si on a des changements qu’on ne veut pas forcément directement sur le master.

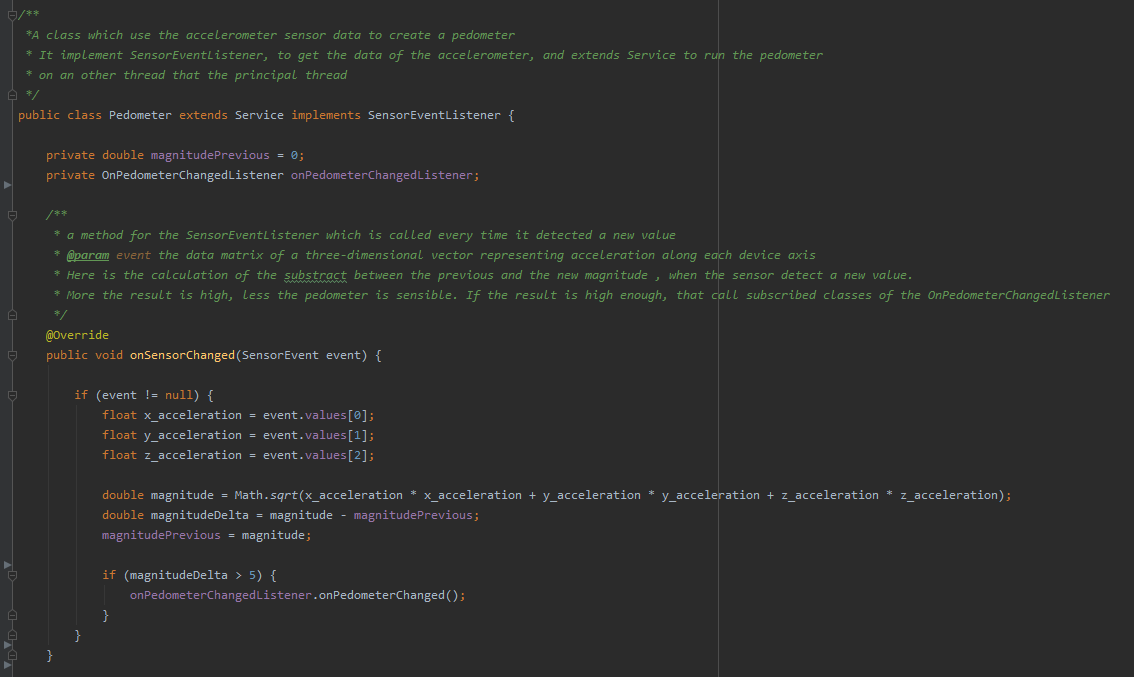
Application :

*Je sais développer une application sans utiliser de librairies externes*

*Je sais développer une application publiable sur le store.*

À venir

*Je sais utiliser l’accéléromètre et ou le gyroscope*



Classe Pedometer dans model/sensor