Compte rendu Audit Consultatif

AC sur le code Android avec Matthias BRUN / ProSE

Appel à l'ordre :

Date de la réunion : 12/05 **Heure début et fin :** 8h ⟶ 9h30 **Lieu :** FLOYD A206

Réunion organisée par : Matthias Brun **Type de réunion :** Audit Consultatif sur le code de l'inc 1 **Rédacteur :** Camille Lenne **Point météo :** Très nuageux avec de la pluie	**Participants :** Théo Bénard, Matthias Brun, Camille Lenne, Gabriel Marquette.

Objectif de la réunion :

Cet audit consultatif permet de faire un point poussé sur l'avancement du code Android. Il nous permet d'avoir un retour complet et surtout des points d'améliorations pour progresser.

Sujets abordés et actions menées :

- Organisation du dépôt à revoir, méthodes de MainActivity à renommer en gui, et frame lien avec gui à réévaluer pour l'incrément 2.
- Gestion de la socket dans ConnectionCANdroid à améliorer, gestion statut et timeout de connexion, utilisation du pattern observateur, gestion des erreurs avec try-catch.
- Protocole à coder, explication du protocole entre la partie C et Java, utilisation de méthodes plus efficaces pour la lecture et l'encodage des messages.
- Amélioration de la visibilité et de la structure du code, éviter les pratiques non recommandées (ex. IP et port en dur), utilisation de Singleton pour la communication, revoir les directives de visibilité et les pansements de code.

Commentaires :

Pastille orange/rouge, 30 % d'avancement

Prise de note complète de la réunion

Audit consultatif (CR) du 12 mai

État d'avancement :

80 % d'avancement → état réel : 30 % de l'incrément 1

-organisation du dépôt à revoir

-les méthodes de mainActivity (à renommer en gui) ne sont pas déclarées dans la conception

-frame lien avec gui peut-être à revoir pour l'incrément 2

-préfixe, à voir puisque pas commun, ça déroute…

ConectionCANdroid
---

dans getSocket() à ne pas faire comme ça :
État de la connexion doit être géré différemment, il ne faut pas exposer la socket.

Gestion du status de la socket à revoir.

Dans connect() :
sur la connexion, ce serait bien d'avoir un timeout
et un pattern observateur, (se mettra en écoute de la communication) ça appellera le callback, onConnexionOk ou onConnexionFail
pareil, avec le pattern observateur

→ throw, il ne sert à rien, il faut faire du catch et pattern observateur
gestion d'erreur à affiner, ça peut être bien avec une pattern observateur, avec unconnect, par exemple pourvoir si c'est good pour la connexion
valable partout, pas que sur connect(), sur écriture, read()

read() à revoir
là, c'est okay pour le throw, juste, il faut le déclarer par un ajout de throw dans la signature
pourquoi changer la nature de l'exécution, c'est uneIOException, pas une RUNtimeException
il faut juste mettre throw e; donc il y aura un pattern observateur et un throw

taille de ce qu'on lit,
byte[1024] ça peut ne pas être suffisant. On fait une boucle et analyser jusqu'à ce qu'on ait un caractère de fin de protocole/ jusqu'à un certain volume d'information.

Il faut un doc pour expliquer le protocole, de manière générale, entre la partie C et celle en java.

toByteArray, pourquoi pas pour encodeMessage()
c'est un peu "artisanal".

Le second try, de read puis write,
est un peu couteux, à revoir, la façon générique, "lis-moi un truc de taille x"
Pourquoi pas direct, utiliser des primitives qui permettent de lire des chaines de caractère.

Pour la lecture : readline, avec des buffers... comme dans felix et camix

sinon, on peut faire en binaire, préfixé par la taille ou par un caractère de fin de chaine
→ revoir l'api utilisé

!!! protocole à coder !!!

ProtocolCANdroid
---

l'objet métier enverra au dispatcher qui lui va dispatcher,
protocole, décodage à faire

tout pourrait être statique pour gagner du temps

LoggerCANdroid

à faire

DispatcherCANdroid

à faire

CommunicationCANdroid
---
ip et port pas en dure dans le code, c'est une mauvaise pratique

soit à remplir dans un fichier de préférence/ soit par l'utilisateur

passer la communication en singleton

normalement, il n'y a que la communication qui a une visibilité publique, les autres packages protected
toujours, la socket ne doit pas être exposé

receiveFrame() ça ne doit pas être publique

directives de visibilité à revoir

pas top le pansement du catch de receiveFrame

run() lancé par
implement runable

on pourrait/devrait faire onConnected lancé, this.start, qui créera un thread qui se branchera qur le run()

AppCompactActivity
---
injection du dispatcher à la comm ici n'est pas utile, ligne 16 et 24 (fait en live)

Point d'attention particulier :
Structuration de la comm, branchement avec le reste de l'appli, singleton.