# LIVRABLE PROJET ??

BLOC 2 – Concevoir une application informatique

Mathis VALLIER

Mika THOMAS

1ère Année DI24

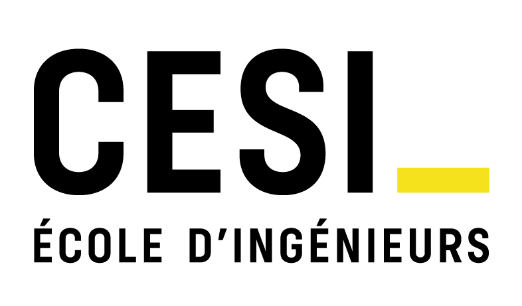


Table des matières

[LIVRABLE PROJET ?? 1](#_Toc189166012)

[1. Introduction 3](#_Toc189166013)

[1.1 – Présentation du projet 3](#_Toc189166014)

[1.2 – Objectifs du dossier 3](#_Toc189166015)

[2. Besoins et contexte 3](#_Toc189166016)

[1.1 – Problème initial (ancien modèle trop coûteux) 3](#_Toc189166017)

[1.2 – Solution proposée (nouveau modèle décentralisé) 4](#_Toc189166018)

[3. Environnement technique 4](#_Toc189166019)

[1.1 – Langages de programmation utilisés 4](#_Toc189166020)

[1.2 – Frameworks et bibliothèques 5](#_Toc189166021)

[1.3 – Outils et logiciels nécessaires 6](#_Toc189166022)

[4. Matériel utilisé 6](#_Toc189166023)

[1.1 – Liste des composants 6](#_Toc189166024)

[1.2 – Rôle de chaque composant 6](#_Toc189166025)

[5. Architecture et branchements 6](#_Toc189166026)

[1.1 – Schéma de l’architecture 6](#_Toc189166027)

[1.3 – Explication des connexions et interactions entre composants 6](#_Toc189166028)

[6. Base de données 7](#_Toc189166029)

[1.1 – Type de base de données 7](#_Toc189166030)

[1.2 – Structure et organisation des donnéesss 7](#_Toc189166031)

[7. Interface utilisateur 7](#_Toc189166032)

[1.1 – Présentation de l’IHM (interface homme-machine) 7](#_Toc189166033)

[1.2 – Fonctionnalités principales 7](#_Toc189166034)

[8. Axes d’amélioration et industrialisation 7](#_Toc189166035)

[1.1 – Limites du prototype 7](#_Toc189166036)

[1.2 – Evolutions possibles pour une version commerciale 7](#_Toc189166037)

## 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

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

L’entreprise pour laquelle ce projet à 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 utilisateur 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.

### 1.2 – 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.  
  
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.

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

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

### 1.1 – 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.

### – 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 |

### 1.3 – Outils et logiciels nécessaires

VS Code pour coder

MySQL Workbench pour gérer la base de données

Postman pour tester l’API

## 4. Matériel utilisé

### 1.1 – Liste des composants

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

### – 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

### 1.1 – Schéma de l’architecture

<https://fritzing.org/>

### – Explication des connexions et interactions entre composants

Parler du bus informatique I2c

## 6. Base de données

### 1.1 – 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.

### 1.2 – Structure et organisation des données

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

Structure de la table mesures

|  |  |  |  |
| --- | --- | --- | --- |
| Nom du champ | Type | Contraintes | Description |
| id |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## 7. Interface utilisateur

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

### 1.2 – Fonctionnalités principales

## 8. Axes d’amélioration et industrialisation

### 1.1 – Limites du prototype

### 1.2 – 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.