INSTITUT UNIVERSITAIRE DE TECHNOLOGIE
de BLAGNAC –TOULOUSE II
Département Informatique
1, place Georges Brassens –BP 73-
31703 BLAGNAC Cedex

IRIT
118 Route de Narbonne
F-31062 TOULOUSE CEDEX 9

Dossier d'Analyse et de Conception

AppliConso

Application d'estimation de consommation energétique

Projet Tutoré

Destinataires :
M. Da Costa
Mme Stolf

Réalisé par :
Emmanuel Tocabens
Charles Thiebaud
Clément Lopez

Année 2016/2017

Sommaire

Introduction

L'Institut de Recherche en Informatique de Toulouse (IRIT) est une unité de recherche mixte travaillant selon 4 grands axes :

- *Informatique pour la santé et l'autonomie*
- *Masses de données et calcul*
- *Systèmes sociotechniques ambiants*
- *Systèmes embarqués critiques*

L'IRIT est le plus grand institut de recherche français en informatique. 690 personnes y travaillent en tant qu'enseignants-chercheurs, chercheurs, doctorants, post-doctorants ou encore personnel administratif et technique.

Notre client et notre superviseur travaillent à l'IRIT en tant que chercheur et enseignant-chercheur. M. Da Costa a créé ce projet dans le cadre d'un développement durable, écologique, dans le secteur de l'informatique.

1 Organisation actuelle et future

La surveillance de la consommation énergétique des ordinateurs et des processus est un enjeu majeur. Il est en effet compliqué d'obtenir des informations fiables sur ces consommations.

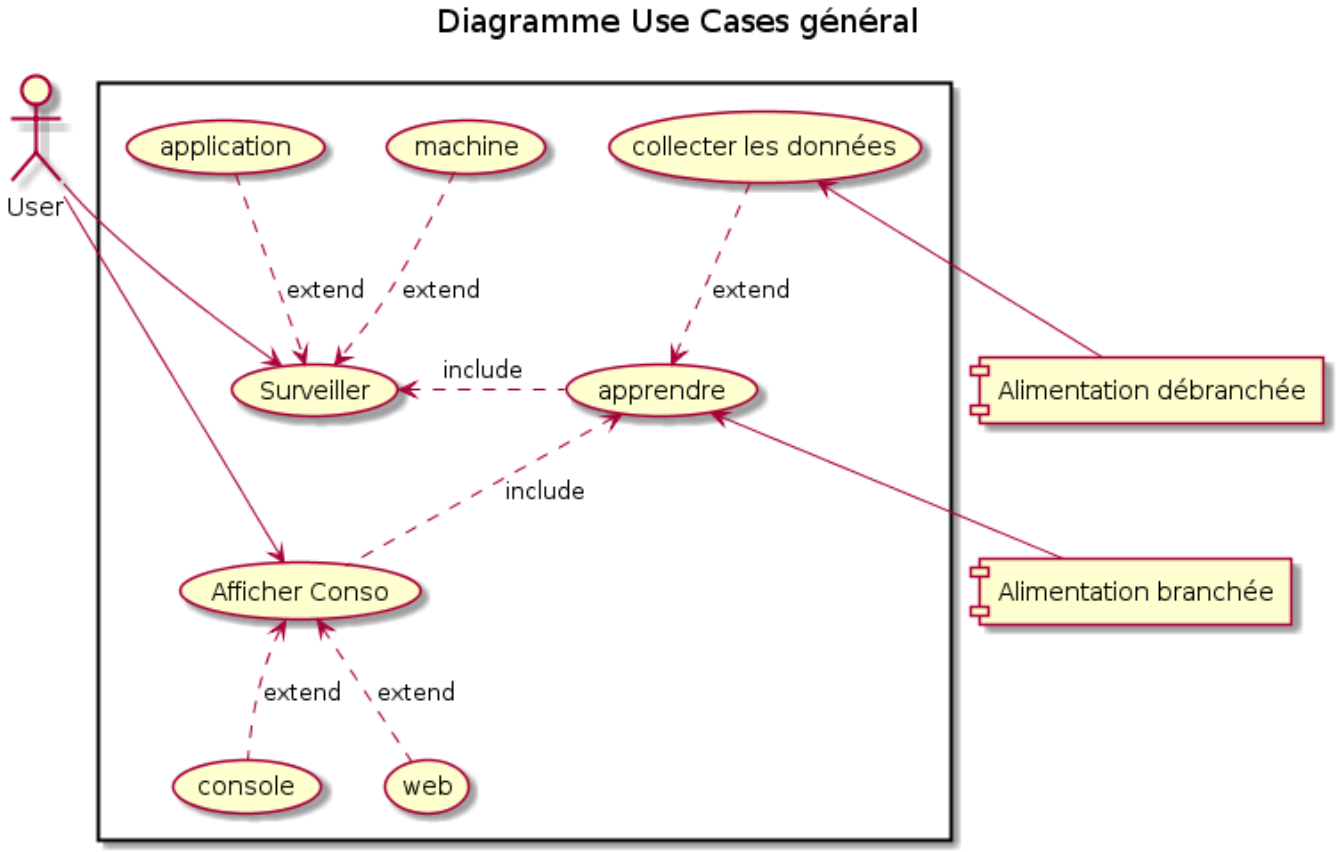
Le projet AppliConso a pour but d'automatiser et de simplifier l'accès à ces données en affichant directement à l'utilisateur et en temps réel la consommation de son système et de ses applications.

Pour faciliter l'utilisation de ce logiciel, il sera donc Open-Source et développer en Python3.5, langage portable.

De plus afin d'économiser l'énergie quand l'ordinateur n'est pas branché, les phases gourmandes en énergie (apprentissage par tensorflow) seront réalisées une foi l'ordinateur rebranché.

2 Analyse

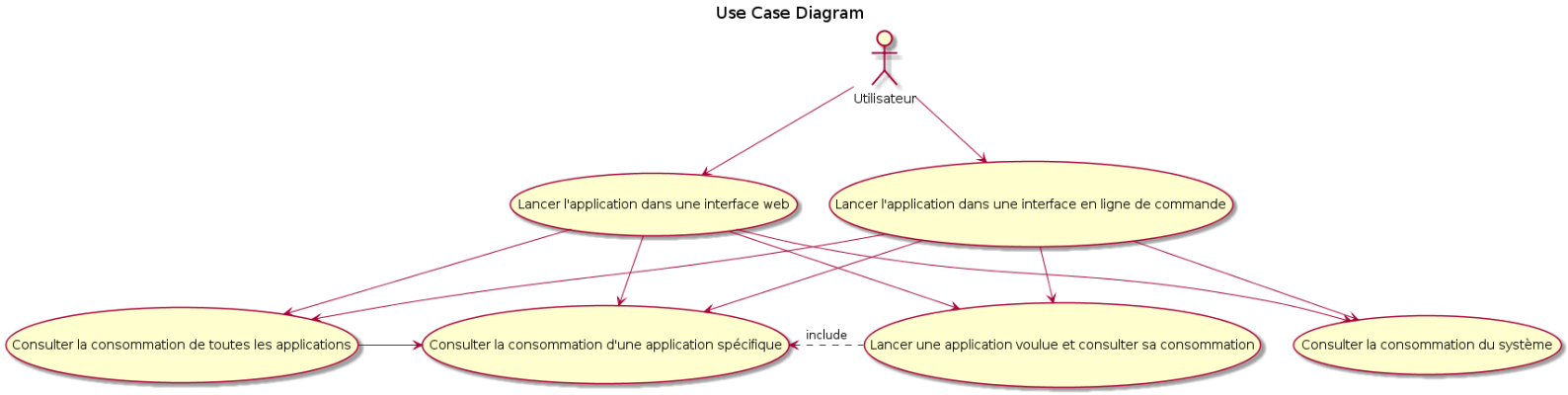
2.1 Diagramme UC Général



AppliConso est une application qui peut se résume en 2 parties : une partie d'interaction avec l'utilisateur et une partie d'apprentissage et de collecte de données.

L'utilisateur peut demander de surveiller la consommation du système entier ou d'une application en particulier puis d'en afficher la consommation qui pourra être calculée/prédite par AppliConso à partir des données collectées et apprises.

2.2 Diagramme UC pour l'interaction utilisateur et Cas d'utilisation



AppliConso est une application qui a pour fonction de permettre à l'utilisateur de consulter la consommation d'énergie sur sa machine.

Cependant, en fonction des données voulues par l'utilisateur et de l'interface d'affichage voulu, plusieurs cas d'utilisations se distinguent.

- Cas n°1 : L'utilisateur veut un affichage dans une interface en ligne de commande

- Scénario 1.1: L'utilisateur veut uniquement la consommation du système.

Dans ce cas, AppliConso va récupérer et stocker les données lorsque la machine n'est pas branchée sur secteur. Ensuite, AppliConso va prédire/calculer, en fonction des données stockées, la consommation du système lorsque la machine est branchée. Les données, ainsi calculées sont affichées à l'utilisateur dans l'interface voulue.

- Scénario 1.2: L'utilisateur veut regarder la consommation d'une application spécifique

Dans ce cas, l'utilisateur va renseigner le PID de l'application dont il veut la consommation. Ensuite, l'apprentissage des données et le calcul de la consommation du système global se fait comme lors du scénario 1.1 sauf qu'au lieu d'afficher la consommation du système on va récupérer la part de l'application en question dans la consommation globale et ainsi afficher la consommation de l'application dans l'interface voulue.

- Scénario 1.3: L'utilisateur veut lancer une application pour consulter sa consommation

Dans ce cas, l'utilisateur va renseigner le chemin d'accès de l'application. AppliConso va ensuite lancer cette application en récupérant son PID pour être suivre ensuite la même procédure que lors du scénario 1.2

- Scénario 1.4: L'utilisateur veut consulter la consommation de toutes les applications ouvertes

Dans ce cas, AppliConso récupère la liste de toutes les applications en cours d'exécution et leur PID. Après l'apprentissage des données et le calcul de la consommation du système, AppliConso va récupérer la part dans la consommation du système de chacune des applications ouvertes, ce qui va permettre de calculer la consommation de chaque application, et ainsi, de pouvoir afficher le résultat dans l'interface voulue.

- Sous-scénario 1.4.1: L'utilisateur veut afficher la consommation d'une application

Il a la possibilité de sélectionner l'application voulue parmi celles affichées, et d'ainsi n'afficher que la consommation de cette application (c'est une passerelle vers le scénario 1.2).

- Cas n°2: L'utilisateur veut un affichage dans une interface web

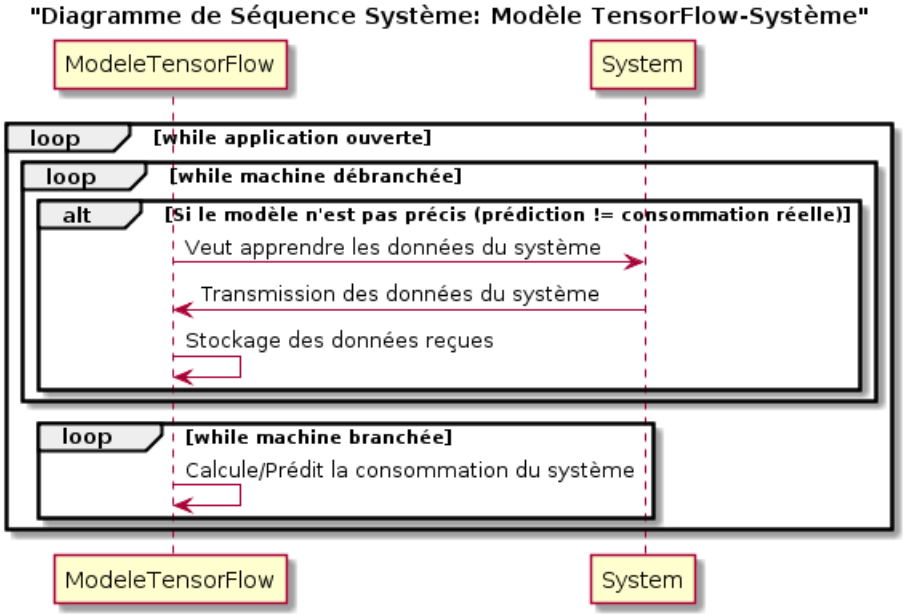
Dans ce cas de figure, les scénarios sont en tous points identiques à ceux du cas n°1 (les données à afficher seront les mêmes) à l'exception de l'affichage qui se réalisera dans une interface web au lieu d'une interface en ligne de commande.

Variante non-retenue : variante du scénario 1.2 (et 2.2)

Une autre idée pour trouver l'application dont l'utilisateur veut la consommation aurait été de ne pas recevoir le PID mais le nom de l'application en question. Cette solution, quoique plus intuitive pour l'utilisateur, n'a pas été retenue. En effet, cela posait problème dans le cas où il y aurait 2 exécutions simultanées de la même application : laquelle doit-être prise en compte ? La solution de recevoir seulement le PID a donc été retenue, après consultation du client.

2.3 Diagrammes de Séquence Système

2.3.1 DSS d'apprentissage et de collecte de données



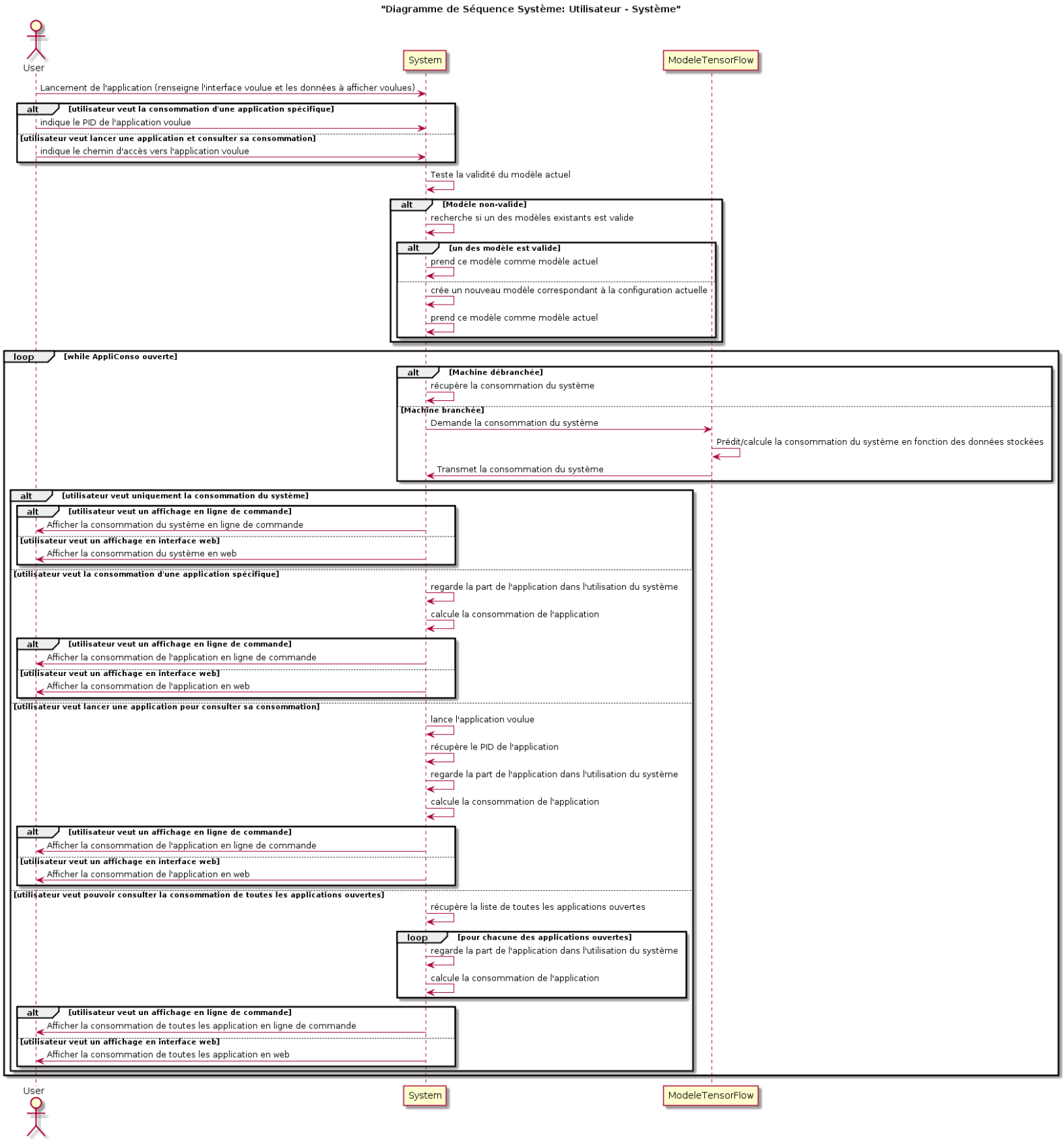
Le modèle TensorFlow d'AppliConso peut se résumer en deux étapes :

Si la machine est débranchée, on récupère un maximum de données utiles à l'évaluation de la consommation. Cela, tant que le modèle n'est pas précis, c'est-à-dire tant que la prédiction que l'on peut réaliser sur la consommation à partir des données collectées n'est pas assez proche de la consommation réelle (connue en récupérant les données de la batterie).

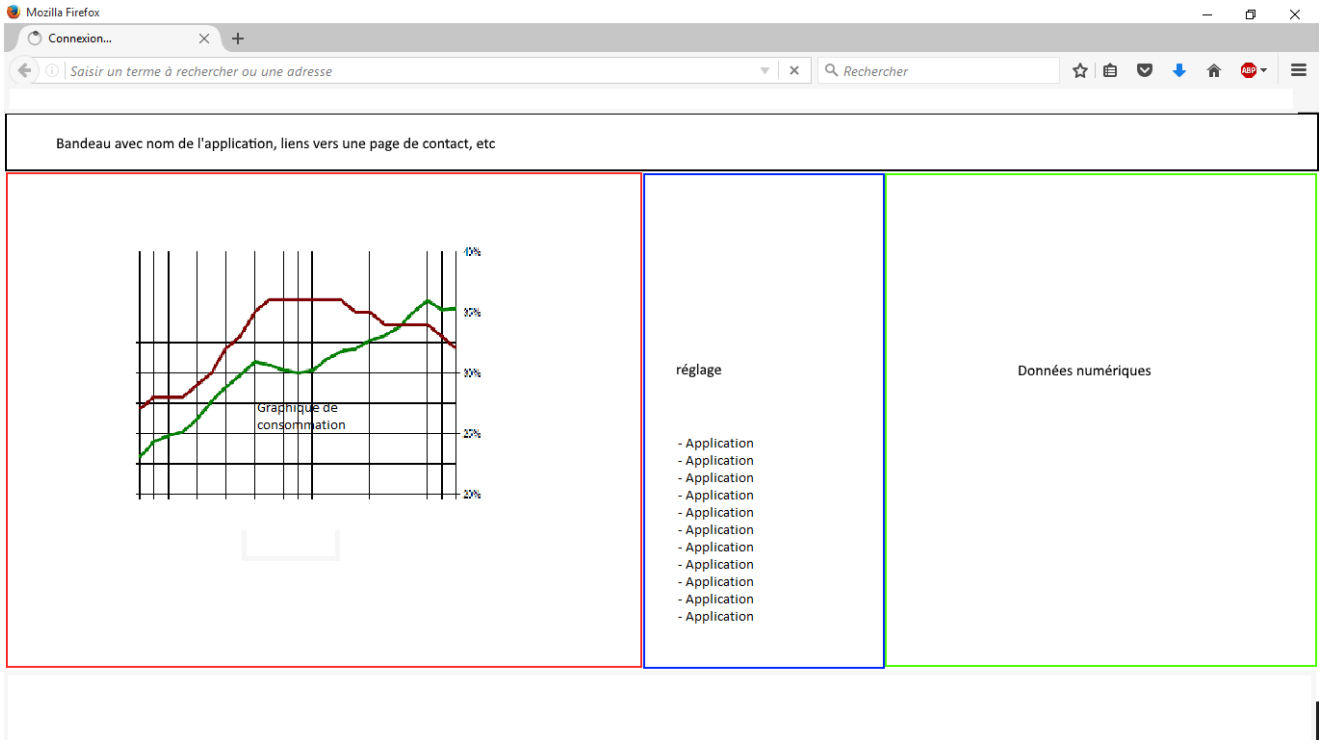
Si la machine est branchée en revanche, les données que l'on peut récupérer sur la consommation réelle sont faussées du fait de la recharge de la batterie, on va donc chercher à prédire la consommation à partir des données stockées et des données de la configuration actuelle de la machine récupérées sur les capteurs (ex : luminosité, ect…).

2.3.2 DSS interaction utilisateur

Ce diagramme reprend les différents cas et scénarios d'utilisation vus précédemment en en détaillant le contenu. On remarque également qu'à chaque lancement, AppliConso teste la validité de son modèle (le nombre de capteurs pouvant différer en fonction du système d'exploitation, l'utilisateur pouvant ajouter/supprimer des capteurs, ect…) certaines données pouvant être manquantes ou de trop, dans ce cas on recherche si un modèle créé antérieurement peut convenir, sinon on crée un nouveau modèle.


"Diagramme de Séquence Système: Utilisateur - Système"
User
System
ModeleTensorFlow
Lancement de l'application (renseigne l'interface voulue et les données à afficher voulues)
alt [utilisateur veut la consommation d'une application spécifique]
indique le PID de l'application voulue
[utilisateur veut lancer une application et consulter sa consommation]
indique le chemin d'accès vers l'application voulue
Teste la validité du modèle actuel
alt [Modèle non-valide]
recherche si un des modèles existants est valide
alt [un des modèle est valide]
prend ce modèle comme modèle actuel
crée un nouveau modèle correspondant à la configuration actuelle
prend ce modèle comme modèle actuel
loop [while AppliConso ouverte]
alt [Machine débranchée]
récupère la consommation du système
[Machine branchée]
Demande la consommation du système
Prédit/calcule la consommation du système en fonction des données stockées
Transmet la consommation du système
alt [utilisateur veut uniquement la consommation du système]
alt [utilisateur veut un affichage en ligne de commande]
Afficher la consommation du système en ligne de commande
[utilisateur veut un affichage en interface web]
Afficher la consommation du système en web
[utilisateur veut la consommation d'une application spécifique]
regarde la part de l'application dans l'utilisation du système
calcule la consommation de l'application
alt [utilisateur veut un affichage en ligne de commande]
Afficher la consommation de l'application en ligne de commande
[utilisateur veut un affichage en interface web]
Afficher la consommation de l'application en web
[utilisateur veut lancer une application pour consulter sa consommation]
lance l'application voulue
récupère le PID de l'application
regarde la part de l'application dans l'utilisation du système
calcule la consommation de l'application
alt [utilisateur veut un affichage en ligne de commande]
Afficher la consommation de l'application en ligne de commande
[utilisateur veut un affichage en interface web]
Afficher la consommation de l'application en web
[utilisateur veut pouvoir consulter la consommation de toutes les applications ouvertes]
récupère la liste de toutes les applications ouvertes
loop [pour chacune des applications ouvertes]
regarde la part de l'application dans l'utilisation du système
calcule la consommation de l'application
alt [utilisateur veut un affichage en ligne de commande]
Afficher la consommation de toutes les application en ligne de commande
[utilisateur veut un affichage en interface web]
Afficher la consommation de toutes les application en web
User
System
ModeleTensorFlow

2.4 Maquettes d'écrans et d'états imprimés



Sur la gauche, un graphique dynamique qui affiche la consommation d'une ou plusieurs applications
Au centre un panneau de réglage avec options, etc.
Sur la droite, des valeurs numériques supplémentaires (précision du modèle, etc)

3 Conception

Notre application fonctionne en multithread, il est par conséquent de représenter toutes les étapes sur un même diagramme.

3.1 DS : Cas général

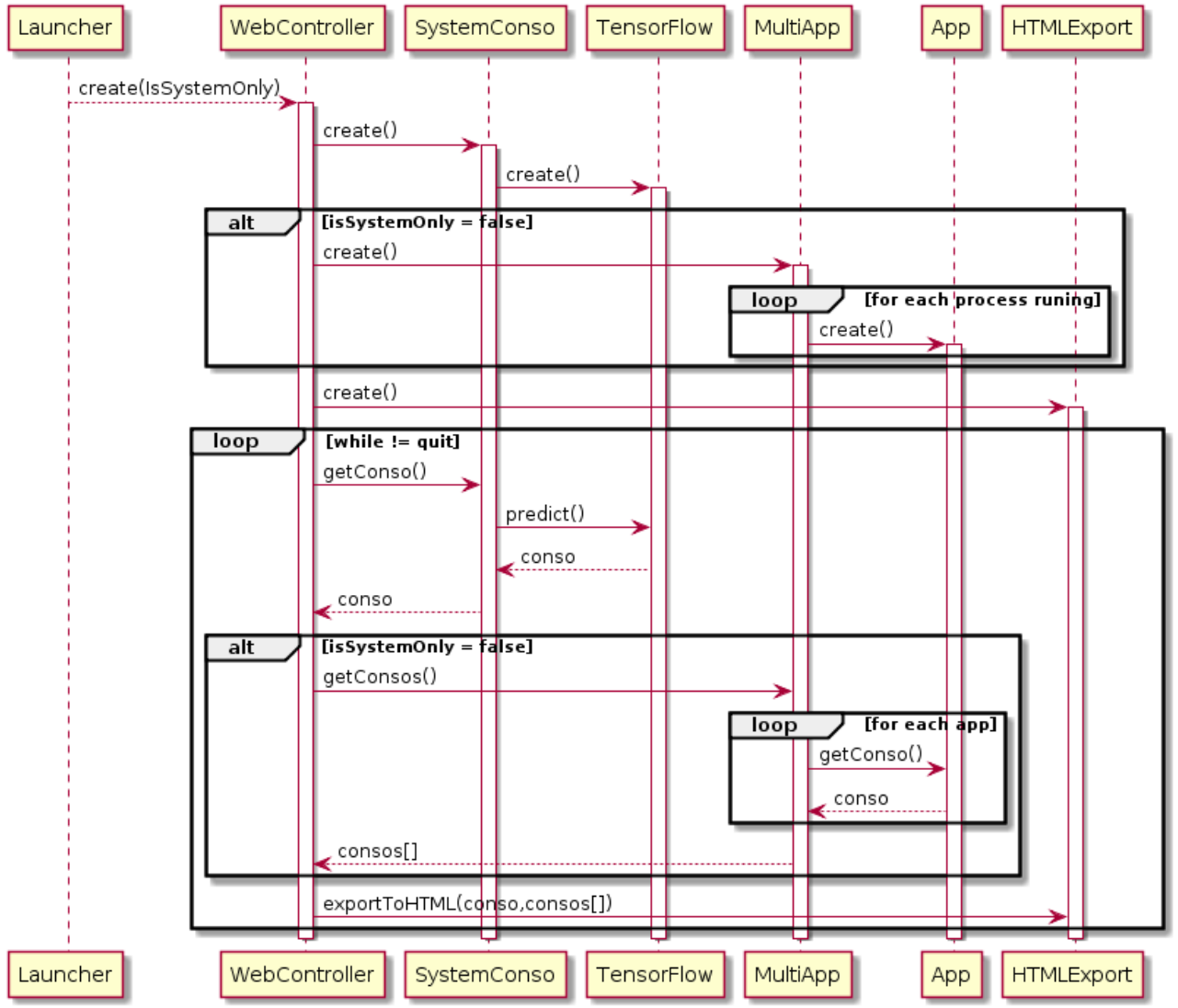


Diagramme 1: DSS General

Le diagramme ci-dessus représente le fonctionnement général du système dans le cas où l'utilisateur lance l'application en web et en surveillant tous les processus. (Le fonctionnement du programme en console est similaire exception faite du la partie HTML qui n'existe pas)

La partie apprentissage de TensorFlow et son fonctionnement en général est ici omis car il fonctionne sur un thread différend.

Le Launcher créer le WebController qui lui-même crée SystemConso, MultiApp(si l'utilisateur l'a demandé) et HTMLExport.

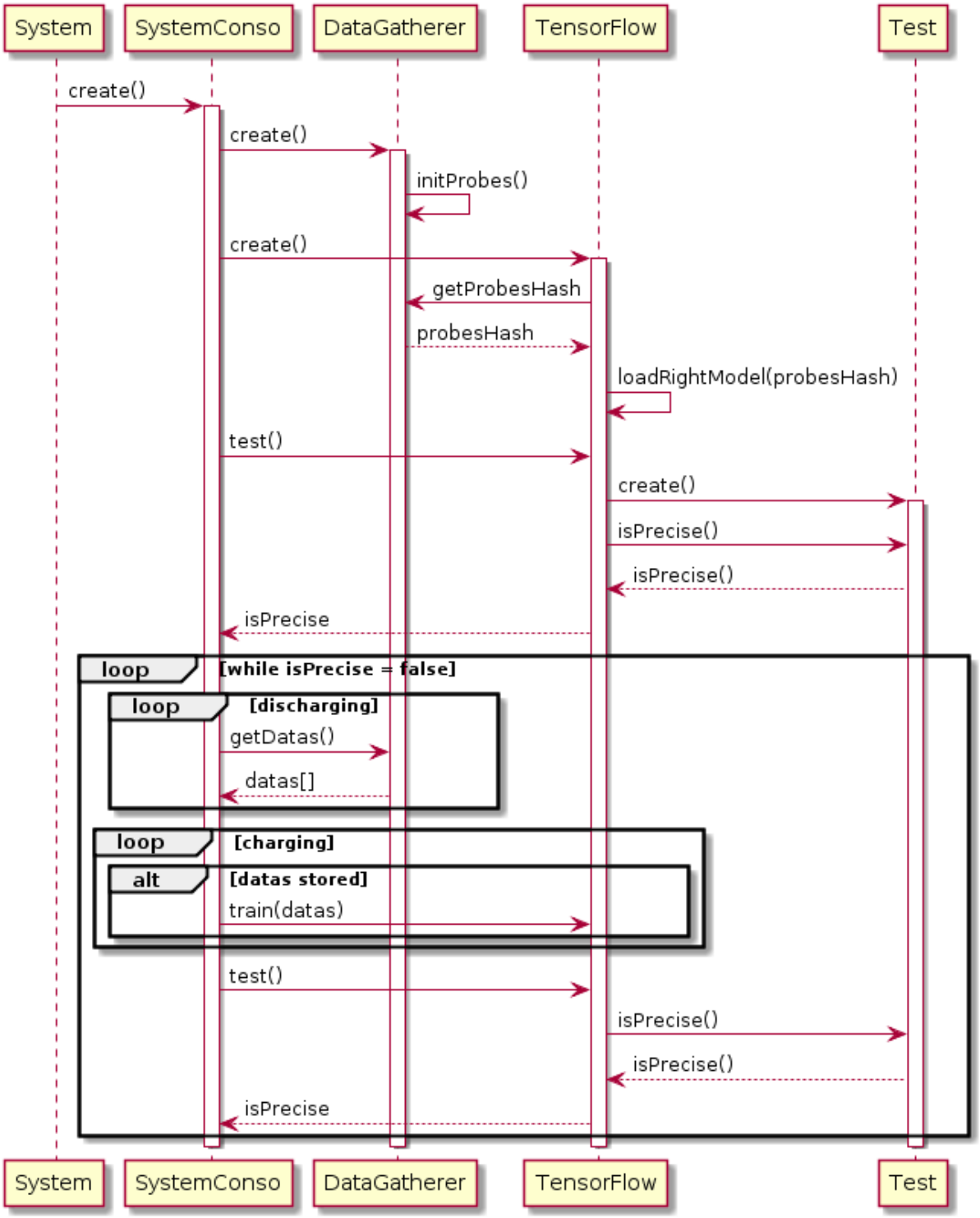
SystemConso crée TensorFlow qui lui permettra de connaître sa consommation.

MultiApp crée une App pour chaque processus actif du système et les ajoute à sa liste.

Tant que l'utilisateur ne quitte pas l'application, le WebController récupère la consommation de SystemConso et si l'utilisateur a demandé de surveiller tous les processus, il récupère aussi la consommation de chaque App sous forme de tableau.

Il donne ensuite ces données au HTMLExport afin de créer une page HTML qui formate les données pour pouvoir les afficher à l'utilisateur sous forme de page web.

3.2 DS :Apprentissage de TensorFlow



Cette séquence se déroule sur un thread qui lui est propre. Elle représente le fonctionnement de l'apprentissage de TensorFlow.

A la création de SystemConso, DataGatherer et TensorFlow sont créés.

TensorFlow récupère le modèle associé à la combinaison des sondes mises à disposition et s'il n'existe pas le crée.

Il crée alors Test qui lui permet de tester la précision de son modèle.

Tant que le modèle n'est pas précis, le système collecte les données pendant la phase de décharge ou apprend ces données collectées pendant la phase de charge.

L'apprentissage effectif des données ne se fait que pendant la charge car ceci est gourmand en énergie et il est nécessaire de ne pas raccourcir la durée de la batterie pour l'utilisateur.

3.3 SNI

L'application n'a pas d'interface à proprement parler. En effet, tout passe par les arguments au lancement du programme, arguments donnés au launcher puis transmis aux différents contrôleurs.

Pour la partie web, il faut se référer aux maquettes présentes dans la partie analyse.

3.4 Architecture MVC

Modèle : -TensorFlow

-Sondes (implémentables par l'utilisateur)

- App (représente un processus)

Contrôleurs : - WebController

- ConsoleController

- MultiApp (gère toutes les App)

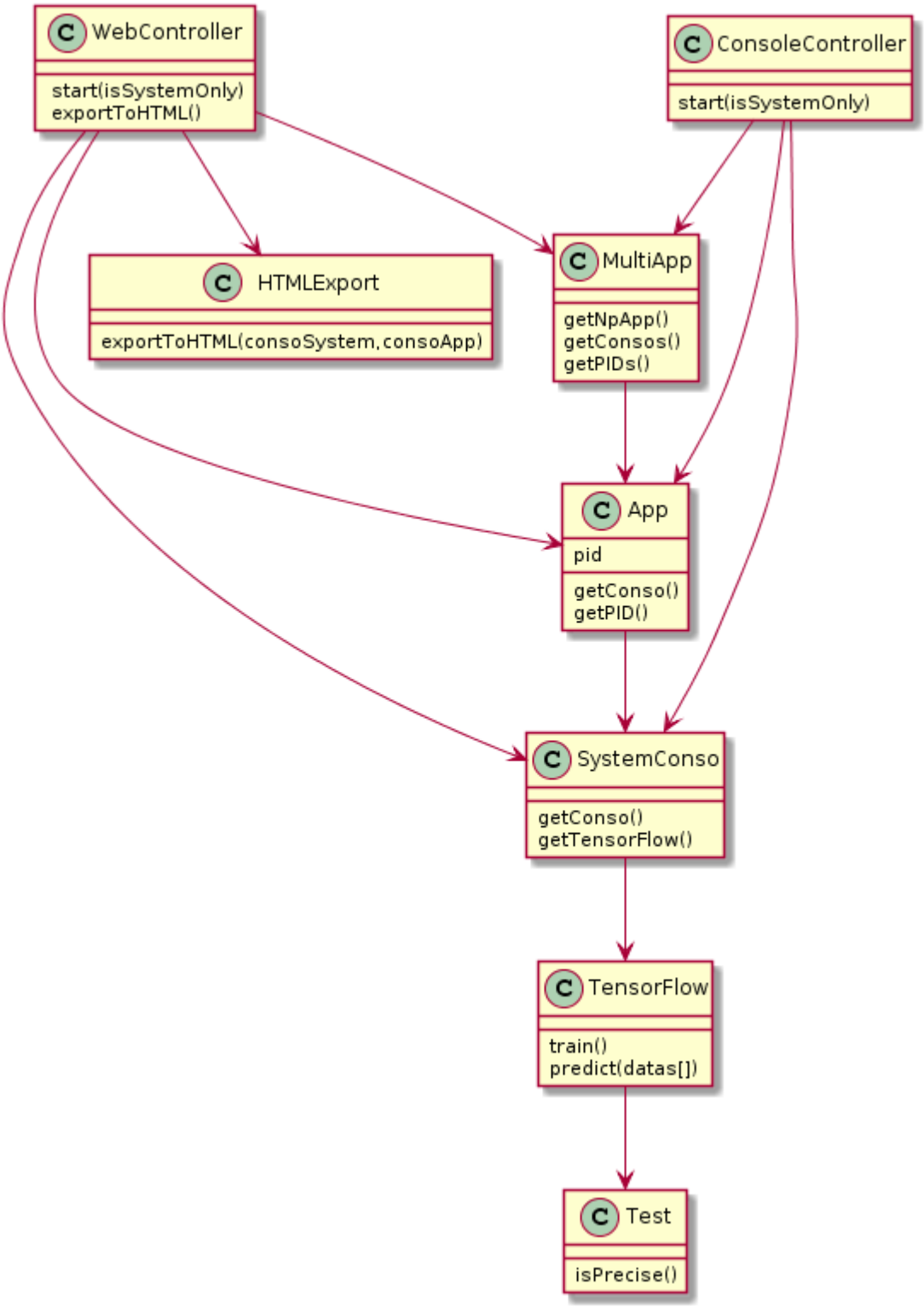
- SystemeConso(récupère la consommation générale du système)

- Test (permet de tester la précision de TensorFlow)

Vue : -HTMLExport

Il n'y a pas de partie vue pour le cas console car c'est le ConsoleController qui s'en charge

3.5 Diagramme des classes contrôleur



Ici App et TensorFlow ne sont pas des classes contrôleur mais est indispensable dans ce schéma.

a) WebController

**start(isSystemOnly) :*

Permet de lancer le programme et précisant si on surveille le système seulement ou les processus aussi

b) ConsoleController :

**start(isSystemOnly) :*

Permet de lancer le programme et précisant si on surveille le système seulement ou les processus aussi

c) HTMLExport

**exportToHtml(ConsoSystem,ConsoApps) :*

Permet de générer la page html avec les données

d) MultiApp :

**getConsos() :*

Permet d'obtenir un tableau qui regroupe les consommations des processus actifs

**getPIDs() ;*

Permet d'obtenir un tableau qui regroupe les PIDs des processus actifs

e) Test :

**isPrecise() :*

Permet de tester si le modèle est précis

4 Spécificités techniques

Le client impose l'utilisation de certains outils :

- *Librairie TensorFlow*
- Python NoteBook
- Dépôt GIT
- Utilisation de logiciels libres de droit

5 Prototype

Description de la V0 :

Le programme est capable :

- *D'estimer la consommation et de collecter des données en temps réel lorsque la machine est débranchée*
- *De traiter les données collectées, puis d'estimer la consommation en temps réel lorsque la machine est branchée*
- *D'admettre de nouveaux capteurs en les ajoutant dans les dossiers WindowsProbes et LinuxProbes, en leur donnant un nom adéquat*
- *De sauvegarder un modèle tensorflow*

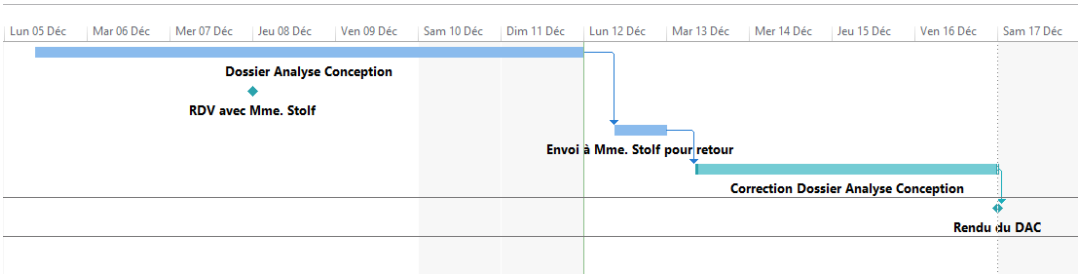
6 Plan d'assurance qualité

Afin de vérifier la qualité et la précision du logiciel fourni, nous allons utiliser deux méthodes :

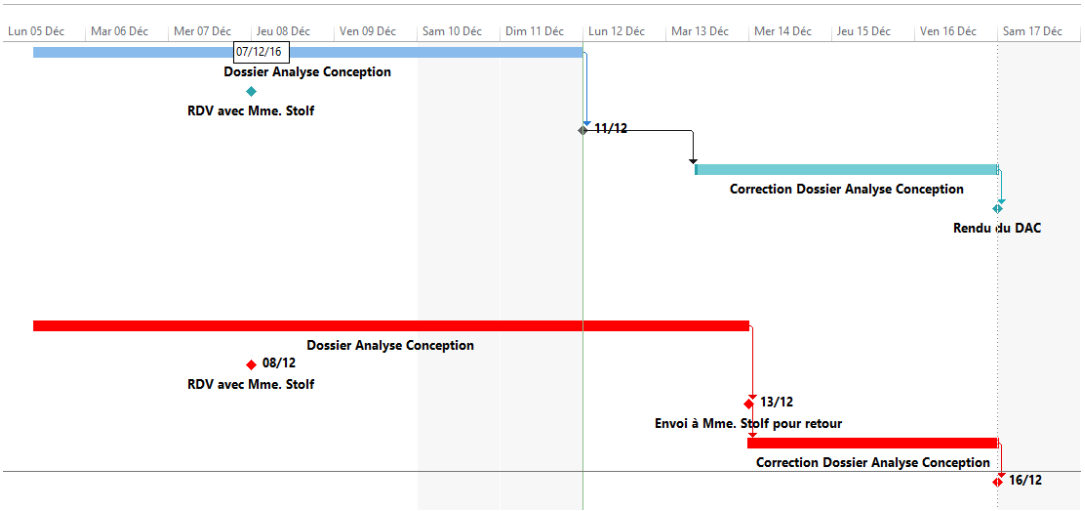
- *Le logiciel sera capable de s'autotester, avec des valeurs prises aléatoirement lors de la collecte des données, gardées pour les tests, afin de déterminer sa précision.*
- *Nous allons utiliser un voltmètre fourni par M. Da Costa afin de tester si les valeurs obtenues par le logiciel sont identiques ou semblables à celle relevées par le voltmètre*

7 Organisation mise en œuvre dans l'étape DAC

Nous avons eu rendez-vous avec Mme Stolph le 8 Décembre à l'IUT. Nous avons, suite à ce rendez-vous envoyé un mail de compte-rendu et y avons demandé des informations quant à la conception dans le cadre du DAC.



Gant prévisionnel



Gant réel

Conclusion

Grâce à cette architecture, cette application est capable de fournir à l'utilisateur une estimation précise de la consommation en énergie de son système et/ou de ses processus en temps réel. De plus, l'utilisation de cette application est simplifiée pour l'utilisateur et à un affichage intuitif.

Annexes

Annexe1 : Mail faisant suite à la réunion avec Mme. Stolf

Bonjour,
Suite à notre rendez-vous avec Mme. Stolf, nous avons plusieurs questions concernant la conception de l'application :

1) Lorsque le nombre de capteurs change (suite à l'ajout ou la suppression d'un capteur), le modèle TensorFlow n'est plus valide (ceci pose aussi problème si l'utilisateur peut choisir au lancement de l'application les capteurs qui doivent être activés). Doit-on prévenir l'utilisateur qu'il a besoin de repasser par la phase d'apprentissage (rapide à implémenter) ou doit-on récupérer les valeurs déjà apprises ? (Long a implémenter si faisable, peut-être long à l'exécution)

2) Pour que notre modèle soit précis , les valeurs des différents capteurs ont besoin d'avoir une amplitude élevée lors de l'apprentissage (ex: utilisation cpu: 0% => 100%).Doit-on simplement prévenir l'utilisateur qu'il doit les faire varier ou doit-on forcer ces valeurs a varier (en simulant une appli gourmade par exemple) sachant que cette variation ne se fera qu'une seule foi (sauf changement du nombre de capteurs) mais pendant la phase de récupération de valeurs (phase débranchée) et risque donc d'être gourmande en énergie

3) Afin de surveiller une application particulière, il nous faut récupérer son pid. Nous avons plusieurs propositions :

-l'utilisateur nous donne le nom de l'application (ou le lien vers l'exécutable) à lancer et notre application se charge de lancer ce programme (au lancement de l'application)

-l'utilisateur nous donne le pid de l'application à surveiller (au lancement du programme)

-nous affichons l'ensemble des processus actifs (au lancement du programme) et l'utilisateur sélectionne l'application à surveiller

4)Vous nous avez demandé de pouvoir surveiller la consommation de tous les processus en même temps et de les afficher. Ceci risque de consommer beaucoup d'énergie (non testé) car nous devons tous les parcourir et les traiter un à un. Gardons-nous cette fonctionnalité ou limitons nous à la surveillance d'un processus et du système (consommation globale du système) ?

5)Que doit faire le site web? S'agit-il d'une interface servant seulement d'affichage ou doit-elle permettre de contrôler l'application ? Si oui, comment peut-on faire communiquer le Python et le web ?

6)Pour évaluer la précision de notre modèle TensorFlow, nous pensons récupérer un petit nombre de données lors de la phase de récolte de celle-ci et les mettre de côté afin de tester si les valeurs prédites par TensorFlow sont correctes (à une précision près). Que pensez-vous de cette méthode ? Avez-vous une autre proposition ?

7)Nous rencontrons des difficultés avec Python Notebook, est-ce un élément indispensable ?

Cordialement
Charles Thiebaud