Hofmann Jonny

Documentation projet module 6

Division Technique du CEJEF

ILBSYS



**Table des matières**

[Introduction 4](#_Toc55562032)

[1 But de l’application 4](#_Toc55562033)

[2 Esquisse 4](#_Toc55562034)

[2.1 Cahier des charges fonctionnel 4](#_Toc55562035)

[2.2 Esquisse 4](#_Toc55562036)

[2.3 Gestion des fichiers 4](#_Toc55562037)

[2.4 Planification 4](#_Toc55562038)

[3 Maquette 5](#_Toc55562039)

[3.1 Use case UML 5](#_Toc55562040)

[3.2 Maquette 5](#_Toc55562041)

[4 Gestion du projet 5](#_Toc55562042)

[4.1 Versionning et code source 5](#_Toc55562043)

[4.2 Remontée et suivis des bogues 5](#_Toc55562044)

[5 Classes 6](#_Toc55562045)

[5.1 InfluxDB 6](#_Toc55562046)

[5.1.1 Variables membre 6](#_Toc55562047)

[5.1.2 Constructeur 6](#_Toc55562048)

[5.1.3 Fonction setCurrentServerAddress 6](#_Toc55562049)

[5.1.4 Fonction getRamUsage 6](#_Toc55562050)

[5.1.5 Fonction getCPUUsage 6](#_Toc55562051)

[5.1.6 Fonction getUptime 7](#_Toc55562052)

[5.1.7 Fonction doInfluxRequest() 7](#_Toc55562053)

[5.1.8 Fonction getCPUOverTime 7](#_Toc55562054)

[5.1.9 Fonction getRAMOverTime 8](#_Toc55562055)

[5.2 Server 8](#_Toc55562056)

[5.2.1 Variables membre 8](#_Toc55562057)

[5.2.2 Constructeur 8](#_Toc55562058)

[5.3 Utilities 8](#_Toc55562059)

[5.3.1 Variables membres 9](#_Toc55562060)

[5.3.2 Fonction getAllServersAsString 9](#_Toc55562061)

[5.3.3 Fonction addServer 9](#_Toc55562062)

[5.3.4 Fonction setCurrentServer 10](#_Toc55562063)

[5.3.5 Fonction getCurrentServerIndex 10](#_Toc55562064)

[5.3.6 Fonction getCurrentServer 10](#_Toc55562065)

[5.3.7 Fonction deleteServer 10](#_Toc55562066)

[5.3.8 Fonction editServer 10](#_Toc55562067)

[5.3.9 Fonction saveData 11](#_Toc55562068)

[5.3.10 Fonction loadData 11](#_Toc55562069)

[6 Version 1.0 11](#_Toc55562070)

[6.1 Maquette 11](#_Toc55562071)

[6.2 Diagramme de classe 0](#_Toc55562072)

[7 Sprint 1 0](#_Toc55562073)

[7.1 Fonctionnalités 0](#_Toc55562074)

[7.1.1 Uptime 0](#_Toc55562075)

[7.1.2 Modification de serveurs 1](#_Toc55562076)

[7.1.3 Suppression de serveurs 1](#_Toc55562077)

[7.2 Protocole de tests 2](#_Toc55562078)

[7.2.1 Uptime 2](#_Toc55562079)

[7.2.2 Modifications de serveurs 2](#_Toc55562080)

[7.2.3 Suppression de serveurs 2](#_Toc55562081)

[8 Sprint 2 3](#_Toc55562082)

[8.1 Fonctionnalité 3](#_Toc55562083)

[8.1.1 Accès depuis l’extérieur 3](#_Toc55562084)

[8.1.2 Jauges pour les pourcentages 3](#_Toc55562085)

[8.2 Protocole de test 4](#_Toc55562086)

[8.2.1 Accès depuis l’extérieur 4](#_Toc55562087)

[8.2.2 Jauges pour les pourcentages 4](#_Toc55562088)

[9 Sprint 3 4](#_Toc55562089)

[9.1 Issue #2 4](#_Toc55562090)

[9.2 Graphique pour la température du processeur 4](#_Toc55562091)

[9.3 Graphique utilisation RAM 6](#_Toc55562092)

[9.4 Réorganisation de la page d’informations 7](#_Toc55562093)

[9.5 Protocol de test 7](#_Toc55562094)

[9.5.1 Issue #2 7](#_Toc55562095)

[9.5.2 Graphique température CPU 7](#_Toc55562096)

[9.5.3 Graphique utilisation de RAM 8](#_Toc55562097)

|  |
| --- |
| ILBSYS |
| Projet module 6 |

# Introduction

Dans le cadre du module 6 nous devons réaliser une application Android avec Android studio. Ce module a été décaler en avant parce qu’il est plus que de faire un cours sur les microcontrôleurs à distance.

# But de l’application

Le but de l’application sera d’afficher des informations et des statistiques de différents postes. Ces informations sont stockées dans une base de données InfluxDB qui est nourrie avec Telegraf. Pour adapter le projet à la durée du projet, les fonctionnalités ont été réduite. De ce fait ce logiciel doit plus tôt être vu en tant que proof of concept au lieu d’une application complète.

# Esquisse

## Cahier des charges fonctionnel

* Ajout de serveur
* Connection automatique au serveur
* Choix des serveurs avec des noms définit à l’ajout
* Afficher l’utilisation CPU
* Afficher l’utilisation de la RAM

## Esquisse

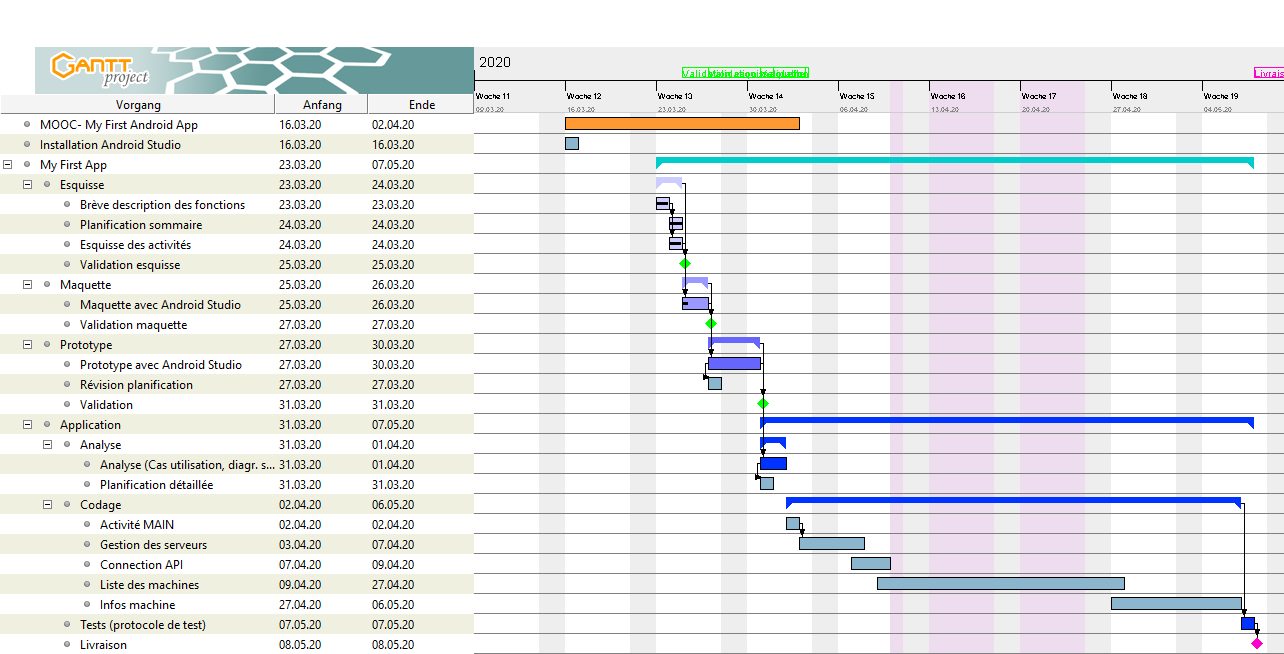
Ceci est la première Esquisse faite pour avoir une vue globale sur la structure fonctionnelle qu’aura l’application.

## Gestion des fichiers

Tous les fichiers sont sauvegardés sur mon GitHub personnel. [ILikeBananas/ILBSYS\_android](https://github.com/ILikeBananas/ILBSYS_android)

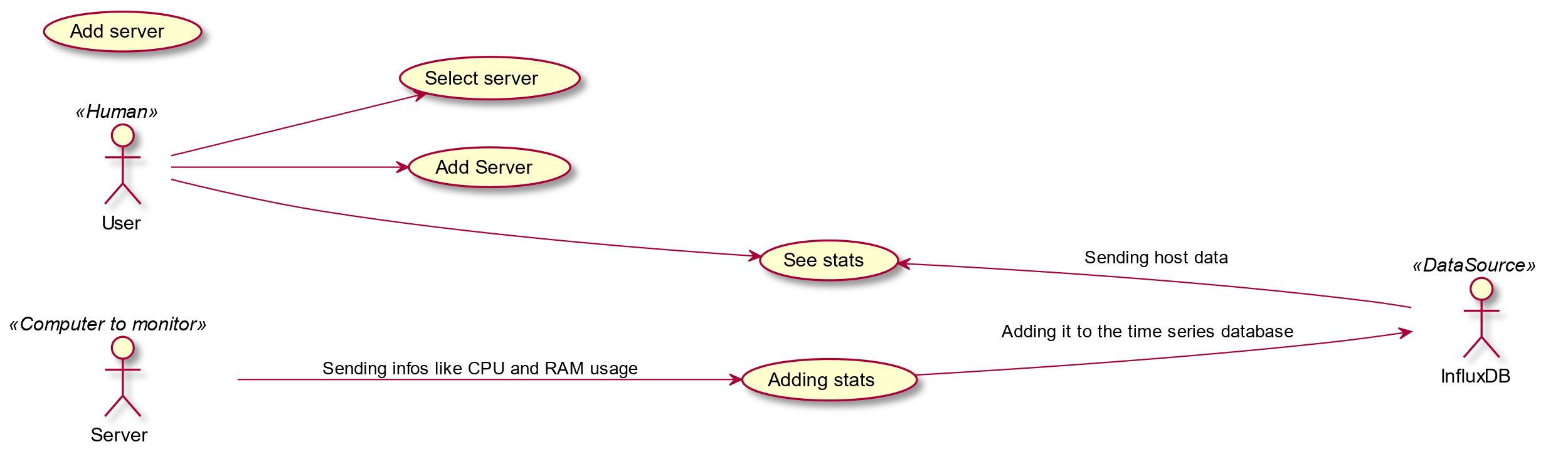
## Planification

La planification se situe dans le dossier « doc » sur GitHub.



# Maquette

## Use case UML



Ce use case est également disponible sous « doc/diagrams/out/useCases.svg » dans le repo GitHub.

## Maquette

La maquette est faite avec le logiciel Balsamiq Mokup et le fichier se trouve dans le dossier « doc » sur GitHub. Elle a été fortement modifiée et n’est plus à jour.

# Gestion du projet

## Versionning et code source

Les versions et le code source sont gérés dans Github ce qui permet de voir tous les changements, quand elles ont été faites et pour quelle cause elles ont été faites.

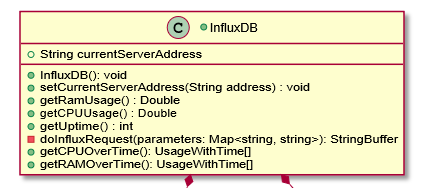
## Remontée et suivis des bogues

Les bogues trouvés durant les tests qui ne sont pas corrigeable directement sont noté et documenté sur GitHub dans la partie Issues. Ceci permet d’avoir un suivit et de lier le commit qui corrige l’erreur.

# Classes

## InfluxDB

La classe Influx gère toues les requêtes vers le serveur InfluxDB.



### Variables membre

|  |  |  |
| --- | --- | --- |
| **Type** | **Nom** | **Description** |
| String | currentServerAddress | Adresse du serveur couramment utilisé |

### Constructeur

**Description :**

Initie la classe InfluxDB et applique les paramètres de thread policies.

### Fonction setCurrentServerAddress

**Description :**

Change l’adresse du serveur utilisé actuellement.

**Paramètre :**

|  |  |  |
| --- | --- | --- |
| **Type** | **Nom** | **Description** |
| String | address | Adresse du nouveau serveur actuel |

### Fonction getRamUsage

**Description :**

Récupère dans la base de données la dernière valeur enregistrée de l’utilisation de RAM pour le serveur actuel.

**Retour :**

|  |  |
| --- | --- |
| **Type** | **Description** |
| Double | Dernière utilisation de RAM, en pourcentage |

### Fonction getCPUUsage

**Description :**

Récupère dans la base de données la dernière valeur enregistrée de l’utilisation du CPU pour le serveur actuel.

**Retour :**

|  |  |
| --- | --- |
| **Type** | **Description** |
| Double | Dernière utilisation de CPU, en pourcentage |

### Fonction getUptime

**Description :**

Récupère dans la base de données le temps en ligne du serveur actuel.

**Retour :**

|  |  |
| --- | --- |
| **Type** | **Description** |
| Int | Temps en ligne en secondes |

### Fonction doInfluxRequest()

**Description :**

Exécute la requête sur la base de données et retourne le resultat.

**Paramètre :**

|  |  |  |
| --- | --- | --- |
| **Type** | **Nom** | **Description** |
| Map<string, string> | parameters | Paramètres de la requête |

Parameters doit contenir au moins deux valeurs qui sont des maps de deux strings. Le premier désigne le nom du paramètre pour la requête et le deuxième la valeur pour ce paramètre.

|  |  |
| --- | --- |
| **Nom** |  |
| db | Nom de la base de données (généralement telegraf) |
| q | Requête |

**Retour :**

|  |  |
| --- | --- |
| **Type** | **Description** |
| StringBuffer | Réponse de la base de données |

### Fonction getCPUOverTime

**Description :**

Récupère dans la base de données les pourcentages d’utilisation du CPU pour les 6 dernières heures pour le serveur actuel.

**Retour :**

|  |  |
| --- | --- |
| **Type** | **Description** |
| UsageWithTime[] | Utilisations CPU des 6 dernières heures |

### Fonction getRAMOverTime

**Description :**

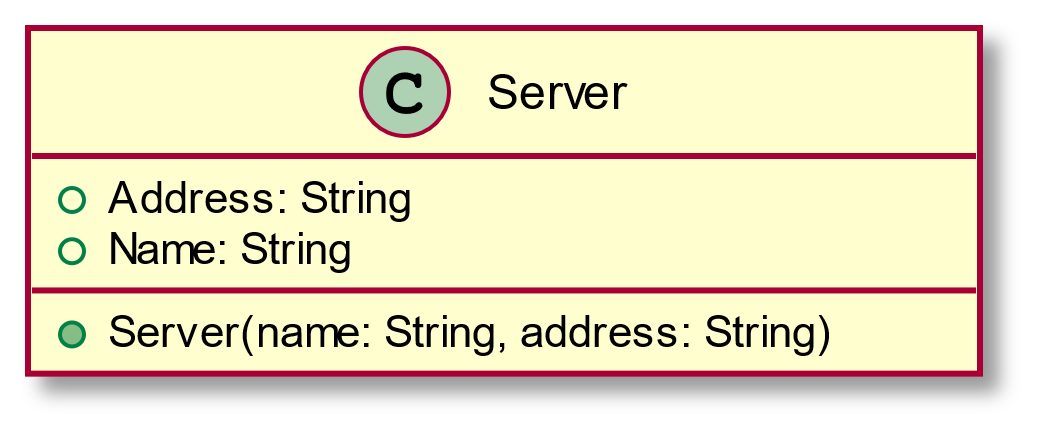
Récupère dans la base de données les pourcentages d’utilisation de la RAM pour les 6 dernières heures pour le serveur actuel.

**Retour :**

|  |  |
| --- | --- |
| **Type** | **Description** |
| UsageWithTime[] | Utilisations RAM des 6 dernières heures |

## Server

Ceci est juste une petite classe pour pouvoir stocker de manière facile les données d’un serveur. Au début une structure simple aurait été voulue mais java, ce langage magnifique ne supporte pas ça.



### Variables membre

|  |  |  |
| --- | --- | --- |
| **Type** | **Nom** | **Description** |
| String | Address | Adresse du serveur |
| String | Name | Nom du serveur |

### Constructeur

**Description :**

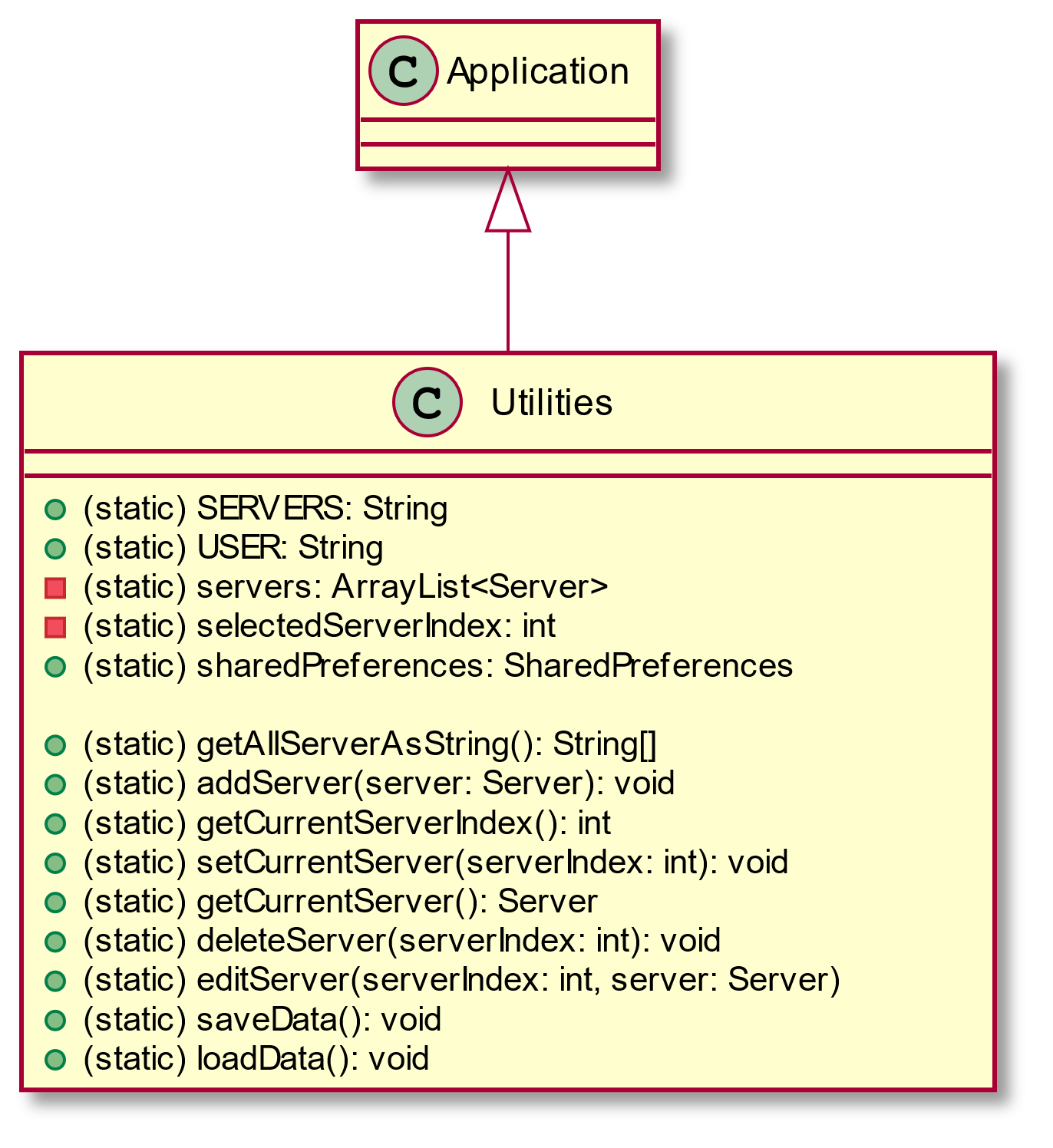
Crée un nouvel objet Serveur et affecte ses variables membre.

**Paramètre :**

|  |  |  |
| --- | --- | --- |
| **Type** | **Nom** | **Description** |
| String | name | Nom du serveur |
| String | address | Adresse du serveur |

## Utilities

Cette class static gère les données qui doivent être passé d’une activité à l’autre et gère également le stockage de donnée dans les shared preferences.



### Variables membres

|  |  |  |
| --- | --- | --- |
| **Type** | **Nom** | **Description** |
| String | SERVERS | String contenant « servers » pour le paramètre des SharedPreferencies |
| String | User | String contenant « USER » pour le paramètre des SharedPreferencies |
| ArrayList<Server> | servers | La liste de tous les serveurs du programme |
| Int | selectedServerIndex | Index du serveur actuel dans la liste servers |
| SharedPreferences | sharedPreferences | S’occupe de la sauvegarde des données |

### Fonction getAllServersAsString

**Description :**

Retourne un tableau de chaînes de caractères avec tous les noms des serveurs.

**Retour :**

|  |  |
| --- | --- |
| **Type** | **Description** |
| String[] | Tableau avec tous les noms des serveurs |

### Fonction addServer

**Description :**

Ajoute un nouveau serveur dans la liste.

**Paramètre :**

|  |  |  |
| --- | --- | --- |
| **Type** | **Nom** | **Description** |
| Server | server | Serveur à ajouter |

### Fonction setCurrentServer

**Description :**

Change le serveur actuel.

**Paramètre :**

|  |  |  |
| --- | --- | --- |
| **Type** | **Nom** | **Description** |
| Int | serverIndex | Index du serveur à choisir |

### Fonction getCurrentServerIndex

**Description :**

Retourne l’index du serveur actuel.

**Retour :**

|  |  |
| --- | --- |
| **Type** | **Description** |
| Int | Index du serveur qui est actuellement sélectionné |

### Fonction getCurrentServer

**Description :**

Retourne le serveur actuel

**Retour :**

|  |  |
| --- | --- |
| **Type** | **Description** |
| Server | Serveur actuel |

### Fonction deleteServer

**Description :**

Change le serveur actuel.

**Paramètre :**

|  |  |  |
| --- | --- | --- |
| **Type** | **Nom** | **Description** |
| Int | serverIndex | Index du serveur à supprimer |

### Fonction editServer

**Description :**

Modifie le serveur.

**Paramètre :**

|  |  |  |
| --- | --- | --- |
| **Type** | **Nom** | **Description** |
| Int | serverIndex | Index du serveur à modifier |
| Server | server | Serveur qui remplacera l’actuel |

### Fonction saveData

**Description :**

Sauvegarde les données des servers dans les shared preferences.

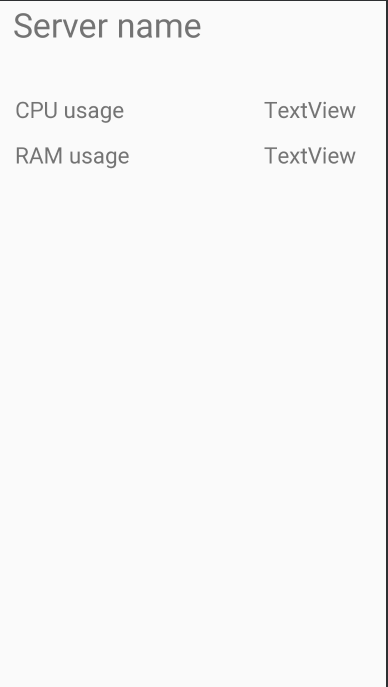
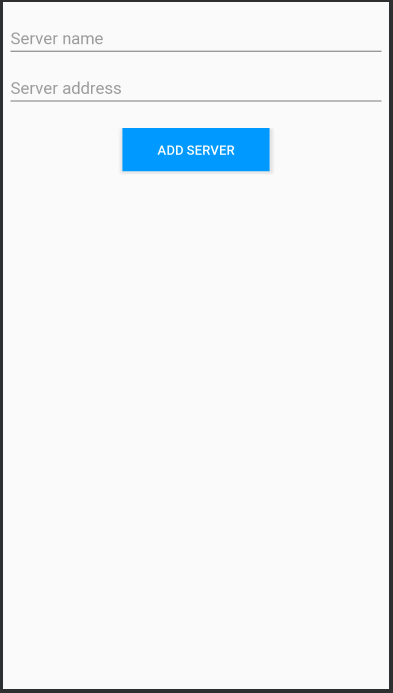
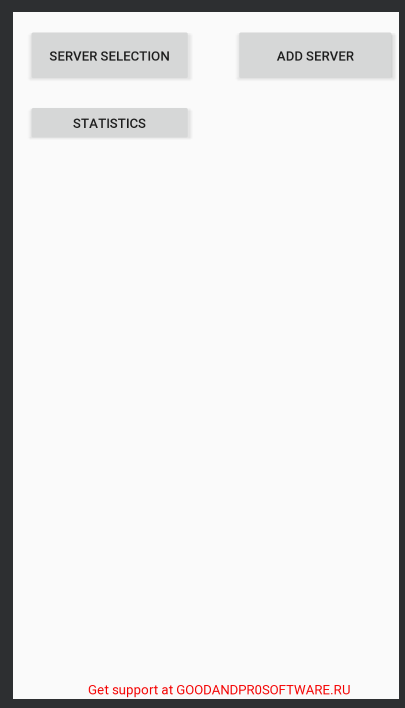
### Fonction loadData

**Description :**

Charge les données des shared preferences dans la liste des serveurs.

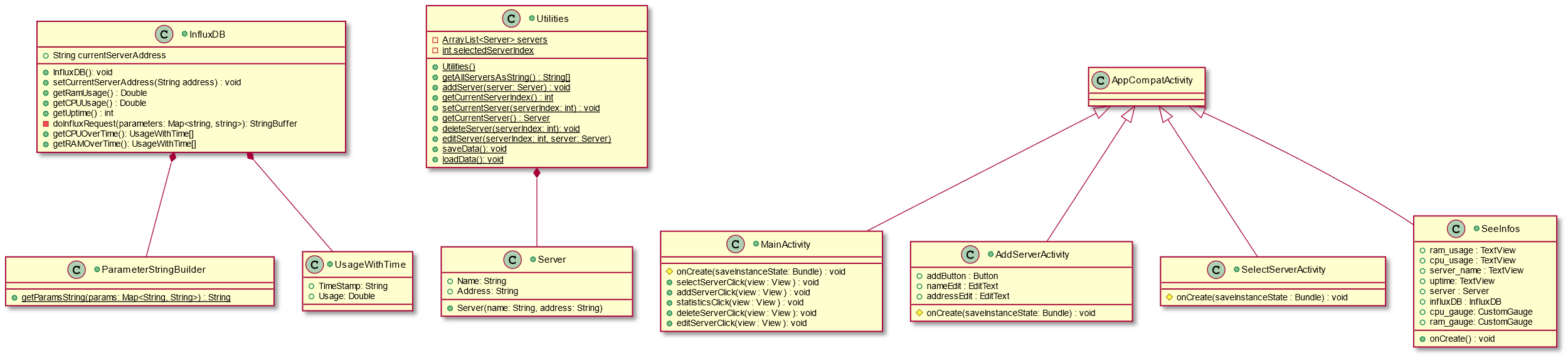
# Version 1.0

## Maquette



## Diagramme de classe

Le diagramme de classe a été réalisé avec l’outil plantuml. Le fichier source et l’image sont dans le dossier « doc/plantuml diagrams ».

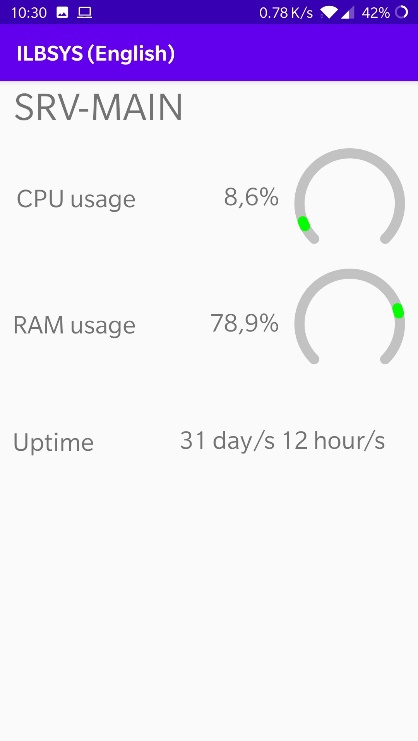


# Sprint 1

## Fonctionnalités

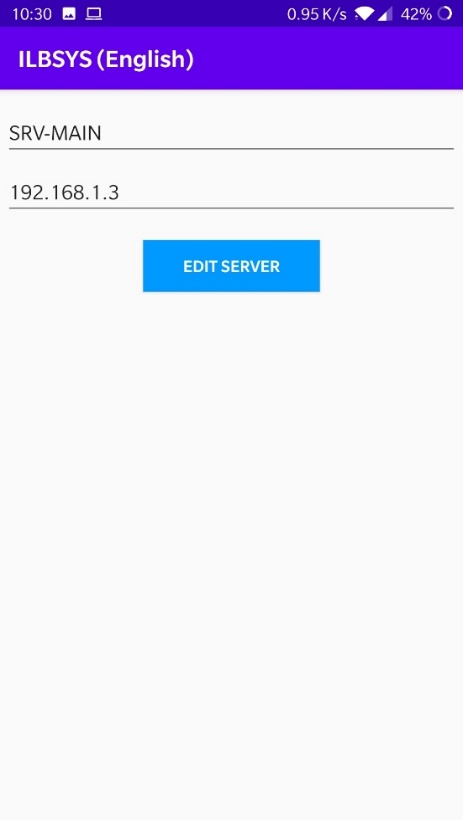
### Uptime

Ajout du temps en ligne depuis le dernier redémarrage est désormais dans la vue des informations serveur. La base de données rend l’uptime en secondes depuis le démarrage.



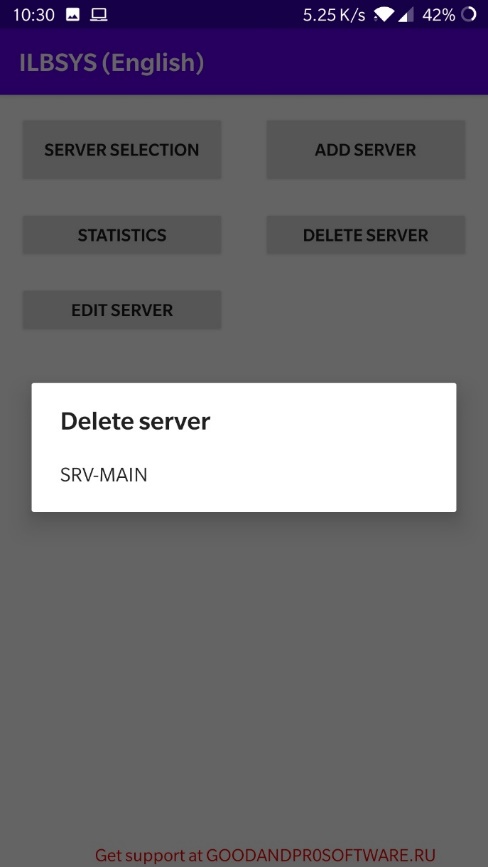
### Modification de serveurs

Modifier les données d’un serveur après coup est très utile dans le cas d’une erreur ou d’un changement d’adresse. Il suffit d’aller sur l’activité de modification de serveur, changer les entrées et ensuite cliquer sur sauvegarder.



### Suppression de serveurs

Ouvre une modale comme pour la sélection d’un serveur mais le supprime quand il est choisi.



## Protocole de tests

Afin de s’assurer que les modifications apportées dans ce sprint soient fonctionnelles, les tests suivants doivent être réalisés et des corrections devront être apportés pour chaque test échoué.

### Uptime

Pour vérifier que le temps en ligne s’affiche correctement et affiche la bonne durée il faut effectuer ces tests :

* Regarder le temps en ligne depuis une source fiable (interface web du serveur, grafana)
* Vérifier si le temps en ligne affichée dans l’application correspond

### Modifications de serveurs

Pour vérifier si la modification de serveurs fonctionne correctement il faut effectuer ces tests :

* Ajouter un serveur avec une adresse fausse
* Retourner dans le menu principal
* Aller dans le menu de modification de serveurs
* Modifier le nom et corriger l’adresse
* Aller dans la liste des serveurs pour vérifier que le nom soit bien modifié
* Aller dans les statistiques pour vérifier le nom et que les données sont chargées, si les données ne sont pas chargées l’adresse n’a pas été correctement modifiée

### Suppression de serveurs

Pour vérifier que la modification des serveurs fonctionne et ne casse pas les listes il faut effectuer les tests suivants :

#### Suppression du premier serveur

Le premier élément d’une liste peu avoir des comportements différents des autres éléments d’une liste, pour cela il faut le vérifier séparément.

* Ajouter plusieurs serveurs
* Supprimer le premier serveur
* Aller dans la sélection des serveurs pour vérifier qu’il ne soit plus là
* Vérifier la sélection du serveur qui est à la première position de la liste après la suppression
* Vérifier qu’on peut afficher les statistiques correctement

#### Suppression d’un serveur au milieu de la liste

Le cas classique est la suppression au milieu de la liste, pour vérifier ce comportements il faut effectuer les tests suivants :

* Ajouter au moins 3 serveurs
* Supprimer l’élément du milieu
* Sélectionner le deuxième serveur
* Vérifier dans l’affichage des statistiques si le bon serveur est choisi

#### Suppression du dernier serveur

Suppression du dernier serveur pourrait provoquer des comportements spécifiques à ce cas d’utilisation. Pour vérifier le bon fonctionnement il faut effectuer les tests suivants :

* Ajouter au moins 2 serveurs
* Supprimer le dernier serveur
* Ajouter un serveur
* Sélectionner le dernier serveur
* Afficher les statistiques pour vérifier que le bon soit choisi

# Sprint 2

## Fonctionnalité

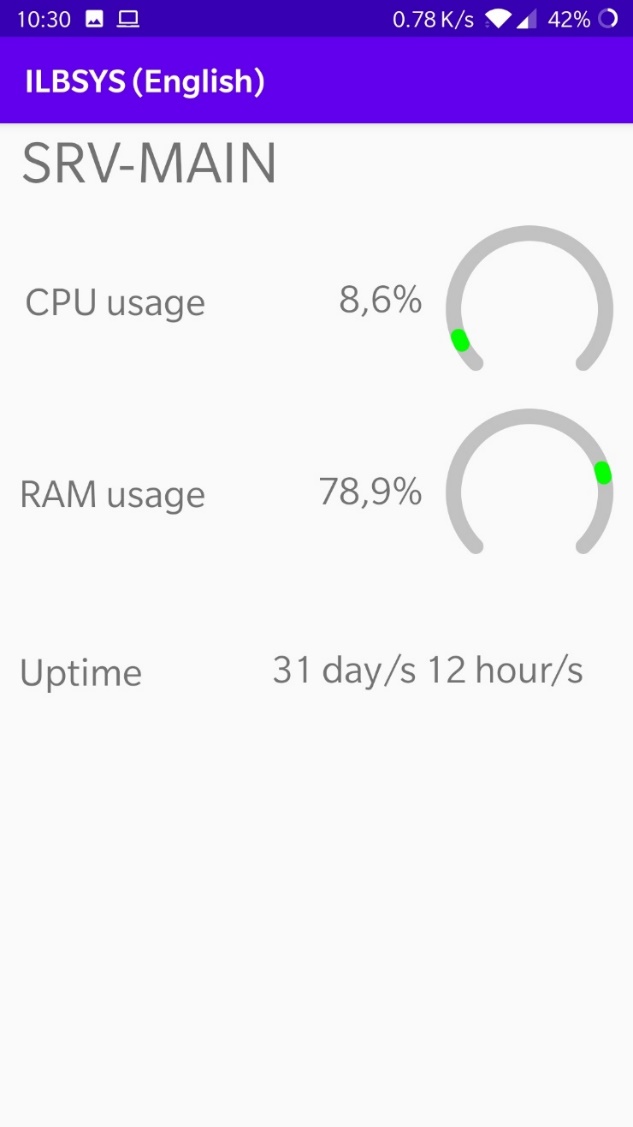
### Accès depuis l’extérieur

Jusqu’à maintenant le serveur InfluxDB était seulement accessible depuis l’intérieur de mon réseau. Une redirection de port a été créer pour pouvoir y accéder depuis l’extérieur. Il s’agit du port 8086 qui est le port par défaut pour InfluxDB.

L’adresse depuis l’extérieur pour faire fonctionner l’application est 185.97.136.203.

### Jauges pour les pourcentages

Pour avoir un affichage plus parlant pour les pourcentages comme pour l’utilisation du CPU et de la RAM des jauges ont été rajoutés. Plusieurs librairies ont dû être testé avant de trouver une qui est compacte et optimal pour les besoins de ce projet.



## Protocole de test

Pour s’assurer que les changements fait durant ce sprint fonctionnent et n’altèrent pas avec les autres fonctionnalités, il faut refaire les tests de l’ancien sprint et ensuite les tests suivants :

### Accès depuis l’extérieur

Pour vérifier que l’accès depuis l’extérieur fonctionne correctement il faut avoir à disposition un téléphone physique qui possède une connexion mobile. Ensuite il faut faire les tests suivants :

* Vérifier que le téléphone n’est pas connecté au wifi
* Ajouter le serveur avec l’adresse 185.97.136.203
* Sélectionner ce serveur
* Aller dans l’affichage des statistiques et vérifier qu’elles chargent correctement

### Jauges pour les pourcentages

Pour la vérification des jauges il faut effectuer les tests suivants :

* Ajouter un serveur
* Aller dans l’affichage des statistiques
* Vérifier si la gauge correspond au pourcentage qui est noté à côté

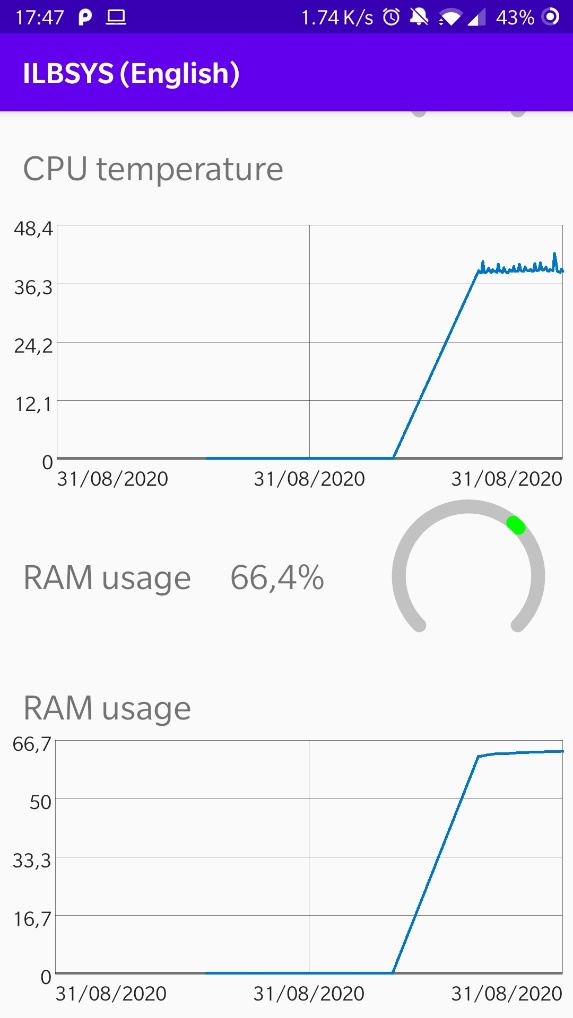
# Sprint 3

## Issue #2

Correction de l’erreur que l’application crashait si aucun serveur était dans les shared preferencies.

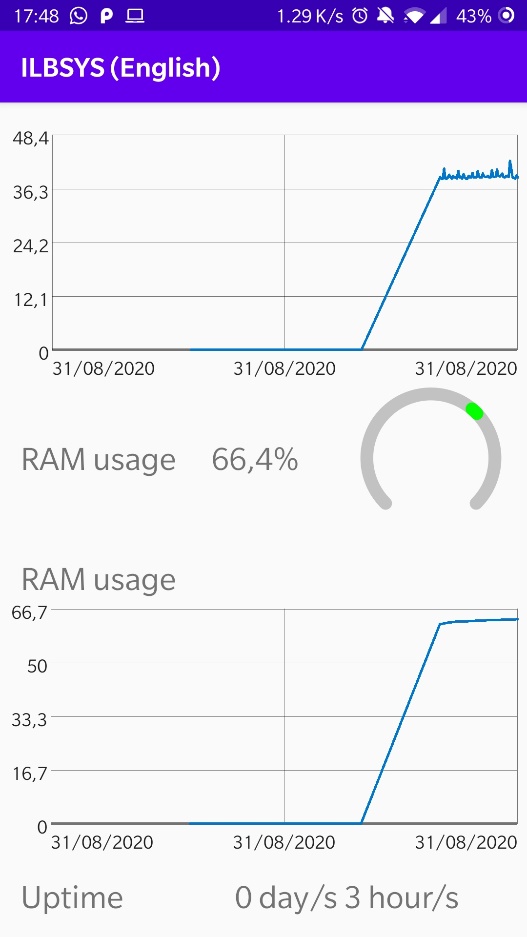
<https://github.com/ILikeBananas/ILBSYS_android/issues/2>

## Graphique pour la température du processeur

Un graphique a été ajouté sur la page des informations qui affiche les températures du processeur sur les derniers 6 heures. Pour cela la librairie « [jjoe64](https://github.com/jjoe64)» a été utilisé, celle-ci fonctionne bien même s’il n’existait pas de la documentation pour tout ce qui a été utilisé.

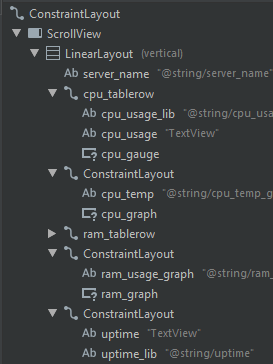
## Graphique utilisation RAM

Comme pour le graphique des températures CPU, la librairie de [jjoe64](https://github.com/jjoe64)a été utilisée.



## Réorganisation de la page d’informations

L’affichage et l’organisation des éléments sur la page d’informations ont été refaits à neuf pour permettre un affichage plus propre, moins de valeurs fixe et une gestion du mode paysage correcte. Une scrollview a été utilisée pour permettre d’afficher plus de données que la hauteur (ou largeur en mode paysage) du périphérique.



## Protocol de test

Pour s’assurer que les modifications de se sprint fonctionnent et qu’elles n’influencent pas les autres fonctionnalités. Il faut d’abord effectuer tous les derniers protocoles de tests et ensuite les tests suivants.

### Issue #2

Pour vérifier que l’application ne plante plus au démarrage si aucun serveur est dans la liste des serveurs il faut :

* Supprimer tous les serveurs de la liste
* Fermer l’application
* Fermer l’application dans le gestionnaire d’application d’Android
* Rouvrir l’application

### Graphique température CPU

Pour vérifier que le graphique de température fonctionne correctement il faut :

* Ajouter un serveur fonctionnel
* Aller sur la page des statistiques
* Vérifier que le graphique affiche des valeurs correctes, si possibles en ayant une source sûre pour comparer

### Graphique utilisation de RAM

Pour vérifier que le graphique d’utilisation de RAM fonctionne correctement il faut :

* Ajouter un serveur fonctionnel
* Aller sur la page des statistiques
* Vérifier que le graphique affiche des valeurs correctes, si possibles en ayant une source sûre pour comparer