Année académique 2018 - 2019

Applications sur Androïd

Monitoring sur A.P.I. et commandes à distance

Cumbo Fabio

Bachelier en Informatique & systèmes orientation réseaux & télécommunications avec option développement

3e année

Enseignant : SCOPEL Fabrice

HAUTE ECOLE EN HAINAUT
Campus technique
8a, avenue V. Maistriau B-7000 Mons
Tél : +32 (0)65 33 81 54
Fax : +32 (0)65 31 30 51
E-mail : tech-mons@heh.be
www.heh.be

Formations de Bacheliers et de Masters organisées par la Fédération Wallonie-Bruxelles
FÉDÉRATION WALLONIE-BRUXELLES
pôle hainuyer

Table des matières

1. Présentation du projet

1.1. Introduction

Le projet a pour objectif la création d'une application sur Androïd permettant de réaliser la supervision et la commande à distance sur deux processus industriels :

- Conditionnement de comprimés, commandé par l'automate S7-1516 2DPPN.
- Asservissement de niveau de liquide, commandé par l'automate S7-1214C.

1.2. Consignes du projet

L'interface visuelle de l'application doit être soignée afin d'afficher les informations d'entrées et de sorties « machine » de façon claire et textuelle ou imagée. Elle doit aussi de pouvoir modifier ces informations aux départs de zone de texte. L'interface doit également suivre le Material Design, il faut donc utiliser au minimum l'API 21 d'Androïd (version 5.0).

Il est nécessaire de prévoir une gestion utilisateur. En effet, les droits de lecture/écriture ne sont pas accessibles à tous, par défaut un utilisateur enregistré n'aura que les droits de lecture sur l'automate. Celui-ci sera représenté par un nom, un prénom, une adresse e-mail et un mot de passe d'au minimum 4 caractères. Les choix des privilèges des utilisateurs sont accessibles via le compte « super-utilisateur » qui est défini au premier démarrage de l'application.

L'application doit être évolutive, donc les paramètres de connexions à l'automate doivent être modifiables dans l'application : adresse IP, rack et slot.

L'application doit pouvoir consulter les données des variables de l'automate via le serveur web disponible et afficher un maximum d'information. Plusieurs champs de textes doivent être prévus afin de modifier les variables spécifiques dans l'automate (numéro de databloc + variable).

1.3. Langages et outils utilisés

Le langage Java, lié au XML (pour l'interface) a été utilisé afin de développer l'application.

Au niveau des outils, j'ai utilisé l'IDE Android Studio qui est basé sur Intellij IDEA et recommandé par Google pour le développement sur Androïd.

J'ai également utilisé Git afin de versionner et suivre efficacement le développement du code source de l'application.

Pour simuler un automate programmable, j'ai utilisé une machine virtuelle Windows XP contenant une suite de logiciel :

- **SIMATIC WinAC RTX :** fournis des solutions d'automatisation et permet d'exécuter des fonctions de commande déterministes en temps réel sur un ordinateur.

- **SIMATIC Manager** : Il a été utilisé afin de permettre de charger un processus industriel avec SIMATIC WinAC RTX.

- **ProcesSim :** ce logiciel permet de simuler le comportement des machines et processus industriels. Celui-ci a donc rendu possible la simulation des deux processus industriels.

2. Base de données

2.1. Conception

La base de données de l'application comprend une seule table, celle-ci permet de stocker les informations relatives à chaque utilisateur enregistré.

TABLE_USER
COL_ID
COL_EMAIL
COL_PASSWORD
COL_LASTNAME
COL_FIRSTNAME
COL_WRITE
COL_ADMIN
id : COL_ID

Schéma conceptuel de la base de données

- **COL_ID :** c'est l'identifiant unique de l'utilisateur. Il est utilisé comme clé primaire de la table et est auto-incrémenté.

- **COL_EMAIL :** représente l'adresse mail indiquée lors de l'inscription de l'utilisateur.

- **COL_PASSWORD :** représente le mot de passe de l'utilisateur. Il doit être composé d'au minimum 4 caractères.

- **COL_LASTNAME :** c'est le nom de famille de l'utilisateur.

- **COL_FIRSTNAME:** c'est le prénom de l'utilisateur.

- **COL_WRITE :** cette colonne permet de définit les autorisations d'un utilisateur.

  - **0** indique qu'il a les droits en lecture uniquement (valeur par défaut utilisé).

  - **1** indique qu'il a les droits en lecture et écriture.

- **COL_ADMIN :** cette colonne donne les privilèges d'un utilisateur

  - **0** indique que c'est un simple utilisateur, il n'a donc pas accès aux modifications de profil des utilisateurs. (par défaut)

  - **1** spécifie un compte « super-utilisateur ». Celui-ci peut donc modifier à sa guise le profil de n'importe quel utilisateur.

Pour l'accès et l'utilisation de la base de données, 3 classes ont été utilisées :

- **User :** c'est une classe permettant de définir un utilisateur

- **UserAccessDB:** c'est la classe permettant d'accéder aux données, elle contient donc des méthodes pour manipuler la base de données

- **UserBddSqlite :** c'est une classe permettant de créer la base de données et la table pour les utilisateurs

2.2. Type de stockage

Afin de créer la base de données et d'envoyer/récupérer des données, la base de données a été réalisée à l'aide du SGBD SQLite.
En effet, celle-ci ne nécessite qu'une petite quantité de mémoire puisqu'elle

s'exécute sans serveur. L'exécution des requêtes se fait donc dans le même processus que l'application et l'utilisation de la base de données est réservée à l'application créatrice uniquement.
Le stockage est donc optimisé et la syntaxe ressemble fortement à celle du SQL.

3. Fonctionnement de l'application

3.1. Inscription

Avant de pouvoir manipuler l'application, la personne doit obligatoirement s'inscrire dans la base de données.
L'utilisateur doit alors indiquer ses informations : *Nom, Prénom, Adresse e-mail et Mot de Passe.*

Des vérifications ont lieu sur certains des champs :

- Tous les champs doivent être remplis.
- L'adresse e-mail doit avoir un bon format (*nom@domaine.xxx).*
- L'adresse e-mail ne doit pas déjà être associée à un utilisateur dans la base de données (vérification de doublons).
- Le mot de passe doit contenir au minimum 4 caractères.

3.2. Connexion

Lorsque l'utilisateur est inscrit, il peut bien évidemment se connecter sur l'activité de connexion.

Celui-ci doit alors entrer son adresse e-mail indiquée lors de l'inscription et son mot de passe associé. Dans le cas où il entre de mauvaises informations, il en sera prévenu.

3.3. Paramètres de connexion à l'automate

N'importe quel utilisateur aura la possibilité de configurer la connexion à l'automate, qui est nécessaire pour pouvoir lire ou écrire des données.
3 options sont alors demandées et nécessaires :

- L'adresse IPv4 de l'automate.

- Le numéro du rack.
- Le numéro du slot.

Tout comme la modification d'utilisateur, on accède à cette activité à partir d'une option sur la barre de menu de l'activité principale pour le choix des automates. Cependant, n'importe quel utilisateur aura la possibilité de modifier ces données, et il pourra les changer à tout moment.

Du point de vue programmation, ces informations sont contenues dans une classe *Configs.java* contenant des assesseurs et mutateurs pour enregistrer et récupérer la configuration de l'automate. Étant donné qu'il n'y a qu'une configuration pour les 2 automates (point de vue à améliorer), cette classe est statique et globale au projet, elle est donc accessible par toutes les classes sans avoir besoin de l'instancier en permanence pour pouvoir utiliser la configuration.

3.4. Modification d'un utilisateur

Le « super-utilisateur » a la possibilité de modifier les informations des utilisateurs. L'activité de modification d'utilisateur est lancée à partir d'une option sur la barre de menu de l'activité principale pour le choix des automates. (Elle s'affiche uniquement si l'utilisateur connecté est un « super-utilisateur »).

La liste des utilisateurs s'affiche alors dans un menu déroulant (composant span) et les composants affichant les informations de l'utilisateur sélectionné se mettent à jour automatiquement.

Il a la possibilité de changer les privilèges administrateur et les droits de lecture/écriture directement depuis des switchs (ON/OFF). Une fois qu'une action a lieu sur ces composants, la base de données est automatiquement mise à jour et on fait appel à la classe *Configs.java* pour mettre à jour également les privilèges et droits en écriture de l'utilisateur.

On retrouve également un champ permettant de modifier le mot de passe de l'utilisateur sélectionné et on laisse la possibilité de le supprimer (une demande de confirmation est alors demandée dans ce cas).

3.5. Automates

Lorsque l'on se trouve dans l'activité du choix d'automates, l'utilisateur a la possibilité de naviguer dans l'activité du processus du conditionnement de comprimés ou de l'asservissement de liquide.

L'interface de ces 2 activités est assez similaire, on retrouve :

- Le titre correspondant au type de processus.

- Un texte est placé tout en haut au centre et affiche le numéro de PLC (référence).

- Les données propres aux automates.

- Un bouton permettant d'initialiser la connexion, par défaut il est un rouge lorsque l'application n'est pas connectée à l'automate et il passe au vert si la connexion a été établie. Si la connexion à l'automate n'a pas pu être établie (par exemple une mauvaise adresse IP en paramètre, le bouton passe alors en orange et on prévient l'utilisateur avec un *Toast*.

- Lorsque c'est un utilisateur avec les droits en écriture qui se trouve dans une de ces activités, il a la possibilité de cliquer sur le bouton écrire (uniquement lorsque la connexion à l'automate est établie, autrement dit, que le bouton soit vert) afin de faire apparaitre des zones de textes et des boutons pour écrire dans les variables de l'automate.

Au niveau de la conception du point de vue de la programmation, ces 2 activités sont liées aux classes *ReadTaskS7* (lecture) et *WriteTaskS7* (écriture).
La conception de ces classes est réalisée à l'aide de Thread et d'Handler afin de pouvoir récupérer en permanence les informations de l'automate et d'effectuer plusieurs traitements en tâche de fond sans bloquer les manipulations faites avec l'application pendant ce moment. Les différents traitements sont effectués avant l'exécution (affichage du PLC), pendant (mise à jour des données et de l'interface) et après (message de déconnexion à l'automate).

3.5.1. Conditionnement de comprimés

Pour la lecture des données dans cet automate, 5 valeurs sont récupérées et affichées avec des composants graphiques :

- **Sélecteur en service :** la checkbox est cochée dans le cas où l'automate est en service, et décochée dans l'autre cas.

- **Arrivées de flacons vides :** la checkbox est cochée si l'arrivée de flacons vides est activée, non dans le cas inverse.

- **Nombre de comprimés demandés :** donne le nombre de comprimés sélectionné à l'aide de 3 boutons, lorsqu'une valeur est sélectionnée, le bouton correspondant à la valeur passe alors en vert sur l'application.

- **Comprimés mis en flacon :** donne le nombre de comprimés mis en flacon à l'instant même.

- **Bouteilles remplies :** affiche le nombre de bouteilles passé par le poste de bouchonnage et rempli (maximum 999).

Lorsque le bouton pour écrire est actionné, les champs de textes et l'écriture dans les variables sont disponibles.
5 valeurs peuvent être entrées par l'utilisateur :

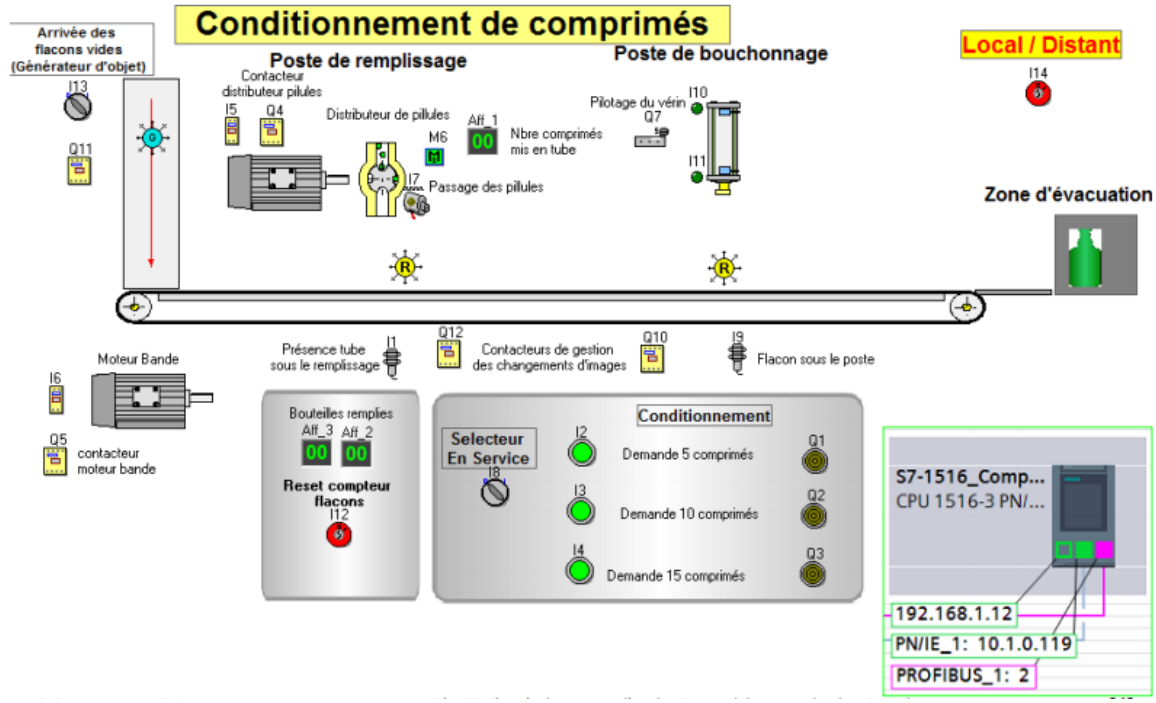
- **DBB5 :** c'est une variable à définir en binaire.

- **DBB6 :** elle possède un fonctionnement similaire à DBB5, il faut entrer une valeur en binaire.

- **DBB7 :** elle aussi possède un fonctionnement proche de DBB5.

- **DBB8 :** c'est un octet formaté en BCD (décimal codé binaire), le format à entrer est un entier.

- **DBW18 :** c'est une variable à entrer sous forme d'entier.



Processus du conditionnement de comprimés

3.5.2. Asservissement de niveau de liquide

Pour la lecture des données de cet automate, 9 valeurs sont récupérées et affichées :

- **Sélecteur de mode :** 2 boutons sont présents et lorsque le sélecteur a été défini dans un des 2 modes, le bouton correspondant au mode se colore en vert automatiquement.

- **État des 4 valves :** Lorsqu'une des valves est ouverte, la checkbox correspondant à la vanne actionnée est cochée, sinon dans le cas contraire, elle est décochée.

- **Niveau d'eau :** affiche le niveau de liquide.

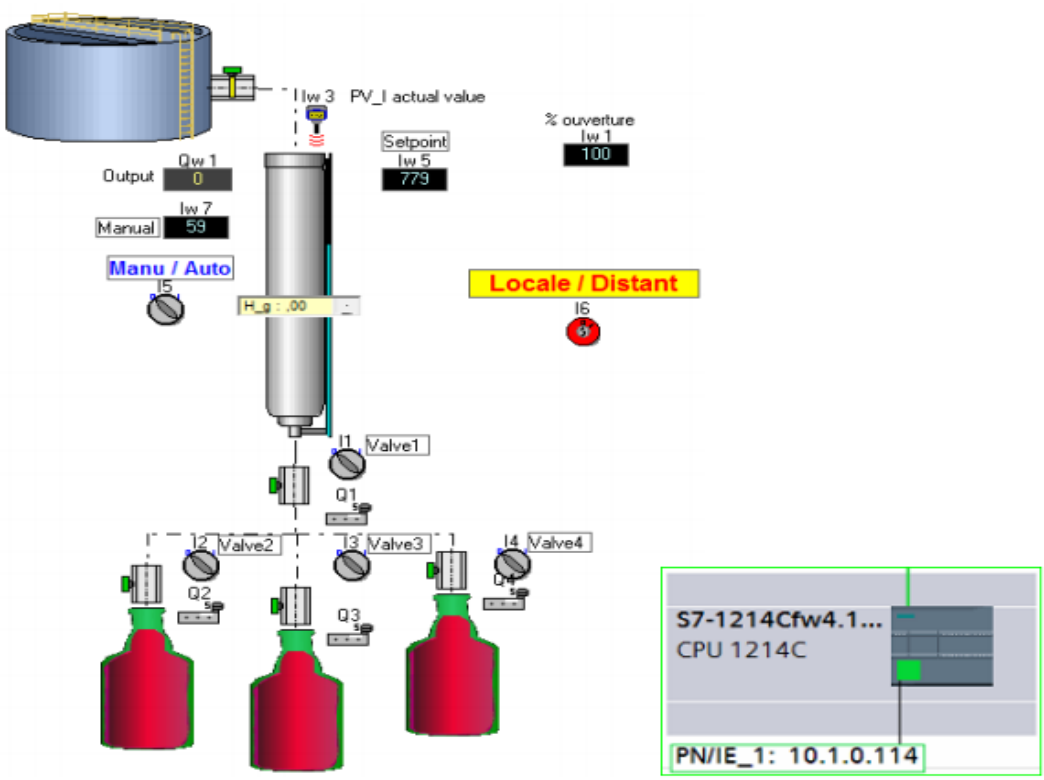
- **Consigne auto :** affiche la valeur de la consigne auto.

- **Consigne manuelle :** affiche la valeur de la consigne manuelle.

- **Mot de pilotage vanne :** affiche la valeur du mot de pilotage vanne.

De la même manière que pour le conditionnement de comprimé, un bouton pour écrire est disponible lorsque l'utilisateur a les droits en écriture. 6 valeurs peuvent être entrées par l'utilisateur :

- **DBB2 :** c'est une variable à entrer en binaire.
- **DBB3 :** elle possède un fonctionnement similaire à DBB2.
- **DBW24 :** c'est une variable à entrer au format entier
- **DBW26 :** elle fonctionne de manière similaire à DBW24.
- **DBW28 :** variable au format entier, similaire à DBW24.
- **DBW30 :** c'est aussi une variable au format entier et similaire à DBW24.



Processus de l'asservissement de liquide

4. Problèmes rencontrés

4.1. Lecture dans les automates

Lors du développement de la fonctionnalité pour lire les données de l'automate, je n'arrivais pas à récupérer toutes les informations.
Après quelques recherches, je me suis rendu compte que le problème se situait au niveau des autorisations de l'application et donc au niveau du fichier « *AndroidManifest.xml* ». La ligne « *android.permission.internet* » était manquante et après ajout de cette ligne, l'application arrivait effectivement bien à récupérer et afficher les données.

J'ai également eu quelques difficultés pour la lecture de la variable DB5.DB15. En effet, après quelques recherches, je me suis rendu compte qu'il ne fallait pas utiliser la méthode « *GetBitAt »* comme pour les autres valeurs.

4.2. Écriture dans les automates

L'écriture m'a aussi posé quelques difficultés. En effet, il a fallu trouver un moyen d'écrire des valeurs de type différent comme demandé dans le cahier des charges. Il a donc fallu créer une méthode spécifique pour chaque type de valeur à écrire. Après plusieurs heures de recherches et d'essai, j'ai compris le fonctionnement de l'écriture dans les variables et j'ai pu commencer à développer la fonctionnalité.

5. Possibilités d'améliorations

Quelques améliorations intéressantes pourraient être apportées à ce projet afin d'optimiser quelques aspects :

- Une table supplémentaire concernant la configuration des automates dans la base de données serait intéressante. En effet, d'une part cela permettrait de ne pas redéfinir à chaque lancement de l'application la configuration d'accès à l'automate. D'autre part cela laisserait la possibilité à l'utilisateur de gérer plusieurs automates avec des configurations différentes sur l'application.

- L'utilisation d'une base de données encryptée pour garantir une meilleure sécurité des utilisateurs.

- Stocker uniquement un hash des mots de passe utilisateur afin d'améliorer encore la sécurité.

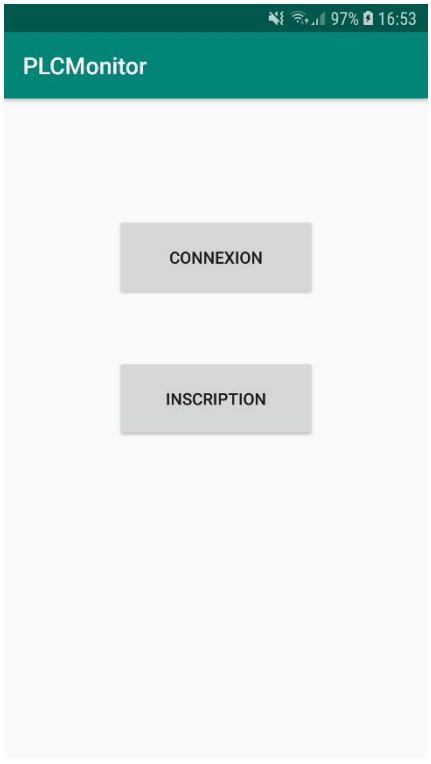
6. Conclusion

L'application permet bien la supervision et la commande à distance des deux processus commandés par les automates.

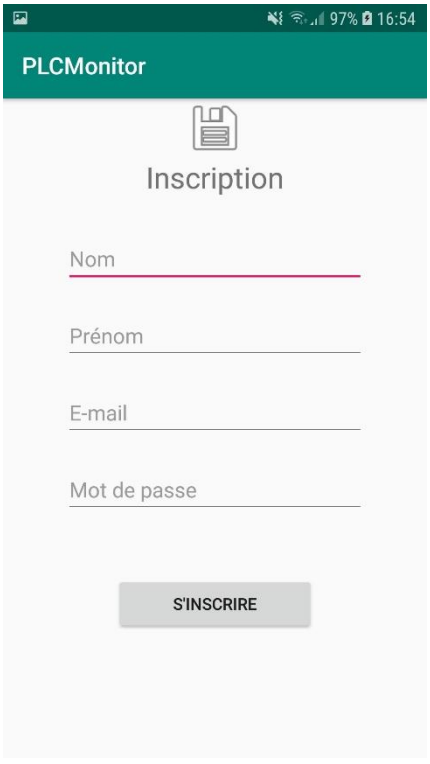
L'interface est simple, le Material design et les layouts ont été utilisés, la lecture et l'écriture dans les automates sont fonctionnelles, les paramètres de connexions sont modifiables et on retrouve bien une gestion utilisateur avec un « super-utilisateur » et des droits d'écritures. Le projet respecte donc bien les engagements énoncés dans le cahier des charges.

Dans l'ensemble le projet s'est assez bien déroulé, il m'aura permis d'en apprendre plus sur le fonctionnement des A.P.I. et sur la conception d'application Androïd avec Java, ce qui me sera utile dans le futur.

7. Captures d'écran de l'application



Menu principal



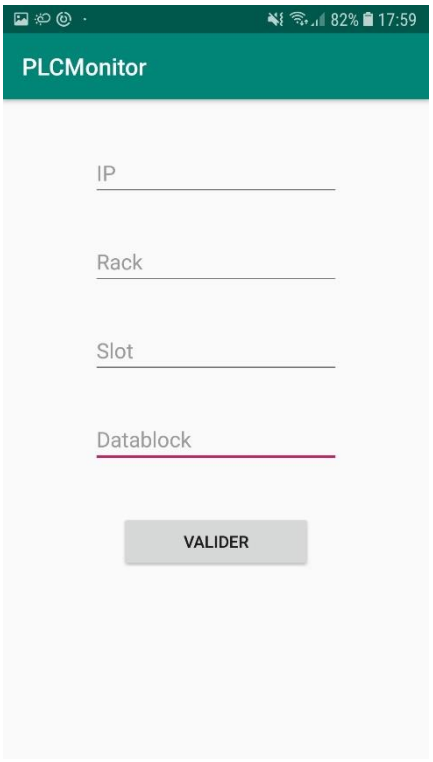
Inscription d'un utilisateur

Connexion d'un utilisateur

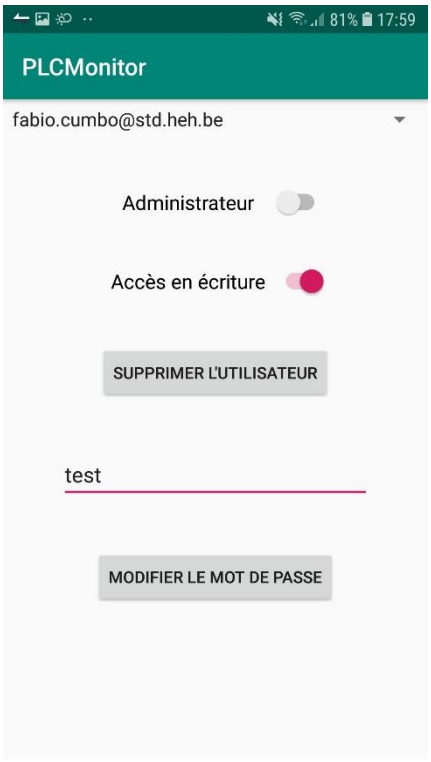
Menu principal du choix des automates

Option disponible dans le menu (profil basique)

Options disponibles dans le menu (admin)



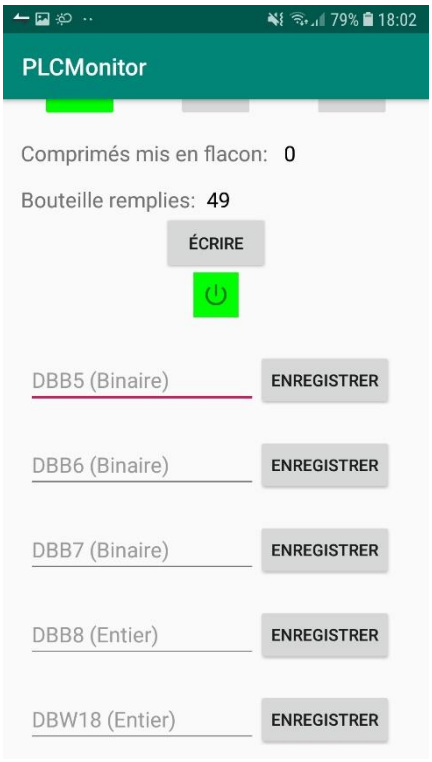
Configuration de l'accès aux automates



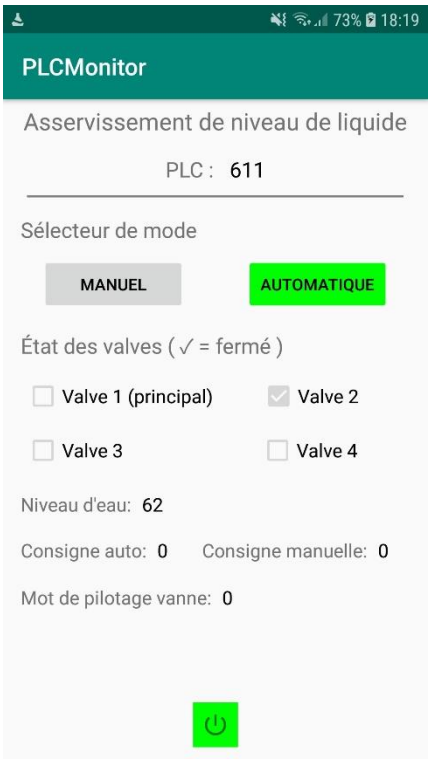
Gestion des utilisateurs (admin)

Conditionnement de comprimés

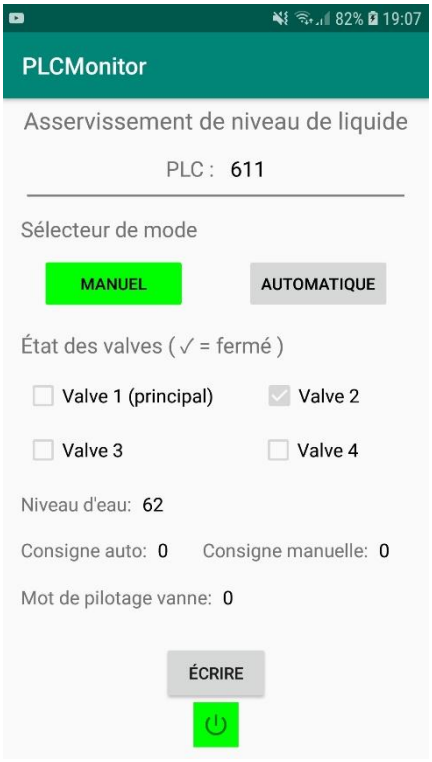
Conditionnement de comprimés (droit en écriture)



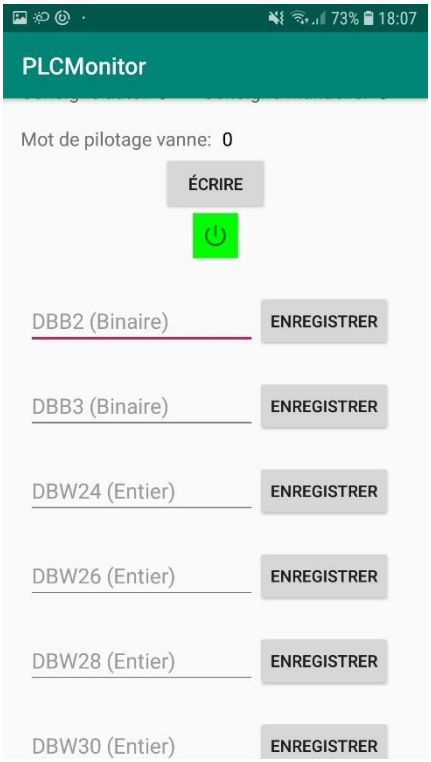
Conditionnement de comprimés (droit en écriture)



Asservissement de niveau



Asservissement de niveau (droit en écriture)



Asservissement de niveau (droit en écriture)