RÉGION ACADÉMIQUE
PAYS DE LA LOIRE
*Liberté*
*Égalité*
*Fraternité*

Lycée TOUCHARD WASHINGTON

# Dossier de conception

## BTS CIEL SESSION 2025 - IR
## E6 – PROJET TECHNIQUE

### Système de Surveillance météorologique connecté

| | | |
|---|---|---|
| Lycée : **Touchard Washington** | Ville : **LE MANS** | |
| Nom du projet : | **Système de surveillance météorologique connecté** | |
| N° du projet : | **TW3** | |
| Projet nouveau | Oui ☐ Non▣ | Projet interne : Oui▣ Non☐ |
| Délai de réalisation | **150 heures** | Statut des étudiants : Formation initiale ▣ Apprentissage ☐ |
| Spécialité des étudiants | ER☐ IR▣ Mixte☐ | Nombre d'étudiants : **3 étudiants** |
| Professeurs responsables | **Philippe CRUCHET**, François MARTIN, Anthony LE CREN, Jilali KHAMLACH, Saïd LAHSIKA. | |

# Sommaire

# ***Introduction***

Après une analyse approfondie menée dans le cadre du dossier d'analyse, ce dossier de conception vise à concrétiser les orientations, besoins et contraintes identifiés. L'objectif est de proposer une solution détaillée, structurée et adaptée aux exigences fonctionnelles, techniques et opérationnelles préalablement définies.

Au sein du dossier d'analyse nous avons pu identifier les éléments nécessaires au déroulé du projet. Maintenant que nous connaissons les différentes données que nous allons traiter, et que nous avons une idée des matériels que nous allons avoir à utiliser, il va être désormais possible de faire des choix techniques en lien avec ceux-ci.

Ce document constitue une étape essentielle du projet, traduisant les conclusions théoriques en une vision concrète et opérationnelle. Il détaille les choix de conception retenus, leur justification, ainsi que leur mise en œuvre prévue.

En s'appuyant sur les conclusions du dossier d'analyse, ce document a pour ambition de poser les bases solides nécessaires à la réalisation et au succès du projet.

**Le code couleur proposé lors du dossier d'analyse pour l'identification des étudiants sera toujours utilisé pour ce document*

# Étude des besoins

La phase d'analyse réalisée en amont a permis d'identifier et de formaliser les besoins liés au projet. Ces besoins constituent la base fondamentale sur laquelle repose la conception de la solution. Ils traduisent les attentes fonctionnelles, techniques et opérationnelles des parties prenantes, ainsi que les contraintes spécifiques à respecter.

Dans cette section, nous récapitulons les besoins essentiels qui guideront la conception et la mise en œuvre du projet. Chaque besoin est présenté de manière structurée afin d'assurer une traçabilité entre les exigences initiales et les choix de conception décrits ultérieurement.

## Serveur Raspberry

- Base de données : la base de données est donc constituée des tables décrites dans le dossier d'analyse. Elle sera stockée dans le serveur Raspberry Pi.
- Traitement des données : le traitement des données sera effectué par le serveur Raspberry Pi. Le logiciel qui fera le traitement devra donc pouvoir communiquer avec des requêtes SQL pour la base de données et être compatible avec la Raspberry.
- Communication : la communication entre les capteurs et le serveur Raspberry sera effectuée grâce à une clé RTL.

## Clients Web & Android

- IHM : les IHM sont une partie essentielle du projet puisque celles-ci permettent une compréhension plus aisée des utilisateurs. Celles-ci vont être différentes en fonction de la plateforme par laquelle l'utilisateur accède aux données.
    - Web: les clients auront accès à une interface qui permettra la lecture des données historiques. Elle sera composée de plusieurs pages qui représentent différentes données (par le biais de différents graphiques).
    - Android : les clients auront accès à une interface qui permettra la lecture en temps réel des informations. Elle s'adapte à la taille de l'écran, permettant une adaptabilité aux différents appareils sous Android.

## Serveurs Web

- Les deux serveurs jouent un rôle d'intermédiaire entre l'interface utilisateur et la base de données, en s'assurant que les requêtes sont correctement traitées et sécurisées.

- Le serveur météo fait la gestion des données météorologiques, il récupère les données du graphique présenté dans la base de données, pour ensuite les exporter vers le client.
- Le serveur Utilisateur fait la gestion des utilisateurs en assurant l'authentification, la création du compte ainsi que l'accès à la session de l'utilisateur.

# Comparaison des outils

## Comparaison QtCreator et Eclipse

| Critères | QtCreator | Eclipse |
|---|---|---|
| **Performances** | ✅ Léger et rapide | ❌ Peut être lourd avec des plugins |
| **Facilité d'utilisation** | ✅ Interface claire, orientée développement **C++** et **Qt**. | ❌ Interface plus complexe, surtout avec de nombreux plugins installés.<br>Plus adapté aux développeurs travaillant sur des projets Java de grande envergure. |
| **Langages supportés** | ✅ C++, QML, Python, JavaScript | ✅ Java, C, C++, Python, PHP, etc. |
| **Plugins et extensions** | ❌ Moins extensible, mais bien intégré à Qt | ✅ Très extensible avec de nombreux plugins |
| **WebSockets** | ✅ Bibliothèques avec WebSockets inclus | ❌ Pas adapté en natif mais possibilité via des plugins |
| **Console** | ✅ Console intégrée, accès aux logs, affichages des résultats d'exécution.<br>- Supporte la sortie console et les messages d'erreur lors du débogage. | ❌ Console pour afficher les logs, erreurs et résultats d'exécution.<br>Débogage et affichage des erreurs plus centrées sur Java. |
| **JSON** | ✅ Supporte **JSON** via la bibliothèque **QJson** facilite la manipulation des fichiers JSON, et l'intégration dans les applications C++.<br>- Parfaitement adapté pour des interactions avec des données JSON (API REST, fichiers de configuration, etc.). | ❌ Pas adapté en natif mais possibilité via des plugins<br>Peut être plus complexe à configurer pour des projets nécessitant une gestion fine de JSON dans des langages autres que Java. |

En vue des problématiques à résoudre, notre choix va se porter vers QtCreator. Ce logiciel est plus adapté, car il intègre toutes les fonctions nécessaires à l'élaboration de notre projet, sans l'ajout d'aucun plugin. De plus, nous avons déjà développé sur ce logiciel, ce qui nous permet de ne pas avoir un temps d'adaptation.

## Comparaison Visual Studio Code et NetBeans

| Critère | VS Code | NetBeans |
|---|---|---|
| **Installation et Configuration** | ✅Léger, rapide à installer et à configurer | ❌Plus lourd, nécessite Java JDK pour fonctionner |
| **Langages supportés** | ✅Très polyvalent (Java, JavaScript, Python, etc.) | ❌Principalement optimisé pour Java |
| **Plugins & Extensions** | ✅Très riche avec une grande variété d'extensions | ❌Extensions limitées mais bien intégrées |
| **Support UML et Diagrammes** | ❌Extensions nécessaires pour UML (ex. "PlantUML") | ✅Outils UML intégrés pour Java |
| **Débogage** | ❌Dépend des extensions et de la configuration | ✅Débogueur intégré pour Java |
| **Intégration avec Git** | ✅Très fluide avec Git et GitHub | ❌Intégré mais moins fluide que VS Code |
| **Complétion de code et IntelliSense** | ✅IntelliSense performant, dépendant des extensions | ✅Très efficace pour Java avec NetBeans IDE |
| **Gestion de projets Java** | ❌Nécessite une configuration manuelle | ✅Gestion automatique avec Maven/Gradle |
| **Performances** | ✅Très léger, rapide | ❌Plus lourd, plus lent au démarrage |
| **Utilisation en entreprise** | ✅Plus flexible, adapté à plusieurs technologies | ❌Utilisé principalement pour Java et applications d'entreprise |

- Si l'implémentation se fait en **Java**, NetBeans est plus adapté car il offre une meilleure gestion des projets Java, un débogueur intégré et des outils UML.
- Si l'implémentation est prévue en plusieurs langages ou nécessite un IDE plus léger et flexible, **VS Code** est un meilleur choix, avec la possibilité d'ajouter les extensions UML et Java nécessaires.

Mon choix va donc se porter sur visual studio code pour un développement plus polyvalent avec plusieurs langages de programmation

# Comparaison MariaDB et MySQL

| Critères | MariaDB | MySQL |
|---|---|---|
| **Performance** | ✅ Plus rapide en lecture/écriture, meilleure gestion du cache | ❌ Performances légèrement moindres sur certaines requêtes complexes |
| **Compatibilité** | ✅ 100 % compatible avec MySQL | ✅ Standard SQL bien respecté |
| **Scalabilité** | ✅ Optimisé pour le multi-threading, meilleur support multi-core | ❌ Moins efficace sur des requêtes massives |
| **Gestion des JSON** | ✅ Support JSON amélioré (utile pour stocker des mesures météo complexes) | ❌ Support JSON mais moins optimisé |
| **Moteurs de stockage** | ✅ Plus de moteurs disponibles (ex: Aria, TokuDB, XtraDB) | ❌ Moins de diversité (ex: InnoDB par défaut) |
| **Sécurité** | ✅ Chiffrement natif, meilleure gestion des accès | ❌ Sécurité correcte mais mise à jour plus lente |
| **Licence** | ✅ Open Source (GPLv2), évolue librement | ❌ Sous Oracle, certaines fonctionnalités payantes |
| **Mises à jour** | ✅ Fréquence élevée, nouvelles fonctionnalités régulièrement | ❌ Moins de mises à jour innovantes |
| **Requêtes sur de grands volumes** | ✅ Plus rapide pour traiter de grandes quantités de données météorologiques | ❌ Peut être plus lent sur certaines jointures complexes |
| **Support de la communauté** | ✅ Grande communauté active, mises à jour fréquentes | ❌ Dépend d'Oracle, support plus restreint |

Le choix se porte plus sur MariaDB notamment pour utiliser JSON pour stocker des mesures météorologiques complexes, la sécurité et si la station météo stocke des millions de mesures (température, pression, vent, etc.)

**BERTAULT Nolhan** **7/25**

**DENEAU Maverick**

## Comparaison Figma et Adobe

| Critères | Figma | Adobe XD |
|---|---|---|
| **Mode de fonctionnement** | Basé sur le cloud | Application desktop (Windows / macOS) |
| **Collaboration en temps réel** | Oui, multi-utilisateurs | Possible, mais moins fluide (via Creative Cloud) |
| **Prototypage** | Inclus avec animations et interactions avancées | Prototypage performant, mais dépendant d'After Effects pour animations complexes |
| **Gestion des composants** | Très avancée, partage facile en équipe | Moins fluide mais efficace |
| **Plugins et Intégrations** | Très large choix (Qt Bridge, FigJam, etc.) | Moins d'options mais intégration avec Adobe Suite |
| **Performance** | Peut être limité sur des fichiers très lourds | Meilleure gestion des fichiers locaux |
| **Prix** | Version gratuite disponible, abonnement abordable | Payant après la période d'essai, inclus dans Adobe Creative Cloud |

**Verdict : Figma ou Adobe XD ?**

✅ Choisir **Figma** si vous travaillez en **équipe**, avec besoin d'un outil **basé sur le cloud**, et souhaitez une collaboration en **temps réel**. Dans mon cas je suis seul donc je peux passer de la **création** de l'IHM **au code directement**.

✅ Choisir **Adobe XD** si vous êtes déjà intégré dans l'écosystème **Adobe** que vous recherchez un outil performant pour un usage **local et plus orienté animations avancées** (via After Effects)

**Globalement, Figma est un excellent choix pour le travail collaboratif, tandis qu'Adobe XD conviendra davantage à ceux qui préfèrent un outil intégré à la suite Adobe.**

# Logiciels choisis

À la suite de cette étude de nos besoins nous avons établi une liste des outils et frameworks nécessaire afin de réaliser notre projet. Nous allons y détailler les points qui ont été la clé de notre choix.

## QT


*Image X : Logo QT*

Qt est un framework de développement logiciel multiplateforme utilisé pour créer des applications avec des interfaces graphiques utilisateurs (GUI), ainsi que des applications non-gui comme des outils de traitement de données ou des serveurs. Développé par The Qt Company, Qt est particulièrement apprécié pour sa capacité à fonctionner sur plusieurs systèmes d'exploitation, notamment Windows, Linux, macOS, et les plateformes mobiles comme Android et iOS.

### Sortie Console

Qt permet de gérer les sorties vers la console en utilisant des classes telles que QDebug, qui facilitent l'envoi de messages de débogage ou d'informations générales directement dans la console. Grâce à cette fonctionnalité, les développeurs peuvent afficher des messages d'état ou des erreurs, ce qui est utile pour le suivi et le débogage des applications pendant le développement. En outre, Qt permet de rediriger les flux de sortie (comme stdout et stderr) vers des fichiers ou d'autres périphériques, offrant ainsi plus de flexibilité dans la gestion des journaux et des messages.

### WebSockets

Qt prend en charge les WebSockets, qui sont utilisés pour établir des connexions bidirectionnelles en temps réel entre un client et un serveur via le protocole WebSocket.

Cette fonctionnalité est particulièrement utile pour les applications nécessitant des échanges de données en continu, comme les applications de chat, les jeux en ligne ou les systèmes de surveillance. En utilisant Qt, les développeurs peuvent facilement implémenter des communications WebSocket, permettant ainsi une interaction

instantanée entre l'interface utilisateur et le serveur sans avoir besoin de recharger la page ou de faire des requêtes HTTP fréquentes.

### Lecture/Écriture JSON

Qt inclut des outils puissants pour la gestion de données en format JSON. La classe QJsonDocument permet de lire et d'écrire des données au format JSON, ce qui est couramment utilisé pour l'échange de données entre une application cliente et un serveur.

Les développeurs peuvent facilement analyser des chaînes JSON, les convertir en objets Qt (comme des QVariant), et vice versa. Cette fonctionnalité est idéale pour les applications qui communiquent avec des API REST ou pour stocker et échanger des données structurées.

Qt facilite ainsi le traitement des données JSON tout en offrant une interface simple et efficace pour l'intégration avec des systèmes externes.

### Lecture de données en entrée d'un port

Qt offre une prise en charge avancée pour la lecture de données provenant de ports série (RS232) ou d'autres ports de communication via la classe QSerialPort. Cette fonctionnalité est utilisée pour interagir avec des périphériques matériels, comme des capteurs, des instruments de mesure, ou des appareils électroniques. Les applications Qt peuvent ainsi lire des données en temps réel à partir de ces périphériques, ce qui est essentiel dans les systèmes embarqués, l'automatisation industrielle, ou pour toute autre application nécessitant la collecte de données depuis un matériel externe. La classe QSerialPort permet de configurer les paramètres du port (comme la vitesse de transmission, la parité, etc.) et de gérer les échanges de données de manière synchrone ou asynchrone.

# QT DESIGN STUDIO


*Image X : Logo Qt Design Studio*

Qt Design Studio est un outil de conception UI/UX permettant de créer des interfaces graphiques interactives et dynamiques pour des applications Qt. Il est spécialement conçu pour les designers et développeurs qui souhaitent travailler ensemble efficacement sur des interfaces utilisateur modernes, notamment pour les applications embarquées, de bureau et mobiles.

**Qt Design Studio & Figma : Une intégration parfaite pour le Design UI/UX**

L'intégration entre Qt Design Studio et Figma permet de convertir rapidement des maquettes UI en interfaces fonctionnelles, en générant automatiquement du code QML à partir des designs. Cela facilite énormément le travail collaboratif entre designers et développeurs, tout en accélérant le processus de développement

**Pourquoi Qt Design Studio et Figma fonctionnent bien ensemble ?**

Figma est un excellent outil pour concevoir des interfaces UI/UX, tandis que Qt Design Studio est spécialisé dans leur conversion en interfaces interactives Qt. Leur compatibilité permet :

- Une transition fluide du design au développement
- L'exportation automatique des éléments graphiques (boutons, images, icônes, textes)
- La conversion du design en code QML sans devoir tout recréer à la main
- Une collaboration optimale entre designers et développeurs

**Processus d'Intégration Figma → Qt Design Studio**

***Étape 1 : Création du Design sur Figma***

- Utilisation des composants UI standards (boutons, cartes, formulaires, etc.)
- Organisation en frames et layers pour structurer l'interface
- Définition des styles (couleurs, typographie, icônes, etc.)

***Étape 2 : Exportation vers Qt Design Studio***

Qt Design Studio propose un plugin officiel pour Figma lui permettant les designs.

1. Installer le plugin Figma → Qt Bridge
2. Sélectionner les composants à exporter (frames)
3. Générer le fichier d'export (format *.qtbridge*)

**Étape 3 : Importation dans Qt Design Studio**

- Ouvrir Qt Design Studio et importer le fichier *.qtbridge*
- Tous les éléments graphiques sont convertis en QML
- Les composants sont réutilisables dans le projet Qt

**Avantages Concrets de cette intégration**

***Automatisation du Workflow***

- Fini le travail manuel de recréation des interfaces
- Le design et l'implémentation restent fidèles sans perte de qualité
- Mise à jour rapide des modifications (pas besoin de tout refaire)

***Accélération du Développement***

- Le code QML est généré directement à partir du design
- Meilleure séparation du design et du code
- Les animations peuvent être ajoutées directement après importation

***Collaboration Facilitée***

- Les designers utilisent Figma sans se soucier du code
- Les développeurs récupèrent un code prêt à l'emploi dans Qt Design Studio
- Moins de friction et d'erreurs entre les équipes

**Cas d'Utilisation**

Application mobile avec interface moderne

**Limitations et Points d'Attention**

⚠️**Pas encore une conversion 100% parfaite :** certains éléments complexes nécessitent des ajustements

⚠️**Animations non exportés de Figma :** elles doivent être recréées dans Qt Design Studio

⚠️**Les contraintes et layouts doivent être bien pensés sur Figma** pour un export propre

# FIGMA

*Image X : Logo Figma*

Figma est un outil de conception UI/UX puissant et largement adopté, notamment pour sa collaboration en temps réel, son accessibilité et sa flexibilité.

**Basé sur le Cloud**

Contrairement à d'autres outils comme Adobe XD ou Sketch, Figma ne nécessite aucune installation : il fonctionne directement depuis un navigateur. Cela permet aux équipes de travailler sur n'importe quel appareil, sans se soucier des mises à jour ou de la compatibilité avec un système d'exploitation spécifique.

**Collaboration en temps réel**

Figma fonctionne comme Google Docs : plusieurs utilisateurs peuvent modifier le même projet simultanément, avec un suivi en direct des modifications. Cela facilite la communication entre designers, développeurs et chefs de projet.

**Compatibilité et intégrations**

- **Plugins variés :** Accès à une vaste bibliothèque de plugins (Icônes, illustrations, générateurs de maquettes, etc..)
- **Exportation simplifiée :** Compatible avec de nombreux formats (SVG, PNG, PDF, etc..) et intègre des outils comme Qt Bridge for Figma pour Qt Design Studio

**Version gratuite et puissante**

Contrairement à Sketch qui est payant, ou Adobe XD qui est limité sans abonnement, Figma propose une version gratuite très complexe avec la possibilité de travailler en équipe sans coût initial.

# MariaDB


*Image X : Logo MariaDB*

MariaDB est un système de gestion de base de données open source créé communautairement sur la base de MySql, ce qui le rend compatible, et est géré par la société Monty Program AB.

**Moteurs de stockage différents**

MariaDB permet de choisir comment les données sont stockées. Le moteur par défaut est InnoDB, qui est rapide et fiable, mais il existe d'autres options comme Aria (plus rapide pour certaines tâches) et Memory (qui stocke les données en mémoire pour des accès très rapides).

**Réplicabilité**

MariaDB permet d'augmenter la capacité de son système, et permet de répliquer les données sur plusieurs serveurs. Cela signifie qu'une copie des données peut être stockée sur différents serveurs pour assurer une meilleure disponibilité et des performances améliorées.

**Performance**

MariaDB est optimisé pour gérer des bases de données volumineuses et des requêtes complexes tout en étant rapide. Elle inclut des outils qui permettent d'améliorer les performances comme des caches pour stocker les résultats de requêtes fréquentes, ce qui réduit le temps nécessaire pour y accéder.

**Facilité d'utilisation** :

MariaDB peut être utilisée avec des outils familiers comme **phpMyAdmin**, qui te permet de gérer la base de données via une interface graphique simple, sans avoir à écrire des lignes de code.MariaDB a ajouté des fonctionnalités qui ne sont pas présentes dans MySQL, comme de meilleures options de recherche textuelle, des outils pour gérer de

grandes quantités de données temporelles, et des moyens d'optimiser encore plus les performances.

# RTL_433

RTL_433 est un logiciel open-source conçu pour recevoir et décoder des signaux sans fil provenant de capteurs variés tels que ceux utilisés pour la météorologie, les systèmes de mesure de consommation d'énergie, les capteurs de mouvement, ou les capteurs de température. Il fonctionne principalement avec des récepteurs SDR (Software Defined Radio), tels que le récepteur RTL-SDR, qui permet de capter une large gamme de fréquences radio, notamment celles proches de 433 MHz.

**Réception de signaux radio :**
Le logiciel permet la réception de signaux sans fil émis par divers appareils, comme des stations météo ou des capteurs d'énergie. Ces signaux sont transmis sur des fréquences telles que 433 MHz, mais RTL_433 peut également gérer d'autres fréquences.

**Décodage de protocoles multiples :**
RTL_433 est capable de décoder une grande variété de protocoles utilisés par des dispositifs sans fil. Cela inclut des dispositifs comme des stations météo, des détecteurs de mouvement, des capteurs de porte, et bien d'autres. La capacité à décoder une multitude de protocoles en fait un outil polyvalent pour de nombreuses applications.

**Interface avec RTL-SDR :**
Le logiciel est conçu pour fonctionner avec un récepteur RTL-SDR, un appareil compact et économique qui se branche à un port USB de l'ordinateur. Ce récepteur capte les ondes radio, permettant à RTL_433 de les analyser et de les décoder.

**Affichage des données :**
Après avoir décodé les signaux radio, RTL_433 affiche les données sous une forme lisible, comme des fichiers texte ou JSON. Cela permet aux utilisateurs de visualiser facilement les informations collectées, telles que les températures, l'humidité, ou la consommation d'énergie, selon le type de capteurs utilisés.

**Compatibilité avec plusieurs systèmes d'exploitation :**
RTL_433 est compatible avec les principaux systèmes d'exploitation, y compris Linux, Windows et macOS, offrant ainsi une grande flexibilité dans son utilisation.

**Communauté active et mises à jour régulières :**
Le projet étant open-source, il bénéficie de la contribution active de sa communauté. Les utilisateurs peuvent soumettre des améliorations, des ajouts de protocoles et des corrections de bugs. Le logiciel reçoit ainsi régulièrement des mises à jour pour étendre la compatibilité avec davantage de capteurs et améliorer ses performances.

# GitHub

*Image X : Logo GitHub*

GitHub est une plateforme de développement collaboratif basée sur Git, un système de contrôle de version distribué. Il permet aux développeurs de travailler ensemble sur des projets logiciels en facilitant le suivi des modifications apportées au code source.

Celui-ci nous offre plusieurs options très intéressante tel que :

**Contrôle de version**

Puisque GitHub est basé sur Git, les options de suivi des versions sont disponibles. Elles nous permettront d'ajouter de nouveau code tout en ayant toujours accès aux anciennes versions (en cas d'erreur de manipulation ou d'oubli de sauvegarde).

**Collaboration**

GitHub permet un travail simultané sur un même projet. La création de branches permet à différents développeurs de travailler sur la même fonctionnalité sans pour autant entrer en conflit. Il offre aussi un outil permettant de signaler des problèmes sur un code pour qu'une amélioration soit suggérée.

**Accès**

GitHub étant un outil disponible sur tous les navigateurs, les projets peuvent être poursuivis à distance tant que vous êtes connecté à Internet. L'accès peut aussi être public (accès à tout le monde) en cas de création d'un code open source par exemple ou en privé.

**Liberté**

GitHub permet le transfert de fichiers, car celui-ci ne restreint pas les fichiers en fonction de leurs type, il est possible de partager du code, des images et du texte sans problème.

# Visual Studio Code


*Image X : Logo VSCode*

Visual Studio est un environnement de développement intégré (IDE) développé par Microsoft.
C'est un outil puissant utilisé par les développeurs pour créer une variété d'applications, notamment des logiciels de bureau, des applications Web, des applications mobiles, des jeux, et bien plus encore. Voici une explication détaillée des principales caractéristiques et fonctionnalités :

**Langages de programmation pris en charge**

Visual Studio prend en charge une large gamme de langages de programmation, notamment C#, Visual Basic, C++, F#, JavaScript, TypeScript, Python, et bien d'autres.
Dans notre cadre, nous allons utiliser le C#.

**Développement multiplateforme**

Visual Studio prend en charge le développement multiplateforme, ce qui signifie que nous pouvons créer notre projet pour une variété de systèmes d'exploitation, y compris Windows, macOS, iOS, Android et Linux.

**Interface utilisateur conviviale**

L'interface utilisateur de Visual Studio est conçue pour être conviviale et intuitive. Elle offre une expérience de développement fluide avec des fonctionnalités telles que l'éditeur de code avec coloration syntaxique, la saisie semi-automatique intelligente, le débogueur intégré, et bien plus encore.

**Outils de développement intégrés**

Visual Studio propose une multitude d'outils intégrés pour faciliter le

développement d'applications. Cela comprend des outils de débogage avancés, des outils de profilage de performances, des outils de test automatisés, des éditeurs de conception pour la création d'interfaces utilisateur graphiques, des gestionnaires de packages, et des outils de déploiement.

**Extensions et personnalisation**

Visual Studio offre une grande extensibilité grâce à son système d'extensions. Les développeurs peuvent personnaliser leur environnement de développement en ajoutant des extensions tierces disponibles dans le Visual Studio Marketplace, ou même en développant leurs propres extensions.

**Collaboration et contrôle de version**

Visual Studio inclut des fonctionnalités de collaboration en équipe, notamment la prise en charge de Git pour le contrôle de version, les outils de gestion de projet agile, et les fonctionnalités de suivi des bogues et des demandes de fonctionnalités.

# ***Adaptation des Diagrammes***

## Diagramme de classe Récepteur Météo

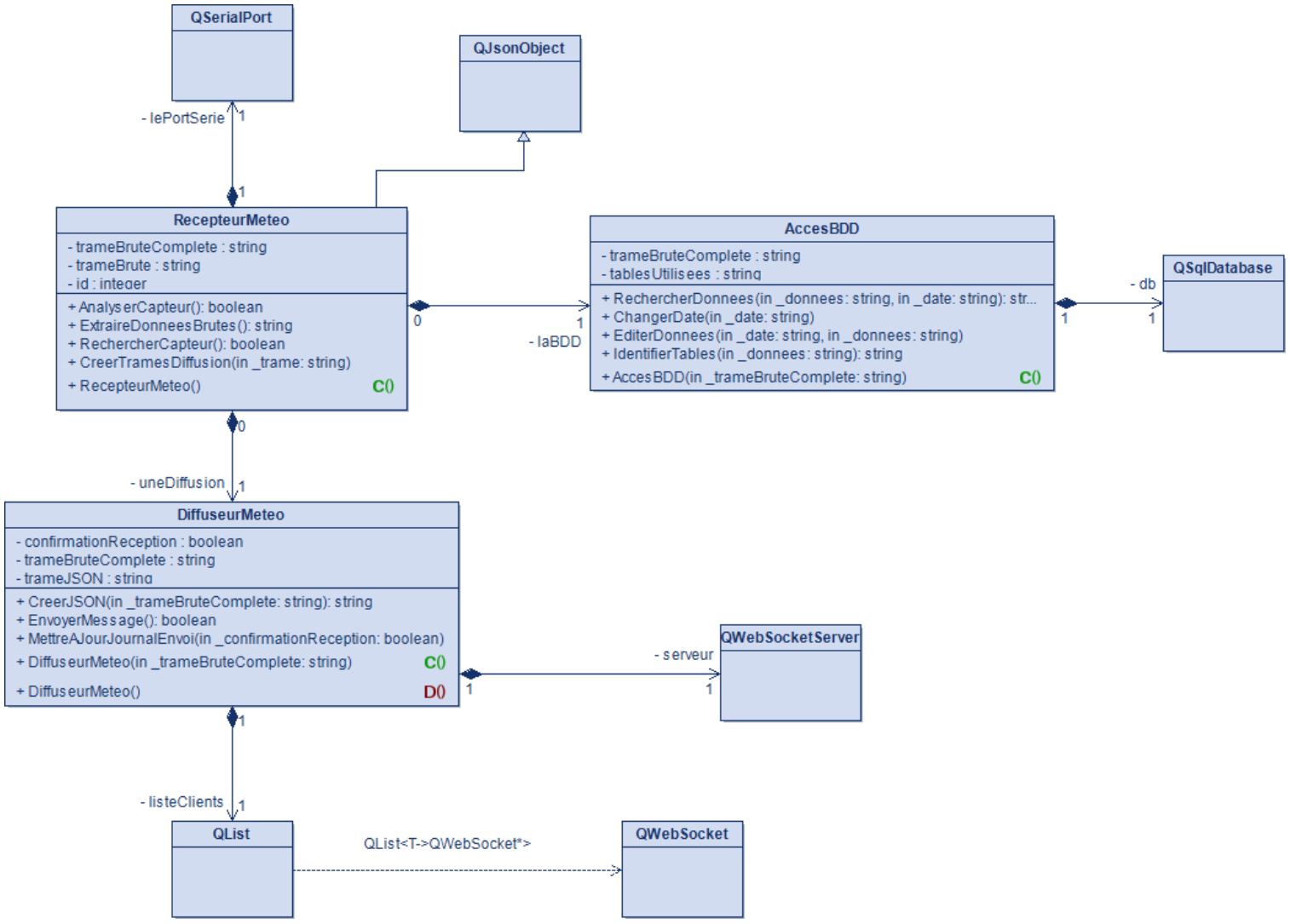


*Image X : Diagramme de classe "Récepteur Météo"*

Puisque nous avons effectué le choix des logiciels, nous pouvons désormais parfaire nos diagrammes afin de les adapter à nos outils. Dans ce diagramme de classe nous avons donc ajouté les classe fournis par le logiciel QtCreator:

- ~~QSerialPort: permet de recevoir les données en provenance du port série de l'hôte~~
- QJsonObject: permet la lecture et l'écriture de trames Json
- QSqlDatabase: permet d'accéder à une base de données et de faire des requêtes SQL

- QWebSocket / QWebSocketServer: permet d'échanger des WebSockets et de gérer un serveur. De plus, la classe QWebSocket est associée à la classe QList, ce qui nous permet de créer une liste d'utilisateurs.

# Diagramme de séquence “Stocker les données”

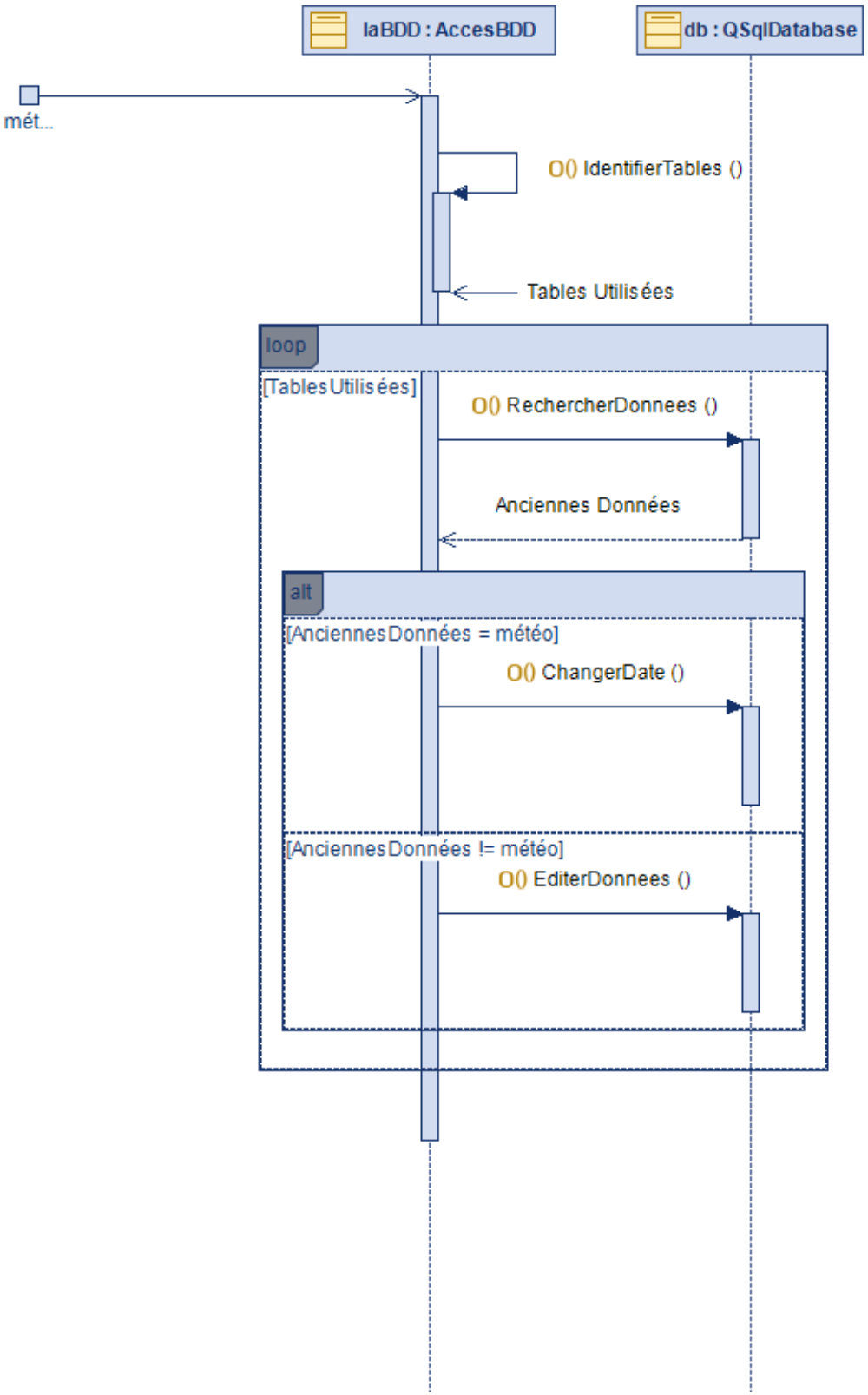

*Image X : Diagramme de séquence “Stocker les données”*

# Diagramme de déploiement UML

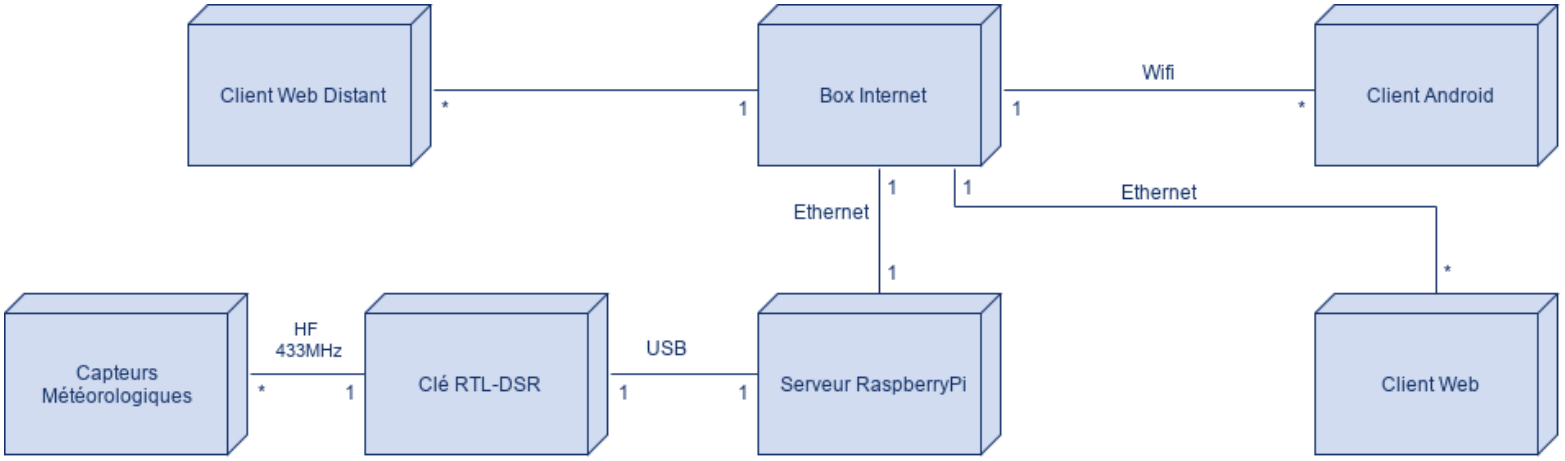


*Image 25 : "Diagramme de déploiement UML"*

Dans ce diagramme de déploiement on peut y voir les différents éléments qui vont constituer notre projet.

Afin de mieux comprendre ce diagramme voici une description des principaux acteurs:

- **Box Internet** : cette box est l'élément central de ce réseau, elle permet la communication entre le Serveur Raspberry et les éléments extérieurs au réseau. Elle est donc unique, mais plusieurs clients (Web et Android) peuvent s'y connecter.
- **Serveur Raspberry** : le Serveur Raspberry permet la connexion entre la Box Internet et les capteurs météorologiques. Il est relié à une clé RTL afin de recevoir les données des capteurs. Celui-ci est unique car il permet de centraliser les données car il a aussi le rôle de base de données.
- **Clients** : il y a différents type de clients:
  - **Clients Android**: se connecte via une application mobile et accède aux informations en direct
  - **Clients Web**:
    - **Local**: les clients locaux peuvent accéder à l'historique des données relevé par les capteurs, en accédant à la base de données.
    - **Distant**: les clients distants ont les mêmes options que les clients locaux, cependant ils doivent s'authentifier avant d'y accéder

# ***Protocoles***

Afin de faciliter la communication avec le serveur Android, la communication se fera à l'aide de WebSocket. Les messages seront écrits sous le format JSON. Lorsqu'un message est envoyé, un timer est lancé, si avant la fin de celui-ci le client android envoi un accusé de réception, rien ne se passe, cependant si il n'en envoi pas une erreur est enregistrée dans un journal.

## JSON

Comme déterminé dans le dossier d'analyse nous allons utiliser le format JSON afin de recevoir les données des capteurs et de les transmettre aux clients android. Le format des données sera sous 2 formes différentes. En ce qui concerne le type des données celui-ci sera le même que dans le dossier d'analyse

### Réception

En ce qui concerne le format de la réception, celui-ci est décidé par le logiciel RTL_433. Nous allons donc nous y adapter afin de traiter les données reçues.
Le format est donc inchangé par rapport au dossier d'analyse. Le format du JSON sera variable, car en fonction du capteur émettant le message sera différent. Seul certains paramètres seront présent sur toutes les trames, et dans l'ordre suivant:

- Date (format: AAAA-MM-JJ HH-MM-SS)
- Marque
- Modèle
- Identifiant du capteur
- Canal
- État de la batterie (batterie ok : 1; batterie faible : 0)

Le reste des données seront les informations transmises par les capteurs.
format : {<Date>, <Marque>, <Modèle>, <Identifiant>, <Canal>, <Batterie> [,<Données>, <Données>]}

### Emission

En ce qui concerne l'envoi des données nous avons choisi le type de format JSON que nous avons estimé le plus approprié. Pour faciliter la construction nous allons nous baser sur le format de la trame reçue du RTL_433. Nous allons simplement retirer les informations inutiles aux clients. Nous allons donc garder les paramètres suivant:

- Date (format: AAAA-MM-JJ HH-MM-SS)
- Modèle
- Canal

- État de la batterie (batterie ok : 1; batterie faible : 0)

Le reste des données seront comme précédemment les informations transmises par les capteurs. Si une des informations n'est pas reçue alors le message ne contiendra pas champ contenant cette données

format : {<Date>, <Modèle>, <Canal>, <Batterie> [,<Données>, <Données>]}

# WebSocket

Puisque nous cherchons à communiquer avec une application Android, à l'aide d'un serveur, et que nous cherchons à pouvoir communiquer dans les deux sens afin de tenir un journal de réception, l'utilisation de WebSocket semble être le meilleur choix.

Les WebSockets sont une technologie essentielle pour les applications Android nécessitant une communication en temps réel avec un serveur. Contrairement aux requêtes HTTP classiques, qui nécessitent une nouvelle connexion pour chaque échange de données, les WebSockets établissent une connexion unique et persistante, permettant un échange bidirectionnel fluide entre le client et le serveur.

L'un des principaux avantages des WebSockets sur Android est leur faible latence, ce qui les rend idéaux pour des applications interactives. Grâce à cette approche, une application peut recevoir instantanément des mises à jour sans avoir à interroger continuellement le serveur, ce qui réduit la consommation de données et l'utilisation des ressources.

Une fois la connexion WebSocket établie, l'application peut recevoir et envoyer des messages en temps réel, garantissant une expérience utilisateur fluide et réactive.

En permettant une communication asynchrone et continue, les WebSockets offrent aux développeurs Android une solution robuste pour améliorer l'interactivité et la réactivité de leurs applications.

Les WebSockets sont donc un outil permettant une communication en temps réel et bidirectionnel entre un serveur et une application Android. Cette connexion est continue et nous permet de communiquer facilement avec les clients. C'est donc sur ce protocole que nous allons nous reposer afin de communiquer.

# Cookies

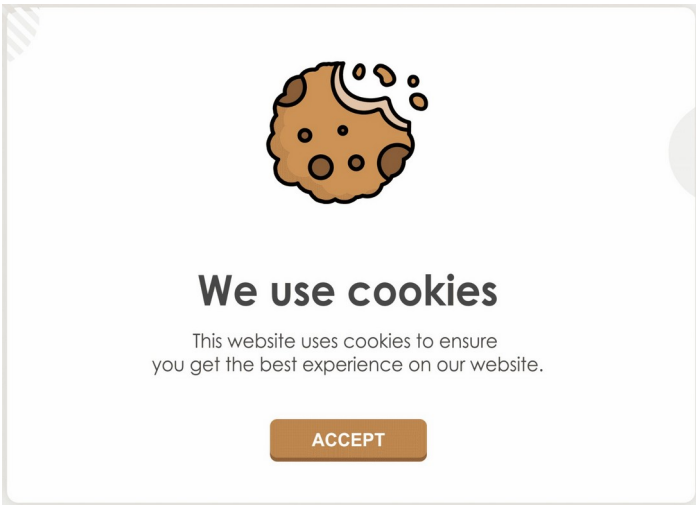


*Image 25 : "Illustration cookies"*

Les cookies sur un site météo nécessitant une connexion peuvent stocker différentes informations en fonction de leur finalité. Voici un aperçu des types de données qu'ils peuvent collecter :

## Cookies essentiels (techniques) :

Ils sont nécessaires au bon fonctionnement du site. Par exemple :

- **Identifiants de session** : Pour maintenir votre connexion active sans avoir à vous reconnecter à chaque page.
- **Préférences utilisateur** : Unités de température (°C ou °F), affichage du graphique (jours, semaines, mois).
- **Données de navigation** : Dernière page visitée.

## Cookies analytiques et statistiques :

Ces cookies collectent des données sur l'utilisation du site pour améliorer l'expérience utilisateur. Ils peuvent enregistrer :

- **Temps passé sur une page** et interactions avec les graphiques.
- **Pages visitées** et navigation entre elles

## Cookies liés aux capteurs et aux données météo

Bien que les cookies eux-mêmes ne stockent pas les données météo en direct, ils peuvent mémoriser :

- **La dernière localisation choisie** pour afficher automatiquement la météo d'une ville.
- **Filtres appliqués** (ex. type de capteur, période d'affichage).
- **Type de graphique préféré** (courbe, histogramme).

Grâce aux cookies nous pouvons avoir plus d'information sur l'utilisateur mais pour

cela il faut modifier notre diagramme MCD de sorte à avoir les informations de connexion et préférence des utilisateurs. Ci-dessous le diagramme MDC après modification

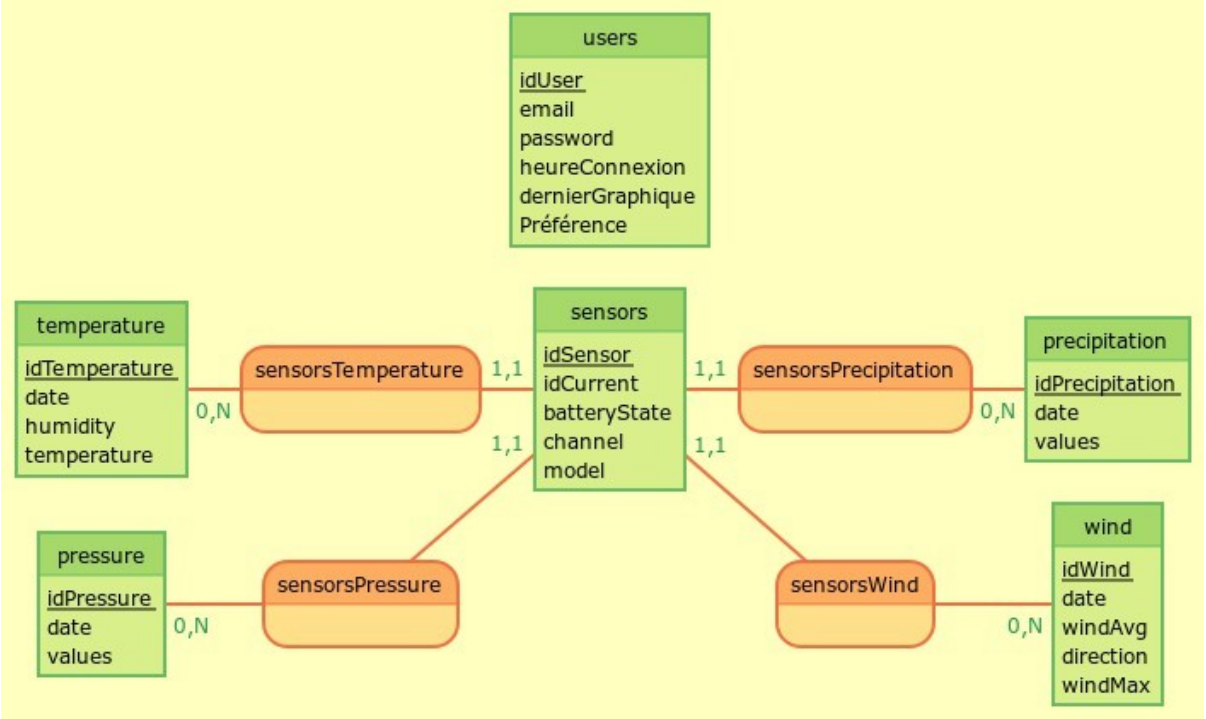


*Image 25 : “Diagramme MCD”*