# LIVRABLE PROJET IOT

BLOC 2 – Concevoir une application informatique

Mathis VALLIER

Mika THOMAS

1ère Année DI24

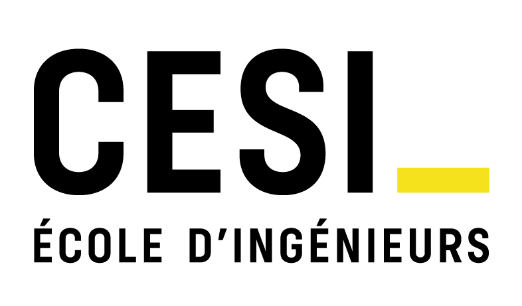


Table des matières

[LIVRABLE PROJET IOT 1](#_Toc189506069)

[1. Introduction 3](#_Toc189506070)

[1.1 – Présentation du projet 3](#_Toc189506071)

[1.2 – Objectifs du dossier 3](#_Toc189506072)

[2. Besoins et contexte 3](#_Toc189506073)

[2.1 – Problème initial (ancien modèle trop coûteux) 3](#_Toc189506074)

[2.2 – Solution proposée (nouveau modèle décentralisé) 4](#_Toc189506075)

[3. Environnement technique 5](#_Toc189506076)

[3.1 – Langages de programmation utilisés 5](#_Toc189506077)

[3.2 – Frameworks et bibliothèques 5](#_Toc189506078)

[3.3 – Outils et logiciels nécessaires 6](#_Toc189506079)

[4. Matériel utilisé 7](#_Toc189506080)

[4.1 – Liste des composants 7](#_Toc189506081)

[4.1 – Rôle de chaque composant 7](#_Toc189506082)

[5. Architecture et branchements 7](#_Toc189506083)

[5.1 – Schéma de l’architecture 7](#_Toc189506084)

[5.2 – Explication des connexions et interactions entre composants 7](#_Toc189506085)

[6. Base de données 9](#_Toc189506086)

[6.1 – Type de base de données 9](#_Toc189506087)

[6.2 – Structure et organisation des données 9](#_Toc189506088)

[7. Interface utilisateur 10](#_Toc189506089)

[7.1 – Présentation de l’IHM (interface homme-machine) 10](#_Toc189506090)

[7.2 – Fonctionnalités principales 10](#_Toc189506091)

[8. Axes d’amélioration et industrialisation 11](#_Toc189506092)

[8.1 – Limites du prototype 11](#_Toc189506093)

[8.2 – Evolutions possibles pour une version commerciale 11](#_Toc189506094)

## Introduction

### 1.1 – Présentation du projet

Ce mini-dossier technique a été réalisé dans le cadre du projet collaboratif CUBES qui vise à penser et réaliser une solution afin de collecter des données météorologiques, avec l’aide des technologies IoT (Internet of Things).

### 1.2 – Objectifs du dossier

L’objectif principal est de remplacer un ancien modèle de station météorologique basé sur une infrastructure centralisée coûteuse par une alternative plus économique et performante.

## Besoins et contexte

### 2.1 – Problème initial (ancien modèle trop coûteux)

L’entreprise pour laquelle ce projet a été réalisé vendait une station météorologique connectée reposant sur un service cloud. Cependant, plusieurs facteurs ont conduit à une baisse des ventes :

* Coûts d’hébergement élevés liés au stockage et au traitement des données sur un serveur distant.
* Manque de flexibilité du modèle centralisé, ne permettant pas aux utilisateurs de gérer leurs données localement.
* Baisse de la rentabilité de l’ancienne solution, forçant l’entreprise à repenser leur modèle vers un système plus économique.

### 2.2 – Solution proposée (nouveau modèle décentralisé)

Pour répondre aux besoins de l’entreprise sur ce projet, nous avons conçu une solution décentralisée basée sur un Raspberry Pi 4 équipé d’une distribution Debian Lite.

Architecture matérielle

L’acquisition des données se fait d’une part via un capteur utilisé pour récupérer les données concernant l’humidité, la température et la pression atmosphérique pour les afficher sur un écran LCD. En parallèle, les mêmes paramètres météorologiques sont récupérés via une API externe afin d’assurer une comparaison avec les données locales et d’évaluer la fiabilité des mesures.

Acquisition des données

Les données récupérées sont stockées dans une base de données locale, assurant ainsi un accès aux historiques sans nécessité de connexion Internet permanente.

Stockage et traitement

Un serveur web est installé sur le Raspberry Pi 4 afin d’offrir à l’utilisateur une interface web ergonomique et interactive.

Le modèle décentralisé nous offre des avantages car il est moins coûteux et dépendant des serveurs cloud et aux services d’hébergement. De plus, il est bien plus autonome car il fonctionne en local et ne nécessite pas d’un accès constant à Internet pour consulter les historiques de données.

## 3. Environnement technique

### 3.1 – Langages de programmation utilisés

Pour développer ce modèle de station météorologique, le langage de programmation Python est utilisé afin d’hydrater la base de données de manière constante et dès le démarrage de la station. Afin d’offrir à l’utilisateur une interface homme-machine ergonomique, le site hébergé en local pour consulter les données est codé avec le langage JavaScript ainsi que certains Frameworks.

### 3.2 – Frameworks et bibliothèques

Technologies utilisées pour le site (frontend React/Next.js)

|  |  |
| --- | --- |
| **Framework/Bibliothèque** | **Description** |
| Next.js | Framework React pour un rendu rapide et optimisé |
| React | Bibliothèque pour la création d’interfaces dynamiques |
| Axios | Permet d’effectuer des requêtes http vers le backend |
| dotenv | Gestion des variables d’environnement |

Technologies utilisées pour le site (backend Node.js/Express)

|  |  |
| --- | --- |
| **Framework/Bibliothèque** | **Description** |
| Express | Framework web pour créer l’API. |
| MySQL2 | Interface pour interagir avec la base de données MySQL |
| dotenv | Gestion des variables d’environnement (exemple : connexion BDD) |
| body-parser | Permet de traiter les requêtes entrantes en JSON |
| Nodemon | Redémarre automatiquement le serveur lors des modifications |

Bibliothèques Python externes

|  |  |  |
| --- | --- | --- |
| **Bibliothèques** | **Description** | **Commande d’installation** |
| board | Accès aux broches de la Raspberry Pi | pip install adafruit-blinka |
| busio | Communication I2C | (Incluse avec adafruit-blinka) |
| adafruit-bme280 | Gestion du capteur BME280 | pip install adafruit-circuitpython-bme280 |
| mysql.connector | Connexion à la base de données MySQL | pip install mysql-connector-python |
| RPLCD | Gestion de l’écran LCD via I2C | pip install RPLCD |
| requests | Requêtes http pour récupérer les prévisions météo | pip install requests |

### 3.3 – Outils et logiciels nécessaires

**VSCode pour la programmation :**

VSCode offre une interface claire et plusieurs extensions utiles. Nous avons utilisé Material Icon Theme pour une gestion des fichiers plus visuelle. Le terminal intégré nous a permis de relancer rapidement le serveur pendant le développement sans devoir quitter l’éditeur.

**L’interface phpMyAdmin pour gérer la base de données :**

phpMyAdmin nous à été suffisant pour la gestion des bases de données, il est rapide et pratique.

**Postman pour tester l’API :**

Postman nous à permis d’envoyer rapidement des requêtes GET et POST pour analyser les réponses JSON. Nous l’avons utilisé pour tester nos endpoints, vérifier les statuts de réponses et affiner la logique de notre backend.

## 4. Matériel utilisé

### 4.1 – Liste des composants

* Raspberry Pi 4
* Capteur de température, humidité et pression BME280
* Ecran LCD 16x2

### 4.1 – Rôle de chaque composant

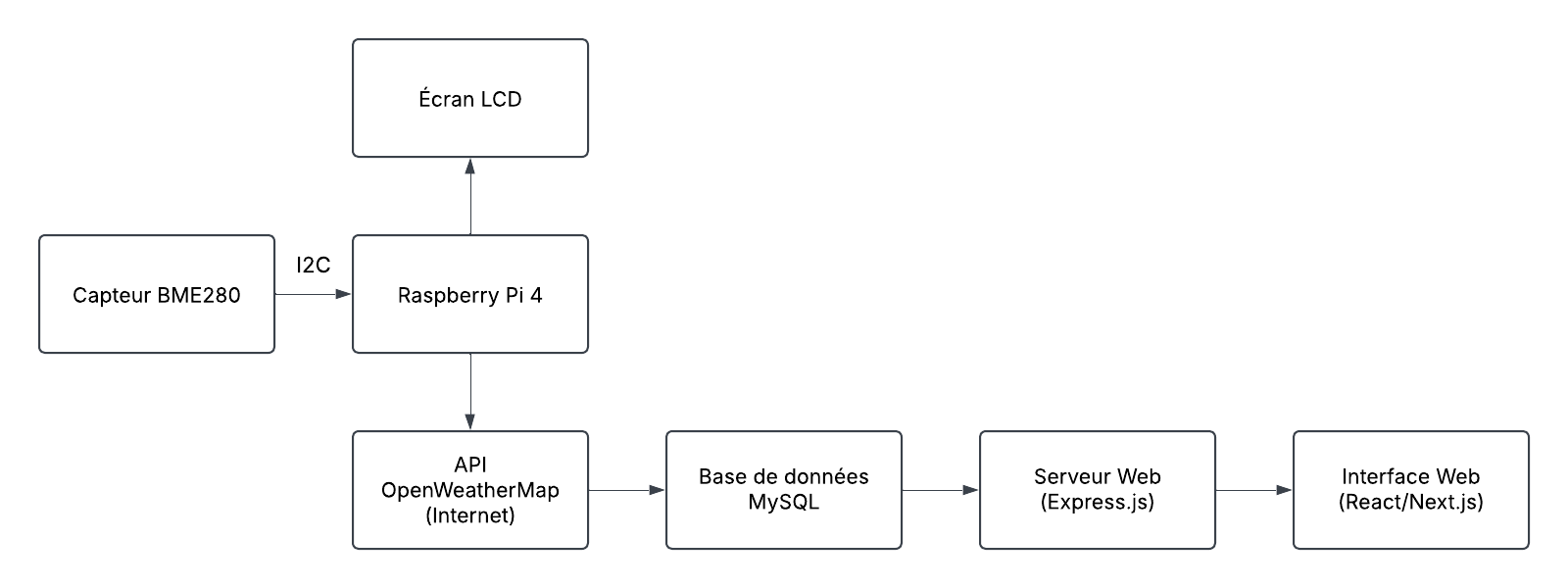
Le Raspberry Pi 4 est le cœur du système, il exécute les scripts et stocke les données.

Le BME280 capture les mesures d’humidité, de température et de pression.

L’écran LCD 16x2 affiche les informations récupérées par le capteur.

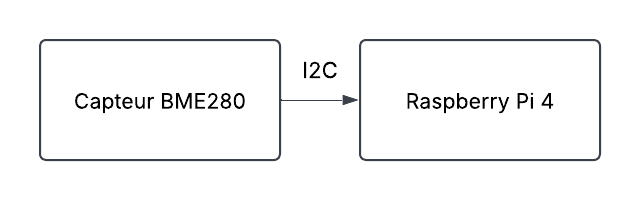
## 5. Architecture et branchements

### 5.1 – Schéma de l’architecture

Architecture du système :

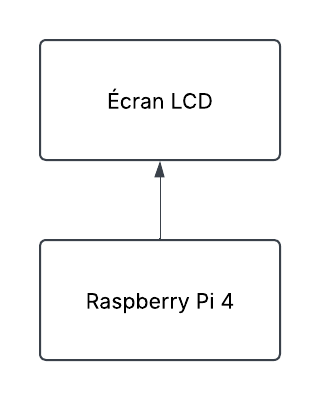
### 5.2 – Explication des connexions et interactions entre composants

Connexions matérielles



Le capteur BME280 mesure la température, l’humidité et la pression atmosphérique.

Il est connecté au Raspberry Pi 4 via un bus I2C (Inter-Integrated Circuit), un protocole de communication série utilisant deux fils pour envoyer les données (pin SDA, la donnée, et pin SCL pour la synchronisation des transmissions).



L’écran LCD affiche les valeurs des mesures et communique également via I2C, ce qui permet au Raspberry Pi de lui envoyer les informations récupérées.

Flux des données et interactions systèmes

Le script Python s’exécute en continu sur le Raspberry Pi pour récupérer et stocker les données :

* Lecture du capteur pour obtenir les mesures locales.
* Requête http vers l’API OpenWeatherMap pour récupérer les données météo externes à partir de la ville où les mesures seront prises.
* Stockage simultané des deux jeux de données dans une base MySQL locale pour garantir l’historique complet.
* Mise à jour de l’écran LCD avec les dernières valeurs locales mesurées.

C’est ensuite l’application web qui permettra une consultation des données locales et externes sous forme de tableaux et graphiques.

## 6. Base de données

### 6.1 – Type de base de données

Pour stocker et gérer les données météorologies, nous avons choisi d’utiliser MySQL, un système de gestion de base de données relationnel en raison de sa fiabilité et de sa robustesse :

* MySQL est compatible avec le backend en Node.js via le package mysql2 ainsi qu’avec Python via le module mysql.connector.
* La gestion des requêtes est efficace même avec des volumes de données importants.

### 6.2 – Structure et organisation des données

La base de données est composée d’une table « mesures » qui stocke les relevés effectués par la station.

Structure de la table mesures

|  |  |  |  |
| --- | --- | --- | --- |
| Nom du champ | Type | Contraintes | Description |
| id | INT(11) | Clé primaire, AUTO\_INCREMENT | Identifiant unique (clé primaire) |
| capteur\_temperature | DOUBLE | NOT NULL | Température mesurée par le capteur |
| capteur\_humidite | DOUBLE | NOT NULL | Humidité mesurée par le capteur |
| capteur\_pression | DOUBLE | NOT NULL | Pression mesurée par le capteur |
| api\_temperature | DOUBLE | NOT NULL | Température rapportée par l’API |
| api\_humidite | DOUBLE | NOT NULL | Humidité rapportée par l’API |
| api\_pression | DOUBLE | NOT NULL | Pression rapportée par l’API |
| horodatage | TIMESTAMP | NOT NULL, DEFAULT current\_timestamp() | Date et heure de l’enregistrement de la mesure |

## 7. Interface utilisateur

### 7.1 – Présentation de l’IHM (interface homme-machine)

[Insérer Screenshot du site une fois terminé]

### 7.2 – Fonctionnalités principales

L’application propose 5 fonctionnalisées pratiques qui assurent des résultats flexibles et précis :

1. Demande de localisation lors de la première utilisation

Lors de la première utilisation de l’application, un formulaire de saisie s’affiche pour demander à l’utilisateur de renseigner le nom de la ville où les mesures seront prises. Cette information permet aussi de modifier dynamiquement l’URL de la requête API afin d’adapter les données à la localisation saisie par l’utilisateur.

[Capture d’écran de la fonctionnalité]

1. Sélection des données par jour et heure

L’utilisateur peut choisir un jour spécifique via une interface de sélection. Ce sélecteur permet de visualiser les relevés météorologiques avec un affichage immédiat des résultats. Les utilisateurs peuvent modifier ces paramètres à tout moment en fonction de leurs besoins.

[Capture d’écran de la fonctionnalité]

1. Visualisation des données sous forme de tableaux

Les relevés météorologiques sont présentés dans des tableaux permettant de consulter facilement les mesures quotidiennes.

[Capture d’écran de la fonctionnalité]

1. Affichage graphique des données

L’utilisateur peut choisir de visualiser les données sous forme de graphiques interactifs, facilitant ainsi l’analyse de comparaison des tendances météorologiques entre les données intérieures et extérieures.

[Capture d’écran de la fonctionnalité]

1. Page d’administration (réinitialisation des données)

Une fonctionnalité réservée à l’administrateur permet de réinitialiser les données météo via une interface sécurisée. Cette option permet de supprimer toutes les données précédemment collectées.

[Capture d’écran de la fonctionnalité]

## 8. Axes d’amélioration et industrialisation

### 8.1 – Limites du prototype

### 8.2 – Evolutions possibles pour une version commerciale

Ce modèle ouvre la voie à des améliorations et des évolutions futures telle que l’utilisation de protocoles comme MQTT ou LoRaWAN pour faciliter l’interconnexion avec d’autres stations météorologiques.