**TP Génie Logiciel, 3IF, 2018/2019**

**Étape 2**

*Historique de modifications par rapport à l’étape 1 :*

* *Reprise des spécifications avec clarification de certains énoncés et fonctionnalités, ajout de fonctionnalités*
* *Idem dans le manuel utilisateur, où nous avons décrit de nouvelles fonctionnalités et mieux détaillé d’autres*
* *Le diagramme de classe, des diagrammes de séquence, d’états transitions et la décomposition en modules ainsi que les descriptions des algorithmes importants ont été ajoutés*

1. Spécification des besoins et plan de tests de validation
   * Spécifications fonctionnelles de l’application :

* Lecture des fichiers

|  |  |
| --- | --- |
| Fonctionnalité F1 | Chargement des fichiers de données |
| Description | Cette fonctionnalité est lancée à l’ouverture de l’application. Elle permet de stocker en mémoire dans l’application les données des fichiers .csv présents sur un serveur central. Ces fichiers contiennent la liste des capteurs, la liste de attributs (ou types de mesures) et les mesures. |
| Données nécessaires | Chemin d’accès de chaque fichier .csv (fichier Capteurs, attributs et mesures). |
| Pré conditions | Les fichiers doivent être conformes au format spécifié dans le sujet du TP (c’est-à-dire chaînes de caractères distinctes séparées par un « ; » par exemple). |
| Post conditions | Le contenu des fichiers est mis en mémoire, il est possible d’y accéder et d’utiliser ces données. |
| Données en sortie | / |
| Tests | Cas d’exception :   * Test d’ouverture d’un fichier vide : exception avec un message d’erreur indiquant que le fichier ne contient rien * Test d’ouverture d’un fichier inexistant ou sans droit de lecture : exception avec un message d’erreur indiquant si le fichier est introuvable ou si une erreur survient lors de son ouverture/lecture   Autre : Test avec des fichiers conformes |

Désormais on considère que les fichiers de données ont été correctement chargés afin de permettre le bon fonctionnement de l’application (on ne considère pas de cas d’erreur où il manque un/les fichiers dans la suite des fonctionnalités). Cette pré-condition (fichiers mis en mémoire correctement) concerne toutes les fonctionnalités ci-après.

* Gestion des capteurs

|  |  |
| --- | --- |
| Fonctionnalité F2 | Consulter la liste des capteurs |
| Description | Cette fonctionnalité affiche la liste des tous les capteurs |
| Données nécessaires | Aucune |
| Pré conditions | Des capteurs sont renseignés, il n’y a pas aucun capteur |
| Post conditions | La liste des capteurs s’affiche |
| Données en sortie | Tableau de chaînes de caractères avec l’ID de chaque capteur, ses coordonnées (latitude et longitude) et sa description |
| Tests | Cas d’exception : Test s’il n’y a aucun capteur dans le fichier d’origine  Autre : Test avec un fichier conforme et affichage de la liste |

|  |  |
| --- | --- |
| Fonctionnalité F3 | Ajouter un capteur |
| Description | Cette fonctionnalité permet d’ajouter un capteur dans le circuit, il est ajouté aux structures de données correspondantes et est rendu fonctionnel (on peut récupérer ses données). |
| Données nécessaires | SensorID, Latitude, Longitude et Description du capteur ajouté |
| Pré conditions | Le SensorID ne doit pas être déjà attribué, les coordonnées doivent être cohérentes. |
| Post conditions | Le capteur est présent dans les structures de données en mémoire de l’application (mais le fichier .csv de base n’est pas modifié). |
| Données en sortie | Boolean (true si l’ajout a fonctionné, false sinon.) |
| Tests | Cas d’exception :   * Indiquer un ID déjà existant * Vérifier que les latitude et longitude du capteur sont possibles * Il est possible de consulter les données du capteur, il est présent en mémoire et fonctionnel |

|  |  |
| --- | --- |
| Fonctionnalité F4 | Supprimer un capteur |
| Description | Cette fonctionnalité permet de supprimer un capteur du circuit, il est supprimé des structures de données correspondantes et n’est plus fonctionnel (on peut plus récupérer ses données ou le retrouver dans la liste des capteurs). |
| Données nécessaires | SensorID |
| Pré conditions | Le capteur doit exister, c’est-à-dire que le SensorID doit correspondre à un capteur existant. |
| Post conditions | Le capteur n’est plus présent dans les structures de données en mémoire de l’application. |
| Données en sortie | Boolean : *true* si ça a fonctionné, *false* sinon. |
| Tests | Cas d’exception : test de la suppression d’un capteur inexistant  Autre :   * Vérifier que le capteur est bien supprimé (impossible de le surveiller, de le trouver dans la liste des capteurs par exemple) * Tester qu’on arrive bien à récupérer les données dans le fichier data même si le capteur est supprimé |

|  |  |
| --- | --- |
| Fonctionnalité F4 | Surveiller un capteur |
| Description | Cette fonctionnalité permet de vérifier la fiabilité d’un capteur : données viables ou non (c’est-à-dire comprises dans des intervalles prédéfinis), dernière date d’actualisation (un capteur ne reste pas 2 jours sans envoyer d’informations). |
| Données nécessaires | SensorID |
| Pré conditions | Le capteur doit exister, c’est-à-dire que le SensorID doit correspondre à un capteur existant. |
| Post conditions | Aucune opération modifiant le capteur n’a été effectué (le capteur est considéré *const* pour cette fonctionnalité). |
| Données en sortie | Boolean : *true* si le capteur est valide, *false* sinon. |
| Tests | Cas d’exception :   * Tester un capteur inexistant * Tester un capteur non fonctionnel (un capteur qui envoie des données aberrantes ou qui n’envoie plus de données depuis 2 jours)   Autre : Tester que tout fonctionne avec un capteur fonctionnel |

|  |  |
| --- | --- |
| Fonctionnalité F5 | Rechercher capteur |
| Description | Cette fonctionnalité permet de trouver l’ID du capteur le plus proche des coordonnées renseignées. |
| Données nécessaires | Latitude, longitude |
| Pré conditions | Les coordonnées doivent être possibles. |
| Post conditions | Affichage de l’ID du capteur |
| Données en sortie | *String* avec SensorID, coordonnées et description |
| Tests | Cas d’exception : Tester sûr des coordonnées erronées  Autre :   * Tests avec 2 capteurs à même distance * Test avec un seul capteur proche |

XXX

|  |  |
| --- | --- |
| Fonctionnalité F6 | Capteurs similaires |
| Description | Cette fonctionnalité permet de trouver les capteurs ayant un comportement similaire à celui du capteur renseigné. Un capteur est similaire à un autre si pour tous les attributs qu’ils ont en commun (un seul attribut peut suffire) toutes leurs valeurs enregistrées sont identiques à l’intervalle de confiance donné près. |
| Données nécessaires | SensorID, intervalle de confiance |
| Pré conditions | Les coordonnées doivent être possibles. |
| Post conditions | RAS |
| Données en sortie | String avec SensorID, coordonnées et description |
| Tests | * Tester sûr des coordonnées erronées * Tests avec 2 capteurs à même distance * Test avec un seul capteur proche |

* Données capteurs

|  |  |
| --- | --- |
| Fonctionnalité F7 | Consulter liste des attributs |
| Description | Cette fonctionnalité permet de consulter le contenu du document avec la liste des tous les attributs |
| Données nécessaires | Aucune |
| Pré conditions | Le document concernant les attributs doit être renseigné à l’application |
| Post conditions | RAS |
| Données en sortie | Tableau de chaînes de caractères avec l’ID de chaque attribut, son unité et sa description |
| Tests | * Test d’ouverture d’un fichier vide : cas d’exception avec un message indiquant que le fichier ne contient rien * Test d’ouverture d’un fichier inexistant ou sans droit de lecture : exception avec un message d’erreur si le fichier est introuvable ou si une erreur survient lors de son ouverture/lecture |

|  |  |
| --- | --- |
| Fonctionnalité F8 | Consulter les données |
| Description | Cette fonctionnalité permet de consulter le contenu du document avec l’enregistrement des données |
| Données nécessaires | Aucune |
| Pré conditions | Le document concernant les données doit être renseigné à l’application |
| Post conditions | RAS |
| Données en sortie | Tableau de chaînes de caractères contenant la date de mesure de chaque donnée, l’ID du capteur l’ayant réalisée, l’ID de l’attribut et la valeur mesurée. |
| Tests | * Test d’ouverture d’un fichier vide : cas d’exception avec un message indiquant que le fichier ne contient rien * Test d’ouverture d’un fichier inexistant ou sans droit de lecture : exception avec un message d’erreur si le fichier est introuvable ou si une erreur survient lors de son ouverture/lecture |

|  |  |
| --- | --- |
| Fonctionnalité F9 | Recueillir données similaires |
| Description | Cette fonctionnalité permet de retourner toutes les données similaires à celle d’un attribut d’un capteur sûr une date précise pour un intervalle de différence donné par l’utilisateur. |
| Données nécessaires | AttributID, SensorID, date, intervalle de différence |
| Pré conditions | Il faut que l’attribut existe et que des valeurs soient renseignées |
| Post conditions | RAS |
| Données en sortie | Toutes les valeurs de l’attribut pour les différents capteur concernés |
| Tests | Cas d’exception :   * Si attributID n’existe pas * Si pas de valeur renseignée   Autre :   * Si une valeur * Si beaucoup de valeurs |

A modifier : il faut une fonctionnalité qui donne la qualité moyenne de l’air à un moment donnée ou sur une période de temps précise + une autre la qualité de l’air a un endroit precis.

|  |  |
| --- | --- |
| Fonctionnalité F9 | Données capteur |
| Description | Cette fonctionnalité permet de trouver les caractéristiques de la qualité de l’air pour un capteur donné ou à un endroit donné. |
| Données nécessaires | SensorID ou coordonnées (longitude et latitude) ou intervalle de coordonnées |
| Pré conditions | Les longitudes et latitudes passées en paramètres doivent être sur le territoire français (car il est couvert par les capteurs) ou le sensorID doit exister |
| Post conditions | RAS |
| Données en sortie | Caractéristiques de la qualité de l’air pour le capteur donnée ou pour celui dont les coordonnées sont les plus proches de celles renseignées et pour la durée ou date donnée s’il y en a une (sinon toutes les mesures du capteur sont affichées) |
| Tests | Cas d’exception :   * Tester si les coordonnées sont impossibles. * Tester que le bon capteur, le plus proche des coordonnées, est sélectionné * Tester que les valeurs renvoyées sont bien celles de la date ou de l’intervalle renseigné (s’il y en a) * Tester si on a pas de données pour le point ou pour l’intervalle de coordonnées   Autre : Vérifier avec des valeurs connues les valeurs retournées. |

* Alertes et seuil

|  |  |
| --- | --- |
| Fonctionnalité F10 | Calculer les données prévisionnelles |
| Description | Cette fonctionnalité permet de calculer à partir des données récupérées des données prévisionnelles |
| Données nécessaires | Au moins un jeu de données récupérées par un capteur |
| Pré conditions | RAS |
| Post conditions | RAS |
| Données en sortie | Données prévisionnelles |
| Tests | * Tester que les données calculées sont un minimum cohérentes. * Tester ce qui se passe si il y a une valeure absurde au milieu de valeur normale sur les prévisions. * Tester à partir de données rentrées par nous-même dans la base que les données prévisionnelles sont bien calculées. |

|  |  |
| --- | --- |
| Fonctionnalité F11 | Données au-delà d’un seuil |
| Description | Cette fonctionnalité permet d’afficher les données de capteurs au-dessus ou en dessous d’un seuil |
| Données nécessaires | AttributID, valeur du seuil, 0 ou 1 (en-dessous ou au-dessus du seuil) |
| Pré conditions | L’attributID doit exister |
| Post conditions | RAS |
| Données en sortie | Chaîne de caractères avec les données (attributID, valeur, date) |
| Tests | * sur un attributID non existant * pour un seuil jamais atteint/dépassé * pour un seuil dépassé par plusieurs valeurs sur un capteur * pour un seuil dépassé par plusieurs capteurs |

|  |  |
| --- | --- |
| Fonctionnalité F12 | Faire des alertes en temps réel |
| Description | Cette fonctionnalité permet d’envoyer à l’utilisateur des alertes si des seuil de danger sont dépassés. |
| Données nécessaires | Au moins un jeu de donnée. |
| Pré conditions | Le jeu de données doit avoir des valeurs supérieures au moins au premier seuil de danger. |
| Post conditions | RAS |
| Données en sortie | Envoie d’une alerte |
| Tests | * Tester que il y a bien une alerte envoyée à chaque dépassement de seuil * Vérifier qu’il y a bien une seule alerte envoyée par dépassement de seuil et pas une autre à chaque donnée supérieur |

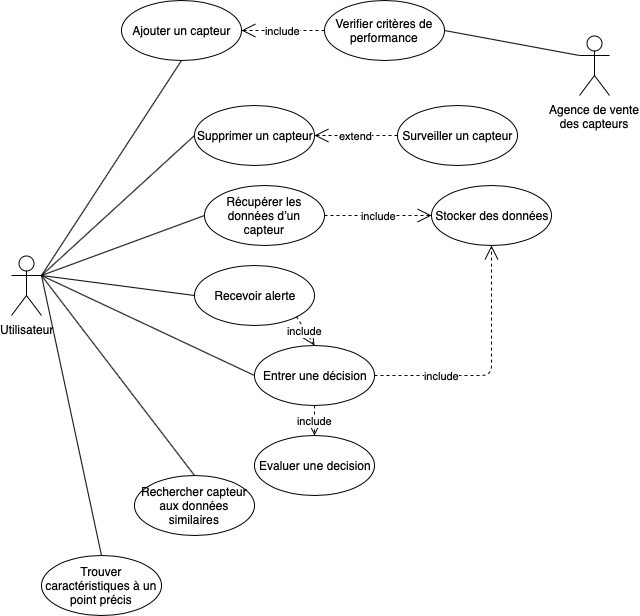
|  |  |
| --- | --- |
| Fonctionnalité F13 | Entrer une décision |
| Description | Cette fonctionnalité permet après une alerte passer à l’utilisateur de rentrer un texte avec la décision qu’il a pris. |
| Données nécessaires | Décision que l’utilisateur a pris. |
| Pré conditions | L’utilisateur doit avoir reçu une alerte. |
| Post conditions | RAS |
| Données en sortie | RAS |
| Tests | * Tester si on rentre 2 fois la même décision. * Tester si on rentre une décision qui n'est pas de type texte. |

|  |  |
| --- | --- |
| Fonctionnalité F14 | Evaluer décision |
| Description | Cette fonctionnalité permet de vérifier si une décision qui a été prise était bien ou non et ainsi d’attribuer une note à la décision. Cette fonction est appelée de manière automatique (on doit déterminer combien de temps après) |
| Données nécessaires | Une décision et des données enregistrées après la décision. |
| Pré conditions | Il faut que le capteur qui envoie les données soit toujours viable après l’envoie de la décision |
| Post conditions | RAS |
| Données en sortie | Note sur la décision |
| Tests | * Tester si on veut noter une décision inexistante * Tester si on a rentré une décision puis que les données enregistrées sont bonne qu’elle soit bien notée. * Tester si on a rentré une décision puis que les données enregistrées sont mauvaise qu’elle soit mal notée. |

|  |  |
| --- | --- |
| Fonctionnalité F15 | Proposer décision |
| Description | Cette fonctionnalité permet de proposer une décision à prendre si l’utilisateur le souhaite lors de l’envoie d’alertes. |
| Données nécessaires | Une base de donnée avec toutes les décision prises dans le passé ainsi que les notes qu’elles ont reçues. |
| Pré conditions | Il faut que l’utilisateur ai reçu une alerte. |
| Post conditions | RAS |
| Données en sortie | Envoie d’une proposition de décision à l’utilisateur |
| Tests | * Tester que l’on n’envoie pas des décision mal notée. * Tester si on a pas de bonne proposition à envoyer. |

|  |  |
| --- | --- |
| Fonctionnalité F16 | Calculer la moyenne des valeurs d'un attribut d'un capteur sur une durée donnée |
| Description | Cette fonctionnalité permet de consulter le contenu du document avec la liste des tous les capteurs et attributs |
| Données nécessaires | ID du capteur et 2 dates (non identiques et existantes) |
| Pré conditions | Le document concernant les données doit être renseigné correctement et non vide |
| Post conditions | RAS |
| Données en sortie | Chaîne de caractères avec l’ID du capteur, l’ID de son attribut concerné, les 2 dates entre lesquelles ont été réalisées les mesures, et la moyenne des valeurs ainsi que le nombre de valeurs prises en compte pour le calcul. |
| Tests | Cas d’exceptions :   * Avec deux dates inexistantes / invalides * Avec deux dates identiques (et donc pas d’intervalle entre les 2) * Avec un capteur invalide (ID inexistant ou aucune mesure effectuée par le capteur) * Avec un attribut non valide  (ID inexistant ou aucune valeur de mesure renseignée pour le capteur donné) |

* + Spécifications non fonctionnelles de l’application :
* Coûts : des capteurs, de l'entretien, de la maintenance …
* Fiabilité : utilisation des indices Atmo pour évaluer les données reçues
* Intégrabilité : facile d’ajouter des capteurs
* Sécurité : les données récupérées ne sont accessibles qu’aux utilisateurs de l'application
* Robustesse : en présence de données invalides par un capteur supposé défaillant
* Performance : actualisation des données fréquentes
* Evolutivité : l’application peut évoluer en ajoutant des fonctionnalités et des attributs aux données des capteurs
* Diagramme de Use Case :



1. Manuel utilisateur

**Manuel Utilisateur application Capt’air**

Version 1, 12/04/2019

**Généralités**

              L’application Capt’air permet de contrôler le bon fonctionnement et d’utiliser les données de capteurs surveillant la qualité de l’air installés sur un territoire. Cette application est destinée à des employés gouvernementaux et s’exécute sur terminal. Une fois exécutée, un menu textuel s’affiche sur le terminal et propose 3 catégories d’actions : 1) Gestion des capteurs, 2) Données capteur et 3) Paramètres d’alertes. L’utilisateur peut renseigner le type d’action qu’il souhaite effectuer en entrant 1, 2 ou 3 sur son clavier. S’affiche alors les différentes fonctionnalités proposées dans celui-ci. Les fonctionnalités de chacune des catégories sont précisées ci-après.

A chaque fois l’utilisateur indique la fonctionnalité qu’il veut utiliser en rentrant le numéro correspondant et renseigne les différents éléments demandés également par une entrée clavier sur le terminal.

Après l’exécution de la fonction, le menu textuel est de nouveau affiché.

A tout moment l’utilisateur peut revenir de l’onglet 1, 2 ou 3 au menu de départ en entrant le numéro correspondant au « Retour au Menu ».

**Gestion des capteurs**

Dans cet onglet, 6 fonctions sont proposées :

* « Consulter liste des capteurs » affiche la liste de tous les capteurs ainsi que leurs coordonnées et description.
* « Ajouter », qui permet d’ajouter un capteur au circuit afin de le rendre fonctionnel et de pouvoir récupérer ses données. Après avoir sélectionné cette fonction, les caractéristiques (SensorID, latitude, longitude et description) du capteur sont demandées afin de pouvoir l’ajouter au circuit. Puis un message de confirmation ou d’erreur indique si l’ajout a été effectué avec succès.
* « Supprimer » permet de supprimer un capteur du circuit. Après avoir sélectionné cette fonction, il suffit de renseigner le SensorID du capteur à supprimer puis de confirmer sa suppression pour le retirer du circuit. Puis un message de confirmation ou d’erreur indique si l’ajout a été effectué avec succès.
* « Surveiller » permet de vérifier la fiabilité d’un capteur. Après avoir sélectionné cette fonction, il faut renseigner le SensorID du capteur dont on veut vérifier le fonctionnement pour que celui-ci soit testé. La viabilité des données du capteur, du capteur lui-même et sa dernière date d’actualisation (qui doit être inférieures à 2 jours avant la date d’exécution de cette demande) sont vérifiés, et un message indique si le capteur est toujours fonctionnel et sinon pourquoi (absence de données, ou pas d’actualisation récente etc.).
* « Rechercher capteur » : on renseigne des coordonnées c’est à dire une valeur de latitude et une valeur de longitude. Le SensorID du capteur le plus proche de ces coordonnées est affiché.
* « Capteurs similaires » : on renseigne un SensorID, un intervalle de confiance et les capteurs (ID, coordonnées et description) ayant un comportement similaire au capteur renseigné à l’intervalle près sont affichés.

**Données capteurs**

Dans cet onglet, 5 fonctions sont proposées :

* « Consulter liste des attributs » affiche l’ensemble des attributs renseignés ainsi que leur unité et description.
* « Consulter les données » affiche l’ensemble des données (date, SensorID, AttributeID ainsi que valeur) présentes dans le fichier des data.
* « Consulter des données » : après avoir sélectionné cette fonction, il y a deux cas possibles, soit on choisit de renseigner le nombre de capteurs soit l’intervalle de latitude-longitude dont on veut consulter les données. On indique ensuite les capteurs ou les coordonnées concernés. Puis on peut décider de renseigner ou non un intervalle de date dans lequel récupérer les données ou bien les attributs dont on souhaite savoir les valeurs (sinon tous les attributs et leur valeur seront affichés). Les données correspondant à notre recherche sont ainsi affichées.
* « Recueillir données similaires » : après avoir sélectionné cette fonction, il faut renseigner un SensorID, un AttributID, une date ainsi qu’un intervalle de différence. Les données similaires à la valeur de l’attribut du capteur à la date précisée pour l’intervalle de différence renseigné seront affichées.
* « Calculer moyenne de valeurs » : après avoir sélectionné cette fonction,  il faut de renseigner un sensorID, un attributID ainsi que 2 dates (non identiques et existantes). La moyenne des valeurs de l’attribut dans l’intervalle de temps et pour le capteur donnés est indiquée.

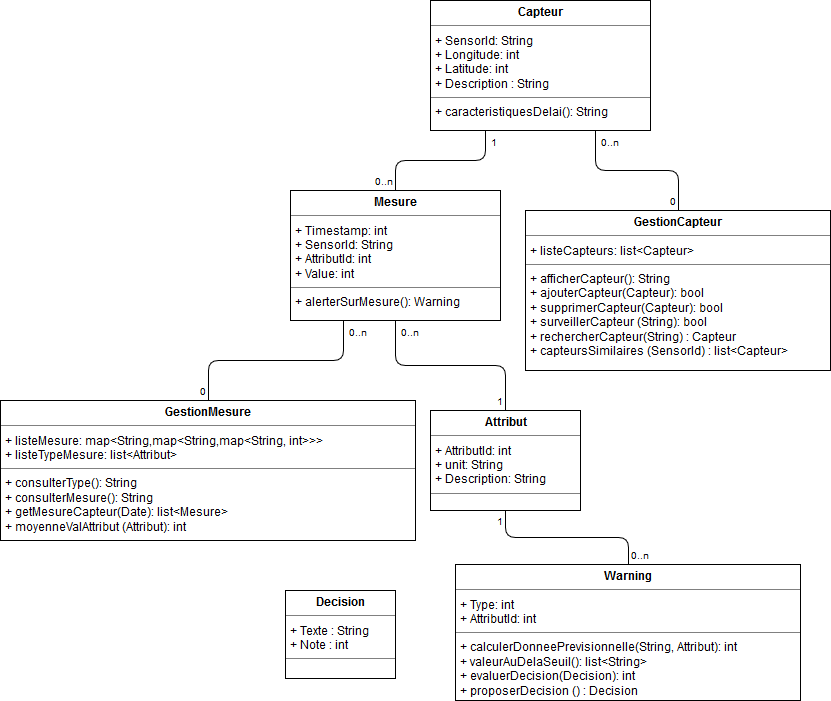
**Alertes et seuil**

Dans cet onglet, les fonctions proposées sont :

* « Calculer les données prévisionnelles » : après avoir sélectionné cette fonction,  il faut renseigner un sensorID et un attributID. Des données prévisionnelles pour cet attribut et ce capteur sont calculées sur les 7 prochains jours et affichés.
* « Données au-delà d’un seuil » : après avoir sélectionné cette fonction,  il faut renseigner un attributID, une valeur seuil et sélectionner si l’on souhaite des valeurs au-dessus ou en-dessous ce seuil. Les données correspondantes, au-delà du seuil pour l’attribut donné, sont affichées.

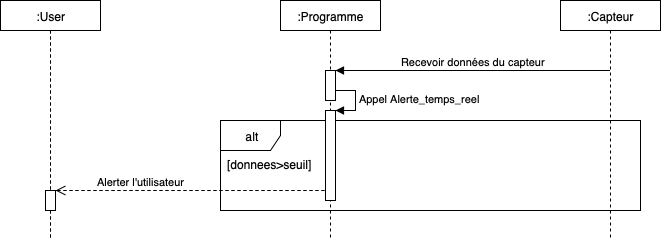
Des fonctionnalités « Faire des alertes en temps réel », « Envoyer une décision », « Évaluer une décision », « Proposer une décision » sont actuellement en réflexion et ne sont donc pas présentées dans cette première version du guide utilisateur.

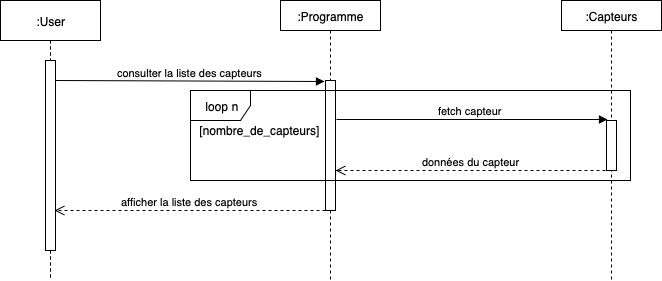
1. Diagramme de classes

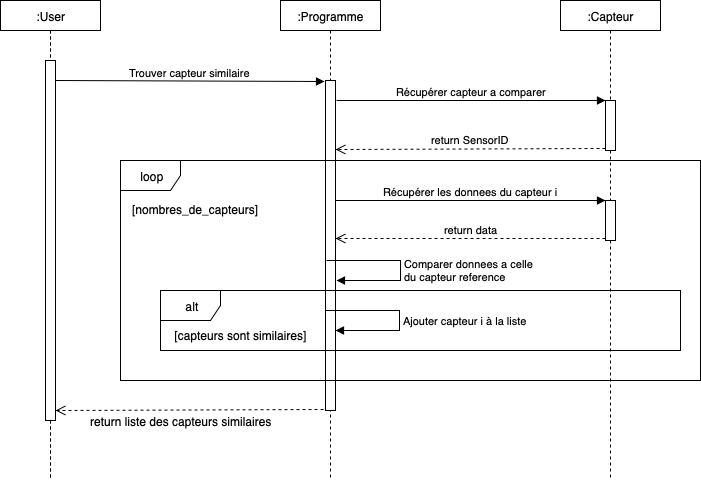


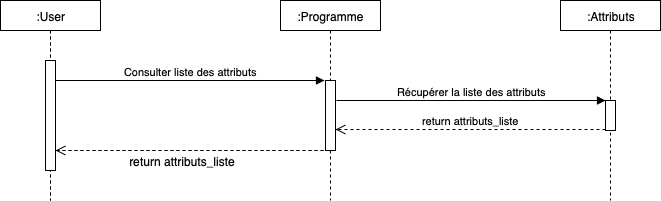
**Diagramme de classe de l’application Capt’Air**

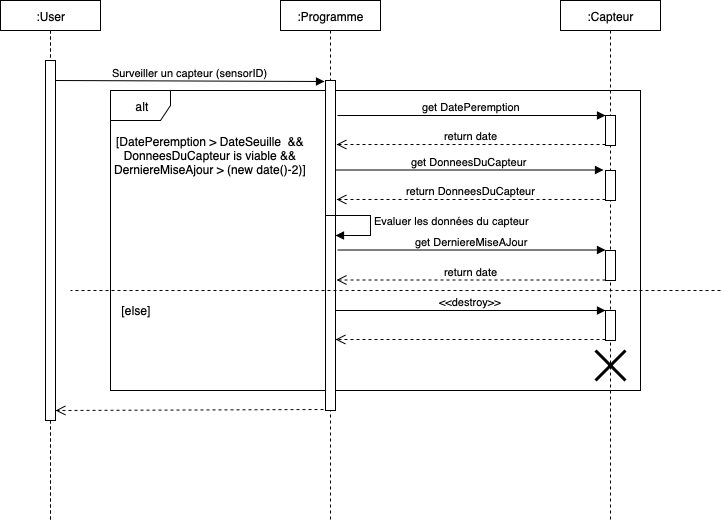
1. Diagrammes de séquence

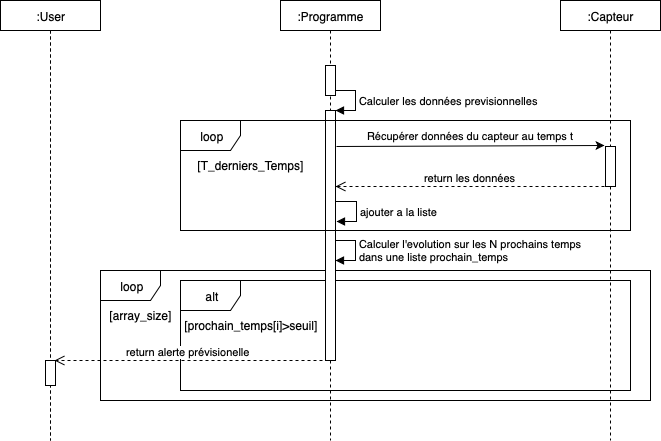


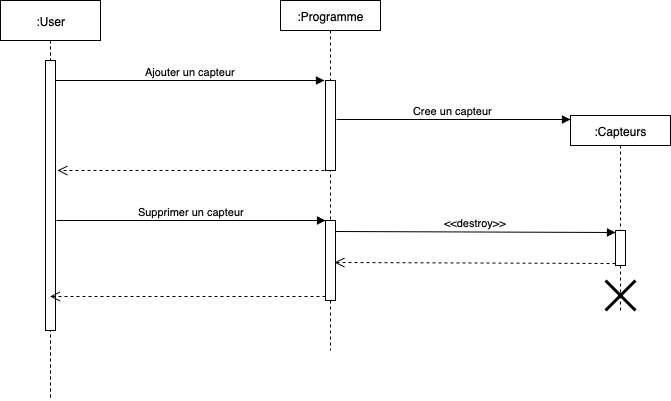




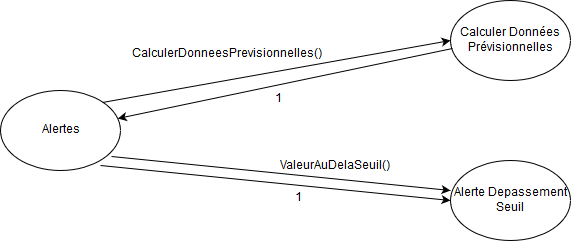


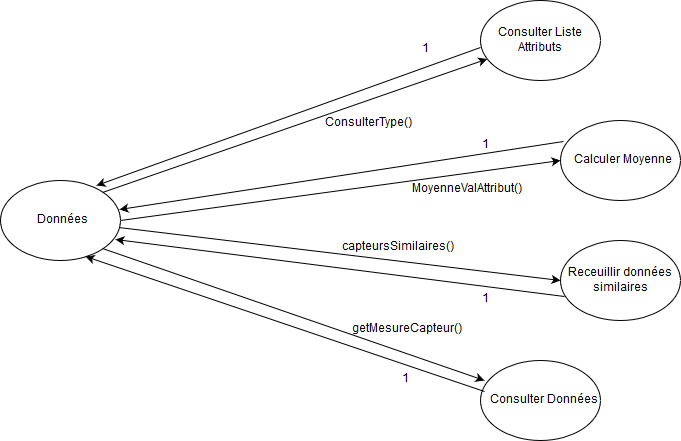


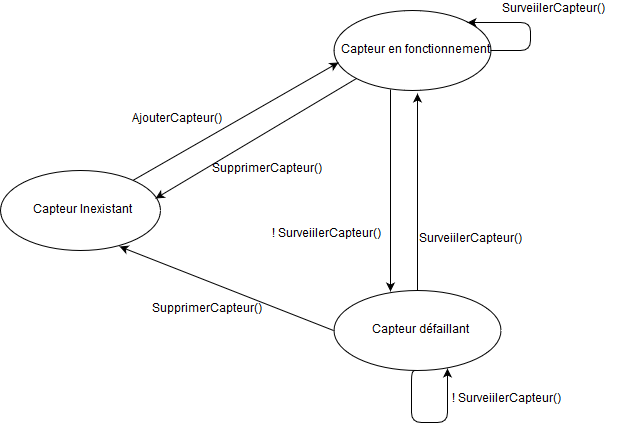


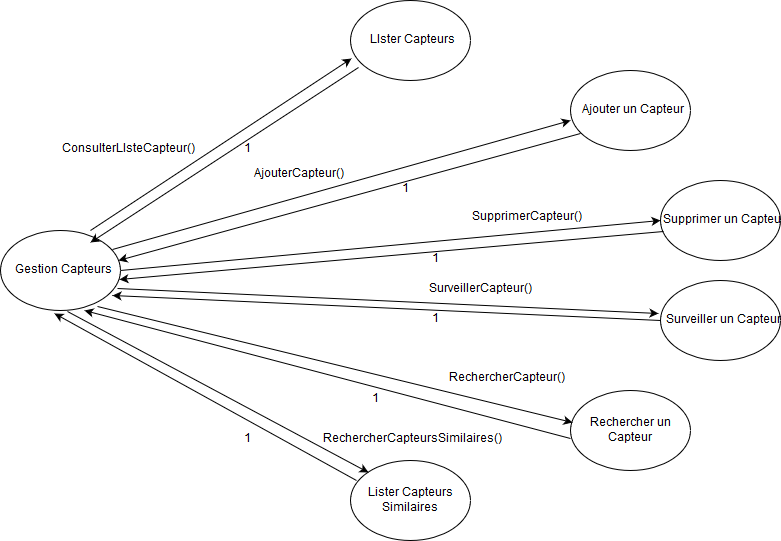


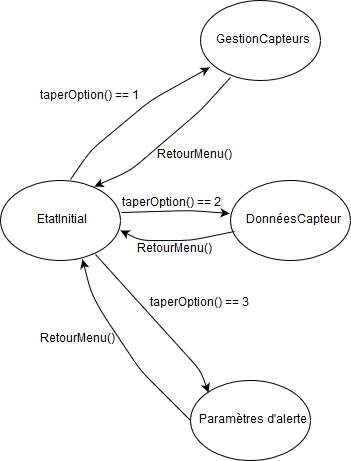
1. Diagrammes d’états transitions











1. Décomposition en modules

Pour une organisation du code optimal, nous divisions notre application en différents modules :

* + Package *Lecture*, qui contient les algorithmes de lecture de fichiers.
  + Package *Données*, qui contient les classes avec les structures de données mises en mémoire dans notre programme.
  + Package *Algorithmes*, avec toutes les fonctionnalités et algorithmes internes à l’application (autre que ceux de lecture).

1. Description des algorithmes importants

**moyenneValAttributs(attribut)**

Entrée : l’ID d’un des attributs des capteurs

Sortie : L’entier indiquant la moyenne des valeurs relevées en rapport avec cet attribut depuis le début des mesures

Si ( attributID n’est pas un des 4 conformes)

    Return -1

Sinon

    Parcourir la map globale pour tous les capteurs, puis pour l’attribut sélectionné, faire la moyenne de toutes les valeurs dont la clé est cet AttributeID

**calculerDonneesPrevisionnelles(String, Attribut)**

Entrée : un String représentant l’ID du capteur à tester

     un Attribut à tester

Sortie : Un entier représentant la prochaine valeur prévue de l’attribut

Récupérer les 5 dernières valeurs avec un reverse\_iterator, puis ensuite récupérer les différences successives entres ces valeurs. Calculer la moyenne de ces différences, pondérées avec leur éloignement dans le temps, et l’ajouter à la dernière valeur.

*Proposition de coefficients :*

D(n+1) = 0.5\*D(n)+0.25\*D(n-1)+0.15\*D(n-2)+0.1\*D(n-3)

avec D(n) la différence entre la dernière (n-ième) valeur et celle qui la précède.

V(n+1) = V(n)+D(n+1)

    avec V(n) la dernière (n-ième) valeur.

**capteursSimilaires(SensorId)**

Entrée : un String représentant l’ID du capteur à tester

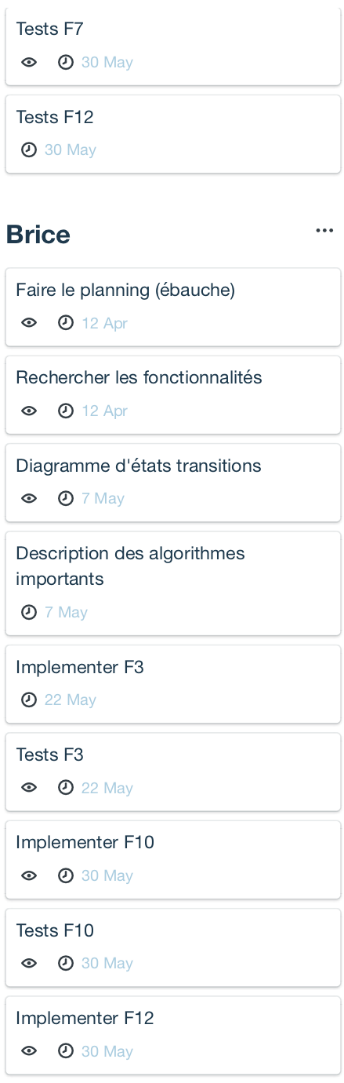
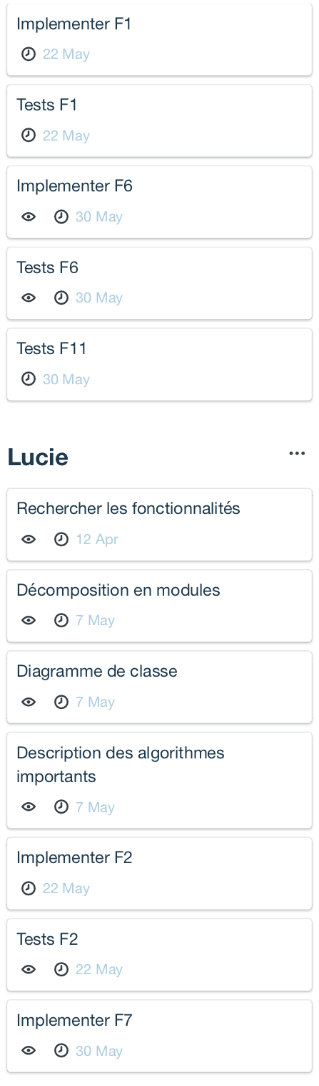
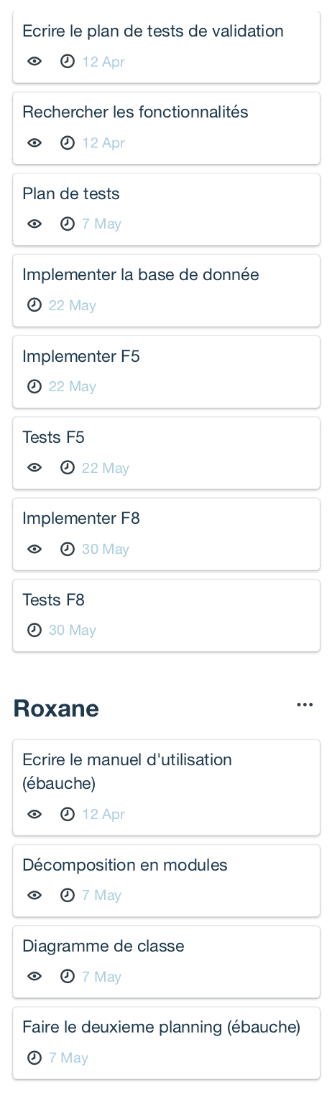
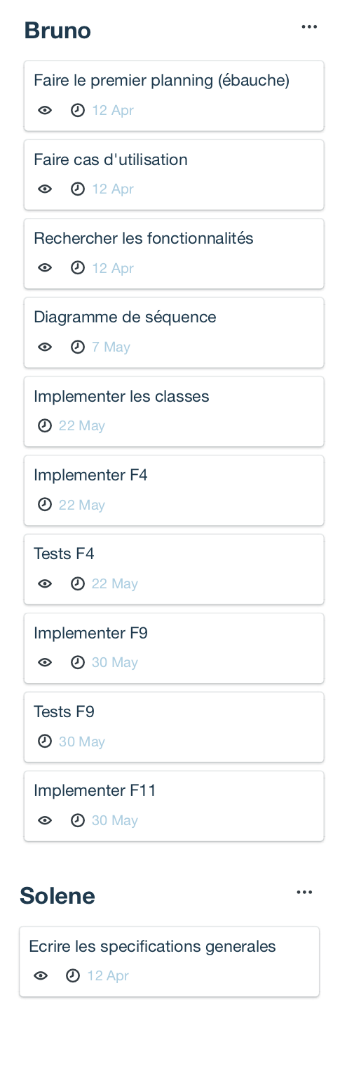
Sortie : liste de Capteurs similaires à ce capteur

Comparer les 5 dernières valeurs avec tous les capteurs pour les attributs, faire la racine carré de la somme des écarts au carré, puis regarder si elle dépasse un seuil (à fixer) pour chaque attribut. Si non, rajouter le capteur testé dans la liste, sinon ne pas le rajouter.

1. Planning

Afin de réaliser le planning provisoire, nous avons choisis de travailler avec un logiciel spécialisé dans la gestion de projet : *Trello*.

Nous avons créé une liste par membre du groupe, où nous avons ajouté des tâches (qui sont commentées et ont toute une date limite). De plus nous avons créé des listes de tâches à faire (actuellement vide) afin de pouvoir rajouter au fur et à mesure des tâches qui ne sont pas encore prévues (réparer un bug par exemple) et les attribuer au membre le plus “libre” ou le plus “motivé”.



A CHANGER

Tous les timestamp etc. en time\_t et #include<time>

Tous les IDs en String et non plus en int.