

Document SRS

Dash Display

Historique des révisions

Table des matières

1 Introduction 6

1.1 Objectif 6

1.2 Portée 6

1.3 Références 7

1.4 Hypothèses et dépendances 7

2 Survol du modèle des cas d’utilisation 8

2.1 Diagramme des cas d’utilisation 8

2.2 Cas d’utilisation 8

2.2.1 CU01 - Changer le mode de l’application 8

2.2.2 CU02 - Recevoir et afficher les données des alarmes et des capteurs 9

2.2.3 CU03 - Changer de page 9

2.2.4 CU04 - Ajouter un capteur ou une alarme 9

2.2.5 CU05 - Modifier la liste des capteurs ou des alarmes 9

2.2.6 CU06 - Afficher les détails d’un capteur ou d’une alarme 9

3 Les acteurs 9

3.1 Ingénieur de piste 9

3.2 Pilote 9

3.3 Le véhicule Formule ÉTS 10

4 Les exigences 10

4.1 Les exigences fonctionnelles 10

4.1.1 EF01 - Configuration de l’application avec un fichier XML 10

4.1.2 EF02 - Configuration de l’application dans les paramètres d’iOS 10

4.1.3 EF03 - Gérer les données reçues en temps réel 10

4.1.4 EF04 - Afficher des couleurs spécifiques pour les RPM 10

4.1.5 EF05 - Afficher des couleurs spécifiques pour la température des pneus 11

4.1.6 EF06 - Afficher des couleurs spécifiques pour la température du moteur 11

4.1.7 EF07 - Afficher des couleurs spécifiques pour la puissance de la batterie 11

4.1.8 EF08 - Afficher des couleurs spécifiques pour les alertes 11

4.1.9 EF09 - Visualiser les alarmes et les capteurs sur l’interface pilote 11

4.1.10 EF10 - Changer l’interface lors de l’appui sur le bouton du volant 11

4.1.11 EF11 - Contenu de la première interface 11

4.1.12 EF12 - Contenu de la deuxième interface 12

4.1.13 EF13 - Contenu de la troisième interface 12

4.1.14 EF14 - Contenu de la quatrième interface 12

4.1.15 EF15 - Affichage en mode paysage pour le mode pilote 12

4.1.16 EF16 - Mettre en veille l’application après 3 secondes sans données 12

4.1.17 EF17 - Visualiser les alarmes et les capteurs sur l’interface ingénieur 12

4.1.18 EF18 - Ajouter une alarme ou un capteur 12

4.1.19 EF19 - Changer l’ordre des alarmes et des capteurs affichés 12

4.1.20 EF20 - Supprimer une alarme ou un capteur affiché 13

4.1.21 EF21 - Afficher les détails de l’alarme ou du capteur 13

4.1.22 EF22 - Gérer les cas d’erreurs de l’application 13

4.2 Les exigences non fonctionnelles 13

4.2.1 ENF01 - Utilisation du visuel de façon intuitive 13

4.2.2 ENF02 - Utilisation du mode pilote doit être très simple 13

4.2.3 ENF03 - Démarrage simple et rapide dans le mode configuré 14

4.2.4 ENF04 - Haut contraste dans les couleurs de l’interface 14

4.2.5 ENF05 - Rafraîchissement de l’écran à une cadence de 10 Hz 14

4.2.6 ENF06 - Période d’utilisation d’au maximum 25 minutes 14

4.2.7 ENF07 - Modification rapide des alarmes et des capteurs 14

4.2.8 ENF08 - Aucun redémarrage de l’application en cas d’erreur 15

5 Documentation en direct pour l’utilisateur et exigences du système d’aide 15

6 Contraintes de conception 15

7 Interfaces 15

7.1 Interfaces utilisateurs 15

7.2 Interfaces matérielles 18

7.3 Interfaces logicielles 18

7.4 Interfaces de communications 19

8 Standards applicables 19

9 Modifications par rapport au document de vision 19

Glossaire 21

Annexes 23

A Spécifications des cas d’utilisation 23

CU01 - Changer le mode de l’application 23

CU02 - Recevoir et afficher les données des alarmes et des capteurs 24

CU03 - Changer de page 25

CU04 - Ajouter un capteur ou une alarme 26

CU05 - Modifier la liste des capteurs ou des alarmes 27

CU06 - Afficher les détails d’un capteur ou d’une alarme 28

B Matrice de traçabilité 29

C Diagramme de classes 31

D Diagramme de séquence 33

# 1 Introduction

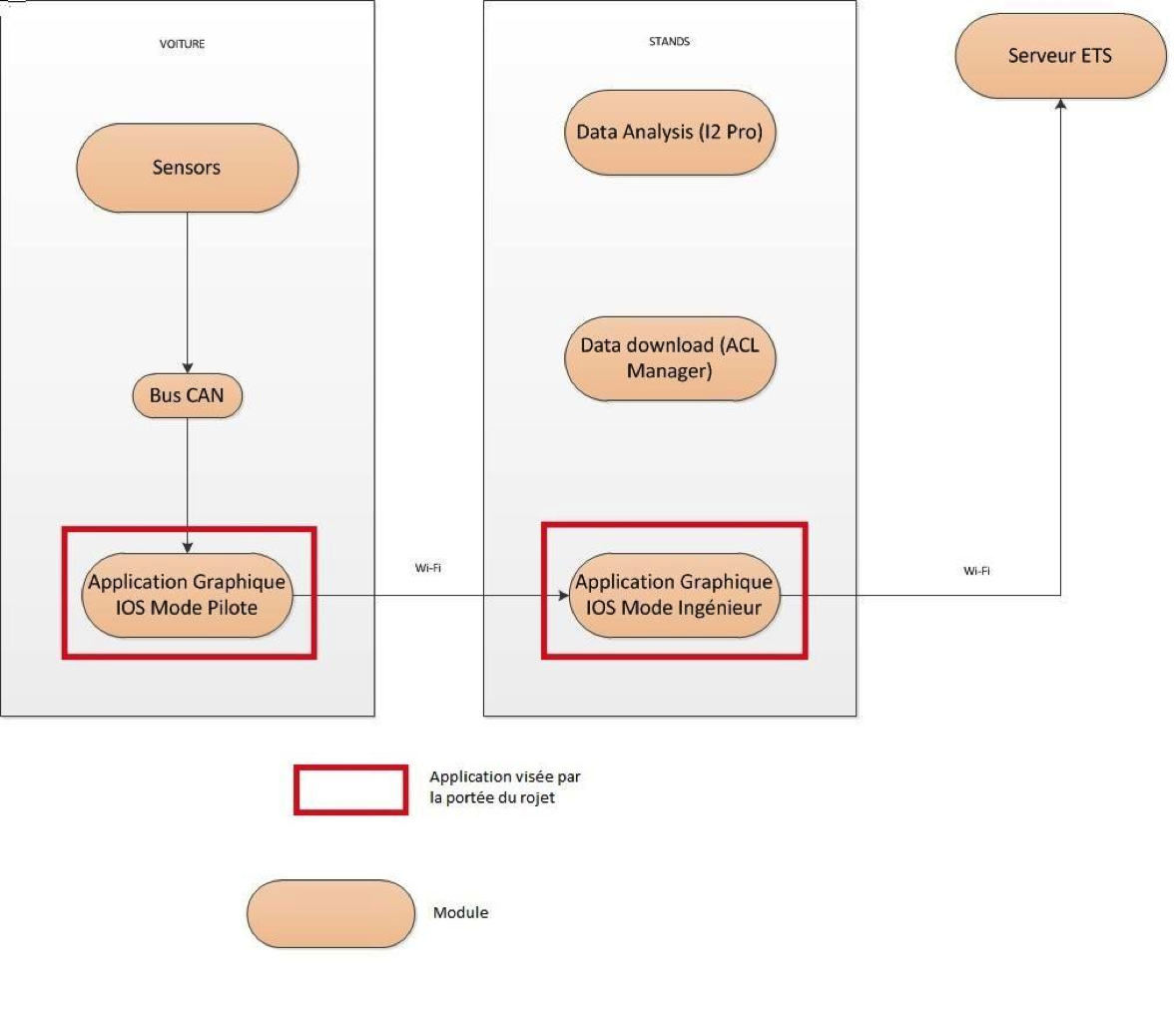
## 1.1 Objectif

Le document de spécification des exigences logicielles (SRS) a pour but de décrire le comportement externe de l’application mobile *Dash Display* destinée à un club étudiant de l’École de technologie supérieure. De plus, les exigences non fonctionnelles, les contraintes et tous les autres éléments nécessaires pour fournir une description complète et compréhensible des exigences logicielles sont présentés.

## 1.2 Portée

Le projet *Dash Display* de la Formule ÉTS est une application mobile fonctionnant sur iOS et qui permet d’effectuer une visualisation des données importantes en temps réel pour le pilote et une présentation des données plus complète pour les ingénieurs supervisant les tests de roulage. Les données consistent en l’état actuel de la voiture comme l’état des pneus, le RPM, les températures, etc. La portée du projet s’étend des données provenant de la voiture jusqu’à leurs affichages sur l’application en mode pilote ou en mode ingénieur de piste.

Ce document porte spécifiquement sur les deux premières phases sur cinq établies du projet *Dash Display*, soit l’interface pilote et l’interface ingénieur de piste. Les autres phases vont être complétées ultérieurement par la Formule ÉTS.



**Figure 1 : la portée du projet *Dash Display***

## 1.3 Références

* REF01 - Le site Web du club Formule ÉTS : <http://www.formuleets.ca/> (consulté le 4 février 2014);
* REF02 - Le site Web du cours LOG410 : <https://cours.etsmtl.ca/log410/private/> (consulté du 24 janvier au 22 février 2014);
* REF03 - Le site Web de MoTeC : <http://www.motec.com/> (consulté le 9 février 2014);
* REF04 - Présentation : *Projet Dash Display Formule SAE ÉTS* : par Benjamin Comeau et Xavier Abram (consulté le 9 février 2014);
* REF05 - Document PDF : *Application mobile : Tableau de bord* : par Benjamin Comeau et Xavier Abram (consulté le 9 février 2014);
* REF06 - Document Excel : *Table des messages CAN* : par la Formule ÉTS (consulté le 9 février 2014);
* REF07 - Document Word : *Réponse aux questions* : par la Formule ÉTS (consulté le 9 février 2014);
* REF08 - Document Word : *Vision de référence* : par l’équipe 21 (consulté le 9 mars 2014);
* REF09 - Document PDF : *Document de réponse client* : par la Formule ÉTS (consulté le 9 mars 2014);
* REF10 - Document PDF : *QUINT2: The Extended ISO Model of Software Quality* : <http://www.cs.uu.nl/wiki/pub/Swa/CourseLiterature/QUINT2.pdf> (consulté le 9 mars 2014).

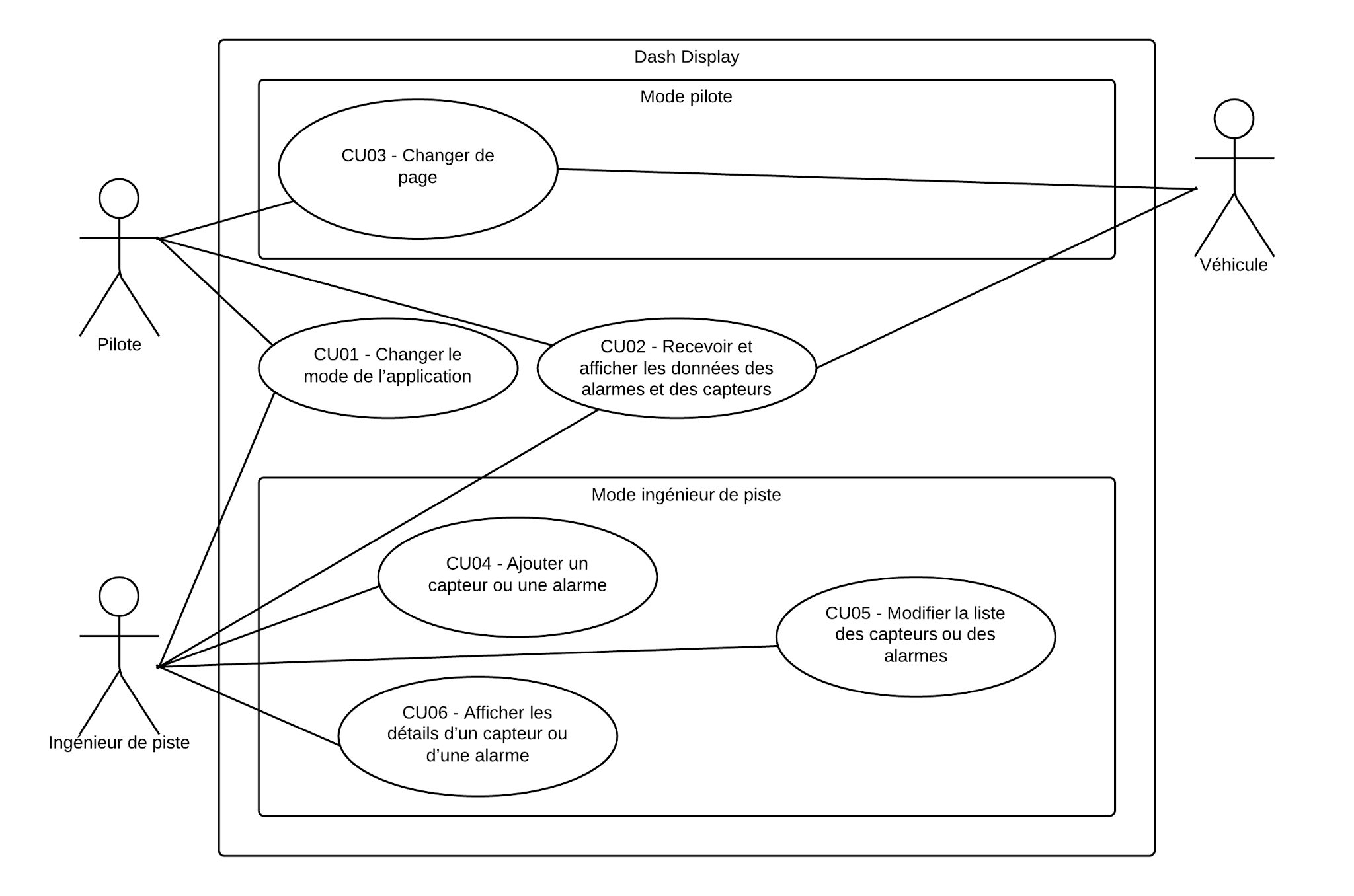
## 1.4 Hypothèses et dépendances

Les hypothèses et les dépendances suivantes sont essentielles pour le développement du *Dash Display* et sont supposées comme valides. Dans le cas de l’invalidité d’un élément de la liste, le projet présenté dans ce document s’en retrouve altéré et son fonctionnement compromit.

|  |  |
| --- | --- |
| **Identification** | **Description** |
| DEP01 | Le document SRS dépend fortement du document de vision (REF08) fourni. Tout changement dans le document de vision peut impacter ce document. |
| DEP02 | Le *Dash Display* dépend des « id » et des « offset » fournies par la table CAN définie par la Formule ÉTS. |
| DEP03 | Les données doivent être transférées par le bus CAN et le protocole UDP via Wi-Fi. |
| HYP01 | Le module CAN à Ethernet (Can2Ethernet) conçu par la Formule ÉTS est fonctionnel. |
| HYP02 | Les futurs utilisateurs de l’application sont à l’aise avec l’utilisation d’un iPod Touch avec iOS 7. |

# 2 Survol du modèle des cas d’utilisation

## 2.1 Diagramme des cas d’utilisation



**Figure 2 : la relation entre les acteurs et les différents cas d’utilisation**

## 2.2 Cas d’utilisation

Cette section présente brièvement les cas d’utilisation reliés au *Dash Display*. Une version plus détaillée de chacun des cas est présentée dans l’annexe A du présent document. À l’annexe C, un diagramme de classes est présenté pour définir comment les différents composants du logiciel sont liés entre elles. De plus, un diagramme de séquence, représentant la mise à jour d’un *widget* affiché suite à la réception d’une donnée du véhicule en mode ingénieur de piste, est présenté à l’annexe D.

Général

### 2.2.1 CU01 - Changer le mode de l’application

L’utilisateur souhaite modifier la configuration de l’application pour modifier le mode, entre pilote et ingénieur de piste, dans lequel l’application doit démarrer.

### 2.2.2 CU02 - Recevoir et afficher les données des alarmes et des capteurs

L’utilisateur souhaite visualiser les données reçues du véhicule sous forme de *widget* spécifique pour chacun des alarmes et des capteurs disponibles.

Mode pilote

### 2.2.3 CU03 - Changer de page

Le pilote désire changer d’interface entre les quatre disponible lorsqu’il appuie sur le bouton situé sur son volant.

Mode ingénieur de piste

### 2.2.4 CU04 - Ajouter un capteur ou une alarme

L’ingénieur de piste désire ajouter une alarme ou un capteur dans la liste déjà affichée à l’écran principal.

### 2.2.5 CU05 - Modifier la liste des capteurs ou des alarmes

L’ingénieur de piste désire changer l’ordre ou supprimer une alarme ou un capteur dans la liste affichée à l’écran principal.

### 2.2.6 CU06 - Afficher les détails d’un capteur ou d’une alarme

L’ingénieur de piste désire obtenir plus de détails à propos d’une l’alarme ou d’un capteur précis comme l’historique des dernières données reçues ou modifier le type d’affichage.

# 3 Les acteurs

## 3.1 Ingénieur de piste

Les ingénieurs de piste utilisent l’application en mode ingénieur de piste. Ils visualisent les données provenant du véhicule durant les essais sur piste et durant les compétitions. Ils se servent de l’application pour conseiller le pilote et pour apporter des modifications mécaniques sur le véhicule dans le but de l’améliorer.

## 3.2 Pilote

Le pilote contrôle le véhicule lors des essais sur piste et des compétitions. Il utilise l’application en mode pilote afin d’accéder aux données du véhicule ce qui permet d’avoir une meilleure compréhension des différents composants et d’améliorer sa conduite.

## 3.3 Le véhicule Formule ÉTS

Le véhicule de la Formule ÉTS possède des capteurs situés sur différentes composantes qui enregistrent des données et les transmet au module Can2Ethernet. Ce module transforme les données en signal CAN afin de les partager par Wi-Fi jusqu’à l’application.

# 4 Les exigences

Dans cette section les exigences fonctionnelles et non fonctionnelles nécessaires pour le *Dash Display* sont présentées. Une traçabilité entre les caractéristiques du document de vision, les exigences, les cas d’utilisation et les contraintes est présentée à l’annexe B du présent document.

## 4.1 Les exigences fonctionnelles

Général

### 4.1.1 EF01 - Configuration de l’application avec un fichier XML

L’application doit utiliser un fichier de configuration, sous le format XML, pour déterminer les alarmes et capteurs disponibles. La liste des alarmes et des capteurs sont définis selon la table CAN fournie par la Formule ÉTS.

### 4.1.2 EF02 - Configuration de l’application dans les paramètres d’iOS

Le système doit permettre de changer quelques configurations directement dans les paramètres de l’application sur iOS. Les configurations doivent inclure, entre autres, le changement de mode entre pilote et ingénieur de piste ainsi que le changement des couleurs de l’interface de pâle à foncé.

### 4.1.3 EF03 - Gérer les données reçues en temps réel

L’application doit constamment recevoir des données du bus CAN via Wi-Fi à partir du module Can2Ethernet et, en comparant avec la table des messages CAN, associer ces données aux capteurs et alarmes pour les afficher.

### 4.1.4 EF04 - Afficher des couleurs spécifiques pour les RPM

L’application doit afficher l’indicateur de RPM avec un code de couleur précis, soit de jaune à rouge en passant par une zone orange visible. De 3000 RPM à 15 000 RPM, l’indicateur doit être dans le spectre de jaune à rouge. À 15 000 RPM et plus, l’indicateur doit être rouge. De plus, le rouge doit changer selon un paramètre calculé par l’ACL lorsqu’un message correspondant au « id » de la table CAN de ce paramètre est reçu.

### 4.1.5 EF05 - Afficher des couleurs spécifiques pour la température des pneus

L’application doit afficher la température des trois capteurs de chaque pneu selon des couleurs spécifiques et avec des transitions fluides. Les capteurs sont situés à l’extérieur, au milieu et à l’intérieur de chacun des pneus. Lorsque la température est de 25°C et moins, la couleur est bleue. Entre 25°C et 65°C, la couleur passe de bleue à jaune. Entre 65°C et 95°C, l’indicateur passe de jaune à rouge. Finalement, en haut de 95°C, la couleur est rouge.

### 4.1.6 EF06 - Afficher des couleurs spécifiques pour la température du moteur

Le *Dash Display* doit afficher la température du moteur à l’aide de quatre couleurs. Lorsque la température est de 70°C et moins, l’indicateur doit être bleu. Entre 70°C et 90°C, la couleur utilisée est le vert. Entre 90°C et 100°C, l’indicateur doit être jaune et, finalement, il doit être rouge lorsque la température dépasse le 100°C.

### 4.1.7 EF07 - Afficher des couleurs spécifiques pour la puissance de la batterie

Le système doit afficher la puissance de la batterie à l’aide de deux couleurs. Lorsque la batterie est à la puissance maximale (14 V) jusqu’à 11.5 V, le fond du capteur est vert. Lorsque la puissance atteint 11.5 V et en dessous, le fond devient rouge.

### 4.1.8 EF08 - Afficher des couleurs spécifiques pour les alertes

Le *Dash Display* doit afficher les alertes de deux façons selon leur statut. Dans tous les cas, elles sont affichées en rouge. Lorsqu’elles sont en cours, elles ont une opacité de 100 %. Sinon, l’opacité diminue à 30 %.

Mode pilote

### 4.1.9 EF09 - Visualiser les alarmes et les capteurs sur l’interface pilote

L’application doit afficher, de manière claire, les informations nécessaires au pilote en provenance des différents capteurs du véhicule. De plus, les messages d’alarmes seront aussi affichés de façon évidente et de façon à ce que le pilote les remarques immédiatement.

### 4.1.10 EF10 - Changer l’interface lors de l’appui sur le bouton du volant

L’application doit changer l’interface affichée lorsque le pilote appuie sur un bouton du volant. Le bouton envoie un message CAN à l’application pour lui indiquer de changer. L’application en mode pilote affichera quatre interfaces différentes en boucle.

### 4.1.11 EF11 - Contenu de la première interface

L’application doit afficher, sur la première interface, les capteurs suivants : la température du moteur, le voltage de la batterie, l’indicateur d’utilisation et d’angle du système de réduction de traînée, les différentes alarmes, la boîte de messages, l’indicateur de vitesse et l’indicateur de révolutions par minute.

### 4.1.12 EF12 - Contenu de la deuxième interface

Le *Dash Display* doit afficher, sur la deuxième interface, les capteurs suivants : la pression et la température des pneus, le biais de frein, l’antiroulis, l’odomètre, le voltage de la batterie, la température du moteur et un indicateur d’utilisation et l’angle du système de réduction de traînée.

### 4.1.13 EF13 - Contenu de la troisième interface

L’application doit afficher, sur la troisième interface, le temps total de course, le temps du tour courant, la différence de temps par rapport au meilleur temps, le meilleur temps de tour de piste et la différence de temps par rapport au dernier de tour de piste.

### 4.1.14 EF14 - Contenu de la quatrième interface

Le *Dash Display* doit afficher, sur la quatrième interface, un schéma de la piste de course et le déplacement de la voiture en temps réel. De plus, les temps suivants doivent être présents : le temps du tour de piste, le meilleur temps et la différence par rapport au meilleur temps.

### 4.1.15 EF15 - Affichage en mode paysage pour le mode pilote

L’application doit afficher les interfaces en mode pilote sous le format paysage.

### 4.1.16 EF16 - Mettre en veille l’application après 3 secondes sans données

L’application doit se mettre en veille si une interruption de données survient et si elle dure plus de trois secondes.

Mode ingénieur de piste

### 4.1.17 EF17 - Visualiser les alarmes et les capteurs sur l’interface ingénieur

L’application doit afficher, sous forme de liste, les différents capteurs et alarmes que l’utilisateur décide d’inclure. Les capteurs et les alarmes sont affichés séparément, les alarmes se trouvant en haut de la liste. Lors de la présentation du prototype, le club formule ÉTS a précisé qu’ils souhaiteraient traiter les alarmes de la même façon que les différents capteurs donc les rendre modifiable sur la page principale.

### 4.1.18 EF18 - Ajouter une alarme ou un capteur

L’application doit permettre à l’utilisateur de sélectionner dans une liste une alarme ou un capteur à ajouter à la liste d’affichage. L’application doit aussi permettre de filtrer la liste des alarmes et des capteurs qui peuvent être ajoutés et d’y effectuer une recherche.

### 4.1.19 EF19 - Changer l’ordre des alarmes et des capteurs affichés

L’application doit permettre à l’utilisateur, une fois en mode édition de la liste, de réorganiser respectivement les alarmes et les capteurs entre eux.

### 4.1.20 EF20 - Supprimer une alarme ou un capteur affiché

L’application doit permettre à l’utilisateur, une fois en mode édition de la liste, de supprimer un capteur ou une alarme.

### 4.1.21 EF21 - Afficher les détails de l’alarme ou du capteur

Le système doit permettre de cliquer sur un capteur ou une alarme affichés afin d’obtenir plus de détails. Lors de la présentation du prototype, cette exigence a été clarifiée. Le client désire avoir la possibilité de modifier l’affichage du *widget* pour d’autres formats ainsi qu’obtenir un historique des dernières données. Les informations apparaissent sous le *widget* principal avec la possibilité d’afficher l’historique en plein écran.

### 4.1.22 EF22 - Gérer les cas d’erreurs de l’application

L’application doit, en cas d’erreurs de l’application, afficher les dernières données reçues. Les cas d’erreurs peuvent être, par exemple, une erreur de transmission de données ou un message d’erreur reçu par une chaîne CAN du module Can2Ethernet.

## 4.2 Les exigences non fonctionnelles

Les exigences non fonctionnelles sont divisées selon les attributs qualité définis par le standard Quint-2 (REF10).

Usability - Understandability

### 4.2.1 ENF01 - Utilisation du visuel de façon intuitive

L’interface doit respecter le fonctionnement natif d’iOS lorsque les diverses actions sont effectuées dans le mode ingénieur de piste. Par exemple, l’ingénieur de piste doit entrer en mode édition pour supprimer ou déplacer une alarme ou un capteur.

Usability - Learnability

### 4.2.2 ENF02 - Utilisation du mode pilote doit être très simple

Les différentes interfaces du mode pilote doivent être simples, claires et précises. Lorsque la voiture est en piste, le pilote ne doit pas avoir à réfléchir pour comprendre et utiliser l’application. Les quatre interfaces disponibles doivent être toutes visibles en trois clics du bouton situé sur le volant puis continuer de cette façon en boucle.

Usability - Operability

### 4.2.3 ENF03 - Démarrage simple et rapide dans le mode configuré

Lorsque le *Dash Display* démarre, il ne doit pas y avoir d’attente ou de commande à effectuer pour que l’application puisse être utilisée. Cette exigence est surtout importante pour le mode pilote qui ne peut pas utiliser l’écran tactile. De plus, le système doit utiliser le mode choisi dans les configurations de l’application directement dans iOS.

Usability - User-friendliness

### 4.2.4 ENF04 - Haut contraste dans les couleurs de l’interface

Les couleurs de l’interface doivent avoir de très haut contraste. De plus, deux versions de couleurs doivent être disponibles dans la configuration de l’application sur iOS : un mode foncé et un mode pâle.

Efficiency - Time behavior

### 4.2.5 ENF05 - Rafraîchissement de l’écran à une cadence de 10 Hz

La fréquence de rafraîchissement des interfaces est de 10 Hz afin que les données affichées soient toujours à jour en temps réel. Avec cette cadence, le véhicule a le temps de transmettre les nouvelles données par Wi-Fi. Cette mesure signifie 10 fois par seconde donc le rafraîchissement est d’une fois à chaque 100 ms.

Efficiency - Resource behavior

### 4.2.6 ENF06 - Période d’utilisation d’au maximum 25 minutes

Les courses ont une durée d’au maximum 25 minutes donc l’application doit être optimisée pour une utilisation sans problème pour ce délai de temps.

Usability - Customizability

### 4.2.7 ENF07 - Modification rapide des alarmes et des capteurs

La liste des alarmes et des capteurs peut facilement être modifiée en 5 minutes et moins. Cette modification est effectuée sur le fichier de configuration XML en fournissant le « id » et l’« offset » définis dans la table CAN. Cette exigence n’était pas précisée de la part du client, mais une précision concernant la table CAN qui peut être modifiée a été faite. Il est donc important que tout le logiciel soit facilement maintenable.

Reliability - Fault tolerance

### 4.2.8 ENF08 - Aucun redémarrage de l’application en cas d’erreur

Lorsqu’une erreur survient, l’application ne doit pas redémarrer seule. Il faut la redémarrer manuellement à chaque fois.

# 5 Documentation en direct pour l’utilisateur et exigences du système d’aide

Le club Formule ÉTS souhaite ne pas avoir de documentation en direct ou de système d’aide pour le *Dash Display.* Plusieurs membres sont des utilisateurs expérimentés des technologies informatiques utilisées et peuvent apporter le support nécessaire.

# 6 Contraintes de conception

|  |  |
| --- | --- |
| **Identification** | **Description** |
| CONT01 | Les configurations pour les catégories des alarmes et des capteurs doivent être définies dans un fichier XML. |
| CONT02 | Les configurations pour le mode par défaut et les couleurs de l’interface sont définies directement dans iOS. |
| CONT03 | La librairie Can2Ethernet développée par le club Formule ÉTS doit être utilisée. |
| CONT04 | L’application doit être en Objective-C sous la plateforme iOS 7 et est destinée aux iPod Touch de 5e génération avec un écran de 4 pouces. |
| CONT05 | Les données sont envoyées par Wi-Fi via le protocole UDP et le bus CAN. |
| CONT06 | Le port UDP est le port par défaut soit 1337. |
| CONT07 | Les « id » et les « offset » des capteurs doivent suivre la table CAN fournie. |
| CONT08 | Les textes de l’application doivent être en anglais. |
| CONT09 | L’application doit être installée et exécutée sur un iPod Touch qui ne nécessite pas un « iOS jailbreaking » |

# 7 Interfaces

## 7.1 Interfaces utilisateurs

Pour l’application, les alarmes et les capteurs des deux modes sont présentés de façon très différente. Les images suivantes présentent le mode pour les ingénieurs de piste alors que le mode pilote consiste en des alarmes et des capteurs sous forme de *widget* avec la seule possibilité de les visualiser. Comme les interfaces pour le mode pilote présentées en croquis par la Fomule ÉTS étaient très claires, seules les interfaces du mode pour ingénieur de piste ont été créées pour clarifier les ambiguïtés.

La première interface présente la liste des alarmes et des capteurs que l’utilisateur a décidé d’afficher. Dans la seconde interface, l’utilisateur est entré en mode édition pour ajouter, déplacer ou supprimer une alarme ou un capteur. Dans la troisième interface, la vue détaillée d’un capteur est présentée sous forme de graphique qui se met à jour en temps réel. Ce graphique représente un historique des dernières données reçues. La page détaillée a été proposée pour clarifier la CAR16 du document de Vision. Plus de détails sur les modifications apportées par rapport au Vision sont donnés dans la section 9 du présent document. Finalement, dans la dernière interface, l’ajout d’une alarme ou d’un capteur est démontré grâce à un sélecteur qui permet de chercher ce que l’utilisateur désire ajouter.

|  |  |  |
| --- | --- | --- |
| srs-1.jpg  **Figure 3 : Interface 1 - liste des alarmes et des capteurs** |  | srs-2.jpg  **Figure 4 : Interface 2 - mode édition de la liste des alarmes et des capteurs** |
| srs-3.jpg  **Figure 5 : Interface 3 - vue détaillée d’une alarme ou d’un capteur** |  | srs-4.jpg  **Figure 6 : Interface 4 - ajout d’une alarme ou d’un capteur** |

## 7.2 Interfaces matérielles

Le *Dash Display* peut recevoir des données provenant de l’appui d’un bouton sur le volant du pilote lorsque l’application est en mode pilote. Ce bouton permet de changer les interfaces entre les quatre possibles en envoyant le signal de la même façon que les capteurs de la voiture c’est-à-dire par le protocole UDP, le bus CAN et le Wi-Fi.

## 7.3 Interfaces logicielles

Aucune interface logicielle n’est requise pour l’application *Dash Display*.

## 7.4 Interfaces de communications

L’application doit gérer les données reçues sous le format CAN transmises par Wi-Fi directement de la voiture de la Formule ÉTS à l’aide du module Can2Ethernet. Les données correspondent à la table CAN définie dans le fichier de configuration XML.

# 8 Standards applicables

Le code source de l’application doit respecter les standards d’Objective-C et du développement d’application sous le système d’opération iOS. Le fichier de configuration de l’application doit suivre la norme XML selon un gabarit défini par le club.

Les données sont transmises des capteurs vers le module Can2Ethernet par des signaux empruntant le bus CAN et le protocole UDP. Le module émet les données par des signaux Wi-Fi vers l’application qui reçoit les messages en texte brut qui suivent le format de la table CAN fournie par la Formule ÉTS.

# 9 Modifications par rapport au document de vision

Lors de la rédaction du présent document, quelques caractéristiques définies dans le document de vision ont été modifiées et corrigées. Entre autres, la caractéristique CAR16 inclut aussi le changement d’ordre, la suppression et l’ajout de nouveaux capteurs et de nouvelles alarmes sur l’interface d’ingénieur de piste afin de personnaliser l’affichage comme demandé par la Formule ÉTS.

Suite à la dernière période de questions (REF09), la caractéristique CAR04 a été retirée puisque les ingénieurs doivent faire défiler la liste de capteurs vers le bas au lieu de changer de page. Pour le changement de mode de la CAR07, l’application détermine le mode en vérifiant un paramètre de l’application dans iOS et non par l’appui d’un bouton. Aussi, la caractéristique CAR17 doit inclure la configuration de l’application dans iOS en plus du fichier XML.

Dans la description de la CAR12, le document de vision fait mention de la couleur des pneus. Ce bout de description n’est pas adapté à cette caractéristique, mais plutôt à la CAR14. Comme une autre caractéristique présente la couleur des pneus, aucun ajout n’est nécessaire de ce côté.

La CAR16 permet de voir plus de données sur un capteur précis comme un historique. Elle est représentée par la page supplémentaire, dans les interfaces, lors du clic sur une alarme ou un capteur dans les interfaces. Suite à la présentation du prototype, les clients ont trouvé que cette proposition était une bonne idée, mais il manquait la possibilité de modifier l’affichage du *widget* selon la configuration du fichier.

Avec le document de questions (REF07), la caractéristique CAR18 a été modifiée par rapport au document de vision pour refléter le changement de format du fichier de configuration de .cfg à .xml. De plus, l’application n’effectue par le passage de .xlsx à .xml, elle ne fait que lire le fichier de configuration sous le format .xml.

La CAR19 a été ajoutée comme les clients ont bien spécifié les couleurs pour la puissance de la batterie. Cette caractéristique était manquante dans le document de Vision.

La CAR20 a été ajoutée pour spécifier que les dernières données reçues par l’application doivent rester affichées en cas d’erreur pour le mode ingénieur de piste. Cette caractéristique était manquante dans le document de Vision.

La CAR21 a été ajoutée pour spécifier que les courses ont une durée maximale de 25 minutes. Cette donnée apporte une exigence supplémentaire comme il faut s’assurer que l’application est optimisée pour cette durée d’utilisation consécutive.

# Glossaire

|  |  |
| --- | --- |
| **Terme** | **Définition** |
| ACL | *Acronyme :* Advanced Central Logger |
| ACL Manager | Un logiciel de la compagnie MoTeC utilisé pour configurer et contrôler le module MoTeC ACL. |
| Android | Un système d’opération de la compagnie Google destiné aux produits tablettes et mobiles qui supportent ce système. |
| A2C | *Acronyme :* Analog To CAN  Une unité similaire au VIM, mais fabriquée par la Formule ÉTS. Il y a un module A2C par roue sur le véhicule. |
| CAN | *Acronyme :* Controller Area Network  Un bus série très populaire dans le domaine de l’automobile. Il est normalisé par la norme ISO 11898 et utilise une approche populaire, soit le multiplexage, pour effectuer la communication entre les composants d’un ordinateur. |
| Can2Ethernet | Un module développé par la Formule ÉTS qui effectue le transfert du bus CAN vers l’Ethernet. |
| ECU | *Acronyme :* Engine Control Unit  Une unité qui gère le moteur et les capteurs reliés à son fonctionnement. |
| Ethernet | Un système servant à connecter un certain nombre de systèmes informatisés pour former un réseau local. Des protocoles sont utilisés pour contrôler les informations partagées et pour empêcher la transmission simultanée de données par deux systèmes ou plus. |
| ÉTS | *Acronyme :* École de technologie supérieure |
| Formule ÉTS | Un club étudiant scientifique de l’École de technologie supérieure qui fabrique une voiture de course monoplace de type formule performante. |
| iPod Touch | Un produit de la compagnie Apple qui sert principalement de lecteur de musique. Il possède aussi la capacité de naviguer sur Internet via le sans-fil et d’utiliser des applications logicielles. |
| iOS | Un système d’opération de la compagnie Apple destiné aux produits tablettes et mobiles qui supportent ce système. |
| iOS jailbreaking | Un processus permettant aux produits Apple qui fonctionnent sous le système d’exploitation iOS d'obtenir un accès complet pour déverrouiller toutes les fonctionnalités du système d'exploitation, éliminant ainsi les restrictions posées par Apple. |
| MoTeC ACL | Un outil d’acquisition de données et de communication hautement configurable de la compagnie MoTeC. |
| MoTeC i2 Pro | Un logiciel de la compagnie MoTeC utilisé pour l’analyse des données provenant du MoTeC ACL |
| MVC | *Acronyme :* Modèle-vue-contrôleur  Un modèle architectural de programmation qui répond aux besoins d’une application interactive en séparant les données, la présentation et la logique en différentes composantes au sein d’une même architecture. |
| Protocole de communication | Un protocole de communication est une définition de plusieurs règles utilisées par un moyen de communication spécifique. |
| Quint-2 | Le standard Quint-2 est une extension des attributs de qualité de la norme ISO 9126. |
| RPM | *Acronyme :* Revolution Per Minute ou Rotation Per Minute  Le tour par minute est une unité de mesure de la vitesse de rotation. Dans le domaine des voitures, cette mesure signifie le nombre de tours par minute du moteur. |
| UDP | *Acronyme :* User Datagram Protocol  Le protocole UDP est l’un des principaux protocoles employés par la communication via Internet. |
| USB | *Acronyme :* Universal Serial Bus |
| VIM | *Acronyme :* Versatile input Module  Une unité qui convertit les signaux des capteurs en signal CAN. |
| Widget | Un *widget* est un composant logiciel en relation avec les interfaces graphiques. Il peut désigner un composant d'interface graphique (un élément visuel d'une interface graphique) ou un gadget interactif (un petit outil qui permet d'obtenir des informations). |
| Wi-Fi | Un ensemble de protocoles de communication sans fil permettant de relier plusieurs systèmes informatiques entre eux pour permettre la transmission d’informations. Le Wi-Fi est régi par la norme IEEE 802.11 (ISO/CEI 8802-11). |
| Xcode | Un environnement de développement pour Mac OS X ainsi que iOS. |

# Annexes

## A Spécifications des cas d’utilisation

### CU01 - Changer le mode de l’application

#### Brève description

L’utilisateur peut modifier les configurations de l’application dans les paramètres iOS pour effectuer le changement entre les deux modes disponibles, soit le mode pilote et le mode ingénieur de piste.

#### Acteurs principaux

* L’ingénieur de piste;
* Le pilote.

#### Précondition(s)

Aucune précondition.

#### Postconditions(s)

L’application démarre dans le mode spécifié par la configuration.

#### Flux d’événements

##### Flux de base

1. L’utilisateur ouvre la configuration de l’application dans les paramètres iOS;
2. L’utilisateur modifie la configuration pour spécifier le mode de l’application;
3. L’application, à l’ouverture, lit la configuration pour savoir quel mode utiliser.

### 

### CU02 - Recevoir et afficher les données des alarmes et des capteurs

#### Brève description

Le véhicule de la Formule ÉTS transmet les données récupérées de ces différents capteurs vers l’application pour interprétation.

#### Acteurs principaux

* Le véhicule;
* L’ingénieur de piste;
* Le pilote.

#### Précondition(s)

Le véhicule transmet les données vers l’application.

#### Postconditions(s)

L’application affiche les informations reçues du véhicule de la manière appropriée.

#### Flux d’événements

##### Flux de base

1. Le véhicule envoie les informations en provenance des différents capteurs à travers le module Can2Ethernet;
2. L’application reçoit les données via Wi-Fi;
3. L’application compare les messages reçus avec la table des messages CAN disponible dans le fichier de configuration XML;
4. L’application affiche les données reçues à travers les *widgets* appropriés au pilote ou aux ingénieurs de piste.

### CU03 - Changer de page

#### Brève description

Le pilote peut changer de page entre quatre interfaces prédéfinies lorsqu’il appuie sur un bouton du volant. La première interface revient lorsque le pilote change l’interface à la dernière disponible.

#### Acteurs principaux

* Pilote;
* Le véhicule.

#### Précondition(s)

L’application est en mode pilote.

#### Postconditions(s)

L’application affiche une page différente de la précédente.

#### Flux d’événements

##### Flux de base

1. Le pilote appuie sur le bouton situé sur le volant véhicule;
2. Le véhicule envoie la commande du bouton à travers le module Can2Ethernet;
3. L’application reçoit les données via Wi-Fi;
4. L’application analyse le message reçu avec la table des messages CAN et détermine que le message est associé à l’appui du bouton;
5. L’application change l’interface affichée pour la suivante.

### CU04 - Ajouter un capteur ou une alarme

#### Brève description

L’ingénieur de piste peut ajouter une alarme ou un capteur dans la liste déjà affichée à l’écran principal.

#### Acteurs principaux

* Ingénieur de piste.

#### Précondition(s)

L’application est en mode ingénieur de piste.

#### Postconditions(s)

Un capteur ou une alarme a été ajouté à la liste d’affichage.

#### Flux d’événements

##### Flux de base

1. L’ingénieur de piste entre en mode « Édition de la liste »;
2. L’ingénieur de piste entre dans le mode « Ajout »;
3. L’ingénieur de piste sélectionne un capteur ou une alarme dans la liste déroulante;
4. L’ingénieur de piste appuie sur le bouton « Ajouter ».

##### Flux alternatifs : 3.[a]

* 1. L’ingénieur de piste se sert de la fonction « Recherche » pour filtrer la liste déroulante;
  2. L’ingénieur de piste sélectionne un capteur ou une alarme dans cette liste filtrée;
  3. Retour à l’étape 4 du flux de base.

##### Flux alternatifs : 3.[b]

* 1. L’ingénieur de piste sélectionne une catégorie de widget pour filtrer la liste déroulante;
  2. L’ingénieur de piste sélectionne un capteur ou une alarme dans cette liste filtrée;
  3. Retour à l’étape 4 du flux de base.

### CU05 - Modifier la liste des capteurs ou des alarmes

#### Brève description

L’ingénieur de piste peut changer l’ordre ou supprimer une alarme ou un capteur dans la liste affichée à l’écran principal.

#### Acteurs principaux

* Ingénieur de piste.

#### Précondition(s)

L’application est en mode ingénieur de piste et la liste contient au minimum deux widgets.

#### Postconditions(s)

La liste d’affichage a été modifiée, soit par la modification de l’ordre des *widgets* ou par la suppression d’un ou plusieurs *widgets*.

#### Flux d’événements

##### Flux de base

1. L’ingénieur de piste entre en mode « Édition de la liste »;
2. L’ingénieur de piste glisse et dépose un *widget* à un nouvel emplacement dans la liste.

##### Flux alternatifs : 2.[a]

* 1. L’ingénieur de piste appuie sur le bouton « Supprimer » d’un *widget*, le supprimant ainsi de la liste.

## 

### CU06 - Afficher les détails d’un capteur ou d’une alarme

#### Brève description

Lors du clic sur une alarme ou un capteur, la page change pour afficher un historique des dernières données reçues ainsi que les différents affichages possibles pour le *widget*.

#### Acteurs principaux

* Ingénieur de piste.

#### Précondition(s)

L’application est en mode ingénieur de piste.

#### Postconditions(s)

L’alarme ou le capteur sélectionné affiche maintenant plus d’informations reliées à son historique de données reçues.

#### Flux d’événements

##### Flux de base

1. L’ingénieur de piste appuie sur l’alarme ou le capteur dont il désire obtenir plus d’informations;
2. Le *Dash Display* affiche maintenant une page complète présentant l’historique des données reçues pour cette alarme ou ce capteur et les autres informations importantes qui ne sont pas affichées dans la liste principale.

## 

## B Matrice de traçabilité

Les contraintes du document de Vision CO02, CO03 et CO04 n’ont aucun EF, ENF ou CU associés du SRS. En effet, les trois contraintes sont aussi des contraintes du SRS et requièrent une implémentation spécifique pour les respecter.

L’exigence non fonctionnelle ENF08 n’a pas été ajoutée à la traçabilité comme elle s’applique pour l’ensemble de l’application.

|  |  |  |  |
| --- | --- | --- | --- |
| **CAR / Contraintes du Vision** | **EF équivalentes** | **ENF équivalentes** | **CU équivalents** |
| CAR01 - Affichage des données pour le pilote | EF03, EF09, EF11, EF12, EF13, EF14, | ENF02, ENF04 | CU02 |
| CAR02 - Affichage des données pour l’ingénieur de piste | EF03, EF17, | ENF01, ENF04 | CU02 |
| CAR03 - Changement d’écran pour le pilote | EF10 | ENF02 | CU03 |
| CAR05 - Nombre de pages pour le mode pilote | EF10 | ENF02 |  |
| CAR06 - Affichage en mode pilote | EF15 | ENF02, ENF04 |  |
| CAR07 - Changement de mode  *(modifications apportées, voir section 9)* | EF02 | ENF03 | CU01 |
| CAR08 - Rafraîchissement de l’écran |  | ENF05 |  |
| CAR09 - Affichage des couleurs de température | EF05, EF06 | ENF02 |  |
| CAR10 - Affichage d’alertes sur les écrans | EF03, EF08, EF09, EF17 | ENF02 |  |
| CAR11 - Transmission des données via WiFi | EF03 |  |  |
| CAR12 - Affichage des RPM | EF04 | ENF02 |  |
| CAR13 - Affichage du temps | EF13 | ENF02 |  |
| CAR14 - Affichage de la température des pneus | EF05 | ENF02 |  |
| CAR15 - Affichage du tracé du circuit | EF14 | ENF02 |  |
| CAR16 - Affichage personnalisé | EF18, EF19, EF20, EF21 | ENF01 | CU04, CU06 |
| CAR17 - Configuration de l’application | EF01, EF02 | ENF03, ENF07 |  |
| CAR18 - Lecture d’un fichier .xml comme fichier de configuration | EF01 |  |  |
| CAR19 - Affichage de la puissance de la batterie  *(caractéristique ajoutée, voir section 9)* | EF07 | ENF02 |  |
| CAR20 - Les dernières données reçues doivent rester affichées en cas d’erreur  *(caractéristique ajoutée, voir section 9)* | EF22 |  |  |
| CAR21 - Durée maximale des courses  *(caractéristique ajoutée, voir section 9)* |  | ENF06 |  |
| CO01 - Éteindre l’affichage | EF16 |  |  |
| CO02 - Fonctionnement sur iPod |  |  |  |
| CO03 - Fonctionnement sous iOS 7 |  |  |  |
| CO04 - Langue de l’application |  |  |  |

## 

## C Diagramme de classes

Le diagramme de classes a été choisi pour bien représenter les différents composants internes du logiciel et leurs liens entre eux. Ce diagramme permet de comprendre comment apporter des solutions aux divers problèmes posés par l’implémentation du *Dash Display*.

Le diagramme de classes représente une implémentation du *Dash Display* avec une architecture de type MVC. Cette architecture a été choisie pour une meilleure gestion des différents *widgets* pour les alarmes et les capteurs et de leur affichage sur les différentes interfaces. Comme plusieurs vues sont nécessaires, il est plus facile d’avoir une gestion de ses vues lorsque le modèle notifie ses observateurs lors d’un changement de données. De plus, des contrôleurs séparés pour les pilotes et les ingénieurs sont nécessaires étant donné que les fonctionnalités sont très différentes.

Une implémentation d’une classe pour lire le fichier de configuration a été ajoutée pour faciliter la lecture du format XML. La classe CANString ont été créées pour gérer la ligne CAN reçue par le module Can2Ethernet et la divisée selon ce qui est demandé dans REF05. Finalement, la classe CANTableLine représente une ligne de la table de configuration en provenance du fichier XML afin d’obtenir d’autres informations comme l’unité et le masque.

## diagramme-de-classes

## D Diagramme de séquence

Pour le second diagramme, un diagramme de séquence a été choisi pour représenter un flux important de l’application : la réception d’une chaîne CAN et la mise à jour des données. Plus précisément, le diagramme de séquence suivant représente la mise à jour d’un *widget* affiché suite à la réception d’une donnée du véhicule. Le diagramme de séquence utilise le mode ingénieur de piste qui implémente une fonction pour mettre à jour le *widget* en question.

Ce diagramme est un complément au diagramme de classes de l’annexe C puisqu’il démontre l’interaction de plusieurs classes entre elles dans un contexte réel.

