

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITÉ DE SOUSSE

INSTITUT SUPÉRIEUR D'INFORMATIQUE
ET DES TECHNOLOGIES DE COMMUNICATION

المعهد العالي للإعلامية وتكنولوجيا الاتصالات



Rapport de stage de fin d'études

Présenté en vue de l'obtention du diplôme de Licence en Ingénierie des
Systèmes Informatiques (Computer Engineering)

Spécialité : Systèmes Embarqués et Internet des Objets

Système de Surveillance Thermique Intelligent

Réalisé par :

Mouhamed Aziz Derouiche

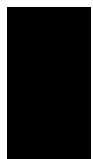
Encadrant académique : Mme. Rahma Kalboussi

Encadrant professionnel : M. Chaker Laaribi

Société d'accueil

XALBOX

Année universitaire : 2023/2024



DEDICATION

Je dédie le fruit des efforts investis dans ce travail à tous ceux qui me sont chères :

À Mon cher père Tarek

Aucune instance ne peut exposer mon profond amour, ma sincère gratitude et ma profonde gratitude pendant les sacrifices que tu as consentis. Que ce travail soit le témoignage de ma reconnaissance et de mon infinie gratitude. Merci pendant tout ce que tu as fait pendant moi et que Dieu te protège et te procure santé et te préserve longue vie.

À Ma chère mère Jamila

Tu m'as donné la vie, la tendresse et le courage de réussir. Tout ce que je peux vous donner ne peut exprimer mon amour éternel et mon respect pour les sacrifices que vous avez faits et l'amour que vous m'avez toujours entouré. Je ferai toujours de mon mieux pour rester votre fierté et ne jamais vous décevoir.

À Mon frère Adem

Toujours présent pour moi, tu es mon ami sincère et mon confident dévoué, prêt à m'aider et à me soutenir tout au long de mon parcours.

À Mes chers Amis, Mes proches et tous ceux qui m'ont aidé à atteindre cette étape

Merci infiniment pour votre soutien moral pendant la préparation de mon projet de fin d'études et pour tous les moments inoubliables que nous avons partagés. Je vous promets de continuer à donner le meilleur de moi-même, à me distinguer et à briller. Mouhamed Aziz restera toujours la personne que vous connaissez.



REMERCIEMENT

Je tiens tout d'abord à remercier ALLAH, le tout puissant et miséricordieux, qui m'a donné La force et la patience d'accomplir ce Modeste travail.

Je tiens à remercier sincèrement mon encadrante **Mme Rahma Kalboussi** , pour ses précieux conseils, ses recommandations judicieuses, et la confiance qu'elle m'a toujours accordée. Son suivi attentif, son aide précieuse, et sa disponibilité ont été d'une grande aide pour moi.

Je tiens à remercier chaleureusement **Mr Chaker Laaribi**, mon encadrant professionnel chez XALBOX pour son accueil et le temps qu'il m'a consacré.

Malgré ses engagements personnels et professionnels, il a toujours été disponible pour m'aider et m'encadrer à tout moment. Ses conseils et son soutien précieux ont été inestimables tout au long de ce stage.

Je souhaite ensuite adresser nos remerciements au corps professoral et administratif de L'institut Supérieure d'Informatique et des Techniques de Communication de Hammam Sousse, pour la qualité de l'enseignement offert et le soutien.

Je souhaite également exprimer mes vifs remerciements aux membres du jury pour l'intérêt qu'ils ont porté à mon projet en acceptant d'examiner mon travail et de l'enrichir par leurs propositions.

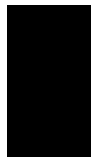


TABLE DES MATIÈRES

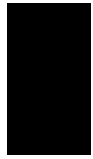
DEDICATION	ii
REMERCIEMENT	iii
LISTE DES FIGURES	ix
LISTE DES TABLEAUX	ix
GLOSSAIRE	x
INTRODUCTION GÉNÉRALE	1
1 ÉTUDE PRÉALABLE	3
1.1 Introduction	4
1.2 Présentation de l'organisme d'accueil "XALBOX"	4
1.3 Présentation du projet	5
1.3.1 Problématique	5
1.3.2 Objectifs du projet	5
1.4 Étude et critique de L'existant	6
1.4.1 Étude de L'existant	6
1.4.1.1 Méthode de gestion des températures	6
1.4.1.2 Les systèmes intelligents de gestion des températures existants	7
1.4.2 Critique de l'existant	10
1.5 Solution proposée	10
1.6 Conclusion	11
2 ÉTAT DE L'ART	12
2.1 Introduction	13
2.2 Environnement de travail	13
2.2.1 Environnement matériel	13
2.2.2 Environnement logiciel	16

2.2.3	Outils de la rédaction du rapport	18
2.2.4	Outils de Conception	19
2.2.5	Outils de développement	20
2.3	Conclusion	22
3	ÉTUDE CONCEPTUELLE	23
3.1	Introduction	24
3.2	Analyse des besoins	24
3.2.1	Besoins fonctionnels	24
3.2.2	Besoins non fonctionnels	25
3.2.3	Besoins matériels	25
3.3	Architecture générale de la solution proposée	26
3.4	Conception	27
3.4.1	Vue statique	28
3.4.1.1	Diagrammes de cas d'utilisation l'interaction entre les utilisateurs et le systeme	28
3.4.1.2	Diagrammes des classes	31
3.4.2	Vue dynamique	32
3.4.2.1	Diagrammes des séquences	32
3.5	Conclusion	35
4	RÉALISATION ET ÉVALUATION	36
4.1	Introduction	37
4.2	Tests et résultats	37
4.2.1	Étapes de configuration	37
4.2.2	Outils de test	46
4.2.3	Scénarios et résultats	47
4.3	Conclusion	51
5	CONCLUSION ET PERSPECTIVES	52
	NETOGRAPHIE	52

LISTE DES FIGURES

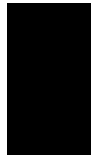
1.1	Logo de la société Xalbox	4
1.2	Site officiel de plateforme WEATHERHUB OBSERVER	8
1.3	Site officiel de Solution Kooklin	9
2.1	Carte ESP32 devkit V1	14
2.2	Capteur de température étanche DS18B20	15
2.3	Câblage des composants du système.	16
2.4	Logo Visual Studio 2022	17
2.5	Logo Arduino IDE	17
2.6	Logo Fritzting	17
2.7	Logo GitHub	18
2.8	Logo stack overflow	18
2.9	Logo de LaTeX	19
2.10	Logo Draw.io	19
2.11	Logo StartUML	20
2.12	Logo Pocketbase	20
2.13	Logo de postman	21
2.14	Logo Xamarin	21
2.15	Logo Csharp	22
2.16	Logo protocole HTTP	22
3.1	Architecture de notre système	26
3.2	Diagramme de cas d'utilisation Global	28
3.3	Diagramme de cas d'utilisation gérer utilisateur	29
3.4	diagramme de classes	31
3.5	Diagramme de séquence « Authentification »	33
3.6	Diagramme de séquence « notification »	34
3.7	Diagramme de séquence « envoyer message »	35
4.1	Téléchargement Pocketbase	38

4.2	Lancement de serveur local pocketbase utilisant cmd	38
4.3	Page d'authentification Pocketbase	39
4.4	Creation collection Pocketbase"	40
4.5	lancement du serveur PocketBase en local	41
4.6	Gérer les paquets NuGet.	41
4.7	Téléchargement le package System.Net.HTTP	42
4.8	Exemple d'utilisation PocketBase avec Csharp	43
4.9	Code arduino	44
4.10	Arduino IDE File→Préférences	45
4.11	Remplissage du champ URL de gestionnaire de cartes supplémentaires.	45
4.12	choisi de carte esp32 devkit V1	46
4.13	Interface d'accueil	48
4.14	Interface de la page d'accueil pour l'administrateur et pour l'utilisateur	49
4.15	Interface notification	50
4.16	interface interface de messagerie	51



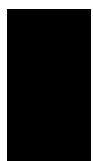
LISTE DES TABLEAUX

1.1	<i>Etude comparative des solutions existantes.</i>	10
2.1	Environnement matériel de travail.	13
3.1	<i>Tableau des Acteurs</i>	27
3.2	Description textuelle du diagramme de cas d'utilisation de l'administrateur	30
4.1	: Les caractéristiques de nos smartphones.	47



LISTE DES ABBRÉVIATIONS

- **API** : Application Programming Interface
- **HTTP** : Hypertext Transfer Protocol
- **IOT** : Internet of Things
- **IDE** : Integrated Development Environment
- **JSON** : JavaScript Object Notation
- **SCAPCB** : Société des Conserves Alimentaires des Producteurs du Cap-Bon
- **SQL** : Structured Query Language
- **UML** : Unified Modeling Language



INTRODUCTION GÉNÉRALE

La surveillance de la température est actuellement l'un des problèmes les plus critiques dans de nombreux secteurs d'activité en plein essor, comme l'usine de conserves alimentaires de tomate fraîches en Tunisie "SCAPCB" [Net 2].

Actuellement, la détection et le contrôle de la température de refroidissement des boîtes de tomates sont entravés par des problèmes de procédure, des contraintes financières, un manque de main-d'œuvre qualifiée et une négligence des responsabilités. Ces défis nous ont motivés à envisager une solution intelligente et abordable basée sur l'Internet des objets (IoT) pour améliorer la surveillance de la température de réfrigération des boîtes de tomates.

Notre projet de fin d'études vise à concevoir une solution intelligente de surveillance de la température de refroidissement des boîtes de tomates, optimisée pour accroître la productivité et réduire les coûts. L'idée principale est d'installer un capteurs IoT capable de communiquer en temps réel les niveaux de température et leurs variations et les représenter sur une application mobile. En nous concentrant sur le contrôle et la surveillance de la température de réfrigération des boîtes des tomates, nous cherchons à garantir la qualité et la sécurité des produits exportés dans le marché tunisien et le marché européen, ce qui est crucial pour la réputation de l'entreprise. Notre solution propose ainsi une approche innovante pour répondre aux exigences strictes de contrôle de la qualité et de la température dans le secteur des conserves alimentaires.

Notre rapport se subdivisera en quatre chapitres, introduction générale et une conclusion générale :

— Le premier chapitre Le premier chapitre présente le contexte général du projet, introduit l'organisme d'accueil. Aussi, l'étude préalable composée de la problématique, l'étude de l'existant et la critique de l'existant. En terminant par la présentation de notre solution proposée.

– Le deuxième chapitre , nous décrirons les technologies, ainsi que les outils utilisés pendant la réalisation du projet, l’environnement logiciel et matériel du système.

– Le troisième chapitre Dédié à la saisie des exigences fonctionnelles et non fonctionnelles de notre système. Nous aborderons l’étape de conception ou nous présenterons Les différents diagrammes d’UML, Ainsi que les architectures de système.

– Le dernier chapitre, nous represontons les interfaces fini de l’application mobile et l’architecture matérielle développées.

ÉTUDE PRÉALABLE

Sommaire

1.1	Introduction	4
1.2	Présentation de l'organisme d'accueil "XALBOX"	4
1.3	Présentation du projet	5
1.3.1	Problématique	5
1.3.2	Objectifs du projet	5
1.4	Étude et critique de L'existant	6
1.4.1	Étude de L'existant	6
1.4.2	Critique de l'existant	10
1.5	Solution proposée	10
1.6	Conclusion	11

1.1 Introduction

Dans ce premier chapitre, nous présentons le projet dans sa globalité. Cette partie est divisée en trois sections distinctes. Pour commencer, nous mettons en avant l'entreprise qui nous a accueillis, à savoir "Xalbox". Ensuite, nous explorons le contexte du projet en identifiant les défis, en examinant les solutions existantes et en proposant une approche innovante. Enfin, nous décrivons la méthodologie de travail que nous avons adoptée pour mener à bien ce projet.

1.2 Présentation de l'organisme d'accueil "XALBOX"

Notre projet a été proposé lors d'un stage de fin d'études afin d'obtenir un diplôme en licence en ingénierie des Systèmes informatiques spécialité : système embarqué et Internet des objets à l'Institut Supérieur d'Informatique et des Technologies de Communication de Hammam Sousse. Ce stage a été effectué au sein de la boîte de développement XALBOX [Net 1].

XALBOX est une nouvelle société qui fournira des services d'ingénierie de haute qualité à ses clients. XALBOX fournit une large gamme de solutions modernes dans les domaines de la Business Intelligence, du Big Data, de la recherche d'entreprise et des logiciels de gestion d'entreprise. Elle a été fondée en 2017 à Bardo, Tunis. Son adresse e-mail est « contact@xalbox.com ».

la Figure 1.1 représente son Logo.



FIGURE 1.1 – Logo de la société Xalbox

1.3 Présentation du projet

1.3.1 Problématique

L'augmentation de la température dans une usine de fabrication de conserve de tomates en boîte peut causer plusieurs problèmes. Tout d'abord, cela peut altérer la qualité des produits finis, affectant leur goût, leur texture et leur couleur. Ensuite, cela peut accroître le risque de contamination des aliments par des bactéries et des moisissures, compromettant ainsi la sécurité alimentaire. Il est nécessaire de passer de l'approche traditionnelle de faire chaque 30 min un contrôle sur niveau de température, qui se concentre sur les municipalités et leur utilisation inefficace de l'énergie, à l'utilisation efficace des technologies de traitement, de contrôle basé sur l'IOT. Ainsi, la problématique principale de ce projet est comment exploiter l'IOT pour assurer une gestion des température efficaces, moderne et à faible coût.

1.3.2 Objectifs du projet

Dans notre projet, l'objectif principal est l'implémentation d'une solution informatique embarquée pour faciliter le travail de l'équipe de la direction de maintenance et laboratoire et les contrôleurs afin de gagner du temps et d'avoir une traçabilité afin d'avoir une vision globale sur l'état de température pour conserve tomate. L'idée générale consiste à proposer :

- Mettre en place un dispositif intelligent de surveillance environnemental pour obtenir des informations pertinentes sur les températures de refroidissement des boîtes de tomates.
- Développer des algorithmes d'analyse des données pour interpréter les informations recueillies par les capteurs et détecter les variations anormales de température.
- Mettre en place un système de surveillance entre les dispositifs IoT et une application mobile, permettant la transmission le nouveau de température et la réception de notifications ont cas dépasse la température leur valeur anormale pour la referoidissement de boîte de tomate en temps réel par toutes les utilisateurs .
- Concevoir une interface utilisateur conviviale dans l'application mobile, offrant aux utilisateurs

la possibilité de surveiller les conditions de température, de recevoir des alertes en cas d'urgence et de communiquer entre eux de manière efficace.

– Assurer la fiabilité, en mettant en œuvre un système de mesures de protection des données et en minimisant les risques de défaillance technique.

1.4 Étude et critique de L'existant

L'une des étapes les plus importantes est l'analyse de l'existant pour la mise en œuvre d'un projet. Il est indispensable d'analyser et de critiquer l'existant avant d'entreprendre la spécification de besoins et la conception de notre solution.

1.4.1 Étude de L'existant

Dans cette section, nous avons étudié l'existant de l'organisme d'accueil ainsi que les solutions existantes dans le marché et qui peuvent être utilisées par le service propre.

1.4.1.1 Méthode de gestion des températures

La collecte des informations sur la température de refroidissement, la concentration de produit tomate fraîche est effectuée manuellement toutes les demi-heures par l'équipe du laboratoire. Cependant, divers imprévus tels que les pannes de machines ou les coupures d'électricité peuvent perturber ce processus. De plus, le contrôle manuel tout au long des 8 heures de travail entraîne des dépenses inutiles. Avec un seul opérateur, il est difficile de surveiller simultanément la température de refroidissement et le poids des produits, prolongeant ainsi le temps nécessaire pour effectuer les vérifications. Parfois, les opérations des contrôles sont réduites pour faire une autre tâche plus importante que le contrôle de température de refroidissement de produit final le tomate conserve, ce qui peut compromettre la qualité des produits finis et entraîner des pertes financières supplémentaires si certains produits doivent être rejetés.

1.4.1.2 Les systèmes intelligents de gestion des températures existants

Le marché offre un grand choix de solutions de gestion des températures. Nous ne pouvons pas tous les énumérer, mais nous donnons quelques exemples.

a) Système intelligent de surveillance de la température TFA :

Le projet vise à mettre en place Le système automatique et économique surveille les températures de -30°C à +60°C, idéal pour les pharmacies, cabinets médicaux, dentaires et maisons de retraite. Utilisez la plateforme WEATHERHUB OBSERVER pour une surveillance et une documentation professionnelles. Le kit de démarrage comprend un capteur de température pour l'intérieur ou les réfrigérateurs, avec la possibilité d'ajouter d'autres capteurs.

Fonctionnement du système TFA :

– Ce système comprend un émetteur de température professionnel pour mesurer la température ambiante dans les pièces intérieures ou les réfrigérateurs et un module de passerelle. L'émetteur de température envoie les valeurs mesurées directement à un serveur via le module de passerelle et l'Internet elle sont stockées dans un serveur cloud. Toutes les données y sont disponibles pendant au moins 90 jours. Grâce à la plateforme WEATHERHUB OBSERVER . La surveillance peut être automatisée et il existe de nombreuses options d'alerte. La plateforme est gratuite et ne nécessite aucun frais d'utilisation ou d'abonnement supplémentaire [Net 3]. la Figure 1.2 représente le Site officiel de plateforme WEATHERHUB OBSERVER .



FIGURE 1.2 – Site officiel de plateforme WEATHERHUB OBSERVER

b) Système de contrôle et surveillez de température frigos de Kooklin :

KookAlert propose une solution innovante de surveillance automatisée de la température pour les équipements frigorifiques. En utilisant des capteurs de température sans fil, le système enregistre automatiquement les relevés de température à des intervalles réguliers. En cas de variation anormale de température, les utilisateurs sont immédiatement alertés via une application mobile, leur permettant ainsi de prendre des mesures correctives rapidement..

• Comment fonctionne le système Kooklin :

- La mise en place des capteurs mesurent la température toutes les 10 minutes, sans que vous ayez à faire quoi que ce soit.
- Si la température devient trop élevée ou trop basse, vous recevez immédiatement une alerte sur votre téléphone.
- Vous pouvez vérifier les relevés de température à tout moment depuis votre téléphone.
- Lors d’inspections ou d’audits, vous pouvez facilement accéder aux données pour montrer que vous respectez les normes de sécurité alimentaire [Net 4].la Figure 1.3 représente le Site officiel de Solution Kooklin.



FIGURE 1.3 – Site officiel de Solution Kooklin

c) Surveillance de la température IoT dans l'industrie de l'énergie DUSUN :

Le système proposé sera capable de surveiller le processus de collecte des données, analysent les données, contrôle automatisé de la température en temps réel à l'aide d'appareils, de capteurs et de protocoles de communication en réseau, ce qui permet d'obtenir des données plus précises et fiables pour une prise de décision cruciale [Net 5].

•Fonctionnement du système de surveillance de température IoT dans l'industrie de l'énergie :

Dans un système de surveillance de la température IoT, ces appareils collectent des données à partir de capteurs de température, analysent les données dans le cloud et envoient des commandes aux actionneurs pour ajuster les systèmes de chauffage et de refroidissement selon les besoins. La passerelle IoT est le hub central pour toutes les collectes et analyses de données. La vanne thermostatique, le thermostat et le contrôleur de la chaudière fonctionnent ensemble pour maintenir la température souhaitée dans le bâtiment ou le système. Le système peut également être configuré pour envoyer des alertes et des notifications au personnel de maintenance si des défauts du système ou des températures dépassent certains seuils.

1.4.2 Critique de l'existant

Le Tableau 1.1 présente les avantages et les inconvénients des solutions précédentes. Ensuite, nous implémentons ensuite notre solution en utilisant les points forts de chaque projet, en évitant au maximum les points faibles.

TABLE 1.1 – Etude comparative des solutions existantes.

	Moyen de connexion	Matériel	Avantages	Inconvénients
XALBOX	–	–	–	- Plusieurs donnée pour le traité. -Nécessité d'une connexion Internet à l'emplacement des capteurs.
TFA	- Wifi - 3G/4G	-Émetteur de température :CHF41.00. -Module de passerelle : CHF 84.00. -Batterie.	-Alerte personnalisable lors de recevoir des notification. -Les données sont stockées sur le serveur pendant au moins 90 jours.	- Les émetteurs de température fonctionnent sur batterie. - Nécessité une connexion internet stable.
DUSUN	- Wifi - 3G/4G	-Thermostat. -Batterie longue duré.	- Meilleure satisfaction client et une rentabilité accrue. - l'efficacité de sécurité.	- Maintenance et de la gestion de ces capteurs. - maintenance et de la gestion de ces capteurs. -Grande quantité de données.
Kooklin	- réseau sans fils	-Capteurs de pression. -Capteurs de température. - Capteurs d'humidité.	- Meilleure satisfaction client et une rentabilité accrue. - l'efficacité de sécurité.	- Maintenance et de la gestion de ces capteurs. - maintenance et de la gestion de ces capteurs. -Grande quantité de données.

1.5 Solution proposée

Sur la base de l'étude et de la critique de l'existant, nous proposons notre projet de fin d'études : une solution intelligente de surveillance et de gestion thermique basée sur l'iot pour

la société de conserve alimentaire des producteurs de tomates fraîches. Cette solution vise à améliorer efficacement l'opérations de surveillance et de contrôler le température referoidissement de produit fini le tomate en boîte. Elle remplace les méthodes obsolètes et les itinéraires prédéfinis de contrôle manuellement de température de referoidissement de l'eau par l'humain pour remplace par notre système qui équipées par un capteur de température connectés avec un serveur locaux qui relie aussi avec une application mobile, permettant de visualiser et de détecter les variations de température.

Ces outils de prise de décision sont entre les mains des gestionnaires, notamment les contrôleurs de qualité des produits et les équipes de maintenance, leur permettant de réagir rapidement et efficacement aux problèmes thermiques.

Le système consiste à la conception et le développement d'une application :

- Une application mobile qui sera utilisée par les équipes des maintenances et laboratoires.

L'application mobile permet de :

- Visualise les différentes températures.
- Surveillance en temps réel des données de température pour identifier les variations anormales.
- En cas de conditions anormales, une application mobile permet de recevoir des alertes par notification.
- Offrir la communiqué textuelle entre les utilisateurs d'application.

1.6 Conclusion

En conclusion de ce chapitre, nous avons débuté par la présentation de notre organisme d'accueil. Ensuite, nous avons réalisé une étude de l'existant afin de cerner les problématiques des sociétés de conserves alimentaires. Il en ressort que l'entreprise souhaite mettre en place une solution de surveillance et de gestion des températures via le développement d'une application mobile. Dans le prochain chapitre, nous détaillerons les spécificités de notre système mobile et procéderons à une étude conceptuelle.

ÉTAT DE L'ART

Sommaire

2.1	Introduction	13
2.2	Environnement de travail	13
2.2.1	Environnement matériel	13
2.2.2	Environnement logiciel	16
2.2.3	Outils de la rédaction du rapport	18
2.2.4	Outils de Conception	19
2.2.5	Outils de développement	20
2.3	Conclusion	22

2.1 Introduction

Après avoir finalisé l'analyse et la spécification des besoins, nous avons précédé à l'implémentation de notre application. Ce chapitre est dédié à la création et à la mise en place de notre application mobile. Dans un premier temps, nous allons présenter le langage de programmation ainsi que l'environnement de travail, par la suite nous allons passer à la présentation de l'architecture globale de l'application et enfin nous allons exposer les interfaces principales développées. L'étape d'achèvement est l'incarnation finale de l'ensemble du processus de conception. Nous allons d'abord vous présenter différents types de logiciels et les langages utilisés pour y parvenir. Ensuite, nous montrerons les différentes fonctions fournies par notre système et cela sera illustré en montrant les différentes parties de notre système

2.2 Environnement de travail

Un environnement fait référence à un ensemble de matériel et de logiciels, y compris un système d'exploitation. Dans cette section, nous décrivons l'environnement matériel et logiciel que nous avons utilisé pour implémenter notre application.

2.2.1 Environnement matériel

Pour développer notre système, nous avons utilisé un ordinateur portable ayant les caractéristiques suivantes mentionner dans le tableau 2.1 :

Marque	ASUS
Processeur	Intel® core™ I5-10300H CPU
RAM	8 GO
Disque dur	512Go SSD
Système d'exploitation	Windows 11 Professionnel

TABLE 2.1 – Environnement matériel de travail.

Dans le chapitre précédent, nous avons mentionné les besoins matériels pour la réalisation du projet. Pour répondre à ces besoins, nous avons choisi d'utiliser les composantes suivantes :

i. Choix de la Carte : Développée par Espressif [1], Le microcontrôleur ESP32 Devkit V1 est un circuit programmable qui fonctionne avec l'environnement Arduino. Il est construit autour du circuit ESP32, qui offre une puissance de traitement élevée grâce à son microprocesseur Dual-Core. Cette carte intègre la connectivité WiFi et Bluetooth, offrant ainsi une grande flexibilité pour les projets sans fil. Avec sa faible consommation d'énergie et ses capacités de stockage adéquates, elle est idéale pour les applications IoT [Net 6]. La Figure 2.1 représente la carte ESP32 devkit V1.

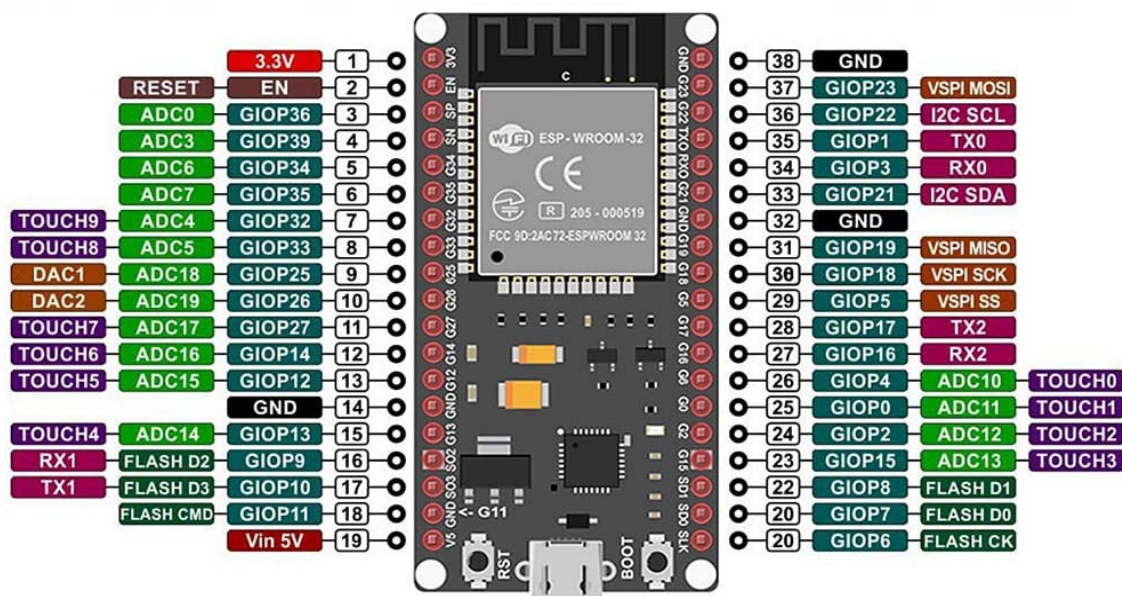


FIGURE 2.1 – Carte ESP32 devkit V1

ii. Choix du capteur de température étanche DS18B20 : Le capteur de température étanche DS18B20 est un choix idéal pour des applications nécessitant une mesure précise et fiable de la température dans des environnements humides ou immergés. Permet de mesurer une température de - 55 à + 125 °C. Elle se raccorde sur un microcontrôleur via une entrée digitale [Net 7]. La Figure 2.2 représente le Capteur de température étanche DS18B20 .

Caractéristiques :

- Alimentation : 5V
- Étendue de mesure Plage de température de fonctionnement : -55 °C à +125 °C

- Alimentation : 3.0V to 5.5V
- Longueur du câble : 100 cm



FIGURE 2.2 – Capteur de température étanche DS18B20

a) Montage du matériel

Dans cette section, nous décrivons la connexion entre nos composants matériels pour mettre en œuvre notre système.

i. Montage et Connexions du Capteur de Température avec l'ESP32 Devkit V1 : Ce montage utilise une carte ESP32 Devkit V1 et un capteur de température étanche DS18B20, ainsi qu'une résistance pull-up de 4,7 kOhm connectée entre le fil de données (jaune) du capteur DS18B20 et l'alimentation 3,3 V de l'ESP32. Les connexions sont les suivantes : le fil rouge (VCC) du capteur DS18B20 est relié au pin 3,3 V de l'ESP32, le fil noir (GND) est relié à un pin GND de l'ESP32, et le fil jaune (données) est connecté à un pin numérique D4 de l'ESP32. La résistance pull-up de 4,7 k est placée entre le fil jaune (données) et le fil rouge (VCC). Ce montage permet à l'ESP32 de lire la température mesurée par le capteur DS18B20 via le protocole OneWire. La figure 2.3 ci-dessous illustre le câblage des composants de notre système.

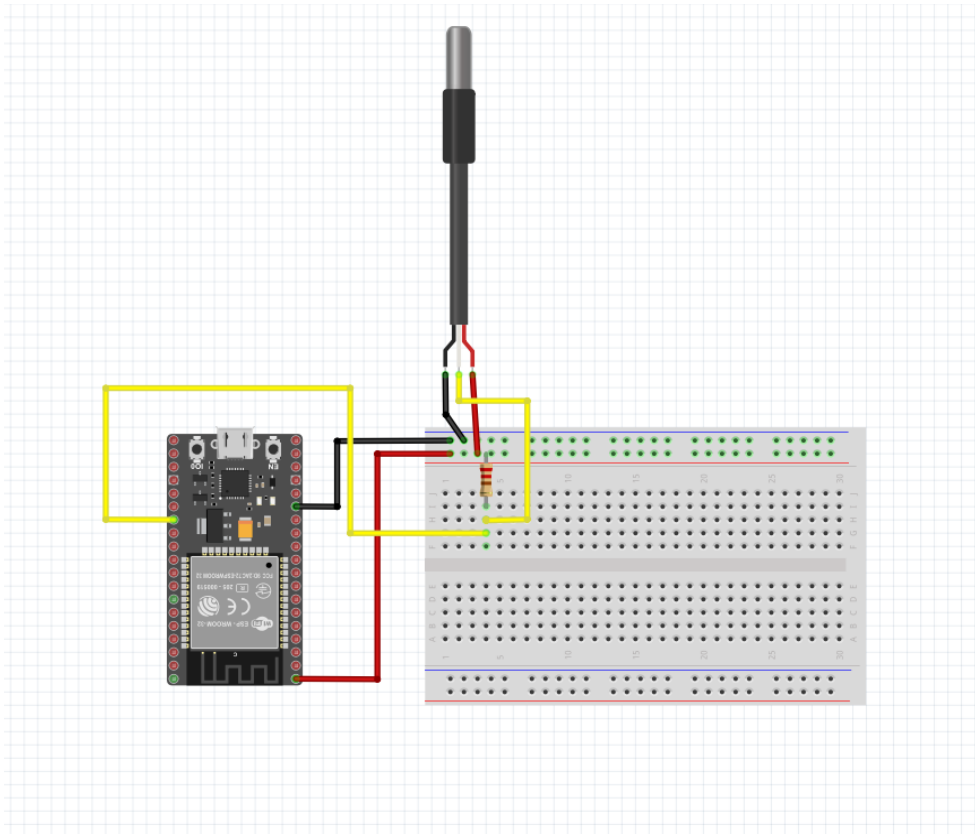


FIGURE 2.3 – Câblage des composants du système.

2.2.2 Environnement logiciel

Dans cette section, nous allons énumérer les logiciels que nous utilisons pour construire notre application.

a) Microsoft Visual Studio 2022

Visual Studio est un outil de développement puissant qui permet d'effectuer l'ensemble du cycle de développement au même endroit. Il s'agit d'un environnement de développement intégré (IDE) complet permettant d'écrire, de modifier, de déboguer et de générer du code, puis de déployer votre application. En plus de l'édition et du débogage du code, Visual Studio comprend des compilateurs, des outils de complétion de code, un contrôle de code source, des extensions et de nombreuses autres fonctionnalités qui améliorent chaque étape du processus de développement logiciel[Net 8]. La figure 2.4 ci-dessous représente le Logo de Visual Studio 2022.



FIGURE 2.4 – Logo Visual Studio 2022

b) Arduino IDE

Arduino IDE est un logiciel open source pour écrire et télécharger du code sur les cartes Arduino. Les applications IDE sont disponibles pour différents systèmes d'exploitation, tels que Windows, Mac OS X et Linux. Il prend en charge les langages de programmation C et C++ [Net 9]. La Figure 2.5 représente le Logo d'Arduino IDE.



FIGURE 2.5 – Logo Arduino IDE

c) Fritzing

Fritzing est un logiciel open source multiplateforme permet de créer des diagrammes. Le circuit que nous avons utilisé avec la carte Arduino Uno [Net 15]. La Figure 2.6 représente le Logo de logiciel Fritzing.



FIGURE 2.6 – Logo Fritzing

d) GitHub

Il vise à coordonner le travail de programmeur, mais peut être utilisé pour suivre modifications de tous les fichiers [Net 10]. La Figure 2.7 représente le Logo de GitHub.



FIGURE 2.7 – Logo GitHub

e) Stack overflow

Stack Overflow est un site web où les développeurs peuvent poser des questions sur la programmation et d'autres sujets techniques, et où d'autres développeurs peuvent fournir des réponses et des conseils. C'est une plateforme communautaire populaire pour résoudre des problèmes de programmation et partager des connaissances dans le domaine de l'informatique [Net 15]. La Figure 2.8 représente le Logo de Stack overflow.



FIGURE 2.8 – Logo stack overflow

2.2.3 Outils de la rédaction du rapport

LaTeX (et parfois les polices LATEX) est un langage de préparation de documents et un système de composition de haute qualité. Il est le plus souvent utilisé pour des documents techniques ou scientifiques de taille moyenne à grande, mais il peut être utilisé pour presque tout type de publication [Net 11]. La Figure 2.9 représente le Logo de LATEX .



FIGURE 2.9 – Logo de LaTeX

2.2.4 Outils de Conception

a) Draw.io

Diagrams.net (anciennement draw.io) est un logiciel de diagramme en ligne gratuit, basée sur le cloud. Vous pouvez l'utiliser comme créateur d'organigrammes, logiciel de diagramme de réseau, pour créer UML en ligne, comme outil de diagramme ER, pour concevoir un schéma de base de données, pour créer BPMN en ligne, comme créateur de schémas de circuits, La Figure 2.10 représente le Logo de logiciel Draw.io.



FIGURE 2.10 – Logo Draw.io

b) StarUML

StarUML est un outil de génie logiciel dédié à la modélisation UML et édité par la société coréenne MKLabs. Il est multiplateforme et fonctionne sous Windows, Linux et MacOS [Net 12]. La Figure 2.11 représente le Logo de logiciel StarUML.



FIGURE 2.11 – Logo StartUML

2.2.5 Outils de développement

Dans cette partie, nous mettons l'accent sur les logiciels utilisés tout au long de la réalisation de notre application mobile .

a) Pocketbase

PocketBase est un backend open source composé d'une base de données intégrée (SQLite) avec des abonnements (Subscribe) en temps réel, une gestion d'authentification intégrée, une interface utilisateur de tableau de bord pratique et une API REST simple [Net 13]. La Figure 2.12 représente le logo de Pocketbase.

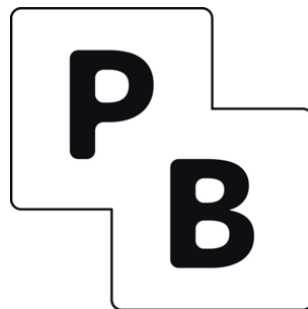


FIGURE 2.12 – Logo Pocketbase

b) Postman

Postman est un outil qui permet aux développeurs de tester et de documenter les API (Application Programming Interface). Il permet de créer des requêtes HTTP (GET, POST, PUT, etc.), de gérer les paramètres et les entêtes, de vérifier les réponses et les statuts de retour, et de

sauvegarder des requêtes pour une utilisation ultérieure [Net 14]. La Figure 2.13 représente le logo de postman .



FIGURE 2.13 – Logo de postman

Les langages de programmation

c) Xamarin

Xamarin est une plateforme open source qui permet de générer des applications modernes et performantes pour iOS, Android et Windows avec .NET. Xamarin est une couche d'abstraction qui gère la communication entre le code partagé et le code de plateforme sous-jacent. La Figure 2.14 représente le Logo du langage Xamarin .



FIGURE 2.14 – Logo Xamarin

d) Csharp

Csharp est un langage de programmation orientée objet, commercialisé par Microsoft depuis 2002 et destiné à développer sur la plateforme Microsoft .NET, au même titre que d'autres langages liés à cette plateforme (ex. : VB .NET, etc.). La Figure 2.15 représente le Logo du langage Csharp .



FIGURE 2.15 – Logo Csharp

Choix du protocole de communication

Le protocole de transfert hypertexte (HTTP) est le meilleur exemple de protocole Web internet des objets. Ce protocole constitue la base de la communication de données réseau. C'est Le protocole le plus couramment utilisé par les appareils IoT lorsqu'il y a une grande quantité de données à publier. Le protocole http est présenté par la figure 2.16.

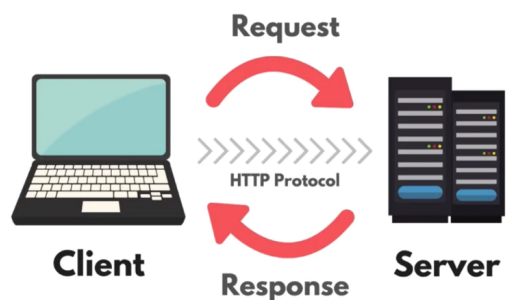


FIGURE 2.16 – Logo protocole HTTP

2.3 Conclusion

Dans ce chapitre, nous avons détaillé les caractéristiques du matériel ainsi que les différents outils nécessaires pour mener à bien le projet. Dans le prochain chapitre, nous décrirons en détail les différentes étapes nécessaires à la mise en place de notre solution.

ÉTUDE CONCEPTUELLE

Sommaire

3.1	Introduction	24
3.2	Analyse des besoins	24
3.2.1	Besoins fonctionnels	24
3.2.2	Besoins non fonctionnels	25
3.2.3	Besoins matériels	25
3.3	Architecture générale de la solution proposée	26
3.4	Conception	27
3.4.1	Vue statique	28
3.4.2	Vue dynamique	32
3.5	Conclusion	35

3.1 Introduction

Dans ce chapitre nous allons expliquer les différents besoins fonctionnels et non fonctionnels dans une première partie. Dans une deuxième partie nous allons aborder l'étude conceptuelle de notre application.

3.2 Analyse des besoins

3.2.1 Besoins fonctionnels

Les exigences fonctionnelles représentent les actions à effectuer par le système. Ils constituent le besoin primaire du client et définissent une fois résolu l'opérationnalité du système. Ainsi que le système doit permettre :

- **Aux utilisateurs** : Les utilisateurs du Système doit être capable de :

- Consulter l'interface du Système.
- Recevoir des alertes de température.
- S'authentifier.
- Gérer message.
- Consulter les informations de leur compte.

- **À l'Administrateur** : Le propriétaire de l'application

- Gérer les Utilisateurs
- Consulter l'interface administrateur de système.
- Recevoir des alertes de température.
- S'authentifier.
- Afficher et modifier les informations de leur compte.

3.2.2 Besoins non fonctionnels

Ces exigences ne sont pas spécifiquement liées au comportement du système, mais définissent plutôt les contraintes internes et externes du système. Ces exigences comprennent :

- **La disponibilité :**
 - Lorsqu'un utilisateur souhaite consulter l'application, elle doit être disponible.
- **Temps de réponse :**
 - Le temps de réponse doit être le plus court possible.
- **La sécurité :**
 - L'obligation d'établir une connexion nécessite une interface d'authentification qui permet à différents utilisateurs de se connecter.
- **Performance :**
 - Temps de réponse.
 - Le nombre d'image pour chaque 10 minute qui devraient pouvoir être analyser et traitées par le système.
- **L'intégrité :**
 - La détection des anomalies est traitée en temps réelle. – L'application doit assurer l'intégrité et la cohérence des données chaque fois que celles-ci sont mises à jour et insérées.
- **Ergonomie et souplesse :**
 - L'application doit fournir une interface conviviale et ergonomique que l'utilisateur peut utiliser en tenant compte de toutes les interactions possibles sur l'écran du stand tenu.

3.2.3 Besoins matériels

- **Au niveau de détection de température. Nous avons besoin :**
 - Un capteur de température capable de mesurer la température de refroidissement de bassin d'eau.

- Un microcontrôleur capable de mémoriser et d'exécuter un programme pour interpréter les entrées afin d'agir sur les sorties.

3.3 Architecture générale de la solution proposée

La figure 3.1 représente la mise en place de la solution qui consiste à la récupération de la carte Esp32 devkit V1 des données collectées par le capteur de température(DS18B20) qui mettre dans le bassin de tour de refroidissement,ce dernier envoyer par la suite à l'application mobile pour affiche les données collecte ou notifie à travers la base de données, ce dernier stocke les données collectées.

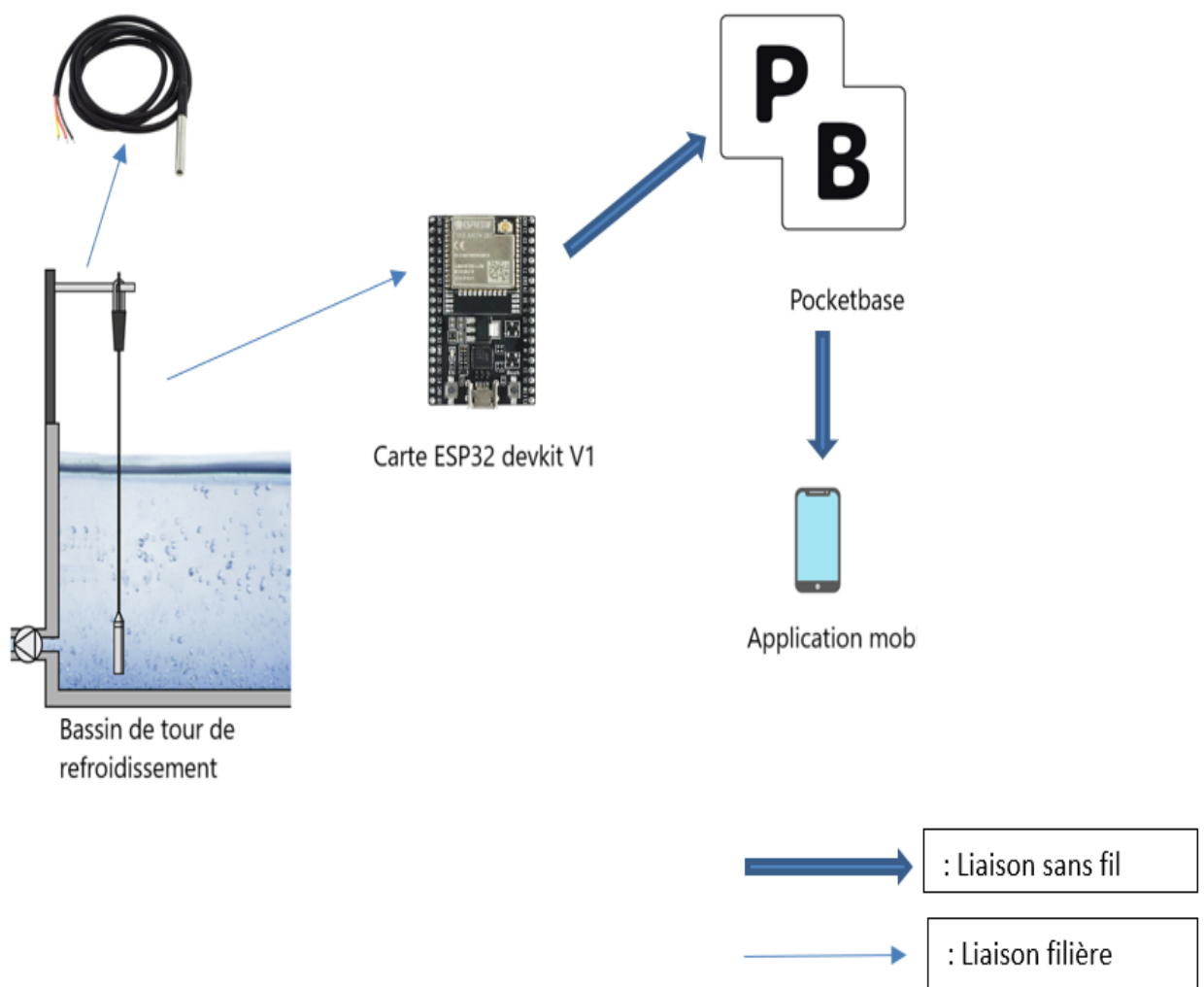


FIGURE 3.1 – Architecture de notre système

3.4 Conception

Identification des Acteurs :

L'identification des acteurs comprend un inventaire des différents utilisateurs qui vont interagir directement ou indirectement avec le système. Le tableau 3.1 identifie les acteurs de notre système. Pour répondre aux exigences fonctionnelles de notre système, nous avons 2 classes d'acteurs qui sont les utilisateurs et l'administrateur. Ils vont vérifier et gérer les fonctionnalités fournies par le système.

TABLE 3.1 – Tableau des Acteurs

Acteurs	Description	Besoin fonctionnelle
Utilisateur	Une personne qui va utiliser l'application mobile	<ul style="list-style-type: none"> • Consulter les différent température • Recevoir une notification en cas d'échec • S'authentifier • Permet de communique avec les différentes utilisateur.
Administrateur	Un superviseur qui va utiliser l'application mobile	<ul style="list-style-type: none"> • Gérer les utilisateurs : Cette fonctionnalité permet à l'administrateur de gérer les comptes d'utilisateurs comme ajouter ou la suppression d'un compte. • Consulter temperaturer : Cette fonctionnalité permet à l'administrateur d'avoir les differént température . • S'authentifier • Permet d'envoyer des message pour les differentes utilisateur • Recevoir une notification en cas d'échec qui represent l'augmentation anormale de température.

3.4.1 Vue statique

3.4.1.1 Diagrammes de cas d'utilisation l'interaction entre les utilisateurs et le systeme

a) Diagramme de cas d'utilisation général

Le diagramme de cas d'utilisation capture le comportement fonctionnel du système. Nous allons présenter la fonctionnalité du système dans un diagramme de cas d'utilisation global afin de simuler les besoins des utilisateurs.

Le diagramme ci-dessous représente les principaux cas d'utilisation de notre système :

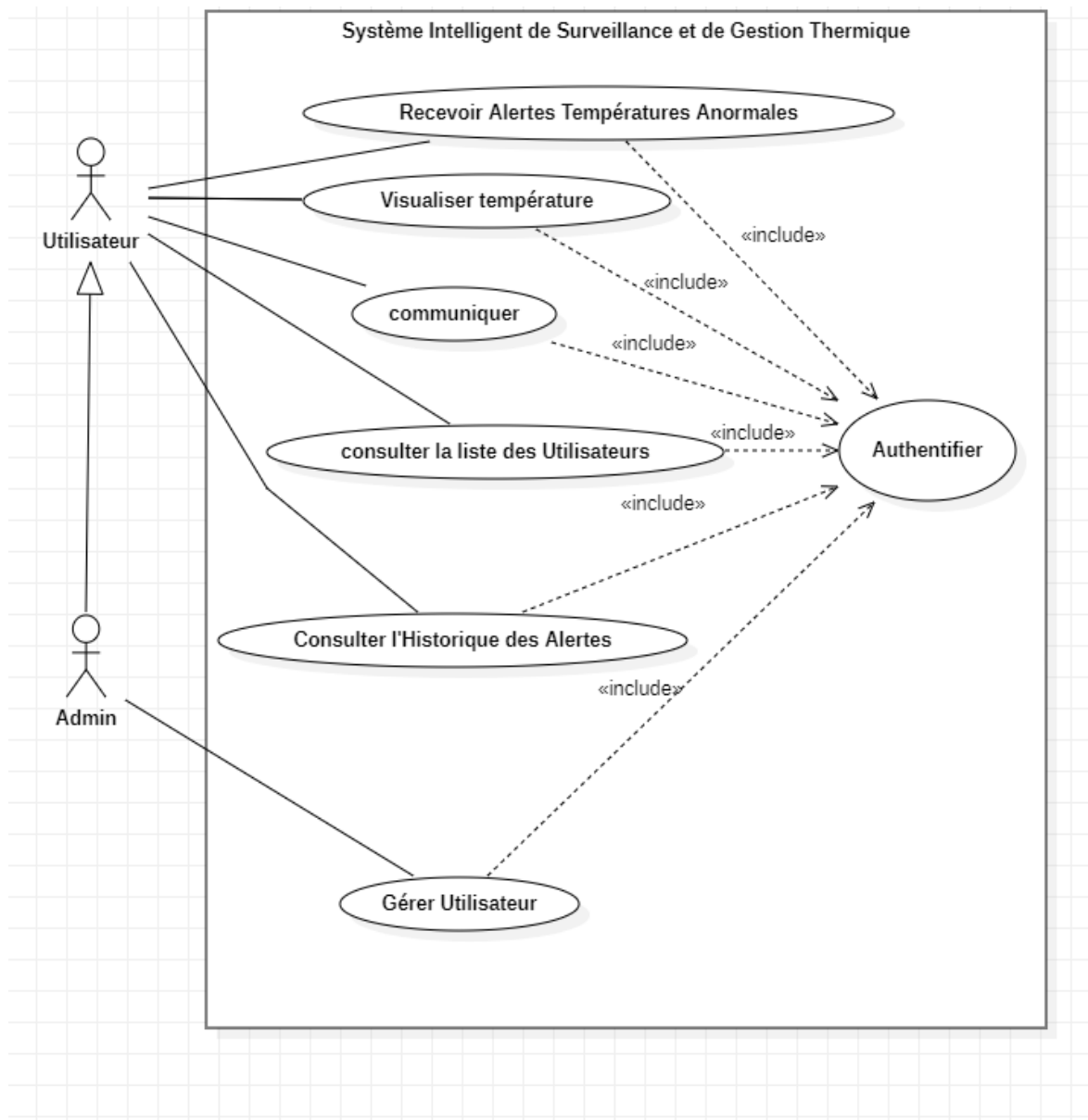


FIGURE 3.2 – Diagramme de cas d'utilisation Global

b) Diagramme détaillé des cas d'utilisation d'administrateur

• L'administrateur est le responsable laboratoire ou maintenance qui gérera l'application. Il doit être authentifié tout d'abord, ceci va lui donner la possibilité d'administrer toute l'application, les vérifications des comptes, modification de l'application. Le scénario est comme suit : le cas d'utilisation commence Lorsque l'administrateur s'authentifie, dès qu'il sera reconnu, il sera autorisé pour gérer la mise à jour du contenu de l'application comme il peut éditer son profil. Si l'email ou le mot de passe sont invalides, l'accès aux ressources demandées est refusé.

• la Figure 3.3 ci-dessous représente le diagramme de cas d'utilisation de ce dernier :

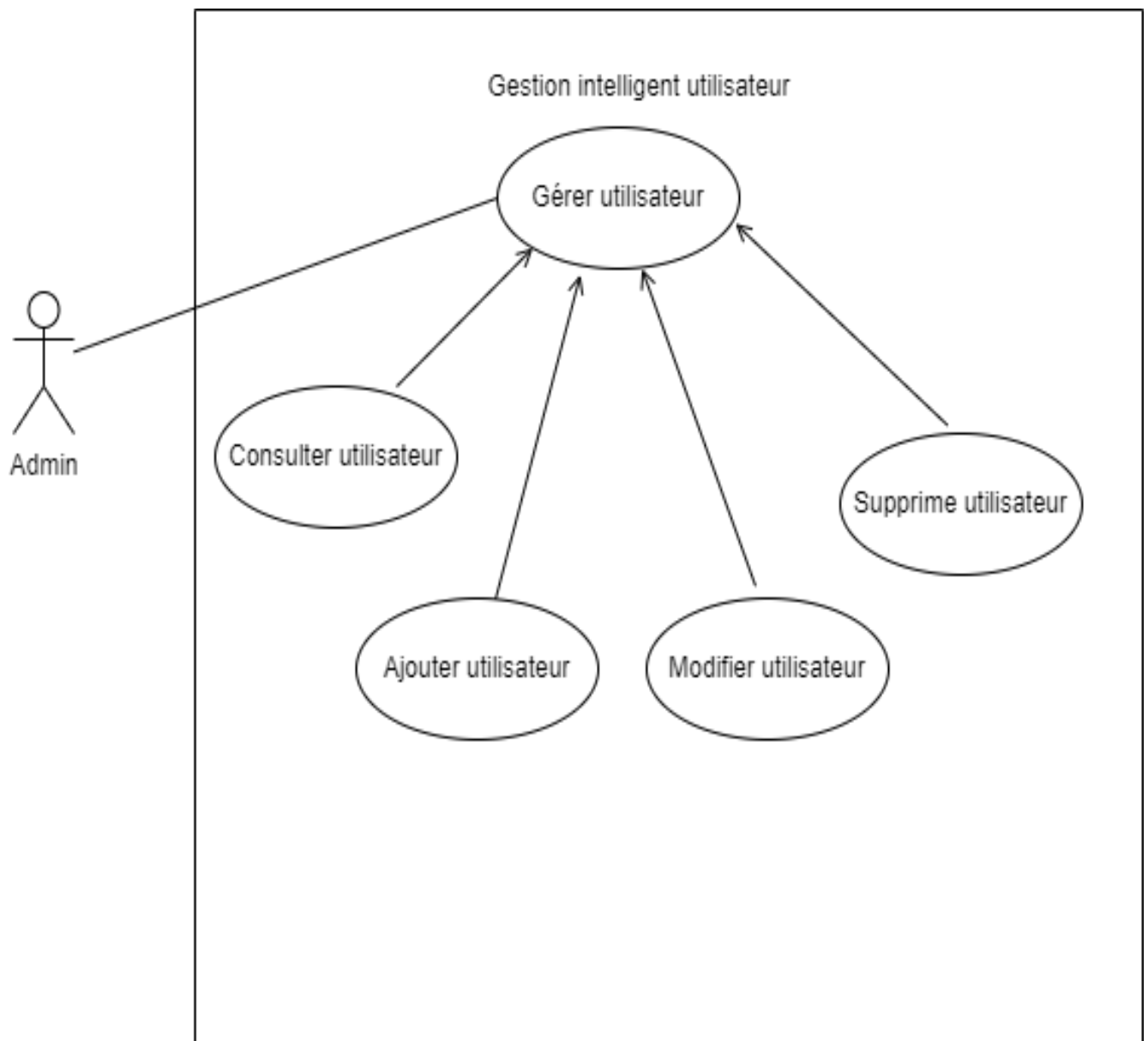


FIGURE 3.3 – Diagramme de cas d'utilisation gérer utilisateur

La description textuelle de ce diagramme est représentée par le tableau 3.2 ci-dessous :

Acteur	Administrateur
Objectif	<ul style="list-style-type: none"> - Le but est de permettre à l'administrateur d'ajouter, supprimer, modifier et consulter les utilisateur. - L'administrateur doit mettre à jour les informations des utilisateurs (Nom utilisateur, mot de passe, Post, Email), et leur compte.
Préconditions	<ul style="list-style-type: none"> - La disponibilité de l'Internet. - L'acteur est authentifié.
Postcondition	<p>Les interfaces :</p> <ul style="list-style-type: none"> • Modifie leur donnée. • Les différents températures. • Messagerie. • Notification
Scénario principal	<ul style="list-style-type: none"> - L'administrateur utilise l'application serveur pour consulter les données après l'exécution de l'application. - L'application demande l'authentification. - L'administrateur saisir leur adresse mail et mot de passe pour s'authentification au système. - L'administrateur clique sur le bouton "Connexion" pour envoyer ses informations d'identification au système d'authentification. . - Le système vérifie que le nom d'utilisateur et le mot de passe sont corrects. - Si les informations sont correctes, l'administrateur est dirigé vers la page homePage d'administration.
Scénario alternatif	<ul style="list-style-type: none"> - L'application continue d'indiquer que les informations de l'identifiant sont incorrectes jusqu'à la fin de la saisie des informations correcte. - Indisponibilité d'internet

TABLE 3.2 – Description textuelle du diagramme de cas d'utilisation de l'administrateur

3.4.1.2 Diagrammes des classes

Après avoir finalisé les diagrammes de cas d'utilisation, nous représentons dans la Figure 3.4 ci-dessous de diagramme de classe de notre système :

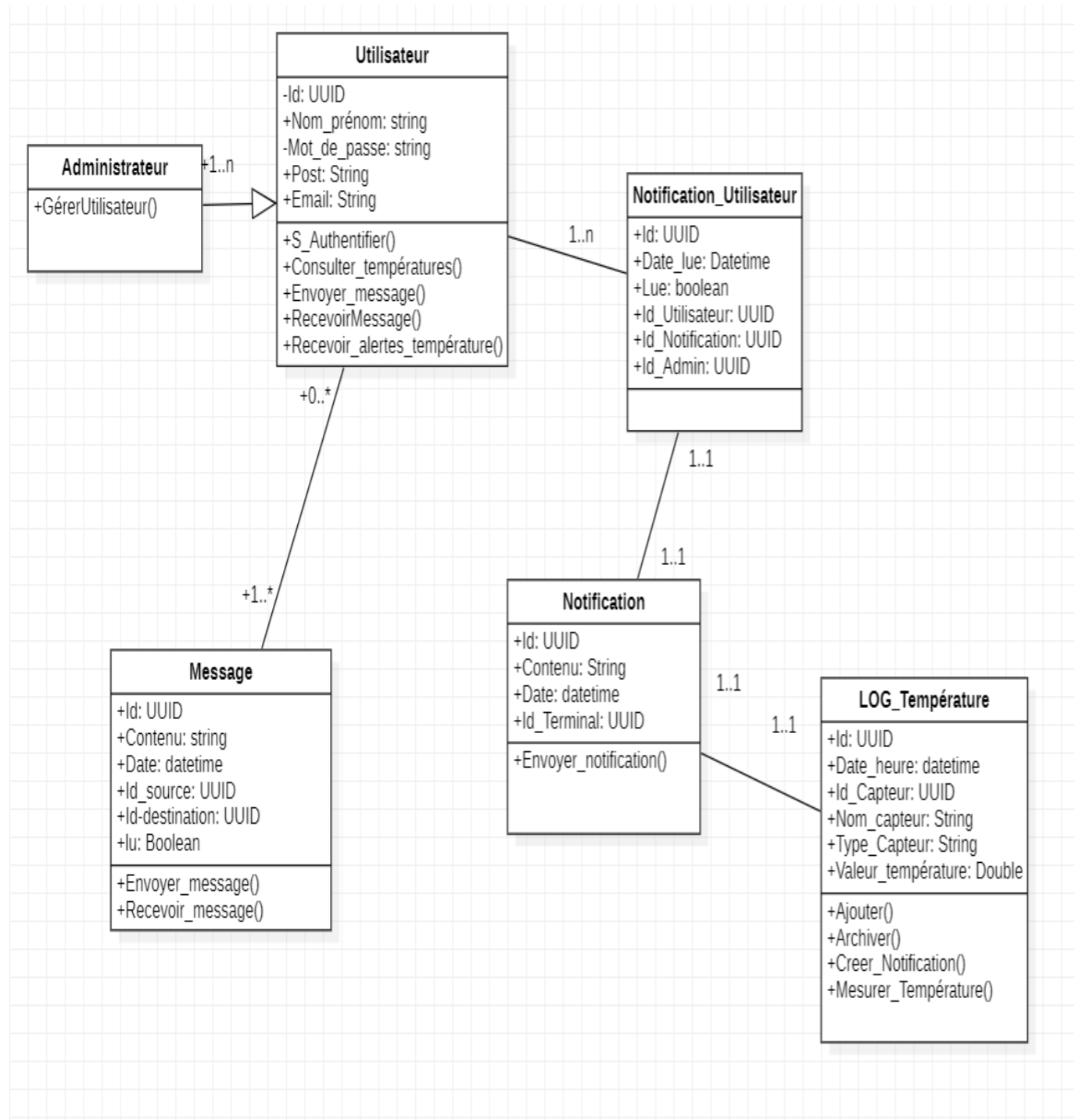


FIGURE 3.4 – diagramme de classes

3.4.2 Vue dynamique

3.4.2.1 Diagrammes des séquences

Les diagrammes de classes sont considérés comme les plus importants dans la modélisation orientée objet. Il montre la structure interne. Il fournit une représentation abstraite des objets du système qui vont interagir pour réaliser un cas d'utilisation. Dans cette section, nous allons montrer tous les diagrammes de séquence de notre application mobile qui représentent la vue dynamique du diagramme de cas d'utilisation. Les diagrammes de séquence nous permettent de visualiser des séquences d'actions dans le temps.

a) Diagramme de séquence « Authentification »

Le diagramme de séquence au- dessous présenté décrit le processus d'authentification d'un utilisateur dans une application mobile. L'utilisateur commence par lancer l'application, ce qui affiche l'interface d'authentification. Il saisit ensuite son login et son mot de passe, et l'interface vérifie le format des champs. Si les informations saisies sont incorrectes, un message d'erreur est affiché à l'utilisateur. Si les champs sont corrects, les identifiants sont envoyés au contrôleur, qui vérifie les informations auprès de la base de données Pocketbase. Si les identifiants sont incorrects, un message d'erreur est renvoyé, mais si les informations sont correctes, le contrôleur distingue si l'utilisateur est un simple utilisateur ou un administrateur. Selon le type d'utilisateur authentifié, il est redirigé vers 'HomePageUtilisateur' s'il est un utilisateur standard ou vers 'HomeAdminPage' s'il est un administrateur. En cas d'authentification réussie, la session utilisateur est initialisée. Ce processus assure une gestion complète de l'authentification, de la saisie initiale des identifiants à la redirection vers la page appropriée ou l'affichage d'un message d'erreur.

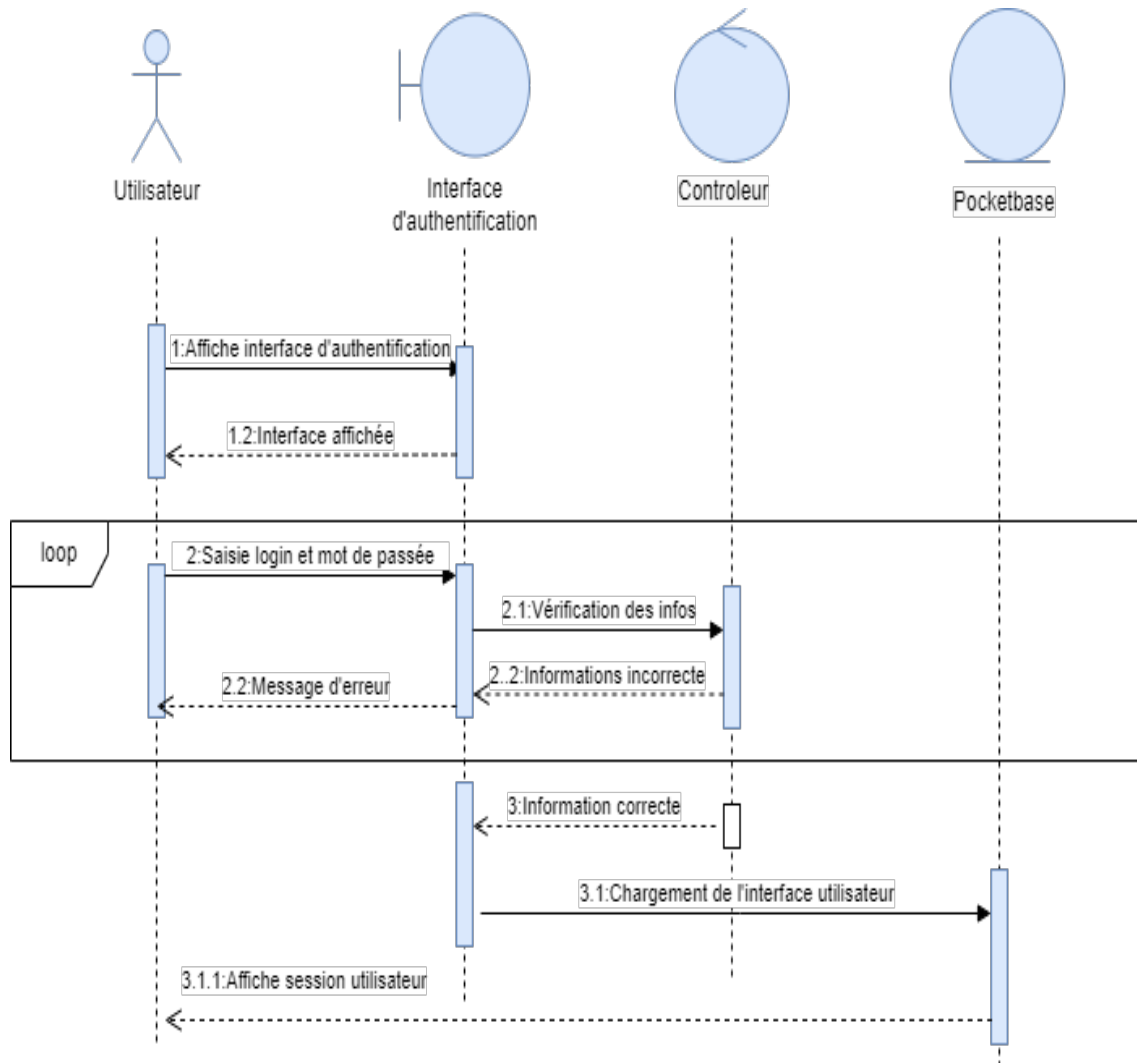


FIGURE 3.5 – Diagramme de séquence « Authentification »

b) Diagramme de séquence « Afficher notification »

Le diagramme de séquence décrit le processus de gestion des notifications dans l'application mobile après l'authentification de l'utilisateur. Sur la page d'accueil, un bouton "Notification" est disponible. Lorsque l'utilisateur clique sur ce bouton, le système envoie une requête à la base de données PocketBase pour récupérer les notifications associées à l'utilisateur authentifié. Les notifications sont ensuite affichées à l'utilisateur. Chaque notification non lue est marquée par un point rouge, tandis que les notifications déjà lues n'ont pas ce point rouge. Ce mécanisme permet à l'utilisateur de distinguer facilement les notifications qu'il n'a pas encore consultées.

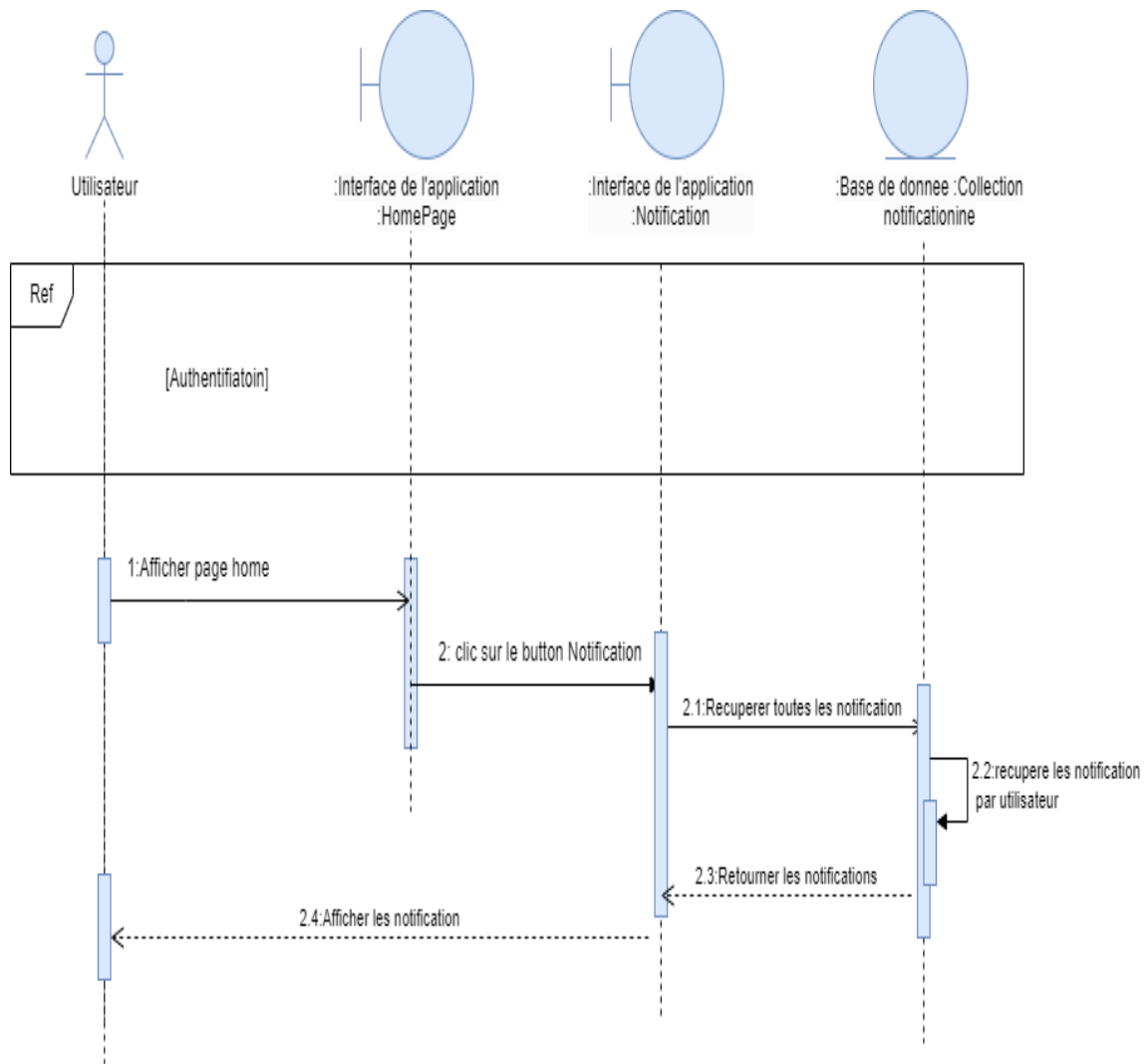


FIGURE 3.6 – Diagramme de séquence « notification »

Diagramme de séquence « envoyer message »

Ce diagramme de séquence illustre le processus d'envoi d'un message par un utilisateur au sein de l'application. Tout d'abord, l'utilisateur consulte l'application, qui récupère et affiche la liste de tous les utilisateurs disponibles. Ensuite, l'utilisateur clique sur le nom de l'utilisateur destinataire, ce qui navigue vers la page "ComposeMessage". Sur cette page, l'utilisateur compose son message et clique sur le bouton "Envoyer". L'application confirme alors l'envoi du message avec succès et le stocke localement. Enfin, l'application envoie le message à la base de données PocketBase pour une mise à jour et un stockage permanent. Ce processus permet à l'utilisateur de composer et d'envoyer des messages à d'autres utilisateurs.

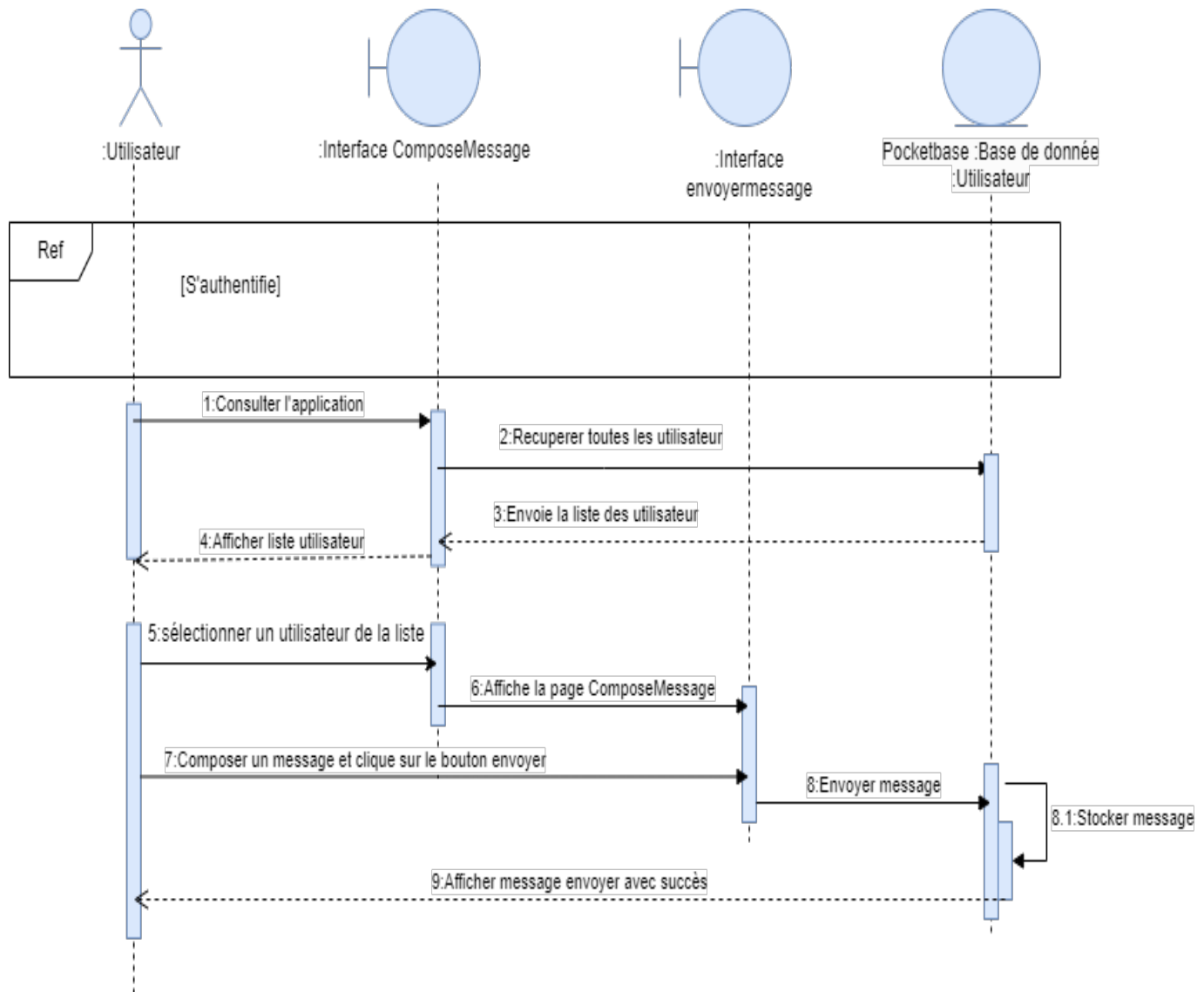


FIGURE 3.7 – Diagramme de séquence « envoyer message »

3.5 Conclusion

Dans ce chapitre, nous commençons par la spécification des exigences des parties prenantes, puis nous présentons les diagrammes de cas d'utilisation, les diagrammes de séquence, le diagramme de classe et le diagramme de déploiement de notre système. Dans le chapitre suivant, nous allons nous positionner sur notre projet, en exposant la réalisation de notre système tel que l'environnement du travail, l'architecture et les interfaces développées.

RÉALISATION ET ÉVALUATION

Sommaire

4.1	Introduction	37
4.2	Tests et résultats	37
4.2.1	Étapes de configuration	37
4.2.2	Outils de test	46
4.2.3	Scénarios et résultats	47
4.3	Conclusion	51

4.1 Introduction

Dans ce chapitre, nous abordons la concrétisation de notre projet de fin d'études qui vise à mesurer le température de referoidissement de produit fini "tomate en boîte" à l'aide d'un capteur étanche WATERPROOF DS18B20 et d'une carte ESP32 devkit v1. De plus, nous avons développé une application mobile permettant aux utilisateurs de visualiser les mesures de température à distance et de recevoir des notifications en cas de dépassement des limites de température prédéfinies. Ce chapitre détaillera la mise en place matérielle, le développement logiciel et les tests effectués pour assurer le bon fonctionnement de notre système.

4.2 Tests et résultats

4.2.1 Étapes de configuration

a) Partie pocketbase :

i. Création d'un projet pocketbase : Notre projet pocketbase nous permet de développer rapidement notre application de qualité, ce qui facilite notre travail, car il permet le partage d'utilisateurs et de données entre différentes plateformes (Android, Windows, iOS). La procédure de création d'un projet Pocketbase est la suivante :

1. Accéder à la documentation pocketbase puis télécharger pocketbase sur < downloadv0.22.12 for ... >. La figure 4.1 représente la documentation de pocketbase :

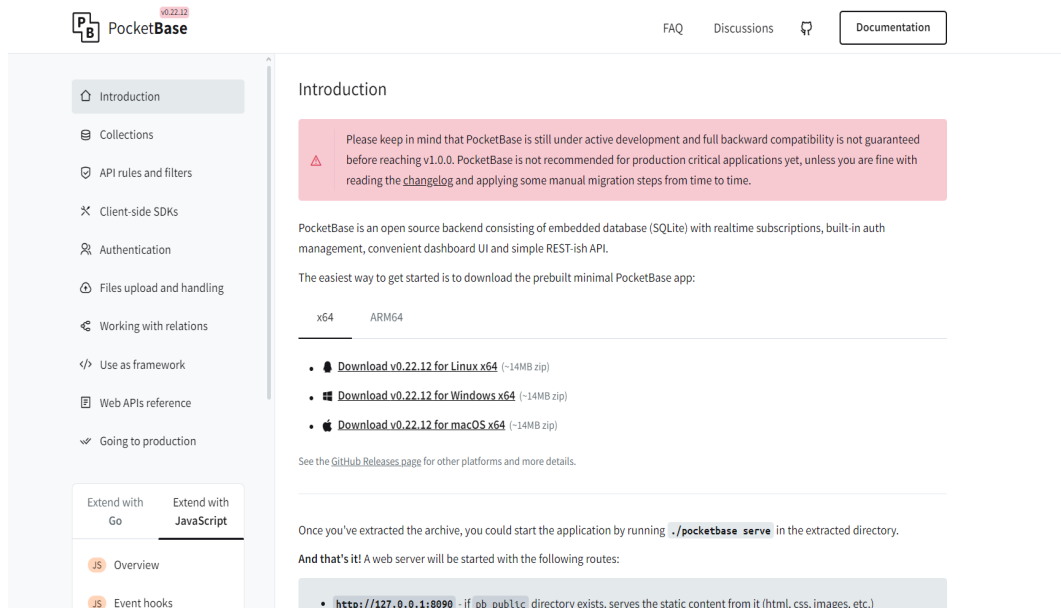


FIGURE 4.1 – Téléchargement Pocketbase

2. Après l'installation de PocketBase sur notre machine, nous allons lancer le serveur PocketBase localement en utilisant la commande `pocketbase serve`, qui permet de démarrer notre serveur localement et de le réserver sur le port 8090. Une fois lancé, le serveur affiche trois liens. Nous travaillerons principalement avec le lien qui fournit l'interface administrateur de PocketBase pour gérer notre base de données. Ensuite, vous nommez votre projet et cliquez sur "Continuer". La figure 4.2 représente le lancement du serveur local PocketBase.

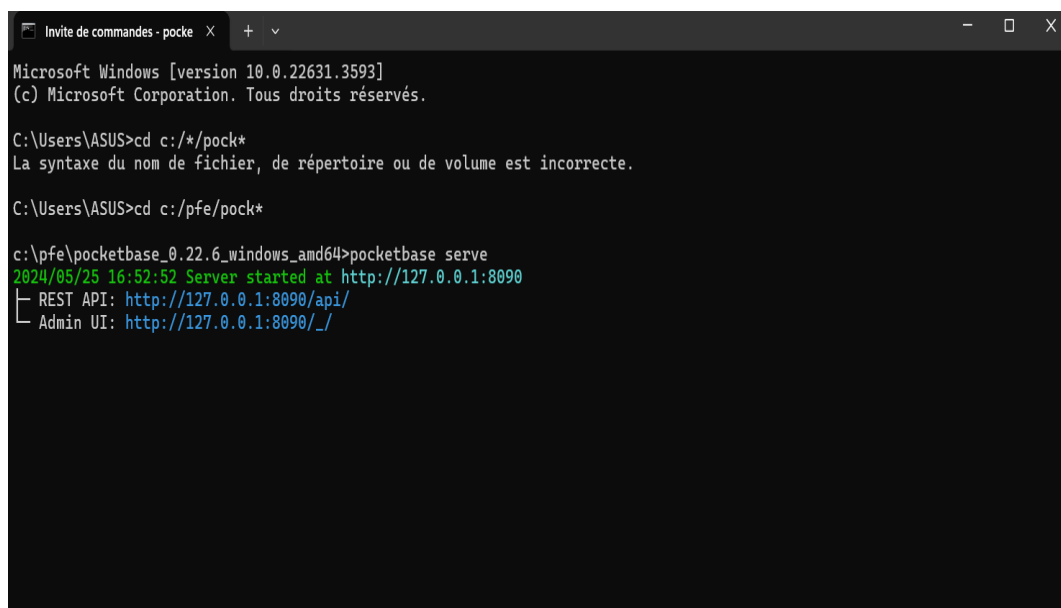


FIGURE 4.2 – Lancement de serveur local pocketbase utilisant cmd

3. Après le lancement de notre serveur PocketBase, nous cliquons sur le lien "Admin UI" pour accéder à la page d'authentification de PocketBase, comme illustré à la Figure 4.3. Ensuite, nous procédons à la création de notre base de données pour l'application Xamarin et C. La Figure 4.4 montre comment créer une collection sur PocketBase.

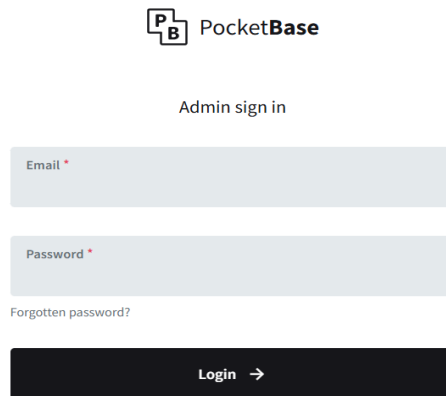
The image shows the PocketBase Admin sign in interface. At the top, there is a logo consisting of a square with 'P' and 'B' inside, followed by the text 'PocketBase'. Below the logo, the text 'Admin sign in' is centered. There are two input fields: the first is labeled 'Email *' and the second is labeled 'Password *'. Below the password field, there is a link that says 'Forgotten password?'. At the bottom, there is a dark button with the text 'Login →'.

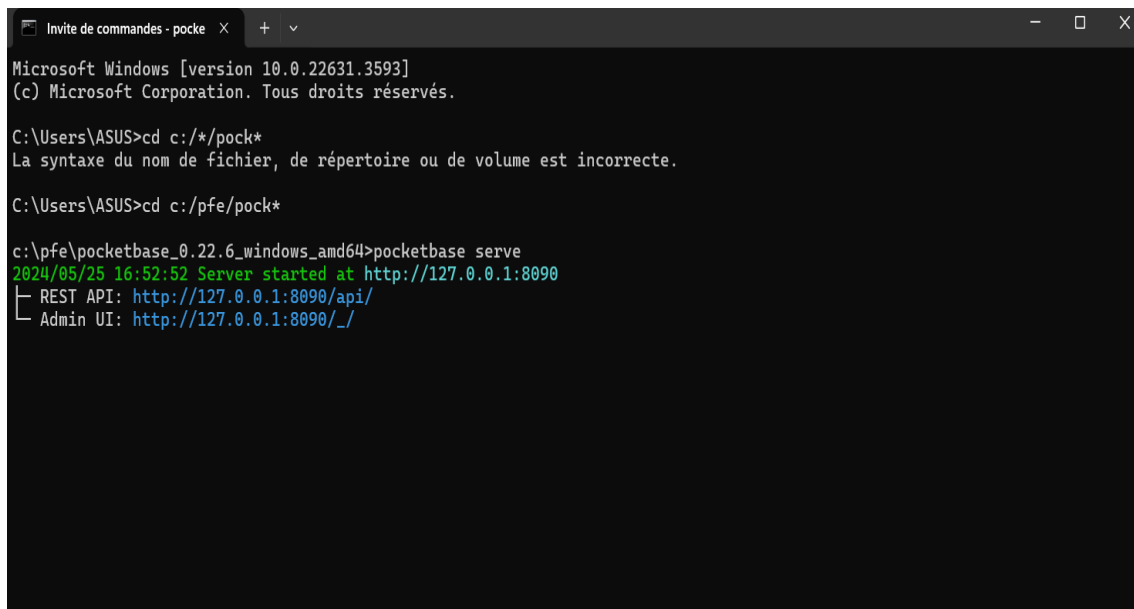
FIGURE 4.3 – Page d'authentification Pocketbase

4. Après l'authentification, cette image montre l'interface de PocketBase pour créer une nouvelle collection. On y définit le nom de la collection (ici "posts") et son type (défini sur "Base"). La section "Champs" permet de définir les champs de la collection, incluant les champs système par défaut "id", "created" et "updated". Les "Règles API" permettent de configurer les accès API, tandis que la section "Contraintes uniques et index" permet d'ajouter des contraintes et des index. Cette interface permet ainsi de configurer en détail une nouvelle collection, créant la structure de données nécessaire pour l'application.

The screenshot shows the 'New collection' form in Pocketbase. At the top, there's a title 'New collection'. Below it, a form field for 'Name' with a red asterisk and a hint 'eg. "posts"' is shown. To the right of the name field is a dropdown menu for 'Type' set to 'Base'. Below the name field, there are two tabs: 'Fields' (active) and 'API Rules'. Under the 'Fields' tab, it lists 'System fields: id, created, updated'. Below this is a button '+ New field'. Further down, it says 'Unique constraints and indexes (0)' with a button '+ New index'. At the bottom right, there are 'Cancel' and 'Create' buttons.

FIGURE 4.4 – Creation collection Pocketbase"

ii. Utilisation pocketbase à une application Csharp 1-Nous allons lancer notre serveur PocketBase localement sur notre machine afin d'exécuter des requêtes sur notre base de données en utilisant l'invite de commande pour démarrer le serveur avec la commande 'pocketbase serve'. La figure 4.5 montre le lancement du serveur PocketBase en local.



```
Invite de commandes - pocke
Microsoft Windows [version 10.0.22631.3593]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\ASUS>cd c:/*/*pock*
La syntaxe du nom de fichier, de répertoire ou de volume est incorrecte.

C:\Users\ASUS>cd c:/pfe/pock*

c:/pfe/pocketbase_0.22.6_windows_amd64>pocketbase serve
2024/05/25 16:52:52 Server started at http://127.0.0.1:8090
└─ REST API: http://127.0.0.1:8090/api/
  └─ Admin UI: http://127.0.0.1:8090/_/
```

FIGURE 4.5 – lancement du serveur PocketBase en local

2-Nous ouvrons Microsoft Visual Studio et ouvrons notre projet. Ensuite, nous faisons un clic droit sur le nom du projet dans l'explorateur de solutions à droite, puis nous sélectionnons l'option "Gérer les packages NuGet". Cette étape est illustrée dans la figure 4.6.

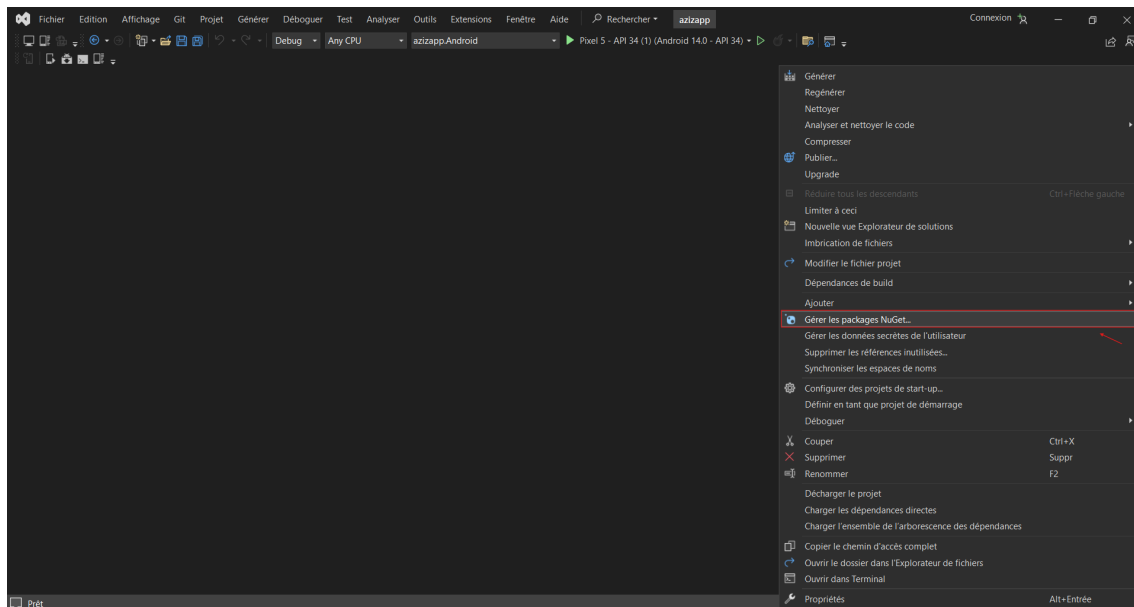


FIGURE 4.6 – Gérer les paquets NuGet.

3-Dans cette étape, nous avons utilisé la barre de recherche pour rechercher le package HTTP dans notre environnement de développement. Après avoir trouvé le package System.Net.Http associé à notre projet, nous avons sélectionné la version compatible et téléchargé le package

en cliquant sur le bouton de téléchargement. Cette action était nécessaire car notre application nécessite une communication avec la base de données PocketBase via des requêtes HTTP, comme illustré dans la figure 4.7.

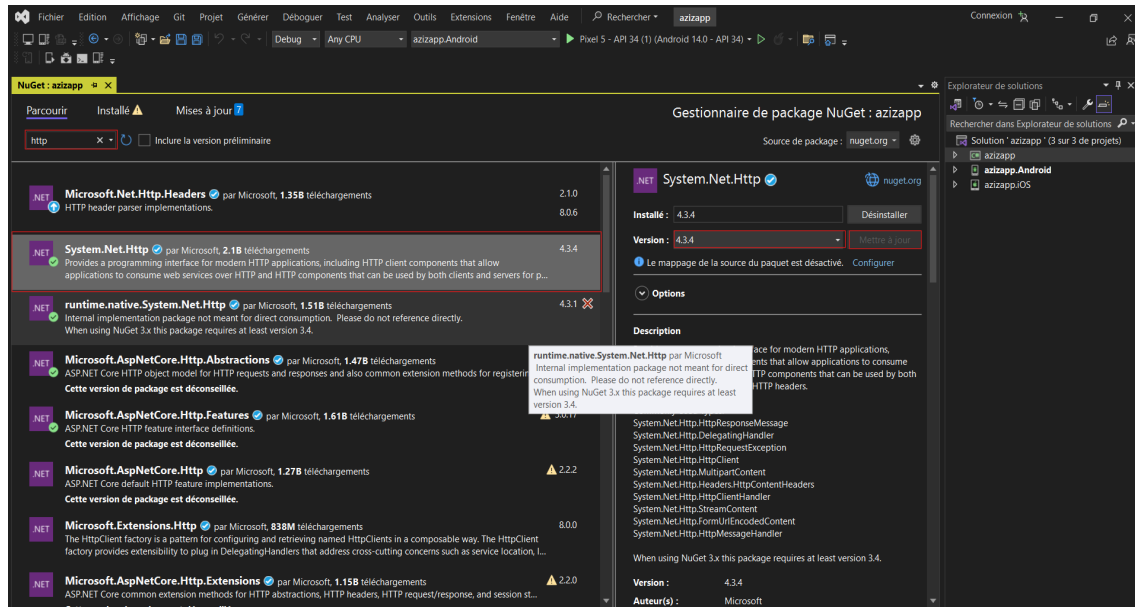


FIGURE 4.7 – Téléchargement le package System.Net.HTTP

4- Ce code Csharp démontre comment créer un nouveau record dans une base de données en utilisant PocketBase via des requêtes HTTP. Il initialise un client HTTP, envoie une requête POST avec les données du nouveau record, puis traite la réponse. Cette approche simple permet d'intégrer facilement PocketBase dans des applications C pour gérer des opérations de base de données. La Figure 4.8 représente un exemple d'utilisation pocketbase avec Csharp :

```

using System.Net.Http;
using System.Text;
using System.Threading.Tasks;

class Program
{
    static async Task Main(string[] args)
    {
        // Création d'un client HTTP
        using (var client = new HttpClient())
        {
            // Définition de l'URL de votre serveur PocketBase pour créer un nouveau record
            string url = "http://localhost:8080/api/data";

            // Création des données à envoyer (par exemple, un objet JSON)
            string jsonData = "{\"name\":\"John\",\"age\":30,\"city\":\"New York\"}";

            // Création du contenu de la requête
            var content = new StringContent(jsonData, Encoding.UTF8, "application/json");

            try
            {
                // Envoi de la requête POST avec les données JSON
                HttpResponseMessage response = await client.PostAsync(url, content);

                // Vérification de la réponse
                if (response.IsSuccessStatusCode)
                {
                    // Lecture de la réponse JSON
                    string jsonResponse = await response.Content.ReadAsStringAsync();

                    // Affichage de la réponse
                    Console.WriteLine("Réponse JSON de PocketBase :");
                    Console.WriteLine(jsonResponse);

                    // Vous pouvez analyser la réponse JSON ici pour extraire des informations si nécessaire
                }
                else
                {
                    Console.WriteLine($"Échec de la création du nouveau record avec le code : {response.StatusCode}");
                }
            }
            catch (HttpRequestException e)
            {
                Console.WriteLine($"Une erreur s'est produite lors de la requête HTTP : {e.Message}");
            }
        }
    }
}

```

FIGURE 4.8 – Exemple d'utilisation PocketBase avec Csharp

b) Code arduino

Le code développé surveille la température avec un capteur DS18B20 connecté à une carte ESP32 Devkit V1. Il se connecte au WiFi, initialise le capteur et, dans une boucle principale, lit la température, affiche la valeur et envoie les données au serveur PocketBase via une requête HTTP. Ce système IoT permet une surveillance en temps réel de la température des boîtes de tomates pour améliorer le contrôle de la qualité et la sécurité des produits. Les figures 4.9 montrent le code Arduino utilisé.

```
const char* serverUrl = "http://192.168.1.21:8090/api/collections/Log_temperature/records";

// GPIO où le capteur DS18B20 est connecté
#define ONE_WIRE_BUS 4

// Informations sur le capteur
const char* id_terminal = "TEMPREFROIDISSIMENT";
const char* nom_terminal = "Capteur de température de refroidissement";
const char* type_terminal = "Capteur";

// Initialisation des instances
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

void setup() {
  Serial.begin(9600);

  // Connexion WiFi
  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println(" connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());

  // Initialisation du capteur de température
  sensors.begin();
}

void loop() {
  // Demande la température au capteur
  sensors.requestTemperatures();
  float temperature = sensors.getTempCByIndex(0);

  // Affichage de la température lue
  Serial.print("Temperature: ");
  Serial.println(temperature);

  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;

    // Préparation de la requête POST
    http.begin(serverUrl);
    http.addHeader("Content-Type", "application/json");

    // Données à envoyer
    String postData = "{\"Valeur_temperature\": " + String(temperature) +
      ", \"id_terminal\": \"\" + id_terminal +
      ", \"Nom_terminal\": \"\" + nom_terminal +
      ", \"Type_terminal\": \"\" + type_terminal + "\"}";

    // Envoi de la requête
    int httpResponseCode = http.POST(postData);

    if (httpResponseCode > 0) {
      String response = http.getString();
      Serial.print("HTTP Response code: ");
      Serial.println(httpResponseCode);
      Serial.println("Response: ");
      Serial.println(response);
    } else {
      Serial.print("Error on sending POST: ");
      Serial.println(httpResponseCode);
      Serial.println(http.errorToString(httpResponseCode));
    }

    // Fin de la requête
    http.end();
  } else {
    Serial.println("Error in WiFi connection");
  }

  // Pause avant la prochaine lecture
  delay(60000); // Lit toutes les 60 secondes
}

```

FIGURE 4.9 – Code arduino

c) Installation de l'esp32 Devkit v1

Il existe plusieurs méthodes pour programmer la carte ESP32 devkit V1, en fait il existe plusieurs outils IDE. Nous avons utilisé l'outil Arduino IDE qui est très facile à utiliser. Programmer la carte ESP32 devkit V1 avec ce logiciel est très simple, il est donc nécessaire de faire quelques préparatifs pour pouvoir programmer la carte ESP32 devkit V1. Veuillez donc suivre les étapes des figures 4.10, 4.11, 4.12. **Étape 1.** Téléchargez le logiciel Arduino.

Étape 2. Ajouter la carte ESP32 à la base des cartes : Dans notre Arduino IDE :

- Ouvrir l'onglet Fichier » Préférences
- Entrer l'adresse suivante dans le champ URL de gestionnaire de cartes supplémentaires

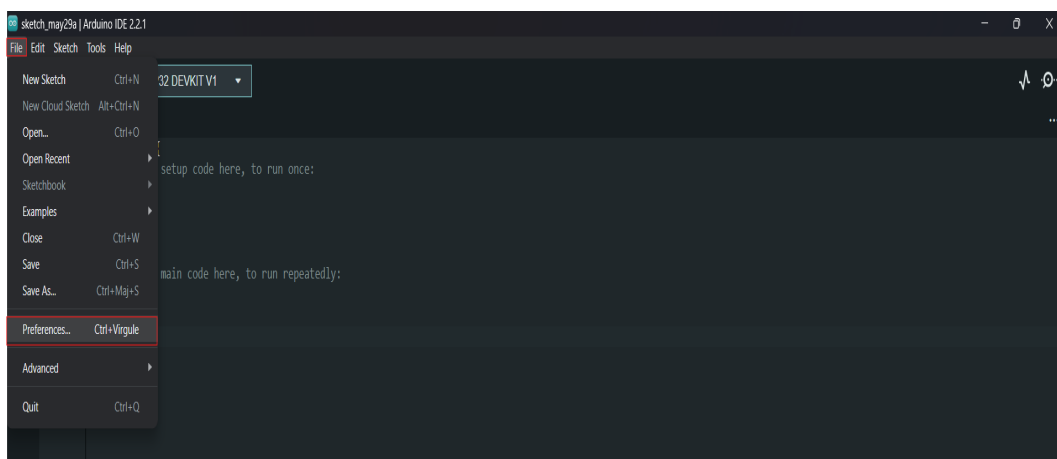


FIGURE 4.10 – Arduino IDE File→Préférences

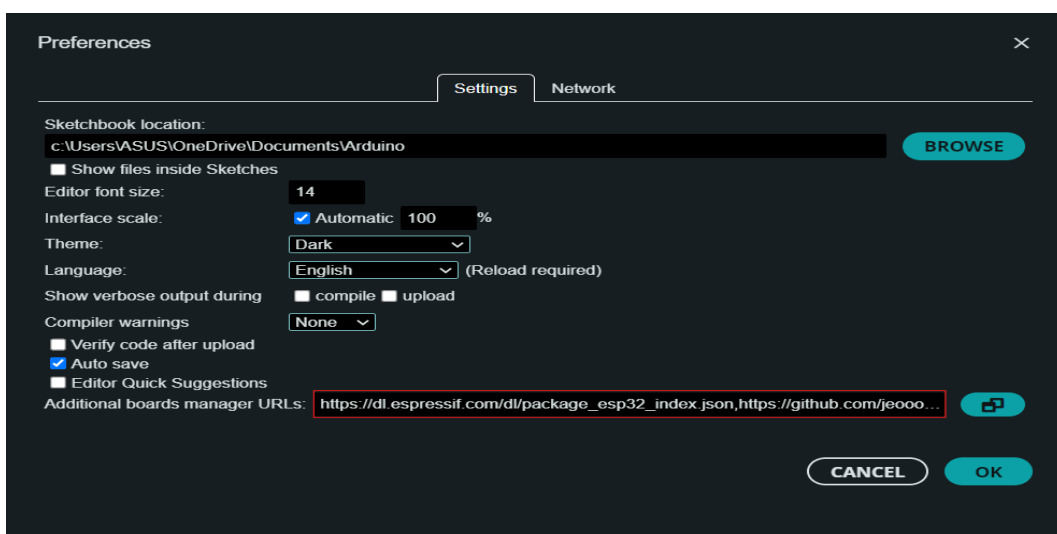


FIGURE 4.11 – Remplissage du champ URL de gestionnaire de cartes supplémentaires.

Étape 3 : Installer la carte ESP32 devkit V1 dans l'IDE Arduino :

Toujours dans notre Arduino IDE :

- Outil » Type de carte » Gestionnaire de carte
- Écrire ESP32 dans la barre de recherche
- Installer le paquet : choisir la dernière version disponible et cliquer sur Installer

Cliquez sur Outils → Carte → esp32, recherchez ESP32 devkit V1 dans la nouvelle fenêtre de dialogue.

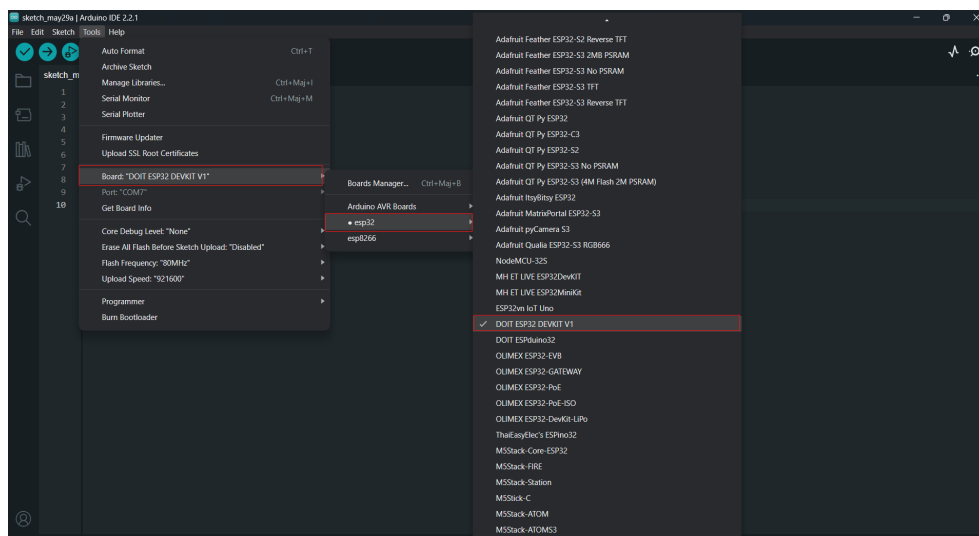


FIGURE 4.12 – choisi de carte esp32 devkit V1

4.2.2 Outils de test

Pour tester le fonctionnement de notre application, nous pouvons utiliser deux méthodes, soit un test réel avec le smartphone, soit par un émulateur.

Émulateur ou test réel

Pour tester notre application, nous avons utilisé deux smartphones : un smartphone réel et un smartphone virtuel "émulateur", dont les caractéristiques sont présentées dans le tableau 4.1.

TABLE 4.1 – : Les caractéristiques de nos smartphones.

smartphone	1	2
propriétaire	emulateur virtuelle	Mohamed Aziz
Marque	pixel 5 - API 34	Samsung A20
Version Android	Android 14	Android 11
RAM	1Go	4Go

4.2.3 Scénarios et résultats

Dans cette partie, nous présentons quelques interfaces de notre application.

a) Interface d'accueil

La figure 4.13 montre l'interface mobile de notre application Android, qui permet à l'utilisateur d'accéder aux différentes fonctionnalités de l'application. L'interface d'authentification permet aux utilisateurs de se connecter en saisissant leur login et mot de passe. Contrairement à la création de comptes, cette interface se limite uniquement à l'authentification, les comptes étant créés uniquement par l'administrateur. Une fois les informations saisies, le système effectue un contrôle de saisie. Si l'utilisateur authentifié est un administrateur, il est redirigé vers la page HomeAdminPage. Sinon, il est dirigé vers la page HomeUtilisateurPage.



FIGURE 4.13 – Interface d'accueil

b) Espace page d'accueil d'admin et utilisateur

La figure 4.14 ci-dessous montre l'interface de la page d'accueil pour l'administrateur ainsi que celle pour l'utilisateur. La page d'accueil de l'administrateur présente plusieurs boutons : un pour revoir les notifications, un pour envoyer des messages à tous les utilisateurs ou à un seul, un pour visualiser la température de refroidissement, un pour consulter les données de leur compte, et une courbe affichant les variations de température échouées au fil du temps pour chaque jour. De plus, un bouton hamburger permet à l'administrateur de gérer les utilisateurs. En comparaison, la page d'accueil de l'utilisateur offre des fonctionnalités similaires, telles

que la réception des notifications, l'envoi de messages, la visualisation de la température de refroidissement, et la consultation des données de leur compte, ainsi que la courbe des variations de température. Cependant, l'interface utilisateur ne comporte pas le bouton de gestion des utilisateurs, restreignant ainsi cette fonctionnalité uniquement aux administrateurs.

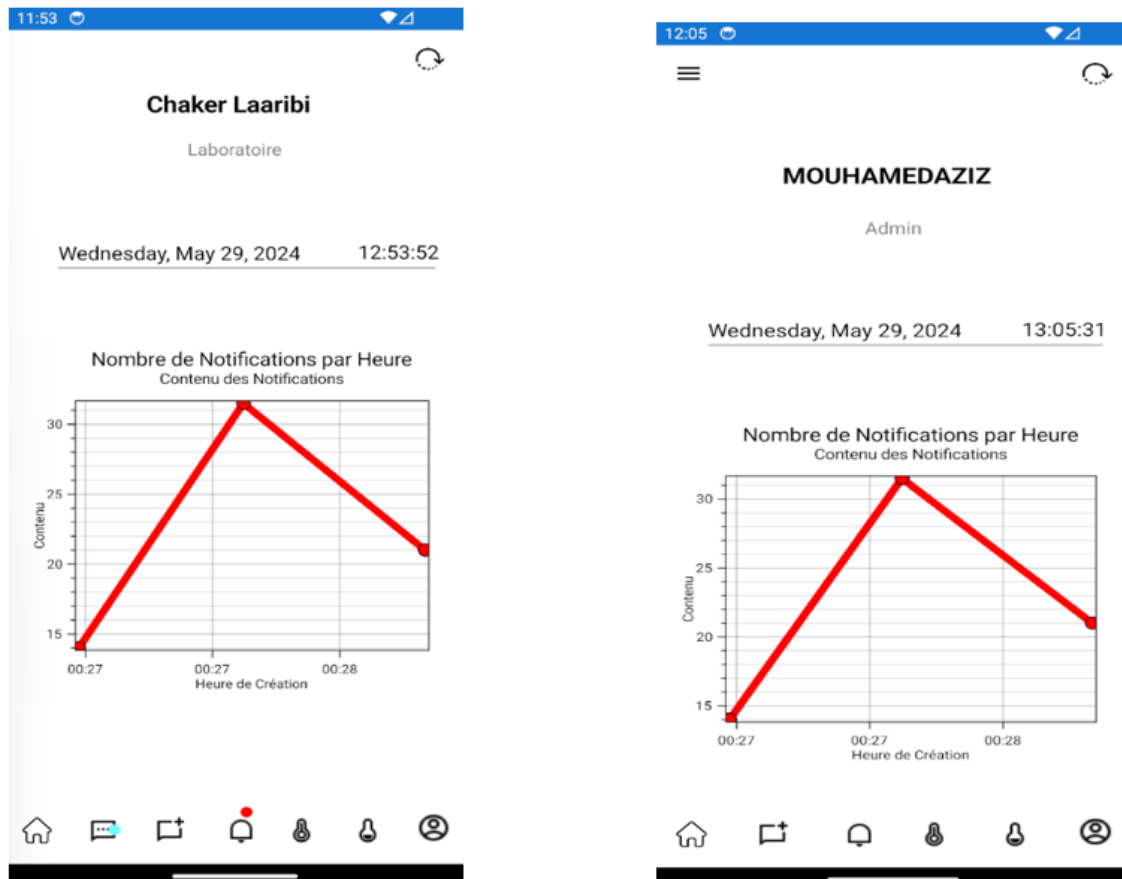


FIGURE 4.14 – Interface de la page d'accueil pour l'administrateur et pour l'utilisateur

c) Espace notification

La figure 4.15 ci-dessous montrent La page de notification affiche toutes les notifications pour l'utilisateur authentifié. Chaque notification comporte un indicateur : un bouton rouge indique que la notification n'a pas été lue, tandis que l'absence de ce bouton signifie que la notification a déjà été lue. Pour lire une notification, l'utilisateur clique dessus et navigue vers une page détaillée, la "DetailNotificationPage". Cette page affiche les détails de la notification

et contient deux boutons : un pour marquer la notification comme lue, et un autre pour répondre (reply) en envoyant un message à la personne responsable de l'incident ou de la maintenance.



FIGURE 4.15 – Interface notification

d) Interface de messagerie de notre application mobile

La figure 4.16 ci-dessous présente L'interface "CreateMessage" permet de récupérer la liste de tous les utilisateurs et offre une fonctionnalité de filtrage selon leur rôle ou poste. Lorsque l'utilisateur clique sur un nom, il est redirigé vers une autre page où il peut écrire et envoyer un message. De plus, cette interface permet à l'utilisateur de voir toutes les discussions entre les utilisateurs, incluant les messages envoyés et reçus. Un point bleu indique que le dernier message a été envoyé par l'autre utilisateur, tandis que l'absence de ce point bleu signifie que le dernier message a été envoyé par l'utilisateur authentifié.

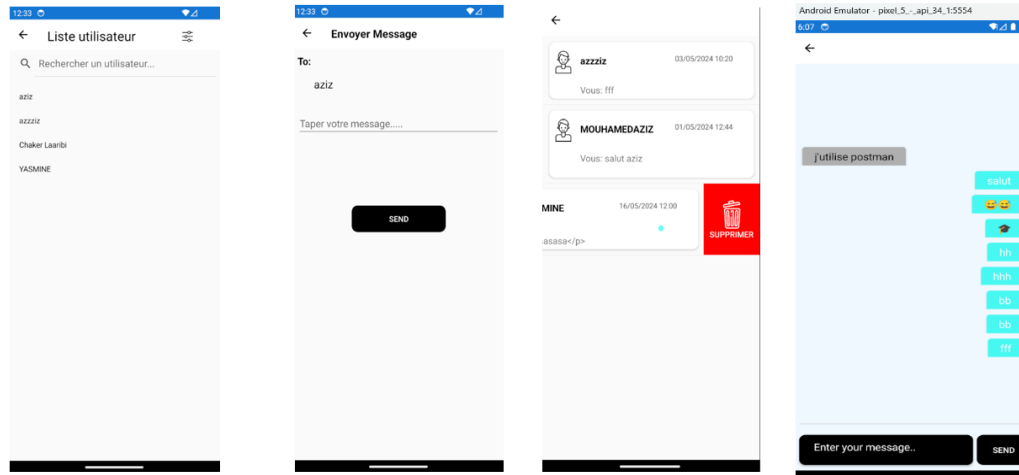


FIGURE 4.16 – interface interface de messagerie

4.3 Conclusion

En ce dernier chapitre, nous avons concentré nos efforts sur la configuration de la carte ESP32 Devkit V1 pour mesurer la température de la boîte de tomates du produit fini. En utilisant le capteur Waterproof DS18B2 et le langage de programmation Arduino, pour lire les valeurs de température et les transmettre via PocketBase. De plus, nous avons présenté les interfaces de notre application mobile, offrant une visualisation pratique des données recueillies.

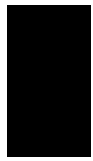
CONCLUSION ET PERSPECTIVES

Ce rapport décrit le développement d'un Système Intelligent de Surveillance et de Gestion Thermique basé sur un système embarqué. J'ai exploré les différentes étapes de conception, de développement et de mise en œuvre, en mettant l'accent sur les aspects matériels, logiciels et l'intégration avec PocketBase.

Le système développé utilise un capteur **Waterproof DS18B2**, une carte **ESP32 Devkit1** et communique avec PocketBase pour surveiller la température de l'eau destinée au refroidissement des boîtes de tomates, tout en envoyant les données à une application mobile. De plus, l'application mobile peut envoyer des notifications en cas d'échec de la température ou en cas d'augmentation ou de diminution de la température au-delà d'une valeur prédéfinie.

Ce projet a été une expérience enrichissante, me permettant d'acquérir des connaissances approfondies en matière de conception de systèmes embarqués, de programmation Arduino et d'intégration avec PocketBase. J'ai pu comprendre l'importance de la collaboration entre les composants matériels et logiciels pour créer un système fonctionnel et fiable.

En conclusion, ce rapport met en évidence les compétences acquises dans le domaine des systèmes embarqués et souligne les possibilités offertes par cette technologie pour résoudre des problèmes du quotidien. Il ouvre également des perspectives intéressantes pour de futures améliorations, telles que l'optimisation de l'interface utilisateur et l'intégration avec d'autres fonctionnalités. Ce projet a été une étape importante dans mon parcours d'apprentissage et m'a inspiré pour explorer davantage ce domaine passionnant.



NETOGRAPHIE

- [Net 1] <http://www.xalbox.com>
- [Net 2] <https://www.scapcb.com/>
- [Net 3] <https://www.wh-observer.de/Account/LogOn?ReturnUrl/>
- [Net 4] <https://www.kooklin.fr/capteur-temperature-automatique/>
- [Net 5] <https://dusuniot.com/fr/blog/iot-temperature-monitoring-in-energy-industry/>
- [Net 6] <https://espacerm.com/webgen/esp32intro/>
- [Net 7] <https://2btrading.tn/capteurs/1494-capteur-temperature-waterproof-ds18b20.html>
- [Net 8] <https://visualstudio.microsoft.com/fr/>
- [Net 9] <https://www.arduino.cc/>
- [Net 10] <https://kinsta.com/fr/base-de-connaissances/base-de-connaissances-github/>
- [Net 11] <https://www.overleaf.com/>
- [Net 12] <https://staruml.fr/download.it>
- [Net 13] <https://pocketbase.io/docs/>
- [Net 14] <https://www.postman.com/company/about-postman/>
- [Net 15] <https://stackoverflow.co>