la tête dans la toile | epsi l'école d'ingénierie informatique

RAPPORT D'ACTIVITÉ

- 2024 -

tb REPRODUCTEUR D'ART

SAISON 1

Ecrit par
Alban MARY

TABLE DES MATIÈRES

REMERCIEMENTS

Je souhaite exprimer ma profonde reconnaissance à toutes les personnes qui ont contribué à la réussite de mon stage et à l'élaboration de ce rapport.

Tout d'abord, je tiens à remercier chaleureusement mon maître de stage, Alan Ferronnière, fondateur de La Tête dans la Toile, pour son accueil chaleureux, sa confiance et le partage de son expertise tout au long de ce stage. C'est grâce à son encadrement que j'ai pu découvrir le fonctionnement de Wordpress et de plusieurs extensions populaires, des éléments clés dans la réalisation de mes missions.

Je désire également exprimer ma gratitude envers mes camarades de stage, Rémi Jegard, Moumine Kone ainsi que Timéo Danois, pour leur aide précieuse dans l'accomplissement de mes missions et pour leur sympathie à mon égard tout au long de ce stage. Leur soutien et leur camaraderie ont grandement contribué à rendre cette expérience professionnelle agréable et enrichissante.

1 - INTRODUCTION

Durant ma deuxième année d'études à l'EPSI, j'ai effectué un stage de 5 semaines dans l'entreprise « La Tête dans la Toile ». Cette société, fondée le 1er juillet 2007 par Alan Ferronnière, est experte dans les domaines du développement web et de la création d'outils métiers.

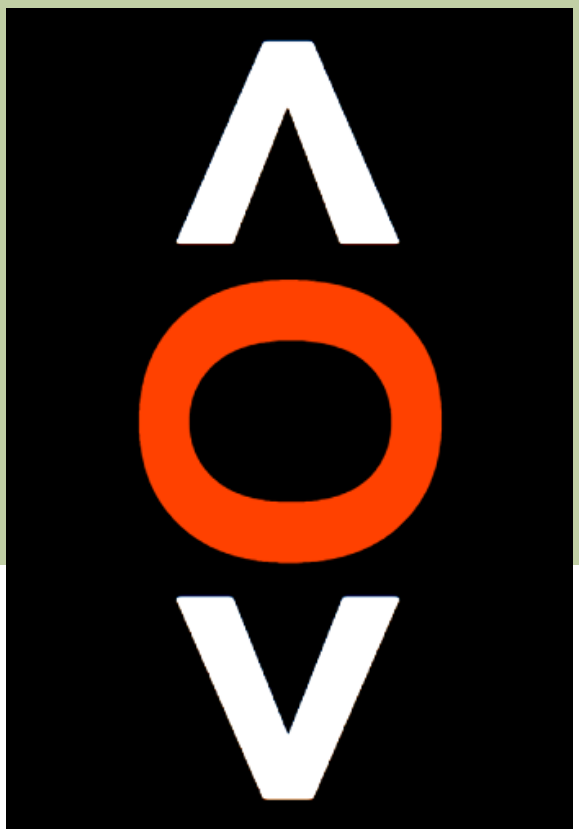
2 - Présentation de l'entreprise

2.1 – Domaine d'activité

Cependant, durant ce stage, j'ai été amené à travailler exclusivement sur le projet Troublanc.

Troublanc est une entreprise fondée en décembre 2022 par Alan et Yvan qui ont une expérience dans les domaines de l'art, de la communication et du web. L'objectif de l'entreprise est de promouvoir les artistes de l'Ouest de la France, notamment les étudiants en art, en proposant une plateforme innovante de diffusion de leurs œuvres.

Dans cette région, de nombreux artistes ont du mal à se faire connaître et à vivre de leur art, en raison du manque de lieux d'exposition adaptés à leurs besoins et à leurs aspirations. De plus, certaines formes d'art, telles que le chara design et les œuvres numériques, ont du mal à trouver



Logo de la tete dans la toile

leur place dans le paysage culturel régional.

Pour répondre à ces problématiques, Troublanc propose une plateforme de diffusion innovante qui permet aux artistes de toucher un public plus large. Les œuvres sont proposées sous forme de tirages d'art adaptés à tous les budgets et tous les goûts, allant de la carte postale au format 50x70 cm. Les reproductions sont réalisées en circuit court avec des imprimeurs locaux reconnus et éco-responsables.

Troublanc propose trois canaux de diffusion pour les œuvres d'art : le web, les réseaux sociaux et les bornes électroniques. Les bornes électroniques, réalisées en partenariat pédagogique avec l'EPSI de Nantes, sont installées dans des lieux de passage publics ou privés (espaces de co-working,

centres culturels, hôpitaux, salles d'exposition, cafés, centres commerciaux, etc.) et permettent au grand public de découvrir les œuvres d'art de manière ludique et interactive.

troublanc sélectionne des œuvres surprenantes, originales et variées, allant du dessin à la peinture, en passant par la photographie, le numérique ou le character design. Les artistes bénéficient d'une plus grande audience et sont rémunérés justement, avec une commission sur chaque reproduction vendue. troublanc ne demande pas l'exclusivité sur les reproductions des œuvres et la participation des artistes au projet est gratuite.

En plus de la plateforme de diffusion, troublanc organise également des expositions physiques plusieurs fois par an pour renforcer le lien entre les artistes et le public. Ces événements proposent des activités de workshop pour initier les visiteurs aux différentes techniques artistiques, ainsi que des moments festifs et conviviaux réunissant les artistes, le public, les partenaires et les médias.

En somme, troublanc a pour objectif de promouvoir les artistes de l'Ouest de la France en proposant une plateforme innovante de diffusion de leurs œuvres, tout en respectant les artistes et en utilisant les nouvelles technologies pour toucher un public plus large.

2.2 - Cadre de travail

Ce stage présente une particularité : mon maître de stage, M. Ferronnière, est également un intervenant à l'EPSI. Il a eu la gentillesse de proposer cette opportunité de stage à l'ensemble de la classe, et c'est avec enthousiasme que j'ai accepté cette offre, en compagnie de trois camarades, Rémi Jegard, Timéo Danois ainsi que Moumine Kone.
Mon objectif principal durant ce stage était de découvrir les méthodes de travail utilisées lors de la réalisation de projets en entreprise, tout en explorant le fonctionnement d'une entreprise comme celle-ci. En effet, j'aimerais devenir freelance.
Pendant ces 5 semaines, j'ai travaillé sur deux projets distincts : d'une part, le site de vente de reproduction d'œuvres d'art Troublanc (www.troublanc.com), et d'autre part, la réalisation d'une application de supervision pour les différentes bornes électroniques utilisées pour la diffusion des œuvres.

Durant mon stage l'entreprise permettait de travailler à distance, j'ai expérimenté le télétravail dans le cadre d'une démarche écologique visant à réduire les déplacements et à encourager un environnement de travail calme et propice à la concentration. J'ai pu travailler depuis mon établissement scolaire et mon domicile, ce qui m'a permis de développer des compétences en gestion du temps, planification autonome, communication à distance et utilisation d'outils technologiques adaptés (Discord, Azure Devops, etc.).

Cette expérience m'a fait apprécier les avantages du télétravail, tels que la réduction du temps de trajet, la possibilité de travailler dans un environnement confortable et familier, et la diminution des interruptions fréquentes dans un bureau traditionnel. Cependant, j'ai également rencontré des défis tels que le maintien d'une discipline personnelle, la communication efficace à distance et la séparation entre vie professionnelle et vie personnelle.

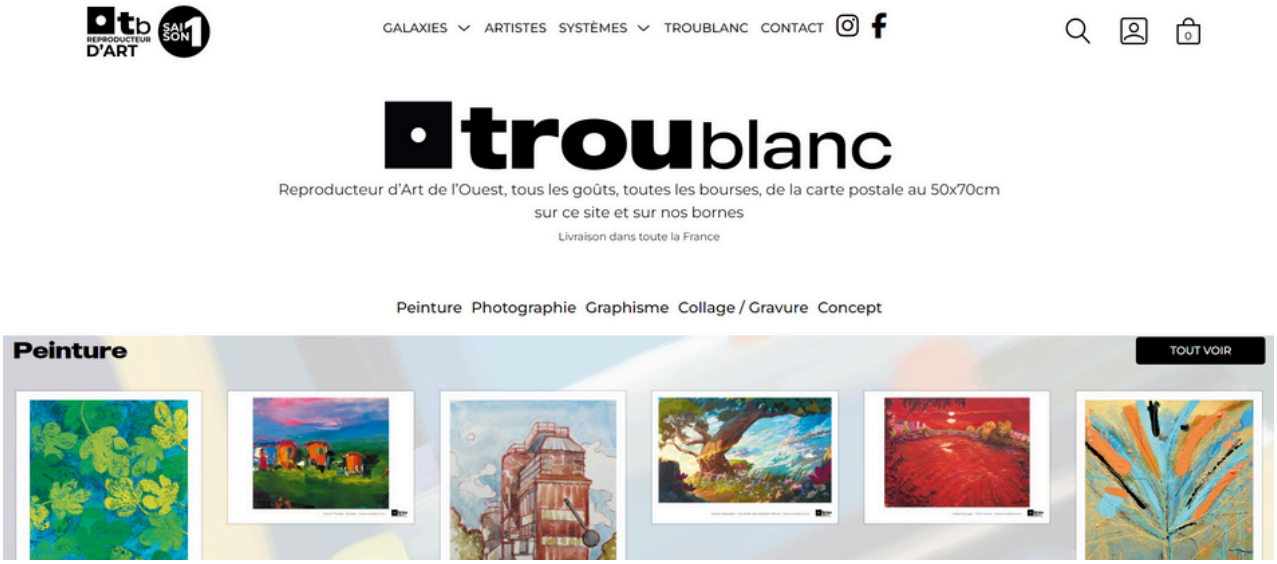
Cette expérience enrichissante m'a permis de développer des compétences clés telles que l'autonomie, la gestion du temps, l'adaptabilité et la résolution de problèmes à distance, tout en renforçant ma capacité à m'organiser efficacement et à travailler de manière indépendante.

3 - Mission principal : Mosaïque de produits

3.1 - Contexte

Le site Troublanc propose la vente de reproductions d'œuvres d'art réalisées par des artistes locaux. Les œuvres sont classées par catégories (graphisme, peinture, etc.) et une sélection aléatoire d'entre elles est affichée sur la page d'accueil, offrant ainsi une visibilité égale à tous les artistes présents sur le site. Il est également possible d'accéder à la page individuelle de chaque artiste pour découvrir l'ensemble de ses œuvres

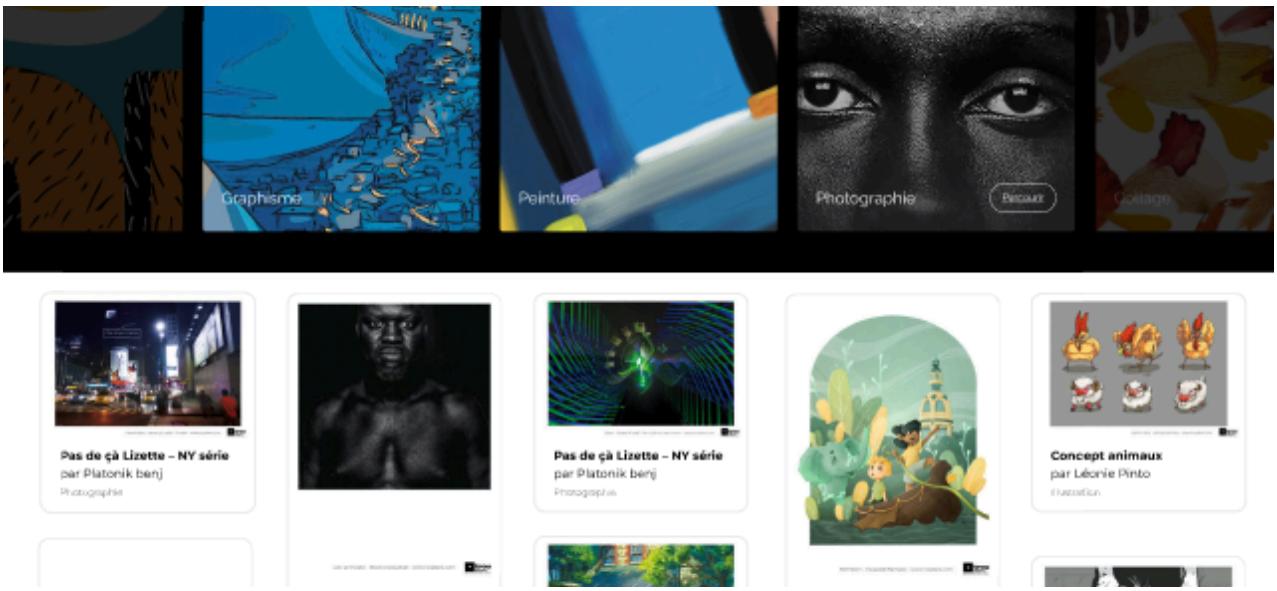
Malheureusement, avec l'augmentation du nombre d'œuvres et d'artistes sur le site, initialement conçu pour une quantité moindre, seule une petite portion d'entre eux était affichée sur la page d'accueil. Les pages dédiées aux différentes catégories rencontraient également ce problème, ainsi qu'un dysfonctionnement dans le code de sélection aléatoire, qui pouvait afficher plusieurs fois la même œuvre.



Site troublanc

Nous avons donc entrepris avec Moumine de redessiner le site pour afficher un plus grand nombre d'œuvres et améliorer l'expérience utilisateur. Pour cela, nous avons commencé par créer une maquette sur Figma, un outil de conception collaborative en ligne largement utilisé dans l'industrie du design pour créer des maquettes de sites web, d'applications et d'interfaces utilisateur. Nous l'avons ensuite soumise à l'approbation d'Yvan, le designer.

Pour améliorer la visibilité en ligne du site Troublanc et attirer un plus grand nombre de visiteurs, nous avons optimisé le design du site pour les moteurs de recherche. Cela comprenait l'ajout du nom de chaque œuvre, de l'artiste et des catégories, ce qui a permis d'améliorer le référencement naturel du site.



Maquette

Le site web Troublanc a choisi d'utiliser Wordpress comme plateforme pour sa boutique en ligne. Cette décision a été prise en raison de la facilité d'utilisation de Wordpress ainsi que de la possibilité de créer rapidement une boutique en ligne grâce à l'extension Woocommerce.

Cependant, malgré la facilité d'utilisation de Wordpress, il est important de noter que cette plateforme présente également des limites et des inconvénients. Tout d'abord, la personnalisation du site peut parfois être limitée, ce qui peut rendre difficile la création d'un site web unique et original. On peut ajouter que des problèmes de performance peuvent aussi impacter le site à cause des nombreux plugins et d'un thème peu performant, cela touche également le site troublanc.

Dans l'ensemble, Wordpress est une plateforme fiable et facile à utiliser pour les boutiques en ligne de petite à moyenne taille. Cependant, il est important de prendre en compte les limites et les inconvénients potentiels avant de prendre une décision quant à l'utilisation de cette plateforme pour un site web donné.

3.2 - Découpage de la mission

Après avoir défini les objectifs de notre projet, nous avons décidé de nous répartir les tâches à l'aide d'Azure DevOps, un outil de développement logiciel collaboratif de Microsoft. Azure DevOps permet de planifier, suivre et discuter du travail avec l'équipe, de coder avec des dépôts Git hébergés dans le cloud ou des repos, et de livrer en continu avec des pipelines CI/CD.

Git est un système de contrôle de version décentralisé qui permet de gérer les différentes versions d'un projet logiciel. Il facilite la collaboration entre les membres d'une équipe en leur permettant de travailler simultanément sur un même projet, de fusionner leurs modifications et de résoudre les conflits éventuels. Git suit l'historique des modifications apportées au code source, ce qui permet de retracer l'évolution du projet et de revenir à une version antérieure si nécessaire.

Pour organiser notre travail et suivre l'avancement du projet, nous avons utilisé un tableau Kanban. Le tableau Kanban est un outil visuel qui permet de suivre l'avancement des tâches à travers différentes étapes de développement. Il se compose de colonnes représentant les différentes étapes du processus de développement, telles que "À faire", "En cours" et "Terminé". Les tâches sont représentées par des cartes que l'on déplace d'une colonne à l'autre en fonction de leur état d'avancement. Cette méthode permet de visualiser facilement l'état d'avancement du projet et d'identifier rapidement les éventuels goulots d'étranglement.

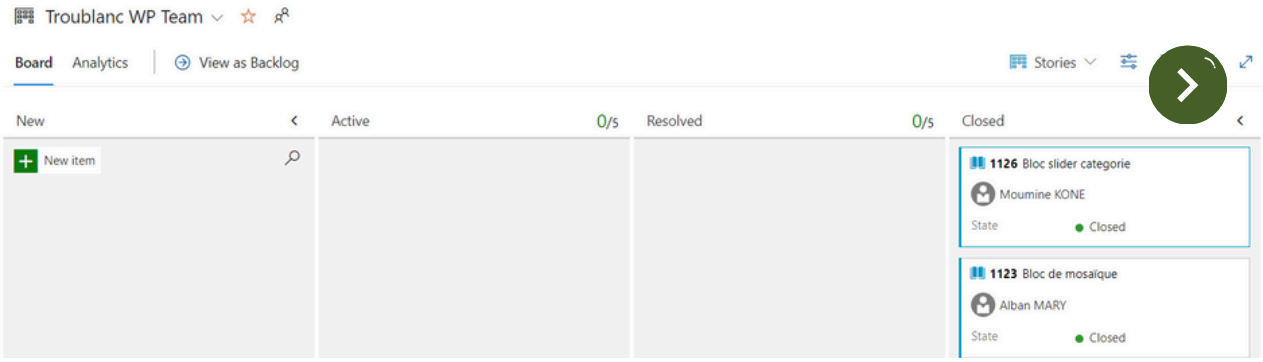


Tableau de Kanban

TABLEAU DE KANBAN

Comme vous pouvez le voir, notre tableau Kanban était divisé en plusieurs colonnes, chacune représentant une étape du processus de développement. Les colonnes étaient les suivantes :

- New : cette colonne contient toutes les tâches que nous avons identifiées comme étant nécessaires pour le projet, mais que nous n'avons pas encore commencé à travailler.
- Active : cette colonne contient les tâches que nous avons sélectionnées pour la prochaine itération et sur lesquelles nous travaillons actuellement.
- Resolved : cette colonne contient les tâches que nous avons terminées et qui sont prêtes à être testées.
- Closed : cette colonne contient les tâches qui ont été testées et approuvées.

Chaque tâche était représentée par une carte sur le tableau Kanban. Les cartes contenaient des informations telles que le titre de la tâche.

Dans notre cas, Moumine était chargé du bloc catégorie, tandis que je devais m'occuper du bloc mosaïque qui permet d'afficher les différentes œuvres tout ayant la possibilité de les trier (par nom d'artiste, nom d'œuvre ou aléatoirement). Ce bloc se trouve également sur les pages des catégories et des artistes.

Grâce à Azure DevOps, nous avons pu suivre l'avancement de notre travail respectif et collaborer efficacement.

3.3 - Mosaïque de produits

Pour créer l'effet de maçonnerie/mosaïque sur le site Troublanc, j'ai commencé par chercher des exemples inspirants sur Codepen et d'autres sites de design. J'ai finalement trouvé un modèle qui me convenait, celui de Pinterest.

Cependant, lorsque j'ai essayé de recréer cet effet avec des données hardcodées en utilisant des bibliothèques existantes, je n'ai pas été convaincu par les résultats. J'ai donc décidé de créer mon propre code en utilisant JavaScript.

Pour créer l'effet de maçonnerie/mosaïque sur le site Troublanc, j'ai utilisé JavaScript pour ajuster automatiquement la taille des éléments afin qu'ils s'adaptent à la grille. Lorsque la page est chargée ou que la fenêtre est redimensionnée, le code calcule la hauteur de chaque élément et ajuste dynamiquement sa taille pour qu'il s'intègre parfaitement dans la grille.

De plus, la fonction imagesLoaded est utilisée pour détecter que toutes les images ont été chargées dans les éléments de la grille, puis ajuste à nouveau la taille de chaque élément pour garantir une disposition cohérente. Cela permet de créer un effet de maçonnerie/mosaïque dynamique et responsive qui s'adapte à la taille de l'écran et à la hauteur variable des éléments du grid.

Pour répondre aux exigences du projet, j'ai étendu mon code initial en ajoutant des fonctionnalités telles que le tri et un bouton "Afficher plus". Pour ce faire, j'ai opté pour l'utilisation d'Advanced Custom Fields (ACF), un outil recommandé et utilisé par Alan. ACF est un plugin WordPress qui facilite la création de champs personnalisés dans l'interface d'administration du site.

Grâce à ACF, j'ai créé des champs personnalisés pour le choix de tri et le nombre d'œuvres à afficher sur la page d'accueil et les pages de catégories. Ces champs sont intégrés au bloc mosaïque qui peut être ajouté n'importe où sur le site, permettant de les modifier sans toucher au code.

Cette approche a rendu la modification des paramètres de tri et de pagination plus accessible et intuitive pour les administrateurs du site, améliorant ainsi l'expérience utilisateur globale. De plus, cela a contribué à réduire le temps de chargement initial des pages, ce qui est crucial pour le référencement et l'expérience utilisateur.

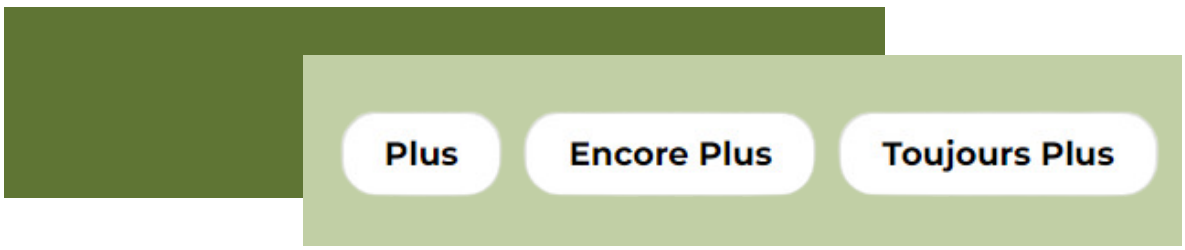
Bloc ACF

J'ai dû considérer l'aléatoire en premier lieu, car cela modifie complètement la manière dont nous approchons la présentation des œuvres. Pour cela, j'ai mis en place une méthode qui génère une liste contenant tous les ids des œuvres disponibles. Ensuite, à chaque chargement de la page ou à chaque fois que l'utilisateur clique sur le bouton "Afficher plus", cette liste est utilisée pour récupérer un autre ensemble d'œuvres.

En pratique, cela signifie que nous pouvons présenter les œuvres dans un ordre aléatoire, évitant ainsi une impression de répétitivité ou de prévisibilité pour l'utilisateur. Cela contribue à maintenir un intérêt constant et à encourager l'exploration continue des œuvres proposées.

Suite à cette mise en place, intégrer le choix de tri de l'utilisateur était une tâche relativement simple. Il suffit de trier la liste d'ids en fonction de ses préférences de tri. En agissant ainsi, nous pouvions garantir que les œuvres étaient présentées selon l'ordre spécifié par l'utilisateur, offrant ainsi une expérience personnalisée et répondant à ses attentes.

Suite aux retours d'Yvan et Alan sur le design, il a été suggéré d'améliorer l'expérience utilisateur en augmentant le nombre d'œuvres affichées. Pour cela, nous avons décidé de donner le contrôle à l'utilisateur en lui permettant de choisir le nombre d'images qu'il souhaite afficher par ligne. Trois boutons ont été ajoutés à cet effet, permettant d'afficher 6, 8 ou 12 images par ligne sur un écran standard de 1920x1080 pixels. L'option sélectionnée est sauvegardée dans le navigateur de l'utilisateur avec un cookie pour qu'il retrouve le même affichage lorsqu'il revient sur le site.



Les 3 boutons

Un autre point soulevé était le fait de devoir cliquer sur le bouton "Voir Plus" pour afficher plus d'œuvres. Pour améliorer cela, Alan a suggéré de remplacer ce bouton par un scroll infini. Le scroll infini est une technique de chargement continu de contenu lorsque l'utilisateur fait défiler la page vers le bas, évitant ainsi le besoin de cliquer sur un bouton pour charger plus de contenu.

Pour mettre en œuvre cette fonctionnalité, j'ai utilisé JavaScript pour détecter lorsque l'utilisateur fait défiler la page vers le bas jusqu'à un certain point, puis charge dynamiquement plus d'œuvres à afficher. Cela a amélioré l'expérience utilisateur en rendant le processus de chargement plus fluide et plus rapide.

Un autre point soulevé était le fait de devoir cliquer sur le bouton "Voir Plus" pour afficher plus d'œuvres. Pour améliorer cela, Alan a suggéré de remplacer ce bouton par un scroll infini. Le scroll infini est une technique de chargement continu de contenu lorsque l'utilisateur fait défiler la page vers le bas, évitant ainsi le besoin de cliquer sur un bouton pour charger plus de contenu.

Pour mettre en œuvre cette fonctionnalité, j'ai utilisé JavaScript pour détecter lorsque l'utilisateur fait défiler la page vers le bas jusqu'à un certain point, puis charge dynamiquement plus d'œuvres à afficher. Cela a amélioré l'expérience utilisateur en rendant le processus de chargement plus fluide et plus rapide.

Un autre point soulevé lors des retours sur le design concernait la taille des images des œuvres, qui pouvaient sembler trop petites pour certains utilisateurs. Pour améliorer cette expérience, j'ai ajouté une fonctionnalité de prévisualisation de l'image. Désormais, lorsque l'utilisateur survole une image avec sa souris, une version agrandie de l'image s'affiche, lui permettant de voir les détails de l'œuvre de manière plus claire.

Prévisualisation de l'image

3.4 - Déploiement sur un environnement de test

Pour travailler sur nos tâches respectives, Moumine et moi avions chacun créé une branche Git distincte. Moumine a travaillé sur la branche "categorie-block" tandis que j'ai travaillé sur la branche "mosaic-block". Nous avons utilisé Azure DevOps pour suivre l'avancement de nos tâches et collaborer efficacement.

Merge de la branche mosaic-block dans blocks

Lorsque nous avons terminé nos tâches respectives, nous avons fusionné nos modifications dans la branche de blocks créer à cet effet. Lors de la fusion de nos branches respectives, nous avons rencontré un conflit Git. Pour résoudre ce conflit, j'ai utilisé l'interface de ligne de commande Git. Tout d'abord, j'ai utilisé la commande "git merge" pour fusionner ma branche avec la branche blocks, ce qui a entraîné un message d'erreur indiquant que des conflits avaient été détectés. Ensuite, j'ai utilisé la commande "git status" pour voir quels fichiers étaient en conflit.

J'ai ouvert ces fichiers dans un éditeur de texte pour résoudre manuellement les conflits. J'ai cherché les marqueurs de conflit Git, qui indiquent les sections de code en conflit, puis j'ai décidé quelles modifications conserver et quelles modifications supprimer. Une fois les conflits résolus, j'ai enregistré les fichiers mis à jour.

Après avoir fusionné nos modifications dans la branche blocks, nous avons déployé le site web dans un environnement de test. Dans cet environnement, nous avons vérifié que le site web fonctionnait correctement et que les modifications apportées par chacun étaient intégrées correctement. Nous avons également effectué des tests manuels pour nous assurer que toutes les fonctionnalités du site web fonctionnaient comme prévu.

Au cours de cette phase de test, nous avons identifié et résolu deux problèmes :
Le premier problème concernait le z-index de la barre de navigation, qui était trop bas et faisait apparaître la prévisualisation de l'image au-dessus de la barre de navigation lorsque l'utilisateur survolait une image. Nous avons résolu ce problème en augmentant le z-index de la barre de navigation à l'aide de la propriété CSS "z-index" et en vérifiant que le z-index de tous les autres éléments de la page web était correctement défini.

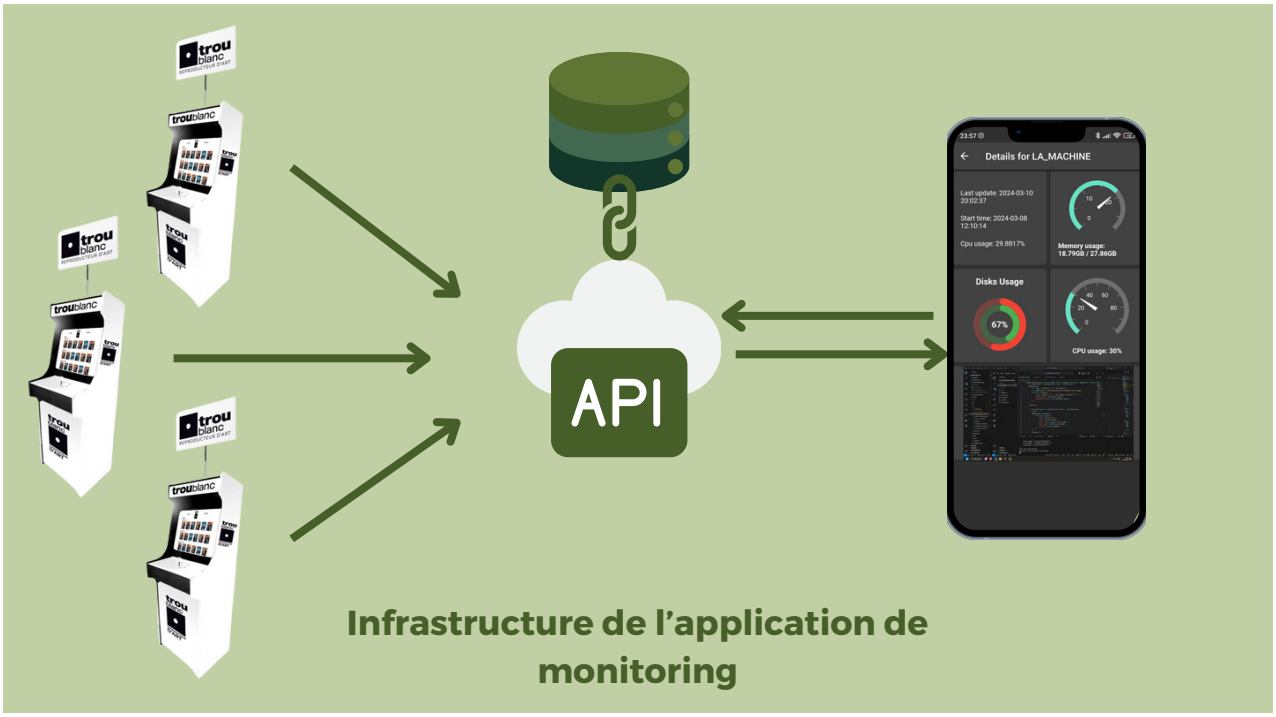
Le deuxième problème concernait une image qui n'était pas commune à toutes les pages du site web et qui n'était pas chargée correctement sur certaines pages. Nous avons découvert que le chemin d'accès à l'image était incorrect dans certains fichiers. Nous avons résolu ce problème en corrigeant le chemin d'accès à l'image dans tous les fichiers concernés et en vérifiant que l'image était correctement chargée sur toutes les pages du site web.

4 - Mission Secondaire : Application de Monitoring

4.1 - Contexte

Comme mentionné précédemment, j'ai été chargé de deux missions distinctes pendant mon stage. La seconde consistait à élaborer et mettre en place une application de surveillance visant à contrôler le bon état de fonctionnement des bornes électroniques utilisées pour la diffusion des œuvres d'art. Ces bornes, conçues pour diffuser des œuvres numériques dans des lieux publics tels que les cafés, les hôtels ou les musées, sont équipées d'écrans haute définition affichant le site web dédié aux œuvres. Elles sont assemblées à la main et disposent d'un joystick ainsi que de boutons pour faciliter la navigation de l'utilisateur sur le site.

L'intérêt d'une telle application est évident, étant donné que les bornes électroniques ont tendance à se figer fréquemment. Un récent incident survenu dans un café, où l'une des bornes a perdu sa connexion Wi-Fi sans être détectée pendant une semaine entière, met en lumière l'importance de cette surveillance. Cette application doit donc non seulement s'assurer du bon fonctionnement des bornes, mais aussi détecter toute erreur en prenant régulièrement des captures d'écran. Garantir le bon état de fonctionnement des bornes est essentiel pour offrir une expérience utilisateur optimale et éviter les interruptions de service.



Infrastructure de l'application de monitoring

4.2 - Module

Pendant ma deuxième partie de stage, j'ai eu l'opportunité de plonger dans le développement d'un module en Rust destiné à nos bornes électroniques. Ce module avait pour mission de récupérer les données relatives à l'état de la borne et de les transmettre à une API. Pour moi, c'était une première expérience avec Rust, et elle s'est avérée très formatrice. Malheureusement je ne vais pas pouvoir expliquer toutes les aspéritées de Rust car j'en aurais pour 30 pages minimum. Cependant j'aimerais l'expliquer rapidement car il a beaucoup d'avantages qui ont d'ailleurs été remarqués récemment.

Rust offre une sécurité accrue grâce à son système de propriétés et à son système de gestion de la mémoire sans garbage collector, conçu dès sa création (pour l'anecdote, son créateur a conçu ce langage après qu'un ascenseur soit tombé en panne à cause d'une erreur de mémoire). Ses performances sont comparables à celles du C ou du C++.

Vous retrouverez le code pour le module à cette adresse (https://github.com/Kidoly/monitorflowmodule/) ou à l'annexe n°1.

Ayant peu de connaissances préalables, j'ai commencé par me former et faire quelques tests. J'ai ensuite appris à utiliser les packages avec Cargo, qui diffère de ce que l'on trouve avec mon langage préféré Python.

J'ai débuté le développement du module par l'écriture d'une fonction principale qui récupère et affiche les informations système obtenues à l'aide du package Sysinfo. Cette fonction a été encapsulée dans une boucle pour permettre l'affichage des données à chaque itération, avec un court délai entre chaque.

Par la suite, j'ai amélioré la fonction pour récupérer uniquement les informations pertinentes et les rendre plus lisibles. J'ai créé une fonction pour convertir ces données en format JSON, couramment utilisé pour transmettre des données sur Internet.

L'étape suivante a été d'intégrer l'envoi de ces données à une API de test via une requête HTTP "POST". Bien qu'il aurait été plus sécurisé que l'API récupère d'elle-même les informations, les bornes pouvant se trouver n'importe où, modifier la configuration NAT de chaque box internet n'était pas une option viable.

Un défi supplémentaire a été de mettre en œuvre la capture d'écran. J'ai utilisé le package Xcap pour cela. Toutefois, pour faciliter son envoi via la requête POST, j'ai dû convertir l'image capturée en format base64.

Enfin, j'ai facilité la configuration du programme à partir d'un fichier ".env", contenant les paramètres nécessaires tels que l'adresse de l'API, la clé API, et l'intervalle entre chaque envoi de données.

J'ai créé une documentation pour Alan afin de simplifier le déploiement du module sur les bornes, en incluant des instructions d'installation ainsi que des étapes pour configurer son démarrage automatique à chaque redémarrage de la machine.

Un problème persistant que j'expérimente concerne la capture d'écran sur les distributions Linux telles que Debian et Ubuntu. Le module utilisé pour prendre des captures d'écran est nommé gnome-screenshot, et le package xcap dépend de ce module. Cependant, un inconvénient majeur se présente : les développeurs de ce module ont ajouté un flash lors de la prise de captures d'écran, sans option pour le désactiver. Malgré les nombreuses plaintes exprimées sur les forums spécialisés, aucune action n'a été entreprise pour résoudre ce problème. La seule solution actuelle consiste à télécharger le module, le modifier manuellement, puis le compiler individuellement sur chaque machine concernée.

Dans l'ensemble, j'ai trouvé que le développement de ce module était une expérience intéressante et enrichissante. Cela m'a permis d'en apprendre davantage sur Rust et sur le développement de modules pour les bornes électroniques. J'ai également apprécié de travailler avec l'API pour envoyer les informations collectées par le module.

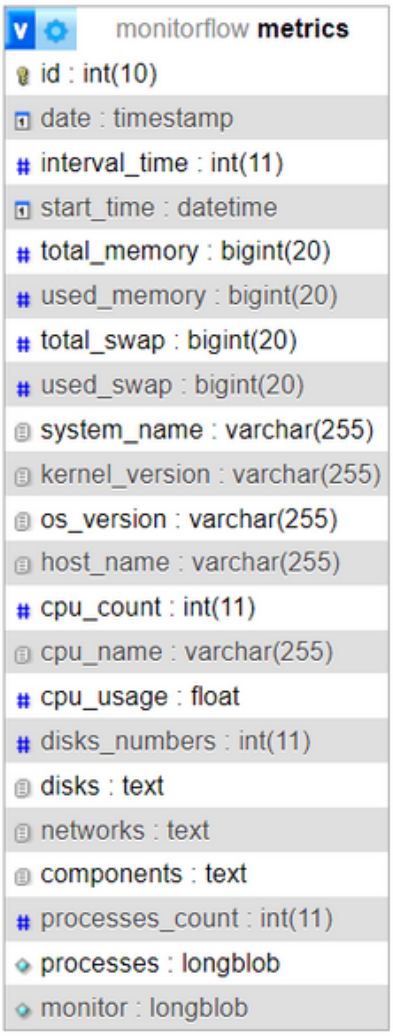
4.3 - API

Le module que j'ai développé est capable d'envoyer des informations. Cependant, il est fastidieux et peu pratique de demander à chaque module d'envoyer les données à une nouvelle adresse IP chaque fois qu'une personne souhaite utiliser l'application mobile pour surveiller les bornes. Cette méthode ne fonctionne plus dès lors que l'on quitte le réseau ou que l'adresse IP privée change. De plus, cela engendrerait de nombreux envois inutiles et nécessiterait de modifier les paramètres NAT de chaque box internet, ce qui n'est pas du tout viable.

Alors, comment les services multiplateformes permettent-ils à une application mobile de communiquer avec un navigateur web ou même avec un réfrigérateur ? Ils utilisent une API. Une API est un intermédiaire qui reçoit des données standardisées et les stocke généralement. Lorsqu'une application a besoin de données, elle ne les demande pas directement à la source, mais à l'API.

En utilisant une API, les applications peuvent communiquer entre elles de manière efficace et sécurisée, sans se soucier des détails d'implémentation spécifiques à chaque plateforme ou appareil. Cela permet également de centraliser la gestion des données et de simplifier le processus de communication entre les différentes applications et appareils.

J'ai donc choisi d'adopter cette méthode, qui présente de nombreux avantages et est relativement simple à mettre en œuvre avec une technologie que je maîtrise, à savoir PHP. Toutefois, il est essentiel de pouvoir stocker toutes ces données. À cette fin, rien de mieux qu'une base de données. Au cours du développement, j'ai apporté quelques modifications, mais la structure principale est restée la même : une table unique nommée "metrics", qui stocke chaque requête envoyée par les bornes.



J'ai choisi de stocker l'image directement dans la base de données, ce qui n'est pas une pratique recommandée. Il aurait été préférable de la stocker dans un dossier séparé. Cependant, cette méthode m'a permis de gagner du temps et l'application n'étant pas soumise à de fortes contraintes, cela reste acceptable.

Il est nécessaire d'avoir une base de données car il faut pouvoir stocker les informations comme ça si une borne ne fonctionne plus on peut tout de même avoir un historique de ce qu'il s'est passé avant la panne.

L'API se divise en deux parties distinctes (annexes n°2 et N°3) : l'une chargée de récupérer les données et l'autre chargée de les transmettre en réponse à une requête. La première partie est relativement simple à décrire : elle récupère les données passées en paramètre POST, puis les stocke méthodiquement dans la base de données, en veillant à les placer correctement à leur emplacement approprié.

J'ai ensuite fait la partie de l'api que l'on peut appeler pour que l'application mobile puisse récupérer les informations et les afficher. Il suffit à cette partie de récupérer les paramètres passés dans le lien de call envoyées par l'application mobile, exemple: https://monitorflow.com/api/api_call.php?host_name=debian où debian correspond au nom de la machine sur lequel on demande des informations, et de faire une commande SQL spécifique à ce qui est demandé avant d'afficher la réponse.

J'ai développé deux requêtes distinctes : l'une sans paramètre, qui renvoie uniquement un JSON contenant tous les serveurs sans détails supplémentaires d'informations :

[{"host_name":"debian","memory":"0.4733","system_name":"Debian GNU\/Linux"},
{"host_name":"LA_MACHINE","memory":"0.5479","system_name":"Windows"}]

Et l'autre requête qui renvoie tous les détails de la borne spécifiée.

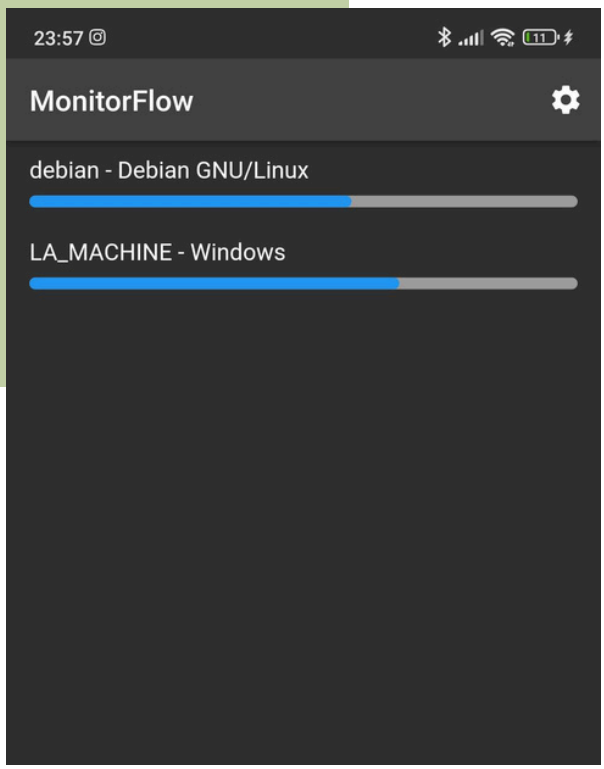
Pour renforcer la sécurité, éviter la saturation de la base de données et prévenir la divulgation de données sensibles, j'ai mis en place un mécanisme de clé d'API limitant l'accès aux seules entités autorisées à envoyer ou récupérer des données.

4.4 - L'application mobile

Toujours désireux de me former, j'ai poursuivi l'exploration de technologies inconnues. J'ai tenté de créer une application web avec Next.js, mais le manque de temps m'a vite rattrapée, avec seulement quelques jours devant moi. Malgré cela, cette expérience m'a permis d'acquérir des connaissances précieuses. En explorant Flutter Flow, un outil en ligne pour créer des applications Flutter sans coder, j'ai décidé de me lancer dans le développement d'une application mobile. Cependant, j'ai constaté que Flutter Flow est payant et qu'en fin de compte, il est plus lent de créer une application avec cet outil que de coder directement avec Flutter.

Flutter est un kit de développement d'applications mobiles créé par Google. Il permet de créer des applications natives pour iOS et Android à partir d'un seul code source, écrit en langage de programmation Dart. Flutter se démarque par sa capacité à offrir une expérience utilisateur fluide et réactive, grâce à son moteur de rendu graphique performant et sa bibliothèque de widgets personnalisables. En somme, Flutter est un outil pratique et efficace pour développer des applications mobiles de qualité supérieure en un temps record.

Flutter est cross plateform



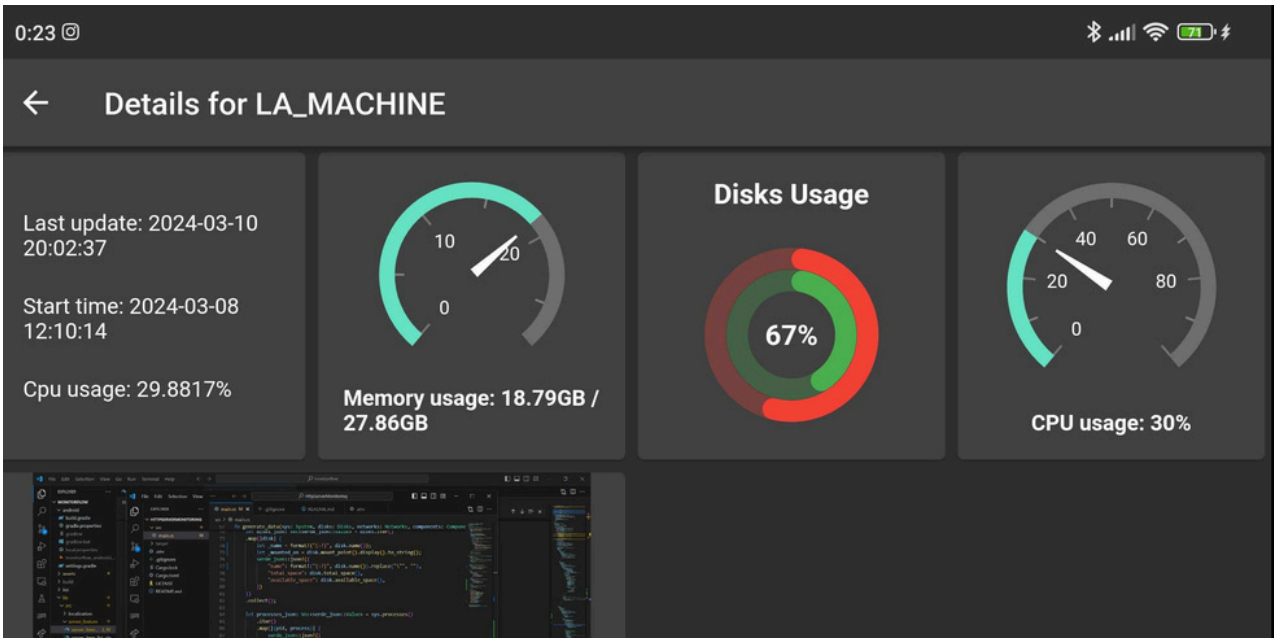
Page qui liste les bornes

Nous disposons d'une multitude d'informations sur les bornes grâce aux modules, mais il est crucial de les organiser de manière efficace. Pour ce faire, j'ai choisi une approche d'interface utilisateur extrêmement simple : une page affichant la liste de toutes les bornes , et lorsqu'une borne est sélectionnée, une autre page s'ouvre présentant en détail les informations spécifiques à cette borne.

La deuxième étape s'est révélée plus complexe que prévu, car je cherchais à créer une application réactive, adaptée aux ordinateurs, tablettes et téléphones (Flutter étant désormais capable de créer des applications pour PC).

Cette étape s'est avérée relativement simple grâce à Flutter, qui offre la possibilité d'initialiser un projet avec une liste d'éléments pré-faits. Cependant, j'ai dû adapter le code pour que les informations affichées sur cette page soient le résultat d'une requête vers l'API. Pour ajouter une touche de design, j'ai inclus une barre de progression sous le nom, représentant la mémoire RAM utilisée par la borne. Une amélioration potentielle consisterait à rendre cette représentation plus intuitive pour l'utilisateur, sans nécessiter d'explications écrites supplémentaires.

Pour permettre l'ouverture de la page de détails lorsqu'on clique sur l'une des bornes, il est nécessaire de transmettre le nom de la borne en tant que paramètre dans la route. Ainsi, sur la page de détails, il suffit de récupérer ce paramètre et de l'utiliser dans une nouvelle requête à l'API pour obtenir les informations nécessaires. Sur Flutter, une route est un moyen de naviguer entre différentes pages ou écrans dans une application. Elle permet de définir la structure et la logique de navigation de l'application, en spécifiant comment chaque écran est lié aux autres et comment les données sont transférées entre eux. En passant le nom de la borne comme paramètre dans la route, nous permettons à la page de détails de savoir quelle information spécifique elle doit afficher, facilitant ainsi la gestion et l'affichage des données pour chaque borne sélectionnée.



Page détails d'une borne

Je pense qu'il est crucial d'avoir une interface claire pour ces informations, car elles fournissent des données essentielles sur le fonctionnement des bornes. Par exemple, afficher l'heure et la date de la dernière information envoyée ainsi que celles du démarrage de la borne permet aux utilisateurs de comprendre la fraîcheur des données et l'état actuel de la borne. De plus, l'utilisation de jauges pour visualiser de manière concise et intuitive l'utilisation de la mémoire RAM et du CPU offre une expérience utilisateur plus intuitive et efficace.

J'ai opté pour un indicateur de progrès circulaire, similaire à celui utilisé sur les montres connectées, pour représenter les différents disques car c'est un design simple. Chaque disque est associé à un cercle indiquant sa capacité utilisée. Au centre, le pourcentage total utilisé sur la machine est affiché. Bien que cela puisse sembler moins pertinent pour les bornes pilotées par des Raspberry Pi, avec comme unique stockage une carte SD, j'ai envisagé une évolution future dans la conception de l'interface.

Le dernier élément que l'on retrouve est la capture d'écran, j'utilise une fonction spécifique pour décoder ces images à partir de leur format base64. L'image est affichée sur l'écran dans un petit rectangle similaire au design pour les jauges et à l'indicateur circulaire. L'utilisateur peut ouvrir l'image en plus grand en cliquant dessus, elle peut être alors zoomée pour voir les détails plus clairement. Cela améliore l'expérience utilisateur en offrant une façon pratique d'explorer les images, surtout lorsqu'elles contiennent des informations importantes.

4.5 - Et après ?

J'ai considéré plusieurs pistes d'amélioration et d'évolution pour cette application :

Premièrement, la mise en place d'une suppression automatique des données périmées lors de l'ajout de nouvelles informations dans la base de données serait bénéfique. Cette fonctionnalité empêcherait l'accumulation excessive de données en éliminant celles qui sont devenues obsolètes ou inutiles.

Je propose également d'intégrer un chiffrement des données transmises, étant donné que leur interception constitue une préoccupation majeure en raison de leur caractère sensible.

En outre, l'ajout de fonctionnalités dédiées à la capture et à l'analyse des échanges réseau des bornes serait avantageux. Cette démarche permettrait une meilleure compréhension des performances du réseau et faciliterait la détection d'éventuels problèmes ou goulets d'étranglement.

Une autre suggestion consiste à migrer l'API PHP vers un serveur Node.js utilisant WebSocket, afin de permettre une transmission des données en temps réel. Cette transition vers une architecture plus réactive offrirait des mises à jour en temps réel sur l'application mobile, améliorant ainsi son efficacité et sa réactivité.

Il serait également très intéressant d'intégrer un système de pipeline CI/CD ou une technologie similaire, qui permettrait de mettre à jour le module sur les bornes à chaque fois qu'une nouvelle est ajoutée.

D'autre part, je recommande de repenser l'interface utilisateur de l'application pour la rendre plus conviviale et intuitive. Cela pourrait inclure des améliorations visuelles, une organisation plus efficace des informations et l'ajout de fonctionnalités interactives pour optimiser l'expérience utilisateur.

Enfin, l'intégration de notifications dans l'application permettrait d'informer les utilisateurs des événements importants ou des mises à jour du système.

Comme évoqué précédemment, la création d'une application web avec Next.js pourrait être une approche intéressante. Cette solution offrirait un contrôle accru sur l'accès aux informations et supprimerait la nécessité de télécharger une application distincte.

5 - Évaluation des réalisations et des compétences mobilisées

5.1 - Analyse des résultats par rapport aux objectifs fixés

Au cours de ce stage, j'ai travaillé sur deux projets principaux : le site de vente de reproduction d'œuvres d'art Troublanc et la réalisation d'une application de supervision pour les différentes bornes électroniques utilisées pour la diffusion des œuvres. Les objectifs fixés pour ces projets ont été atteints.

Pour le site Troublanc, j'ai participé à la refonte du site en créant une maquette sur Figma et en optimisant le design pour les moteurs de recherche. J'ai également développé un bloc mosaïque permettant d'afficher les différentes œuvres tout en pouvant les trier. Ce bloc se trouve sur les pages des catégories et des artistes. L'objectif principal était d'améliorer l'expérience utilisateur et la visibilité en ligne du site, ce qui a été accompli avec succès. Malheureusement je n'ai pas pu rendre accessible le nouveau design du site aux commandes de navigation des bornes électroniques.

Pour l'application de supervision, j'ai développé un module en Rust destiné à récupérer les données relatives à l'état de la borne et à les transmettre à une API. J'ai également créé une API pour stocker ces données dans une base de données et développé une application mobile pour surveiller l'état des bornes. Les objectifs de ce projet étaient de garantir le bon état de fonctionnement des bornes et d'offrir une expérience utilisateur optimale, ce qui a été réalisé avec succès.

5.2 - Évaluation des compétences techniques acquises ou mobilisées

Au cours de ce stage, j'ai acquis et mobilisé plusieurs compétences techniques. Tout d'abord, j'ai développé mes compétences en développement web en travaillant sur le site Troublanc. J'ai utilisé des outils tels que Figma pour créer des maquettes, et j'ai optimisé le design du site pour les moteurs de recherche. J'ai également développé un bloc mosaïque en JavaScript, ce qui m'a permis de renforcer mes compétences dans ce langage.

Ensuite, j'ai développé mes compétences en développement d'applications en créant un module en Rust pour les bornes électroniques. Cette expérience m'a permis de découvrir un nouveau langage de programmation et de développer mes compétences en matière de gestion de la mémoire et de sécurité.

J'ai également acquis des compétences en développement d'API en créant une API pour stocker les données des bornes dans une base de données. J'ai utilisé PHP et MySQL pour développer cette API, ce qui m'a permis de renforcer mes compétences dans ces technologies.

Enfin, j'ai développé mes compétences en développement d'applications mobiles en créant une application pour surveiller l'état des bornes. J'ai utilisé Flutter pour développer cette application, ce qui m'a permis de découvrir ce framework et de développer mes compétences en matière de développement d'applications multiplateformes.

5.3 - Réflexion sur les compétences transversales développées

Au cours de ce stage, j'ai également développé plusieurs compétences transversales. Tout d'abord, j'ai renforcé mes compétences en gestion de projet en travaillant sur deux projets distincts et en utilisant des outils tels que Azure DevOps pour organiser mon travail et suivre l'avancement des tâches.

J'ai également développé mes compétences en communication en travaillant en étroite collaboration avec mon maître de stage et mes camarades de classe. J'ai appris à communiquer efficacement à distance en utilisant des outils tels que Discord et à travailler en équipe pour atteindre des objectifs communs.

Enfin, j'ai développé mes compétences en résolution de problèmes en rencontrant et en résolvant plusieurs défis techniques au cours de ce stage. J'ai appris à rechercher des solutions en ligne, à demander de l'aide lorsque cela était nécessaire et à persévérer jusqu'à ce que je trouve une solution.

6 - Conclusion

6.1 – Bilan du stage

Ce stage m'a permis de découvrir les méthodes de travail utilisées lors de la réalisation de projets en entreprise, tout en explorant le fonctionnement d'une entreprise spécialisée dans le développement web et la création d'outils métiers. J'ai travaillé sur deux projets distincts : le site de vente de reproduction d'œuvres d'art Troublanc et la réalisation d'une application de supervision pour les différentes bornes électroniques utilisées pour la diffusion des œuvres.

6.2 – Retour d'expérience personnel

Ce stage a été pour moi une expérience enrichissante, tant sur le plan professionnel que personnel. J'ai pu développer des compétences en gestion du temps, planification autonome, communication à distance et utilisation d'outils technologiques adaptés, grâce à la mise en place du télétravail. De plus, la réalisation de l'application de supervision m'a permis d'approfondir mes connaissances en développement d'applications sécurisées, en prenant en compte les enjeux de sécurité liés à la collecte et au stockage des données. En plus de cela, j'ai envie de continuer son développement sur mon temps personnel car c'est un outils qui avec des améliorations pourrait me permettre de mieux protéger des serveurs sous ma surveillance.

6.3 – Perspectives d'évolution et d'amélioration

Ce stage m'a permis d'explorer un domaine en dehors de ma volonté de projet professionnel initial. Bien que j'aie travaillé sur des projets différents de la cybersécurité, cette expérience a renforcé mon intérêt pour ce domaine. En effet, mon stage précédent chez Kereis, où j'ai eu l'opportunité de travailler sur des problématiques de cybersécurité, m'a donné un aperçu de l'importance de ce secteur et m'a motivé à poursuivre dans cette voie.

Mon intérêt pour la cybersécurité s'est donc confirmé durant ce stage, et je souhaite désormais poursuivre dans cette voie. Je compte approfondir mes connaissances en sécurité informatique, en suivant des formations spécialisées et en m'informant régulièrement sur les dernières avancées technologiques dans ce domaine. Je suis conscient que la cybersécurité est un domaine en constante évolution, et je suis prêt à m'investir pleinement pour relever les défis qui se présenteront à moi.

Dans le cadre de mon parcours professionnel, je souhaite travailler sur des projets liés à la sécurisation des systèmes d'information et à la protection des données. Je suis particulièrement intéressé par la sécurité des applications web et mobiles, ainsi que par la protection des infrastructures réseau. Mon objectif est de devenir un expert en cybersécurité, capable de proposer des solutions innovantes et adaptées aux besoins des entreprises et des organisations.

Enfin, je souhaite mettre à profit les compétences acquises durant ce stage pour apporter ma contribution à des projets open source dans le domaine de la cybersécurité. Je suis convaincu que l'échange et le partage des connaissances sont essentiels pour faire face aux défis de la sécurité informatique, et je suis déterminé à m'impliquer activement dans cette communauté.

7 - Annexes

7.1 - Annexe n°1

```rust
use std::{io::Cursor, thread::sleep, time::Duration};
use sysinfo::{System, Disks, Networks, Components};
use xcap::{Monitor, image::RgbaImage};
use std::env;
use dotenv::dotenv;

fn main() -> Result<(), Box<dyn std::error::Error>> {
    dotenv().ok(); // Load .env file

    let api_key = env::var("API_KEY").expect("API_KEY not set in .env file");
    let api_url = env::var("API_URL").expect("API_URL not set in .env file");
    let interval: u64 = env::var("INTERVAL")
        .expect("INTERVAL not set in .env file")
        .parse()
        .expect("INTERVAL must be a valid number");

    loop {
        let mut sys = System::new_all();
        let disks = Disks::new_with_refreshed_list();
        let networks = Networks::new_with_refreshed_list();
        let components = Components::new_with_refreshed_list();
        let monitors = Monitor::all()?;
        sys.refresh_all();

            if let Some(monitor) = monitors.first() {
            let data = generate_data(sys, disks, networks, components, monitor, &api_key, &interval);

            let req = ureq::request("POST", &api_url);
            println!("{:#?}", data);

            // Send a POST request
            let response =  req.set("Content-Type", "application/json")
                .send_string(&data.to_string());

            match response {
                Ok(response) => {
                    if response.status() == 200 {
                        println!("Data sent successfully");
                        let response_body = response.into_string().expect("Failed to read response body");
                        println!("Response: {}", response_body);
                    } else {
                        println!("Failed to send data: {}", response.status());
                        let error_body = response.into_string().expect("Failed to read response body");
                        println!("Error details: {}", error_body);
                    }
                },
                Err(e) => eprintln!("Failed to send data: {}", e),
            }
        } else {
            eprintln!("No monitors found");
        }
        // Wait for the next iteration
        sleep(Duration::from_secs(interval));
    }
}

```

7.1 - Annexe n°1

```rust
fn generate_data(sys: System, disks: Disks, networks: Networks, components: Components, monitor: &Monitor, api_key: &str, interval: &u64) -> serde_json::Value {
    let avg_cpu_usage = sys
        .cpus()
        .iter()
        .map(|cpu| cpu.cpu_usage())
        .sum::<f32>() / sys.cpus().len() as f32;

    let monitor_image = match monitor.capture_image().map(|image| to_base64(image)) {
        Ok(image) => image,
        Err(e) => {
            eprintln!("Failed to capture monitor image: {}", e);
            String::new()
        }
    };

    let disks_json: Vec<serde_json::Value> = disks.iter()
    .map(|disk| {
        let name = disk.name();
        let mounted_on = disk.mount_point().display().to_string();
        serde_json::json!({
            "file_system": disk.file_system(),
            "total_space": disk.total_space(),
            "available_space": disk.available_space(),
        })
    })
    .collect();

    let processes_json: Vec<serde_json::Value> = sys.processes()
        .iter()
        .map(|(pid, process)| {
            serde_json::json!({
                "pid": pid.as_u32(),
                "name": process.name(),
                "start_time": process.start_time(),
                "cpu_usage": process.cpu_usage(),
                "memory": process.memory(),
            })
        })
        .collect();

    serde_json::json!({
        "api_key": api_key,
        "interval_time" : interval,
        "start_time": System::boot_time(),
        "total_memory": sys.total_memory().to_string(),
        "used_memory": sys.used_memory().to_string(),
        "total_swap": sys.total_swap().to_string(),
        "used_swap": sys.used_swap().to_string(),
        "system_name": System::name(),
        "kernel_version": System::kernel_version(),
        "os_version": System::os_version(),
        "host_name": System::host_name(),
        "cpu_count": sys.cpus().len(),
        "cpu_name": sys.cpus()[0].brand(),
        "cpu_usage": avg_cpu_usage,
        "disks_numbers": disks.len(),
        "disks": disks_json,
        "networks": networks.iter().map(|network|{format!("{network:#?}")}).collect::<Vec<_>>(),
        "processes_count": sys.processes().len(),
        "processes": processes_json,
        "monitor": monitor_image,
    })
}

fn to_base64(image: RgbaImage) -> String {
    let mut c = Cursor::new(Vec::new());
    image.write_to(&mut c, image::ImageOutputFormat::Png).unwrap();
    base64::encode(c.into_inner())
}
```

7.2 - Annexe n°2

```
<?php
require_once 'config.php';

// Connect to the MySQL database
$host = DB_HOST; // or your database host
$db   = DB_NAME; // your database name
$user = DB_USER; // your database username
$pass = DB_PASS; // your database password
$charset = DB_CHARSET;

$dsn = "mysql:host=$host;dbname=$db;charset=$charset";
$options = [
    PDO::ATTR_ERRMODE            => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
    PDO::ATTR_EMULATE_PREPARES   => false,
];

try {
    $pdo = new PDO($dsn, $user, $pass, $options);
} catch (\PDOException $e) {
    throw new \PDOException($e->getMessage(), (int)$e->getCode());
}

// Read the JSON POST body
$json = file_get_contents('php://input');
$data = json_decode($json, true);

// Assuming you have a table called 'server_info' with appropriate columns
// You should adjust the table name and columns according to your database schema
$sql = "INSERT INTO metrics (interval_time, start_time, total_memory, used_memory,
total_swap, used_swap, system_name, kernel_version, os_version, host_name,
cpu_count, cpu_name, cpu_usage, disks_numbers, disks, networks, components,
processes_count, processes, monitor) VALUES (?,?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
$stmt = $pdo->prepare($sql);

// Replace 'your_api_key' with your actual API key
$api_key = API_KEY;

if ($data['api_key'] !== $api_key) {
    // If the API key is wrong, return an error message and stop the script
    http_response_code(403);
    echo "Forbidden: Wrong API key";
    exit;
}
```

7.2 - Annexe n°2

```
$start_time = date("Y-m-d H:i:s", $data['start_time']);

// Bind values from the JSON to the INSERT statement
$stmt->execute([
    $data['interval_time'],
    $start_time,
    $data['total_memory'],
    $data['used_memory'],
    $data['total_swap'],
    $data['used_swap'],
    $data['system_name'],
    $data['kernel_version'],
    $data['os_version'],
    $data['host_name'],
    $data['cpu_count'],
    $data['cpu_name'],
    $data['cpu_usage'],
    $data['disks_numbers'],
    json_encode($data['disks']), // Storing as JSON string
    json_encode($data['networks']), // Storing as JSON string
    json_encode($data['components']), // Storing as JSON string
    $data['processes_count'],
    json_encode($data['processes']), // Storing as JSON string
    json_encode($data['monitor']) // Storing as JSON string]
]);

echo "Data inserted successfully";
?>
```

7.3 - Annexe n°3

```
<?php
require_once 'config.php';

// http://albanmary.com/api/api_call.php?api_key=your_secret_api_key_here
$your_api_key = API_KEY;

// Check if an API key has been set in the request
if (!isset($_GET['api_key'])) {
    http_response_code(401); // Unauthorized
    echo json_encode(['error' => 'No API key provided']);
    exit; // Stop script execution
}

// Verify the API key
$request_api_key = $_GET['api_key'];
if ($request_api_key !== $your_api_key) {
    http_response_code(403); // Forbidden
    echo json_encode(['error' => 'Invalid API key']);
    exit; // Stop script execution
}

// Connect to the MySQL database
$host = DB_HOST; // or your database host
$db = DB_NAME; // your database name
$user = DB_USER; // your database username
$pass = DB_PASS; // your database password
$charset = DB_CHARSET;

$dsn = "mysql:host=$host;dbname=$db;charset=$charset";
$options = [
    PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
    PDO::ATTR_EMULATE_PREPARES => false,
];

try {
    $pdo = new PDO($dsn, $user, $pass, $options);
} catch (\PDOException $e) {
    throw new \PDOException($e->getMessage(), (int)$e->getCode());
}
```

7.3 - Annexe n°3

```
// Check if host_name is provided in the request
if (isset($_GET['host_name'])) {
 $host_name = $_GET['host_name'];

 // Query to fetch all information for the specified host_name
 $sql = "SELECT id, date, interval_time, start_time, total_memory, used_memory,
total_swap, used_swap, system_name, kernel_version, os_version, host_name,
cpu_count, cpu_name, cpu_usage, disks_numbers, disks, processes, processes_count,
monitor FROM metrics WHERE host_name = :host_name ORDER BY id DESC LIMIT 1";
 $stmt = $pdo->prepare($sql);
 $stmt->execute(['host_name' => $host_name]);
 $result = $stmt->fetch();

 if ($result) {
 header('Content-Type: application/json');
 echo json_encode($result);
 exit; // Stop script execution after outputting the result
 } else {
 http_response_code(404); // Not Found
 echo json_encode(['error' => 'Host not found']);
 exit; // Stop script execution
 }
}

// Original query to select all unique host_names with memory usage
$sql = "SELECT r1.host_name, (used_memory/total_memory) AS memory,
system_name FROM metrics AS r1 INNER JOIN (
 SELECT r2.host_name, MAX(id) AS max_id
 FROM metrics AS r2
 GROUP BY host_name
) AS latest_entries ON id = latest_entries.max_id;";

$stmt = $pdo->prepare($sql);
$stmt->execute();

$results = $stmt->fetchAll();

// Outputting the results as JSON
header('Content-Type: application/json');
echo json_encode($results);

?>
```