WAREHOUSE ENVIRONMENT MONITORING SYSTEM

PROJET

LOUIS ET ADRIEN

CONTENUE DU CAHIER DE CHARGES

I CONTEXTE ET ENJEUX

II LES TECHNOLOGIES DÉPLOYÉES ET L’ARCHITECTURE MATÉRIELLE

III FONCTIONNALITÉS

IV RÉPARTITION DES TACHES

V DICTIONNAIRE DES MOTS TECHNIQUE **(UN ASTÉRISQUE?)**

I) LE CONTEXTE DU PROJET ET SES ENJEUX.

A)LE CONTEXTE

Nous avons été contacté par l'entreprise **404 error** afin de développer et mettre en place une **surveillance fiable** et simple d'utilisation qu'ils pourront installer dans **les différents entrepôts** qu'ils possèdent.

B) OBJECTIF

Le principal objectif de ce projet est de mettre en place un système de surveillance de la **température** et de **l'humidité** dans **un entrepôt informatique**. De plus, le système doit également être utilisable dans d'autres cas d'utilisation qui pourraient être affectés par des problèmes d'humidité et de température nécessitant un suivi régulier.

C)POURQUOI UNE TELLE DEMANDE ?

Surveiller la température et l'humidité dans un centre de données ou un entrepôt informatique est **crucial** pour **garantir un fonctionnement optimal et fiable** des équipements informatiques. EN Voici quelques raisons:

premièrement:

Prévenir la surchauffe des équipements : les serveurs, les routeurs, les commutateurs et les autres équipements informatiques génèrent beaucoup de chaleur. Si la température ambiante de l'entrepôt informatique est trop élevée, cela peut entrainer une surchauffe des équipements, ce qui peut causer des dommages permanents ou une panne complète. La surveillance de la température peut aider à prévenir ces problèmes en alertant les responsables de l'entrepôt informatique lorsque la température dépasse un seuil critique.

secondement:

Éviter les variations d'humidité : des niveaux d'humidité élevés ou faibles peuvent également affecter négativement les équipements informatiques. Une humidité excessive peut causer de la condensation, ce qui peut endommager les composants électroniques sensibles. D'un autre côté, une humidité insuffisante peut provoquer des décharges électrostatiques qui endommagent également les équipements informatiques. La surveillance de l'humidité peut aider à maintenir un niveau d'humidité optimal pour les équipements informatiques.

troisièmement:

Prévenir les pertes de données : si les équipements informatiques ne fonctionnent pas correctement en raison de la surchauffe ou de l'humidité excessive, cela peut entrainer des pertes de données importantes. La surveillance de la température et de l'humidité peut aider à prévenir ces problèmes et à assurer la disponibilité des données pour les utilisateurs.

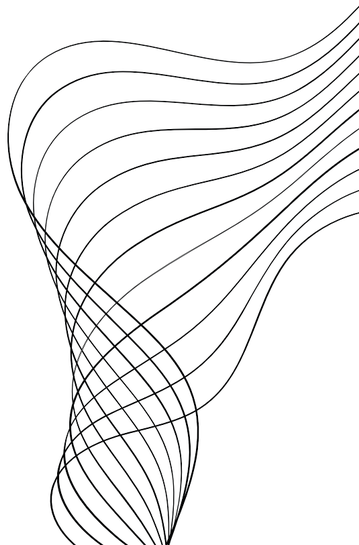
conclusion:

En somme, surveiller la température et l'humidité dans un entrepôt informatique est donc essentiel pour garantir un fonctionnement fiable et optimal des équipements informatiques, éviter les pertes de données et prolonger la durée de vie des équipements.

II) LES TECHNOLOGIES DÉPLOYÉS ET L'ARCHITECTURE MATÉRIELLE

Cette partie a pour objectif de présenter **les choix techniques** qui ont été faits pour le développement de notre solution. Nous y décrirons **les langages de programmation et les frameworks** utilisés pour concevoir **l'architecture logicielle** de notre système.

Nous expliquerons également les décisions prises concernant **l'architecture matérielle** de la solution, notamment en ce qui concerne les serveurs, les **bases de données** et les **équipements réseau** utilisés pour assurer un déploiement optimal de notre application.



A) ARCHITECTURE MATÉRIELLE

récepteur et traitement des données

GitHub page

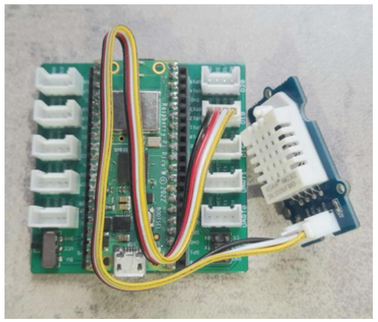
appel API

envoie API

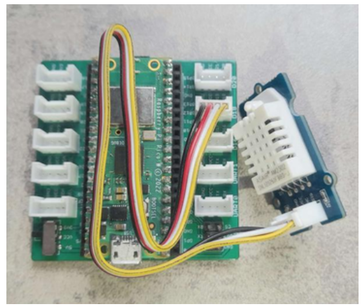
websocket* notif

{temperature:36.3,humidity:35.2,macAddress:"C3-D5-57-A6-92"}

collecteurs et transmetteurs d'informations



client (Python)



client (Python)

serveur (NODE JS)

envoie de la de la page

demande de la page

clients web (javascript natif)

B) LES LANGAGES ET TECHNOLOGIES

Node.JS:

Nous permet de faire des applications JavaScript coté serveur. nous l'utilisons pour faire le serveur BACK END (L'API REST) qui récupère les envoie du Pico et du serveur web, traite les données reçus(enregistre les données en base de données, fait des vérifications tierces, envoie des SMS et bien plus) et pour finir envoie les données nécessaire au fonctionnement correcte de l'application web

ADONIS JS* :

le choix du framework* est totalement subjectif. Il va nous faciliter le développement de L'API REST*

Python:

nous l'utilisons sur les PICOs WIFI (microcontrôleurs*) pour collecter les données du capteur (sensor pro v1.3) ainsi qu'envoyer les données au serveur (Raspberry Pi) sous format JSON. Nous utilisons python pour cette partie pour son utilisation simple et les nombreux tutoriels disponible sur internet pour coder sur un pico.

JavaScript natif :

permets de faire toute l'interaction utilisateur sur le navigateur(animations, boutons...) ainsi qu'envoyer des informations et de recevoir les informations de L'API.

HTML, CSS :

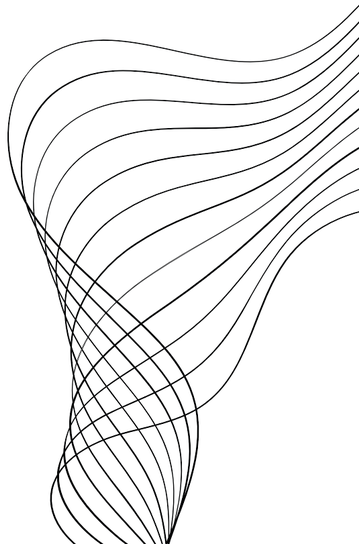
HTML langage de balisage qui va nous permettre de faire toute la partie visible de l'application web, CSS langage de stylisation qui va permettre de rendre agréable et jolie la partie visible de l'application web.

I) FONCTIONNALITÉES

A)collecteur d'informations

Nous allons mettre en place **un formulaire** afin de saisir les informations requises, ces informations seront enregistrées dans un **fichier JSON**. Tout ceci permettra d'éviter d'avoir à accéder au code lors de la configuration d'un nouveau collecteur d'informations AINSI QUE DE RENDRE SIMPLE LE TRAVAIL DU TECHNICIEN.

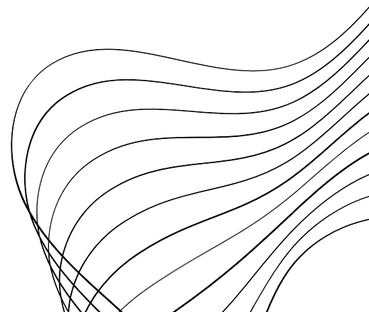
Le programme sera conçu pour que le microcontrôleur (Pico) se connecte **en wi-fi** renseigné lors de la configuration. par LA SUITE IL récupèrera **les données du capteur** branché et les envoie à l'API (Raspberry pi) sous forme de données **JSON** toutes les x secondes.



A) Site Web

Page de connexion : La page de connexion qui fera aussi office de page d'accueil permettra de se connecter au serveur via un formulaire qui requiert une IP et un port. Un message d'erreur apparaitra si l'IP n'est pas renseigné ou incorrect.
Lorsque les informations entrées sont correctes vous serait rediriger sur la prochaine page après que les informations renseignées soient enregistrés dans le localstorage* pour être réutilisé sur le reste de l'application. Vous verrez alors un loader présent le temps du chargement de celle-ci. La page de connexion fera aussi office de page pour afficher les erreurs sous forme de notifications au cas où, l'utilisateur essaie de faire une action interdite, ou que la connexion au serveur ne s'effectue dans le temps imparti.

Page d'information sur les capteurs : Cette nouvelle page nous proposera un affichage sous forme de cartes montrant les capteurs connectés au serveur avec quelques informations utiles comme la latitude et longitude du capteur (si renseigné), une information sur la connectivité du capteur, son nom et une brève description qui renseignera la date de mise en place).
Vous aurez accès aussi à une barre de recherche qui vous permettra de rechercher un capteur en fonction de son nom. Lorsque vous cliquerez sur une carte, vous serait automatiquement redirigé sur une page d'informations supplémentaires concernant le capteur sélectionné.



Page d'information supplémentaire des capteurs : elle vous affichera la température (en °C) et l'humidité (en %) renvoyés en temps réel par le capteur sélectionné au préalable. Des graphiques linéaires seront présents sur la page pour une meilleure visibilité de la température et de l'humidité sur un temps donné (24 dernières heures) . Vous aurez aussi accès à un bouton qui vous emmènera sur une page permettant d'ajouter un nouveau capteur à l'application ainsi qu'un second bouton permettant de suivre la météo du jour afin de pouvoir prédire des variations de température et d’humidité.

Formulaire d'ajout de capteur : L'utilisateur devra renseigner le nom du capteur, sa description ainsi que la latitude et la longitude (facultatif) de plus vous devrez indiquer un seuil à ne pas dépasser pour l'humidité et la température(si ce seuil viens à être dépassé vous serez contacté par SMS . Un message d'erreur apparaîtra si des renseignements son manquants ou incorrecte. Une fois enregistré vous verrez une notification en haut de l’écran qui vous indique que tout c'est bien passé

Informations sur la météo du jour : Cette page vous permettra le suivi de la météo de la semaine (sur quatre jours), permettant ainsi la prévention. vous aurez affiché, en grand la température et une image représentante de la météo du jour, de plus vous aurez aussi la météo sur 3 jours qui sera situé juste en dessous.

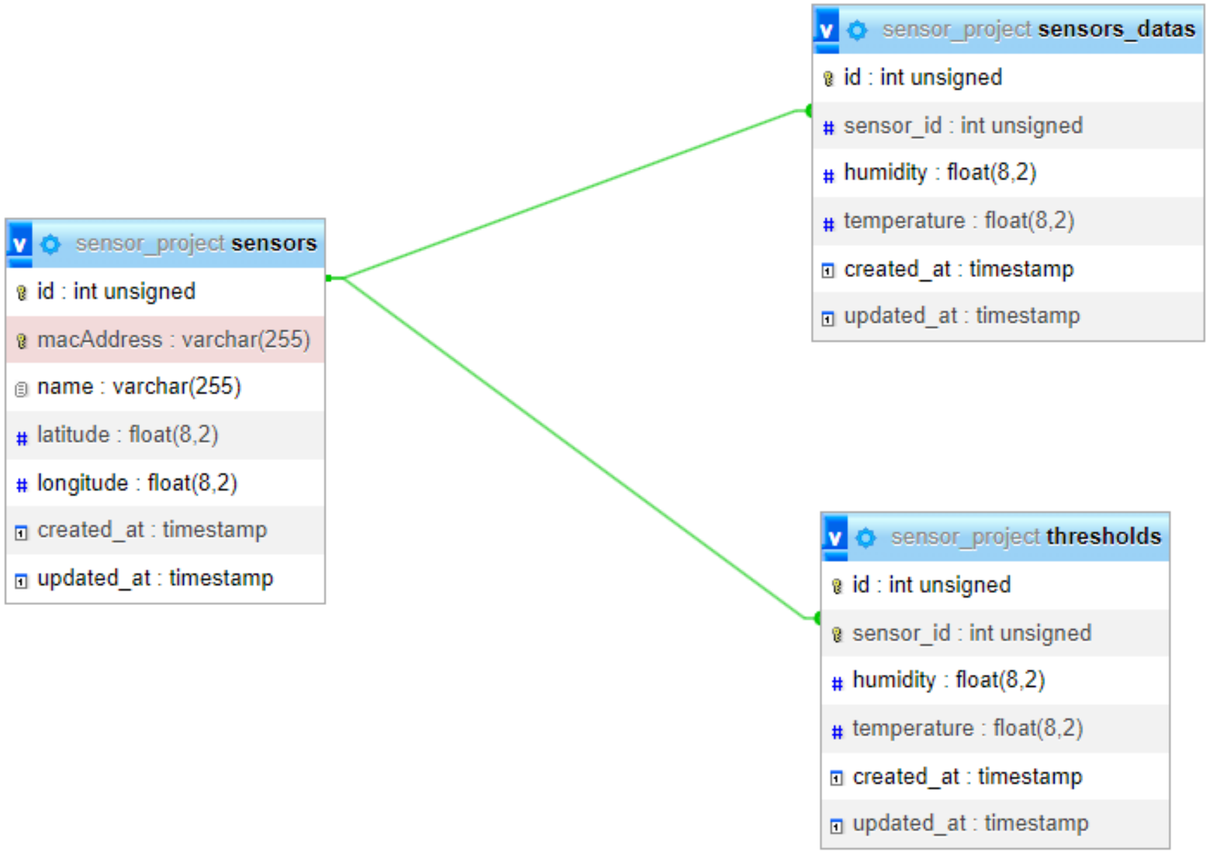
page de modification de seuil : Sur cette page vous disposerez d'un formulaire et deux entrées, le seuil d'humidité et de température du capteur sélectionné. Ces deux entrées seront auto-complétés grace à une requête à l'api. vous pourrez alors y modifier l'information que vous souhaitez. Une fois la modification apportée vous aurez une notification en haut de la page qui vous indique que la base de données a bien pris en compte les changements effectués.

B)récepteur et traitement des données

Dans cette partie nous allons vous présenter toutes les routes http* que nous souhaitons mettre en place pour notre api en expliquant leur but et ce qu'elles permettrons d'accomplir. Mais avant ca regardez les tables que nous avons décidé de creer dans la base de données MYSQL*:

SQL SENSOR_PROJECT TABLES:

on utilise l'adresse mac pour identifier chaque capteurs



sensors routes :

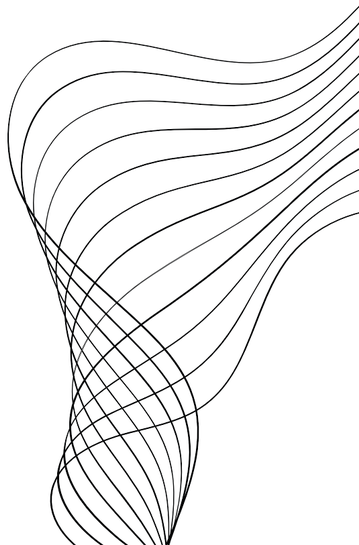
-post: cette route va permettre d'enregistrer un capteur dans la base de données avec la vérification ci dessous des données en entrée. Un fois le capteur enregistré la route renvoie l'identifiant de celui-ci

```
public schema = schema.create({
  macAddress : schema.string({},[
    rules.unique({ table: 'sensors', column: 'macAddress',}),
  ]),
  name: schema.string(),
  latitude: schema.number.optional(),
  longitude: schema.number.optional(),

})
```

-Get: Cette route renverra les capteur en fonction de la page avec le paramètre **page=** avec une limite de 10 et possibilité de faire une recherche par nom avec le paramètre **q=**

-get:id: route renverra les informations d'un capteur en fonction de son identifiant



threshold routes:

-post: cette route enregistrera un seuil dans la base de données en verifiant les données en entrées comme ci dessous. Une fois le seuil enregistré l'identifiant de celui-ci est renvoyé

```
public schema = schema.create({
    humidity: schema.number(),
    temperature: schema.number(),
    sensorId: schema.number([
      rules.unique({ table: 'thresholds', column: 'sensor_id',}),
      rules.exists({ table: 'sensors', column: 'id' })
    ]),
})
```

-patch:id : cette route permettra de mettre à jour un seuil en fonction de son identifiant, les données attendus sont representé si dessous:

```
public schema = schema.create({
  temperature : schema.number.optional(),
  humidity : schema.number.optional(),
})
```

-get:sensor_id : cette route permettra de récupérer le seuil d'un capteur en fonction de l'identifiant de celui-ci

SENSORS_DATAS ROUTES:

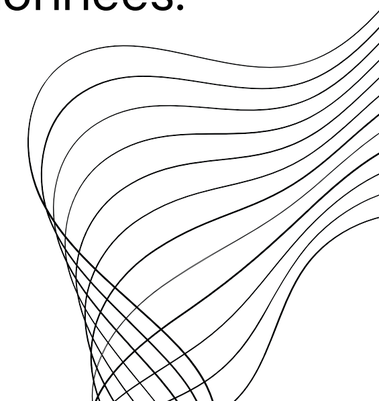
-post: cette route va permettre de créer un sensor_data, elle envoie ensuite la donnée au client websocket (au navigateur). et pour finir on vérifie que le seuil du capteur n'a pas était dépassé, si oui on envoie un sms au client. Une fois toute les actions effectuées la route renverra l'identifiant de l'enregistrement.

```
public schema = schema.create({
  macAddress: schema.string(),
  temperature: schema.number(),
  humidity: schema.number()

})
```

-get:sensor_id: on récupère les données du capteur renseigné avec sensor_id. les données sont récupérées la moyenne sur les 24 dernières heures.

-get:sensor_id: cette route permet de récupérer la dernière données enregistré par le capteur renseigné par le paramètre sensor_id qui est l'identifiant du capteur dont on souhaite récupérer les données.



IV) RÉPARTITION DES TACHES

Adrien: a essayé 10 minutes mais n'a finalement rien fait

Louis: administration de la base de données MYSQL et de toute l'application back end (api) . Développement de toute la partie front end. développement du code des collecteur et transmetteur D'informations. mise en place du cahier des charges.

V) DICTIONNAIRE DES MOTS TECHNIQUES

API RESTFULL:
Une API REST (également appelée API RESTful) est une interface de programmation d'application (API ou API web) qui respecte les contraintes du style d'architecture REST et permet d'interagir avec les services web RESTful.

websocket:
Une L'API WebSocket est une technologie évoluée qui permet d'ouvrir un canal de communication bidirectionnelle entre un navigateur (côté client) et un serveur.

ADONIS JS:
AdonisJS est un framework Node.js de type MVC (Modèle-Vue-Contrôleur) qui permet de développer des applications web en utilisant des concepts modernes tels que les Middlewares, l'injection de dépendances, la gestion des sessions et la création d'API REST. AdonisJS est inspiré de Ruby on Rails et Laravel, et propose un ensemble d'outils et de fonctionnalités pour faciliter le développement d'applications web robustes et évolutives. Le framework est basé sur une architecture modulaire qui permet aux développeurs de choisir les composants dont ils ont besoin pour leur projet.

FRAMEWORK:
Un framework est un ensemble d'outils, de bibliothèques et de conventions de développement qui permettent de construire plus rapidement des applications.

microcontrôleur :
Un microcontrôleur est un composant électronique intégré qui combine sur une seule puce un processeur, de la mémoire, des entrées/sorties (E/S) et d'autres périphériques utiles pour les applications électroniques.

localstorage :
Le localStorage est une fonctionnalité offerte par les navigateurs web modernes, qui permet aux sites web de stocker des données localement sur l'ordinateur de l'utilisateur. Les données sont stockées sous forme de paires clé-valeur, où la clé est une chaîne de caractères et la valeur peut être de différents types de données, tels que des chaînes de caractères, des nombres, des tableaux ou des objets JavaScript.

routes http :
les routes HTTP de l'API sont des URLs qui permettent d'accéder aux différentes ressources de l'API en utilisant les méthodes HTTP appropriées pour interagir avec ces ressources.

mysql :
MySQL est un système de gestion de base de données relationnelles open-source développé par Oracle Corporation. Il est l'un des systèmes de gestion de bases de données les plus utilisés dans le monde, connu pour sa rapidité, sa fiabilité et sa facilité d'utilisation. MySQL est utilisé pour stocker et gérer de grandes quantités de données, et est couramment utilisé dans les applications web