# REMMERCIEMENTS

**A**vant tout développement de ce rapport, nous rappelons que « une seule main ne peut attacher un fagot de bois » dit la sagesse africaine. Ainsi, il apparait opportun de commencer nos propos de remerciements à ceux-là qui de près ou de loin ont concouru au bon déroulement de notre formation académique. Gratitude tout d’abord à l’éternel Dieu.

Le travail ainsi élaboré est notamment le fruit de la collaboration de nombreuses personnes qui m’ont soutenu et encourager. Il m’est donc agréable de témoigner de ma gratitude, ma reconnaissance et mon profond respect :

Merci à Monsieur ***Armand Claude ABANDA,*** Représentant Résident de l’IAI-Cameroun, centre d’excellence technologique Paul BIYA qui ne manque jamais l’occasion de nous prodiguer ses précieux conseils nous permettant d’éviter les pièges pouvant nuire au bon déroulement de notre formation.

* Merci à ***Monsieur OSSE Belinga***, directeur de **Open-Secure-Fundation** qui nous a accueilli au sein de cette structure où nous avons passé notre stage.
* Merci à ***OTTOU PROSPERE Blaise***, pour la dispensation des différents modules nous ayant permis de produire ce projet
* Merci à Monsieur ***ONGUENE Aimée*** et pour leur soutient et leur accompagnement tout au long de la **production** de ce proect.
* *A la famille* ***TITI***
* *A la famille* ***NACK***
* A Mon Mère et Ma Mères pour tous les encouragements et les efforts.
* A ma sœur et amie ***Alicia Biloa*** qui n’a jamais sans cesse cesser de me soutenir et de croire en moi.
* A tout le personnel **Open-Secure-Fundation** pour leur encadrement et leur soutien lors du déroulement de notre stage.
* A mon encadreur académique M. **NOMO AWONO Francois**.
* A tous les enseignants de l’IAI-Cameroun pour leurs enseignements durant notre parcours.
* A tous mes amis et camarades de l’IAI-Cameroun pour leurs soutiens.

# SOMMAIRE

# TABLEAUX

# Glossaire des Termes Techniques

* **API (Application Programming Interface)** : Interface de programmation qui permet à deux applications de communiquer entre elles. Dans le contexte de ce rapport, il s'agit de l'API REST de WordPress.com.
* **CI/CD (Continuous Integration / Continuous Delivery)** : Méthodologie de développement logiciel qui automatise les étapes de construction, de test et de déploiement d'une application.
* **Défacement** : Attaque informatique qui consiste à modifier l'apparence ou le contenu d'une page web sans l'autorisation de l'administrateur.
* **GitHub Actions** : Plateforme d'automatisation des workflows CI/CD directement intégrée à GitHub. Elle permet d'exécuter des scripts ou des tâches sur des « runners » virtuels.
* **Hashlib** : Bibliothèque standard de Python pour la création de hachages sécurisés, utilisée ici pour vérifier l'intégrité du contenu.
* **Pipeline** : Séquence d'étapes automatisées dans un processus CI/CD.
* **Requests** : Bibliothèque Python simplifiant l'envoi de requêtes HTTP/1.1 vers des serveurs web et l'exploitation de leurs réponses.
* **REST API** : Style architectural pour la conception d'API. L'API REST de WordPress.com permet la manipulation de ressources (articles, pages, commentaires) via des requêtes HTTP.
* **Runner** : Serveur virtuel ou physique où s'exécute un job dans un workflow GitHub Actions.
* **Secrets de GitHub** : Stockage sécurisé et crypté de données sensibles (clés d'API, mots de passe) dans un dépôt GitHub pour utilisation dans les workflows.
* **Smtplib** : Bibliothèque intégrée à Python pour l'envoi de courriels en utilisant le protocole SMTP.
* **Timeout (Délai d'attente)** : Période maximale accordée pour l'achèvement d'une opération. Une erreur de timeout survient lorsque cette période est dépassée.
* **Twilio** : Plateforme de communication dans le cloud qui permet aux développeurs de programmer des fonctions de messagerie, y compris l'envoi de messages WhatsApp et SMS.
* **UML (Unified Modeling Language)** : Langage de modélisation unifié standard pour la visualisation, la spécification, la construction et la documentation des artefacts d'un système logiciel.
* **WordPress.com** : Plateforme d'hébergement web basée sur le logiciel WordPress. Il s'agit d'un service géré (SaaS) où l'utilisateur a un accès limité à l'infrastructure sous-jacente.
* **WordPress.org** : Version auto-hébergée du logiciel WordPress, nécessitant que l'utilisateur gère lui-même son propre hébergement et ses fichiers.

# RESUME

Ce rapport documente la conception et l'implémentation d'un protocole automatisé pour la sécurisation et la restauration d'un site hébergé sur la plateforme WordPress.com. Face aux menaces cybernétiques omniprésentes, la mise en place d'une solution proactive de surveillance et de sauvegarde est devenue une nécessité critique pour la continuité des activités en ligne. Le projet a été initié pour répondre à cette problématique spécifique à une plateforme SaaS, où les méthodes de gestion de code traditionnelles sont inapplicables.La solution développée contourne les limitations d'accès au système de fichiers et à la base de données de WordPress.com en s'appuyant sur son API REST. Le cœur du système est un script Python qui surveille les modifications de contenu sur des pages stratégiques du site web, telles que la page d'accueil, le flux RSS et les commentaires. Ce script utilise des mécanismes de hachage pour détecter les altérations et déclenche des procédures d'alerte et de sauvegarde en cas de changement. Le protocole est entièrement orchestré par un pipeline d'intégration et de déploiement continus (CI/CD) mis en œuvre avec GitHub Actions, ce qui garantit une exécution automatisée, fiable et sécurisée. Les données de connexion sensibles sont gérées via les secrets de GitHub, ce qui renforce la sécurité de l'ensemble du processus.

L'analyse d'un fichier de log de test a permis d'identifier et de résoudre des défis techniques majeurs, notamment des erreurs de Read timed out et d'authentification. Ces incidents ont conduit à l'intégration d'une logique de réessai et à une gestion rigoureuse des informations d'identification, transformant un dysfonctionnement initial en une opportunité d'améliorer la robustesse du système. Le projet a été documenté par neuf diagrammes UML, des captures d'écran et des extraits de code, illustrant de manière exhaustive la conception et l'architecture de la solution.Ce stage a permis l'acquisition de compétences techniques approfondies en développement Python, en automatisation CI/CD et en sécurité logicielle. Les résultats démontrent la viabilité d'une approche non-conventionnelle pour la sécurisation des plateformes SaaS, ouvrant la voie à de futures améliorations pour un système de monitoring et de restauration encore plus sophistiqué.

ABSTRACT

This report documents the design and implementation of an automated protocol for securing and restoring a website hosted on the WordPress.com platform. In the face of ubiquitous cyber threats, the establishment of a proactive monitoring and backup solution has become a critical necessity for online business continuity. The project was initiated to address this specific challenge on a SaaS platform, where traditional code management methods are not applicable.The developed solution circumvents the limitations of file system and database access on WordPress.com by leveraging its REST API. At the core of the system is a Python script that monitors content changes on strategic website pages, such as the homepage, RSS feed, and comments. This script uses hashing mechanisms to detect alterations and triggers alerting and backup procedures in the event of a change. The protocol is entirely orchestrated by a Continuous Integration and Continuous Deployment (CI/CD) pipeline implemented with GitHub Actions, which ensures automated, reliable, and secure execution. Sensitive connection data is managed via GitHub secrets, strengthening the security of the entire process.

The analysis of a test log file identified and allowed for the resolution of major technical challenges, including Read timed out and authentication errors. These incidents led to the integration of a retry logic and a rigorous management of credentials, transforming an initial malfunction into an opportunity to enhance the system's robustness. The project was documented through nine UML diagrams, screenshots, and code snippets, providing a comprehensive illustration of the solution's design and architecture.This internship has allowed for the acquisition of deep technical skills in Python development, CI/CD automation, and software security. The results demonstrate the viability of an unconventional approach to securing SaaS platforms, paving the way for future enhancements for an even more sophisticated monitoring and restoration system.

# INTRODUCTION GENERALE

Le stage s'est déroulé au sein de [Nom de l'entreprise], une société spécialisée dans le développement de solutions logicielles sur mesure et la cybersécurité. L'entreprise accompagne ses clients, principalement des PME, dans la digitalisation de leurs activités en ligne, en mettant un accent particulier sur la robustesse et la protection de leurs infrastructures. C'est dans ce cadre que s'est inscrit le projet de stage.  
 La mission principale était de concevoir et d'implémenter un protocole de sécurisation et de restauration pour un site web, en se basant sur un cas d'étude concret fourni par l'entreprise : un site vitrine client hébergé sur WordPress.com. Cette problématique a été retenue en raison de la popularité croissante des plateformes SaaS qui, bien que simplifiant la gestion pour les utilisateurs, posent des défis uniques en matière de sécurité et de résilience. Contrairement aux installations traditionnelles de WordPress auto-hébergées (WordPress.org), les plateformes comme WordPress.com limitent l'accès aux fichiers sources et à la base de données, rendant les solutions de sécurité conventionnelles inopérantes. Le stage a donc été l'occasion d'explorer une approche non-conventionnelle, s'appuyant sur l'automatisation et les interactions d'API pour pallier ces contraintes.

# PARTIE I : PHASE D’INSERTION

## RESUME

Dans cette section nous parlerons de notre accueil à **OPEN-SECURE**.  
Nous y fournirons les détails nécessaires qui ont permis l’implantation dans la structure et le déroulement des activités menées à bien.  
(Plan de localisation, organigramme, activités…)

## PLAN

### INTRODUCTION

### ACCUEIL ET INTEGRATION

### PRESENTATION DE LA STRUCTURE

### ORGANIGRAMME

# I-INTRODUCTION

Depuis le début du 21ième siècle l’informatique est devenu un domaine incontournable pour la majeur partie des métiers et activités menés aussi bien en Afrique qu’en occident de ce fait dans le but de l’obtention d’un Diplôme d’Ingénieur de Travaux Informatique que l’IAI Cameroun centre d’Excellence Technologique Paul Biya institue en son sein dans le cursus de formation des étudiants en plus d’une formation à forte valeur théorique un stage académique d’une durée de trois (03) mois qui permettront à leurs apprenants d’associer la théorie à la pratique et de se familiariser avec le monde professionnel.

Ce stage nous permettra de mieux comprendre les réalités de la vie professionnelle et de toucher d’un peu plus prêt les systèmes d’information d’entreprise en occurrence celle de ***OPEN-SECURE***. Lors de notre phase d’insertion, nous avons été entretenus par son Directeur Général avant d’être affecté à la Division des génies logiciel pour une durée allant du 03 juillet au 30 septembre 2025.

Dans le but d’accomplir cette mission, Nous avons effectué différentes tâches. L’insertion dans la structure d’accueil étant préalable, nous présenterons en premier lieu notre rapport d’insertion, suivi du cahier des charges, la présentation du thème, la méthodologie et enfin les tests et résultats.

II-ACCEUIL  
A-NTERGRATION

Nous sommes arrivés le lundi 03 juillet 2023 à la société ***OPEN-SECURE***, il était 07h précise. Après une petite attente à la salle de réception, nous avons été dirigés vers le bureau du responsable des stagiaires de l’entreprise. Après un bref entretien avec ce dernier, nous avons signé une fiche de stage dans laquelle étaient mentionnées les dates de début et de fin de stage, le nom de notre encadrant et celui du directeur. Une fois la fiche signée, nous avons été remis entre les mains de monsieur ***ONGUENE Aimée*** notre encadreur professionnel.

### B-INTECRATION

Durant cette phase d’insertion qui a durée exactement 2 semaines, il a été question pour nous de nous adapter et de nous familiariser avec notre structure d’accueil, en côtoyant quotidiennement les services de la société devant intervenir de près ou de loin à la réalisation de notre projet. Notre séjour à ***OPEN-SECURE*** nous a permis de présenter la quintessence des projets que nous envisageons et de sélectionner le plus réaliste afin d’en faire notre thème de stage et étendre à un projet lucratif.

# II-PRESENTATION DE L'ENTREPRISE : OPEN SOURCE AND CYBER SECURITY FOUNDATION (OPENSECURE)

# A-INTRODUCTION GENERALE

Le document présente la fondation **OPENS-SECURE** (Open Source and Cyber Security Foundation), une organisation à but non lucratif créée le 18 août 2023. Sa mission est de promouvoir l’open source et la cybersécurité en Afrique, et plus particulièrement au Cameroun, dans le but de générer des opportunités économiques et de transférer la propriété économique au plus grand nombre.

Pour atteindre ses objectifs, la fondation propose une gamme de services incluant des campagnes de sensibilisation, des formations certifiantes ou non, des ateliers, des conférences, ainsi que des services de conseil et d’accompagnement. Elle s’adresse principalement au secteur éducatif, aux administrations et aux entreprises. L’organisation mentionne l’Institut Saint Jean comme partenaire et reconnaît des défis tels que des ressources limitées, mais voit une grande opportunité dans le développement embryonnaire du secteur de l’open source dans la région. Le document liste également les principaux dirigeants de la fondation.

Fondée le 18 août 2023, l’Open Source and Cyber Security Foundation, plus connue sous le nom d’OPENSECURE, est une fondation apolitique et à but non lucratif dont la mission fondamentale est de démocratiser l’accès à la technologie et à l’innovation en Afrique. Avec pour slogan « La Liberté par l’open source et la cybersécurité », l’organisation se positionne comme un acteur clé du développement économique et social, œuvrant pour le transfert de la propriété économique des mains de quelques-uns vers le plus grand nombre. Elle vise à autonomiser les populations, les entreprises et les administrations, notamment au Cameroun, en les sensibilisant, en les formant et en leur offrant une expertise de pointe dans des domaines cruciaux pour l’avenir numérique.

## 1. GENERALITES SUR L’ENTREPRISE

* **Dénomination :** Open Source and Cyber Security Foundation (en anglais) ou Fondation Open Source et Cybersécurité (en français).
* **Nom abrégé :** OPENSECURE.
* **Forme juridique :** Fondation à but non lucratif et apolitique. C'est une association qui s'appuie sur ses membres pour son déploiement.
* **Date de création :** 18/08/2023.
* **Slogan :** « La Liberté par l’open source et la cybersécurité » (en français) et « Freedom by open source and cyber Security » (en anglais).

#### 2. HISTOIRE ET MISSION

La Fondation a été créée avec l’idéal de générer des opportunités économiques et créatives pour tous à travers l’adoption de l'open source et de la culture de la cybersécurité. Elle cherche à transférer la propriété économique des mains de quelques-uns vers le plus grand nombre, afin d'améliorer la qualité de vie des populations en Afrique, et particulièrement au Cameroun.

L’Open Source est perçu comme une occasion unique pour les pays en développement, car il supprime les barrières et libéralise les règles du jeu économique, garantissant ainsi l'accès à la connaissance, au savoir-faire et à l'innovation pour tous.

La Fondation a pour objectifs de :

* Promouvoir l’Open Source et la Cybersécurité au Cameroun et en Afrique.
* Sensibiliser, former et accompagner le public, notamment les élèves, les étudiants, les entreprises et les administrations, dans les domaines de l’open source et de la cybersécurité.
* Élaborer et améliorer les cadres organisationnels et juridiques liés à ces domaines.
* Mener des enquêtes et de la recherche.
* Collaborer avec d’autres personnes physiques ou morales qui œuvrent dans le même sens.

## B-SITUATION GEOGRAPHIQUE

Sont renseignés ci-dessous, le plan de localisation de la structure et la fiche signalétique.

#### 3. ORGANISATION ET DIRIGEANTS

La fondation est une association qui s’appuie sur ses membres pour son déploiement. Ses principaux dirigeants sont :

* **Président :** NGUIMBUS Emmanuel.
* **Vice-Président :** OTTOU OTTOU Prosper Simon Blaise.
* **Secrétaire Général :** BOMBA Noah Serge Patrick.
* **Secrétaire Général Adjoint :** WAMBO FOTSO Luc.
* **Trésorier :** TCHATCHOUANG TSIEMI Carolle.
* **Commissaires aux Comptes :** KEYA Henri.

#### 4. PRODUITS ET SERVICES

Les services de la fondation consistent en la sensibilisation, le renforcement des capacités et l'expertise dans les domaines des logiciels libres et de la cybersécurité. Elle déploie ses actions à travers :

* Des campagnes de sensibilisation.
* Des sessions de formation, certifiantes et non-certifiantes.
* L'élaboration de programmes scolaires et académiques.
* Des ateliers, conférences, sommets, colloques et séminaires.
* Des compétitions et prix d’excellence.
* Des services de conseil, d’accompagnement et de prestations.

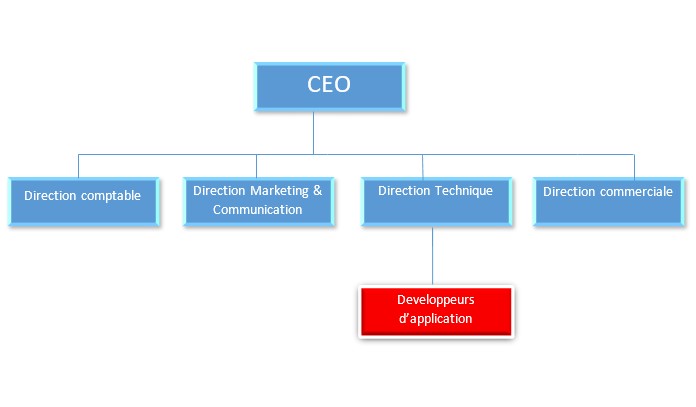
La fondation peut aussi ouvrir des centres multimédias, des centres de formation et de recherche, ainsi que des incubateurs et des pôles d’excellence technologiques.

#### 5. PARTENAIRES, CLIENTS ET ENVIRONNEMENT CONCURRENTIEL

* **Clients cibles :** L’ensemble du secteur éducatif, l’administration, les enseignants et les entreprises.
* **Partenaires :** L’Institut Saint Jean est un partenaire, et d'autres partenariats sont en cours de finalisation.
* **Facteurs influençant l’activité :** Les ressources limitées pour un déploiement à grande échelle.
* **Opportunité :** Le secteur du logiciel libre est encore à un stade embryonnaire dans son exploitation optimale.
* **Menace :** La dépendance vis-à-vis des logiciels propriétaires.

### III-ORGANIGRAMME

Division des Systèmes et Réseaux, telle qu’illustrée ci-dessous.



## IV. RESSOURCES MATERIELLES ET LOGICIELLES

L’entreprise possède de nombreux équipements listés dans le tableau qui suit.

### A-Ressources matérielles

|  |  |
| --- | --- |
| **Cinq(09) ordinateurs de bureau de marque DELL** |  |
| **08 ordinateur de marque LENOVO laptop K5-**  **9152-82FF** |  |
| **4 trousses de maintenance contenant tout un ensemble d’éléments nécessaires pour les différentes interventions rapides** |  |
| **5 imprimantes de marque HP** |  |

### B-Ressources logicielles

|  |  |
| --- | --- |
| **La suite office 2021** |  |
| **Système d’exploitation des ordinateurs de bureau : Windows 10** |  |
| **Un système d’exploitation d’un ordinateur de bureau : Windows 10** |  |
| **Antivirus pour les ordinateurs de bureau : Kaspersky + licence** |  |
|  |  |

V-Conclusion   
   
 Rendu au terme de cette phase d’insertion qui portait non seulement sur la présentation générale de l’entreprise, mais aussi sur notre familiarisation avec l’environnement de travail ; il en découle que cette étape s’est déroulée dans un cadre convivial, ceci avec un esprit de confiance dû au professionnalisme et à l’expérience du maître de stage et de ses collaborateurs. En outre, cette phase nous a été bénéfique du fait de la perception de la nécessité et de l’importance du travail collaboratif, et surtout le plus important de se confronter aux réalités du milieu professionnel.  
 **OPEN-SECURE** est une entreprise citoyenne travaillant dans le domaine de la Cyber-Sécurité. Elle est installée à Yaoundé au Cameroun. Une structure qui met à la disposition une équipe dynamique et de qualifié à la hauteur de vos besoins.

# PARTIE II : PHASE TECHNIQUE

# DOSSIER I : DOSSIER DE L’EXISTANT

Cette partie est essentiellement dans la relation d’un projet. L’existant permet la capture des besoins d’utilisateurs. Elle décrit de façon claire l’étude de l’existant, ses limites, les solutions proposées, le choix de la méthode d’analyse et enfin la modélisation de la future solution.

* ***INTRODUCTION***

1. **PRESENTATION DU THEME**
2. **ETUDE DE L’EXISTANT**
3. **CRITIQUE DE L’EXISTANT**
4. **PROBLEMATIQUE**
5. **PROPOSITION DE SOLUTION**

* ***CONCLUSION***

I-INTRODUCTION  
  
La mission principale était de concevoir et d'implémenter un protocole de sécurisation et de restauration pour un site web, en se basant sur un cas d'étude concret fourni par l'entreprise : un site vitrine client hébergé sur WordPress.com. Cette problématique a été retenue en raison de la popularité croissante des plateformes SaaS qui, bien que simplifiant la gestion pour les utilisateurs, posent des défis uniques en matière de sécurité et de résilience. Contrairement aux installations traditionnelles de WordPress auto-hébergées (WordPress.org), les plateformes comme WordPress.com limitent l'accès aux fichiers sources et à la base de données, rendant les solutions de sécurité conventionnelles inopérantes. Le stage a donc été l'occasion d'explorer une approche non-conventionnelle, s'appuyant sur l'automatisation et les interactions d'API pour pallier ces contraintes.

## A-PRESENTATION DU THEME

Ce projet se concentre sur la création d’un protocole automatisé pour sécuriser et restaurer un site web hébergé sur WordPress.com. Face aux cybermenaces constantes, la solution met en œuvre une surveillance proactive et une sauvegarde pour assurer la continuité des activités en ligne. L’approche est conçue pour contourner les limites d’accès des plateformes SaaS, où les méthodes de gestion traditionnelles ne sont pas applicables. L’ensemble est orchestré par un pipeline CI/CD utilisant GitHub Actions pour garantir une exécution fiable et autonome.

## B-Etude de l’existant

L’existant est un site vitrine sur WordPress.com. La principale faiblesse de cette plateforme est qu’elle limite l’accès aux fichiers sources et à la base de données, rendant les solutions de sécurité conventionnelles inefficaces.

Une analyse d’un prototype de surveillance a révélé des failles critiques à travers son fichier de log (monitor.log). Les problèmes majeurs identifiés étaient :

* Une erreur d’authentification (Authentication Error) avec le service d’alerte Twilio, indiquant une mauvaise gestion des informations d'identification.
* Une erreur de connexion (Read timed out) lors d’une tentative de sauvegarde, montrant que le script n’était pas assez robuste pour gérer les instabilités du réseau ou les lenteurs du serveur.

## C-PROBLEMATIQUE

La problématique du projet vise à répondre aux questions suivantes :

* Comment surveiller efficacement les modifications de contenu sur une plateforme SaaS comme WordPress.com sans accès direct au serveur?
* Comment déployer un système d’alerte instantané et fiable pour l’administrateur en cas d’altération?
* Comment concevoir un mécanisme de sauvegarde et de restauration qui fonctionne uniquement via des interfaces externes comme une API REST?
* Comment automatiser l’ensemble du protocole via un pipeline CI/CD pour assurer une exécution régulière, fiable et autonome?

## D-SOLUTIONS PROPOSEES

La solution développée est un système de surveillance externe automatisé. Le cœur du dispositif est un script Python qui interagit avec le site via son API REST. Ce script utilise des mécanismes de hachage pour créer une empreinte numérique du contenu des pages et détecter les altérations.   
 L'ensemble du processus est orchestré par GitHub Actions, qui exécute le script à intervalles réguliers. Pour corriger les failles du prototype, les informations d’identification sensibles sont gérées de manière sécurisée via les secrets de GitHub. Le script a également été amélioré pour intégrer une logique de gestion des erreurs et une augmentation du délai d’attente afin de le rendre plus résilient face aux problèmes de connexion.

E-CONCLUSION  
  
 Le projet a atteint avec succès ses objectifs en fournissant un protocole de sécurisation innovant et efficace pour une plateforme SaaS. Les difficultés initiales, telles que les erreurs de connexion et d’authentification, ont été des opportunités pour améliorer la robustesse et la sécurité de la solution. Cette expérience a permis de développer des compétences approfondies en développement Python, en automatisation CI/CD et en sécurité logicielle, préparant aux défis de l’ingénierie moderne.

# DOSSIER II : CACHIER DE CHARGES

# A-CONTEXTE ET JUSTIFICATION DE L’ÉTUDE

Dans un contexte où les entreprises dépendent de leur présence en ligne, la sécurité des sites web est devenue une nécessité critique. La popularité des plateformes SaaS comme WordPress.com simplifie la gestion pour les utilisateurs, mais introduit des défis de sécurité uniques en raison de l’accès limité à l’infrastructure. Ce projet se justifie par le besoin de créer une solution de sécurité externe et automatisée, capable d’assurer l’intégrité d’un site web malgré ces contraintes.

## B-EXPRESSION DES BESOINS DE L’UTILISATEUR

L’administrateur du site a besoin d’un système qui répond aux exigences fonctionnelles suivantes :

* Détecter de manière automatisée toute modification non autorisée du contenu public du site.
* Alerter instantanément l’administrateur par email et messagerie (WhatsApp) en cas de détection d’une altération.
* Sauvegarder l’état “stable” du site pour permettre une restauration.
* Restaurer le contenu du site en cas d’incident.
* Automatiser l’ensemble du processus pour garantir une exécution régulière et fiable sans intervention humaine.

## C. PLANIFICATION DU PROJET

Le projet s’est déroulé selon les phases logiques suivantes, telles que structurées dans le rapport:

* Phase d’Analyse (Chapitre 2) : Analyse du fichier de log et présentation des technologies clés.
* Phase de Conception (Chapitre 3) : Modélisation de l’architecture logicielle avec neuf diagrammes UML.
* Phase d’Implémentation (Chapitre 4) : Configuration de l’environnement GitHub, développement du script Python et création des workflows CI/CD.
* Phase de Bilan et Évaluation (Chapitre 5) : Évaluation du protocole, analyse des difficultés et bilan des compétences acquises.

## D. ESTIMATION DU COÛT DU PROJET

Le projet a été réalisé en optimisant les coûts grâce à l’utilisation de technologies gratuites ou open-source :

* Logiciel : Le script est développé en Python, un langage open-source.
* nfrastructure : Le code est hébergé et exécuté sur GitHub, qui fournit une plateforme d’automatisation (GitHub Actions) avec un quota d’utilisation gratuit significatif.

Le principal coût du projet réside dans le temps de développement consacré à l’analyse, la conception et l’implémentation de la solution.

## E-Estimation du cout du projet

### 1-RESSOURCE HUMAINE

Pour mener à bien notre projet, nous aurons besoin d’un analyste qui nous permettra de mieux ressortir les manquements du système en place ainsi que les différentes solutions possibles pour les combler, d’un développeur qui va réaliser l’application de notre solution et de deux chefs de projets qui dirigeront les opérations d’analyse et de développement.

|  |  |  |  |
| --- | --- | --- | --- |
| **Rôle**  **Temps** | | **Coût financier** | **Coût total** |
| **Analyste** | 6 semaines | 150 000 FCFA/  semaine | 900 000 FCFA |
| **Développeur** | 3 semaines | 150 000 FCFA/  semaine | 450 000 FCFA |
| **Chef projet/**  **Testeur** | 10 semaines | 350 000 FCFA/  semaine | 3 500 000 FCFA |
| **Chef projet**  **adjoint** | 10 semaines= | 250 000 FCFA/  semaine | 2 500 000 FCFA |
| **COÛT TOTAL** | |  | 7 350 000 FCFA |

**Source :** mercuriale 2025

### 2-RESSOURCES MATERIELLES

|  |  |  |
| --- | --- | --- |
| Matériel | Description | Prix |
| Ordinateur portable | Coré I5 7eme Génération  1terra de mémoire HDD et 32Gigas de ram | 200.000FR |

**Source :** [https://www.materiel-informatique.fr](https://www.materiel-informatique.fr/)

### 3-RESSOURCES LOGICIELLES

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **LIBELLE**  **LOGO**  **VERSION** **DESCRIPTION**  **PRIX** | | | | |
| **Microsoft**  **Windows 10** |  | 1803 | Système d’exploitation  Développé par  Microsoft | 138 000  FCFA |
| **Microsoft office**  **16** |  | 16.0.7167.2040 | Progiciel de  Bureautique  Composé de nombreux outils de création et d’édition de documents de  plusieurs formats | 186 300  FCFA |
| **Power AMC** |  | 15.1 | Atelier de génie logiciel  permettant la modélisation des applications | 20 000  FCFA |
| **Visual Studio**  **Code** |  | 1.25.0 | Editeur de code multiplateforme, supportant plusieurs langages et outils de programmation | Gratuit |
| **Firefox**  **Developer** |  | 69.0b10 | Navigateur web conçu avec les meilleurs outils pour développeurs | Gratuit |
|  |  |  |  |  |
| **Gant Project** |  | 2.8.10 | Outil de construction de diagrammes de planification de tâches pour des projets | Gratuit |
| **COÛT TOTAL** | | |  | 344 300  FCFA |

***Source :*** *mercuriale 2025*

## F. LES CONTRAINTES DU PROJET

### 1-CONTRAINTES DU COÛT

La réalisation de notre projet nécessitera des dépenses en ressources humaines, matérielles et logicielles pour un total de 8 595 950 FCFA.

### 2-LES CONTRAINTES DE DELAIS

Notre projet devra être réalisé sur une durée de 3 mois, à savoir du 03 juillet 2023 au 02 Octobre 2023

### 3-LES CONTRAINTES DE QUALITES

La plateforme résultante de la réalisation de notre projet devra suivre les contraintes de qualités suivantes :

* Elle devra être robuste, c’est-à-dire devra fonctionner avec le moins d’erreur possible.
* Elle devra être évolutive, c’est-à-dire capable de s’adapter aux nouvelles demandes de l’utilisateur.

Elle devra être ergonomique, c’est-à-dire un bon contraste entre les besoins fonctionnels et non fonctionnels de l’utilisateur, tout en gardant un équilibre entre les performances

## G. LES CONTRAINTES DU PROJET

* Contrainte technique principale : L’impossibilité d’accéder directement au système de fichiers ou à la base de données de WordPress.com. Toute interaction avec le site doit passer exclusivement par son API REST.
* Contrainte de dépendance : La solution dépend de la disponibilité et de la politique des services tiers (API de WordPress.com, GitHub Actions, Twilio).
* Contrainte de sécurité : Une gestion rigoureuse des clés d’API et des secrets est impérative pour ne pas introduire de nouvelles vulnérabilités.

## VI. LES LIVRABLES

Les livrables finaux du projet sont les suivants :

* Le code source complet du script de surveillance Python.
* Les fichiers de configuration YAML pour les workflows GitHub Actions (monitor.yml, restore.yml).
* Le rapport de stage complet, incluant l’analyse, la conception (avec 9 diagrammes UML) et les détails de l’implémentation.

## VII. LES INTERVENANTS

|  |  |  |
| --- | --- | --- |
| **NOMS ET PRENOMS** | **FONCTION**  **RÔLE** | |
| **NOMO Aimée** | Enseignant à l’IAI Cameroun | Encadrant académique |
| OTTOU BLAISE Prospere | Directeur général de OPEN-SECURE | Encadrant professionnel |
| **Le Développeur (Daniel Titi)** | Etudiant en 2ème année  Licence à l’IAI Cameroun | Réalisateur du projet. ET Stagiaire |

## VI-CONCLUSION ET PRINCIPES DE LA SECURITE

|  |  |
| --- | --- |
| **La confidentialité** | *Le but est de s’assurer que seule la personne qui est récipiendaire du message pourra comprendre le message* |
| **L’intégrité** | *Le but est de s’assurer que le message envoyé n’a pas été altéré de manière volontaire ou involontaire* |
| **La disponibilité** | *L’objectif de la disponibilité est de garantir l’accès à un service ou à des ressources* |
| **La non-répudiation** | *Ceci est la garantie qu’aucune des deux parties ne pourra assurer ne pas être l’auteur du message* |
| **L’authentification** | *L’assurance que chaque personne faisant parti de l’échange est bien celui qu’il prétend être.* |

Le dossier de l’existant, est important pour la réalisation d’un projet est une étape qui nous permet de décrire l’existant si cela existe, de montrer ses limites et de proposer une solution méliorative.Cette phase sera notre guide tout au long de l’analyse qui suivra.

# DOSSIER II : CACHIER DE CHARGES

Le cahier de charge est un document qui présente une étude détaillée du projet. Il décrit précisément les besoins auxquels les intervenants doivent répondre (***objectifs, besoins, contraintes,*** etc…). Cette partie permet à l’étudiant de mieux prendre connaissance de son thème qui est dans notre cas <<Conception et Réalisation d’une plateforme de freelance : cas du Cameroun>>.

Plan

### INTRODUCTION

### CONTEXTE ET JUSTIFICATION DE L’ETUDE

### EXPRESSION DES BESOINS DE L’UTILISATEUR

### PLANIFICATION DU PROJET

### ESTIMATION DU COÛT DU PROJET

### LES CONTRAINTES DU PROJET

### LES LIVRABLES

### LES INTERVENANTS

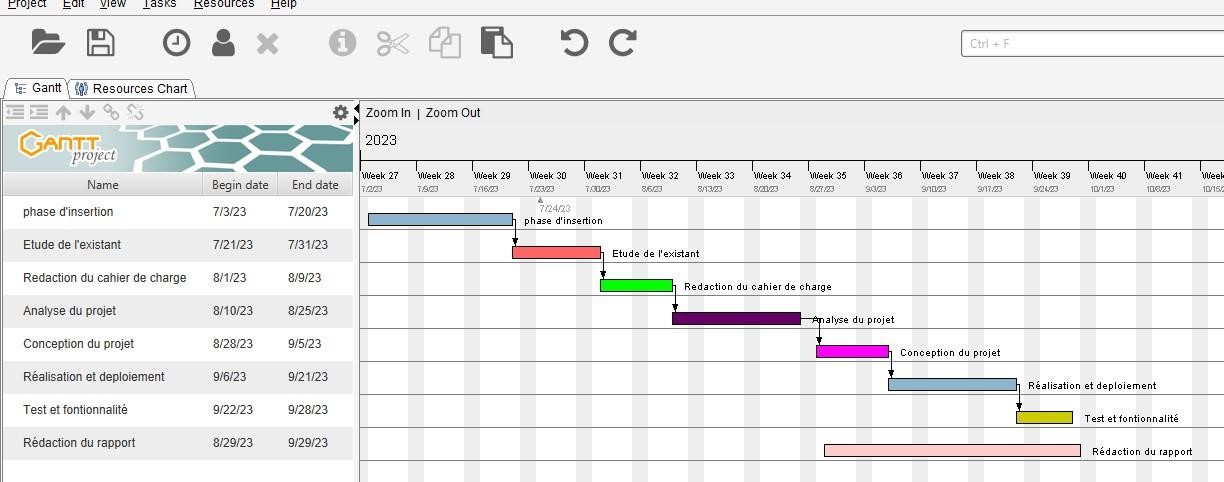
### CONCLUSION

### 3-LES CINQ PRINCIPES DE LA SECURITE SONT RESPECTES :

## **IV-Planification du proje**t

Cette étape nous permettra de représenter la répartition des taches sur des intervalles de temps bien définis. Pour cette planification, nous ferons recours au logiciel Gantt Projet. Nous obtiendrons donc un chronogramme qui représentera de voir l’état d’avancement de nos différentes taches. Elle nous permettra de ressorti, un seul coup d’ail.

### GANTT PROJET



## V-ESTIMATION DU COUT DU PROJET

### 1-RESSOURCE HUMAINE

Pour mener à bien notre projet, nous aurons besoin d’un analyste qui nous permettra de mieux ressortir les manquements du système en place ainsi que les différentes solutions possibles pour les combler, d’un développeur qui va réaliser l’application de notre solution et de deux chefs de projets qui dirigeront les opérations d’analyse et de développement.

|  |  |  |  |
| --- | --- | --- | --- |
| **Rôle**  **Temps** | | **Coût financier** | **Coût total** |
| **Analyste** | 6 semaines | 150 000 FCFA/  semaine | 900 000 FCFA |
| **Développeur** | 3 semaines | 150 000 FCFA/  semaine | 450 000 FCFA |
| **Chef projet/**  **Testeur** | 10 semaines | 350 000 FCFA/  semaine | 3 500 000 FCFA |
| **Chef projet**  **adjoint** | 10 semaines= | 250 000 FCFA/  semaine | 2 500 000 FCFA |
| **COÛT TOTAL** | |  | 7 350 000 FCFA |

**Source :** mercuriale 2025

### 2-RESSOURCES MATERIELLES

|  |  |  |
| --- | --- | --- |
| **Matériel** | **Description** | **Prix** |
| **Ordinateur portable** | **Coré I5 7eme Génération**  **1terra de mémoire HDD et 32Gigas de ram** | **200000FR** |

**Source :** [https://www.materiel-informatique.fr](https://www.materiel-informatique.fr/)

### 3-RESSOURCES LOGICIELLES

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **LIBELLE**  **LOGO**  **VERSION** **DESCRIPTION**  **PRIX** | | | | |
| **Microsoft**  **Windows 10** |  | **1803** | **Système d’exploitation**  **Développé par**  **Microsoft** | **138 000**  **FCFA** |
| **Microsoft office**  **16** |  | **16.0.7167.2040** | **Progiciel de**  **Bureautique**  **Composé de nombreux outils de création et d’édition de documents de**  **plusieurs formats** | **186 300**  **FCFA** |
| **Power AMC** |  | **15.1** | **Atelier de génie logiciel**  **permettant la modélisation des applications** | **20 000**  **FCFA** |
| **Visual Studio**  **Code** |  | **1.25.0** | **Editeur de code multiplateforme, supportant plusieurs langages et outils de programmation** | **Gratuit** |
| **Firefox**  **Developer** |  | **69.0b10** | **Navigateur web conçu avec les meilleurs outils pour développeurs** | **Gratuit** |
|  |  |  |  |  |
| **Gant Project** |  | **2.8.10** | **Outil de construction de diagrammes de planification de tâches pour des projets** | **Gratuit** |
| **COÛT TOTAL** | | |  | **344 300**  **FCFA** |

## VI. LES CONTRAINTES DU PROJET

### 1-CONTRAINTES DU COÛT

La réalisation de notre projet nécessitera des dépenses en ressources humaines, matérielles et logicielles pour un total de 8 595 950 FCFA.

## 2-LES CONTRAINTES DE DELAIS

Notre projet devra être réalisé sur une durée de 3 mois, à savoir du 03 juillet 2023 au 02 Octobre 2023

## 3-LES CONTRAINTES DE QUALITES

La plateforme résultante de la réalisation de notre projet devra suivre les contraintes de qualités suivantes :

* Elle devra être robuste, c’est-à-dire devra fonctionner avec le moins d’erreur possible.
* Elle devra être évolutive, c’est-à-dire capable de s’adapter aux nouvelles demandes de l’utilisateur.

Elle devra être ergonomique, c’est-à-dire un bon contraste entre les besoins fonctionnels et non fonctionnels de l’utilisateur, tout en gardant un équilibre entre les performances

### VII- LES LIVRABLES

**A la fin de ce projet, nous devons fournir :**

* + Un document concernant le projet comportant :
  + Un dossier insertion,
  + Un dossier d’analyse,
  + Un dossier de conception,
  + Un dossier de réalisation,
  + Un guide d’utilisation de l’application.
* **Un DVD comportant :**
  + Le rapport de stage
  + Le PowerPoint de notre travail
  + Le code source de l’application
  + Le script de la base de données

### VIII- LES INTERVENANTS

|  |  |  |
| --- | --- | --- |
| **NOMS ET PRENOMS** | **FONCTION**  **RÔLE** | |
| **NOMO AWONO FRACOIS** | Enseignant à l’IAI Cameroun | Encadrant académique |

## X-CONCLUSION

L’élaboration d’un cahier de charges est une étape très importante dans la mise en place d’une solution informatique. Ce document nous a permis de mieux cadrer notre projet et de connaître de manière exacte le chemin à suivre et les tâches à effectuer dans les délais prescrits pour mener à bien notre projet.

# PARTIE II : FONDAMENTAUX ET ARCHITECTURE DU SYSTEME

# DOSSIER III : DOSSIER DE L’ANALYSE

## RESUME

L’analyse est une étape importante dans tout projet informatique, quel que soit sa taille. Dans cette section nous parlerons de la méthode d’analyse choisie et des diagrammes qui matérialisent la structure de notre projet.

### PLAN

**INTRODUCTION**

1. **METHODOLOGIE**
2. **ETUDE COMPARATIVE UML ET MERISE**
3. **ETUDE COMPARATIVE DES PROCESSUS UNIDIES**
4. **MODELISATION**
5. **DIAGRAMMES DE CAS D’UTILISATION**
6. **DIAGRAMME DE COMMUNICATION**
7. **DIAGRAMME DE SEQUENCE**
8. **DIAGRAMME D’ACTIVITE**

**CONCLUSION**

**INTRODUCTION**

L’analyse est un examen méthodique permettant de distinguer les différentes parties d’un problème et d’en déterminer les solutions en suivant une méthodologie bien définie. L’analyse de notre proje6t sera basée sur la méthodologie du langage ***UML***, selon le processus ***2TUP.***

Par ces derniers, nous tirerons les diagrammes intervenants dans la phase d’analyse :

* ***Diagrammes des cas d’utilisations***
* ***Les diagrammes d’activités***
* ***Les diagrammes de communications***
* ***Les diagrammes de séquence.***

## I-METHODOLOGIE

Notre analyse sera essentiellement basée sur le langage ***UML***, avec le ***processus 2TUP***

## 1-ETUDE COMPARATIVE UML ET MERISE

### 1-1 Généralité sur les méthodes d’analyse UML et MERISE

Une méthode définit une démarche reproductible qui produit des résultats fiables. Une méthode d’élaboration d’application décrit comment modéliser et construire les systèmes logiciels de manière fiable et reproductible. De manière générale, une méthode définie :

* ***Des éléments de modélisation ;***
* ***Une représentation graphique ;***
* ***Du savoir-faire et des règles.***

Il existe plusieurs méthodes d’analyse, mais les plus utilisées sont ***MERISE et UML*** :

* **MERISE** *(Méthode d’Etude et de Réalisation Informatique pour les Systèmes d’Entreprise*) est une méthode d'analyse, de conception structurelle et de réalisation des systèmes d'information qui est élaborée en plusieurs étapes : schéma directeur, étude préalable, étude détaillée et la réalisation.
* **UML** (***U****nified* ***M****odeling* ***L****anguage*), est un langage de modélisation des systèmes standards, qui utilise des diagrammes pour représenter chaque aspect d'un système : statique, dynamique en s'appuyant sur la notion d'orienté objet qui est un véritable atout pour ce langage.

### 1-2-Etude comparative entre UML et MERISE

|  |  |
| --- | --- |
| **MERISE** | **UML** |
| ***Méthode systématique d’analyse et de conception des systèmes d’information :*** | ***Langage de représentation d’un système d’information*** |
| ***Représentation sous forme de model*** | ***Représentation sous forme de diagrammes*** |
| ***Approche systématique*** | ***Approche Objet*** |
| ***Franco-Français*** | ***Internationale*** |
| ***Propose de considérer le système réel selon deux aspects :***  ***Une vue statique (données)***  ***Une vue dynamique*** | ***Propose une approche différente en ce sens qu’elle associe les données et les traitements. Avec UML, les données d’un type et les traitements associés*** |
| ***Avec MERISE, nous avons une étude séparée des données et des traitements*** | ***UML permet de limiter les points de maintenances dans le code et facilite l’accès à l’information en cas d’évolution*** |
| ***Schéma directeur, étude préalable, Etude détaillée et la réalisation*** | ***Langage de modélisation des systèmes standards, qui utilise des diagrammes pour représenter chaque aspect d’un système : statique,***  ***dynamique… en s’appuyant sur la notion d’orienté objet*** |

## 3-PRESENTATION DE LA METHODE D’ANALYSE

### 1-CHOIX DE LA METHODE L’ANALYSE

Au regard de la complexité croissante des systèmes d’informations et de cette étude comparative, **UML** est le mieux adapté dans le cadre de notre projet. UML est un langage unifié de modélisation qui permet de spécifier, visualiser, construire et documenter de manière précise un système d’information. UML grâce à ses langages offre une description statique, dynamique et fonctionnelle du système à réaliser en plus il est idéal pour :

* Concevoir et déployer une architecture logicielle développée dans un langage objet. Certes UML, dans sa volonté ***"unificatrice***" a proposé des formalismes ;
* Pour modéliser les données (l***e modèle de classe réduit sans méthodes et stéréotypé en entités***), mais avec des lacunes que ne présentait pas l'entité relation de Merise ;
* Pour modéliser le fonctionnement métier (***le diagramme de cas d'utilisation***) qui est un formalisme très ancien.

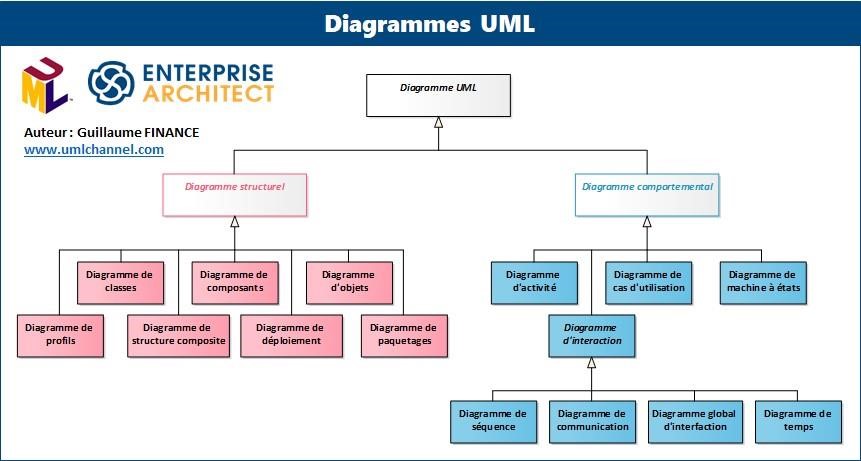
### 2- PRESENTATION DE UML

**UML** signifie *Unified Modeling Language* en français Langage de Modélisation Unifié. ***« Unifié »*** car il regroupe les langages **BOOCH, OMT et OOSE**. Il est utilisé en développement logiciel, et en conception orientée objet. UML est à présent une méthode standard défini par

L’Object Management Group (***OMG).*** La dernière version diffusée par ***l’OMG est UML 2.4.1*** depuis août 2011.

### 3-VERSION

La version d’UML que nous allons utiliser dans notre projet est ***la version 2.5.1 qui comporte 14 diagrammes.***



Source : [www.umlchanel.com](http://www.umlchanel.com/)

### 4-LES DIAGRAMMES D’UML

La modélisation d’un projet informatique effectué à l’aide d’UML version 2.5 présente 14 diagrammes. Ces diagrammes sont regroupés de la manière suivante :

* **Diagrammes comportementaux** : Diagramme de cas d’utilisation, d’activité, diagramme d’état transition et des diagrammes d’interactions (diagramme de séquence, de communication, le diagramme global d’interaction, le diagramme de temps).
* **Diagrammes structurels** : Diagramme de classe, diagramme de composants, diagramme d’objets, diagramme de profils, diagramme de structure composite, diagramme de déploiement et le diagramme de paquetages.

### 5-FORMALISME

* **Les vues** : Les vues sont les observables du système. Elles décrivent le système d’un point de vue donné, qui peut être organisationnel, dynamique, temporel, architectural, géographique, logique. En combinant toutes ces vues, il est possible de définir (ou retrouver) le système complet.
* **Les diagrammes** : Les diagrammes sont des éléments graphiques. Ceux-ci décrivent le contenu des vues, qui sont des notions abstraites. Les diagrammes peuvent faire partie de plusieurs vues.
* **Les modèles d’élément** : Les modèles d’élément sont les briques des diagrammes,UML, ces modèles sont utilisés dans plusieurs types de diagrammes à l’exemple des cas d’utilisation, des classes, des associations.

### 6-LES VUES

Les différentes vues d’UML sont les suivantes :

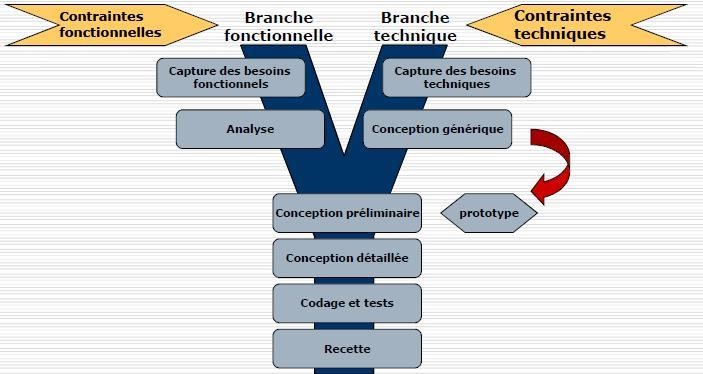
* **Vue des cas d’utilisation** : C’est la description du modèle vu par les acteurs du système. Elle correspond aux besoins attendus par chaque acteur (c’est le QUOI et le QUI).
* **Vue logique** : C’est la définition du système vu de l’intérieur. Elle explique comment peuvent être satisfaits les besoins des acteurs (c’est le COMMENT).
* **Vue d’implémentation** : Cette vue définit les dépendances entre les modules.
* **Vue des processus** : C’est la vue temporelle et technique, qui met en œuvre les notions de tâches concurrentes, stimuli, contrôle, synchronisation, etc.
* **Vue de déploiement** : Cette vue décrit la position géographique et l’architecture physique de chaque élément du système (c’est Le OÙ).

## 2-COMPARAISON DES PROCESSUS UNIFIEES

Un processus définit une séquence d’étapes ordonnées qui concours à l’obtention d’un système logiciel ou à l’évolution d’un système existant. L’objet d’un processus de développement est de produire des logiciels de qualité qui répondent aux besoins de leurs utilisateurs dans des temps et des coûts prévisibles. Le 2TUP (Two Track Unified Process) est un processus de développement logiciel qui implémente le processus unifié. Il propose un cycle de développement qui dissocie les aspects techniques des aspects fonctionnels. Il s’agit de l’étude de l’application et de son implémentation. Le processus 2TUP s’articule autour de trois branches à savoir :

* **Une branche fonctionnelle :** Elle produit le modèle des besoins focalisés sur le métier des utilisateurs. Elle permet de visualiser au plus tôt le risque de produire un système inadapté aux utilisateurs.
* **Une branche Technique :** Elle Recense toutes les contraintes sur les choix de dimensionnement et la conception du système, les outils et le matériel sélectionné ainsi que
* La prise en compte des contraintes d’intégration avec l’existant. Cette étape permet de définir le modèle d’analyse technique.
* **Une branche de réalisation :** La phase de réalisation consiste à réunir les deux branches, permettant de mener une conception applicative et enfin la livraison d’une solution adaptée aux besoins.

## Ceci illustré à travers la figure suivante :



### 

### A-LA BRANCHE FONCTIONNELLE

Les étapes de la branche fonctionnelle se présentent comme suit :

* La capture des besoins fonctionnels : elle produit le modèle des besoins focalisés sur le métier des utilisateurs. Elle permet de visualiser au plus tôt le risque de produire un système inadapté aux utilisateurs. Cette phase a pour objectif de définir :
  + La frontière fonctionnelle entre le système considéré comme une boîte noire et son environnement, c’est le niveau contextuel ;
  + Les activités attendues des différents utilisateurs par rapport au système toujours envisagées comme une boîte noire, c’est le niveau cas d’utilisation.

L’étape d’analyse : elle consiste à étudier précisément les spécifications fonctionnelles de manière à obtenir une idée de ce que va réaliser le système en termes de métier.

### B-LA BRANCHE TECHNIQUE

Les étapes de la branche technique se présentent comme suit :

* L’étape de la capture des besoins techniques : recense toutes les contraintes sur les choix de dimensionnement et de la conception du système, les outils et le matériel sélectionné ainsi que la prise en compte des contraintes d’intégration avec l’existant (pré requis d’architecture technique). Cette étape permet de définir le modèle d’analyse technique.
* L’étape de conception générique : définis ensuite les composants nécessaires à la construction de l’architecture technique. Cette conception est complètement indépendante des aspects fonctionnels. Elle permet de générer le modèle de conception technique ou design**.**

### C-LA PHASE DE REALISATION

La phase de réalisation consiste à réunir les deux branches, permettant de mener une conception applicative et enfin la livraison d’une solution adaptée aux besoins.

## II-MODELISATION

## 1-DIAGRAMME DE CAS D’UTILISATION

L’étude fonctionnelle nous a permis d’y définir les contraintes (ce que notre système doit réaliser en termes du métier) ; on y relevé ainsi la capture des besoins fonctionnels, d’analyse et de spécification.

### A-DEFINITION

Un diagramme de cas d’utilisation capture le comportement d’un système, son rôle est de recueillir, d’analyser et d’organiser les besoins, ainsi que de recenser les grandes fonctionnalités d’un système en représentant de manière graphique toutes les actions possibles d’un acteur dans ce dernier.

Il s’agit donc de la première étape UML pour l’analyse d’un système.

### B- Formalisme

Un diagramme de cas d’utilisation se compose des principaux éléments énumérés dans le tableau suivant :

|  |  |  |
| --- | --- | --- |
| **REPRESENTATION NOM** | | **DESCRIPTION** |
| Acteur | **Acteur** | **Un acteur est un élément externe qui joue un rôle dans le système. Le rôle joué est considéré comme un cas**  **d’utilisation** |
| |  | | --- | | **Systeme** | | **Système** | **Le système représente l’espace étudié. C’est dans celui-ci que l’acteur effectue ses actions, par conséquent c’est dans ce dernier que se trouvent les cas d’utilisation. Il est représenté par un cadre, avec le nom du système placé en haut.** |
| cas d'utilisation | **Cas d’utilisation** | **Il représente une fonctionnalité fournie par le système. Il est décrit sous la forme verbe (infinitif) + objet. Il correspond à un objectif du système motivé par un besoin d’un ou plusieurs utilisateurs.**  **L’ensemble des cas d’utilisation décrit le but du système. Le cas d’utilisation est représenté par une ellipse contenant son nom.** |
|  | **Association** | **Les associations sont utilisés pour lier les acteurs avec les cas d’utilisation.**  **Elles indiquent qu’un acteur participe au cas d’utilisation sous une forme quelconque. Les associations sont représentées par une ligne reliant l’acteur au cas**  **d’utilisation** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Relation** | | | | **Les relations sont utilisées pour lier les cas d’utilisation entre eux. Elles permettent de définir quel cas d’utilisation dépend de l’autre (relation**  **<<include>> ou d’inclusion) ou encore quel cas d’utilisation est une possibilité étendue d’un autre (relation <<extend>> ou d’extension). Les relations sont représentées par une flèche à traits interrompus allant d’un cas d’utilisation à l’autre.** |
|  |  |  | |
|  |
|  | | | **Généralisation** | **Les généralisations sont utilisées pour lier les acteurs entre eux. Elles permettent de dire qu’un acteur effectue tous les cas d’utilisation d’un autre en plus de ce qu’il peut faire. Cette relation est aussi appelé relation d’héritage. Les généralisations sont représentées par une flèche allant d’un acteur (celui qui hérite) à un autre.** |

**Acteur Principal (Entends)** : Administrateur Système

**Déclencheur (Intrend) :** Demande manuelle via l’interface GitHub (workflow\_dispatch)

**Problème :** Le site WordPress a été compromis à la suite d’une attaque ou les données ont été corrompues. Le site est inaccessible ou malveillant.

**Solution :** Exécution manuelle du workflow de restauration qui récupère la dernière sauvegarde valide stockée sur AWS S3 et la restaure sur le serveur.

**Processus Détaillé :**

1. L’administrateur se connecte à GitHub et navigue vers l’onglet Actions.

2. Il sélectionne le workflow Restore et clique sur le bouton Run workflow.

3. Le workflow GitHub Actions se déclenche : a. Se connecte au bucket AWS S3 via les clés d’accès stockées dans les secrets GitHub. B. Télécharge la dernière sauvegarde chiffrée (fichiers + base de données). C. Se connecte en SSH au serveur de production. D. Décompresse et déchiffre la sauvegarde. E. Remplace les fichiers du site et importe la base de données. F. Exécute des commandes WP-CLI pour vider les caches et réinitialiser les permissions.

### C-IDENTIFIANT DES DIFFERENTS ACTEURS DU SYSTEME

Les acteurs se recrutent parmi les utilisateurs d’un système et aussi parmi les responsables de sa configuration et de sa maintenance d’où les acteurs potentiels qui vont interagir avec l’application sont :

* Le visiteur
* Le client
* Le vendeur
* L’administrateur
* L’API de paiement  Un chabot

#### I-DIAGRAMME DE CAS D’UTILISATION

**Cas d’Utilisation 1 : Restauration Rapide après un Incident (Piratage/Corruption)**

**Acteur Principal (Entends) :** Administrateur Système

**Déclencheur (Intrend) :** Demande manuelle via l’interface GitHub (workflow\_dispatch)

**Problème :** Le site WordPress a été compromis à la suite d’une attaque ou les données ont été corrompues. Le site est inaccessible ou malveillant.

**Solution :** Exécution manuelle du workflow de restauration qui récupère la dernière sauvegarde valide stockée sur AWS S3 et la restaure sur le serveur.

**Processus Détaillé :**

1. L’administrateur se connecte à GitHub et navigue vers l’onglet Actions.

2. Il sélectionne le workflow Restore et clique sur le bouton Run workflow.

3. Le workflow GitHub Actions se déclenche : a. Se connecte au bucket AWS S3 via les clés d’accès stockées dans les secrets GitHub. B. Télécharge la dernière sauvegarde chiffrée (fichiers + base de données). C. Se connecte en SSH au serveur de production. D. Décompresse et déchiffre la sauvegarde. E. Remplace les fichiers du site et importe la base de données. F. Exécute des commandes WP-CLI pour vider les caches et réinitialiser les permissions.

4. Une notification de succès ou d’échec est envoyée sur Slack.

5. Le site est restauré à son état antérieur, sain, en moins de 15 minutes (RTO).

### A-DESCRIPTION

UML permet de décrire l’exécution d’un cas d’utilisation de manière textuelle, sous une forme appelée scénario nominal. Un scénario nominal décrit de manière plus détaillée l’exécution d’un cas d’utilisation par un acteur jusqu’à ce qu’il soit entièrement réalisé avec succès.

#### 1-FORMALISME

Une description textuelle d’un cas d’utilisation est représentée sous la forme qui suit :

|  |  |
| --- | --- |
| **NOM DU CAS D’UTILISATION** | |
| **acteur** | Acteur concerné dans le cas d’utilisation |
| **Présupposé** | Ensemble des actions qui doivent être remplies avant le lancement du cas d’utilisation |
| **Précondition** | Ensemble des conditions qui doivent être remplies avant le lancement du cas d’utilisation |
| **Déclencheur** | Elément qui déclenche le cas d’utilisation |
| **Dialogue** | Description du scénario nominal |
| **Scénarii alternatifs** | Description des scénarii alternatifs (cas où le scénario nominal est un échec) |
| **Post condition** | Ensemble des mécanismes pouvant conduire à la fin du cas d’utilisation |
|  |  |

#### 2-DESCRIPTION TEXTUELLE DE S’INSCRIRE

Ce diagramme présente une vue d'ensemble des interactions entre les acteurs principaux (Administrateur, Développeur, Planificateur) et le système CI/CD, ainsi que les fonctionnalités clés qu'ils utilisent.

Explication détaillée : Ce diagramme sert de carte mentale pour comprendre qui utilise le système et pourquoi.L'Administrateur Système est l'acteur chargé des opérations critiques de restauration et de surveillance manuelle, intervenant en cas d'incident. Le Développeur interagit avec le système dans le cadre du cycle de développement normal : il propose du code, le déploie et corrige les vulnérabilités identifiées en amont. Le Planificateur Système représente l'automatisation temporelle ; il n'agit pas manuellement mais configure et supervise les processus déclenchés par cron, comme les audits et sauvegardes périodiques.

Enfin, Slack est un acteur passif mais crucial qui reçoit toutes les notifications, faisant office de centre de contrôle et de journal des événements. Ce diagramme illustre ainsi la philosophie du système : une collaboration entre l'automatisation et l'expertise humaine, où chaque acteur a un rôle bien défini.

Diagrammes

## 2-DIAGRAMMES DE SEQUENCES - RESTAURATION APRES INCIDENT

### 1-DEFINITON

Vue de l’extérieur du système, Les Diagrammes de séquence permettent de montrer les interactions entre les utilisateurs et le système à travers une représentation linéaire et chronologique des cas d’utilisations du système. Chaque colonne correspond à un objet ou éventuellement à un acteur, introduit dans le diagramme des cas d’utilisation. La ligne de vie de l’objet représente la durée de son interaction avec les autres objets du diagramme.

#### 2-FORMALISME

#### Cas d’Utilisation 1 : Restauration Rapide après un Incident (Piratage/Corruption)

**Acteur Principal (Entends) :** Administrateur Système  
**Déclencheur (Intrend) :** Demande manuelle via l’interface GitHub (workflow\_dispatch)  
**Problème :** Le site WordPress a été compromis à la suite d’une attaque ou les données ont été corrompues. Le site est inaccessible ou malveillant.  
**Solution :** Exécution manuelle du workflow de restauration qui récupère la dernière sauvegarde valide stockée sur AWS S3 et la restaure sur le serveur.

#### Processus Détaillé :

1. L’administrateur se connecte à GitHub et navigue vers l’onglet Actions.

2. Il sélectionne le workflow Restore et clique sur le bouton Run workflow.

3. Le workflow GitHub Actions se déclenche : a. Se connecte au bucket AWS S3 via les clés d’accès stockées dans les secrets GitHub. B. Télécharge la dernière sauvegarde chiffrée (fichiers + base de données). C. Se connecte en SSH au serveur de production. D. Décompresse et déchiffre la sauvegarde. E. Remplace les fichiers du site et importe la base de données. F. Exécute des commandes WP-CLI pour vider les caches et réinitialiser les permissions.

4. Une notification de succès ou d’échec est envoyée sur Slack.

5. Le site est restauré à son état antérieur, sain, en moins de 15 minutes (RTO).

#### Cas d’Utilisation 2 : Blocage d’un Déploiement à Risque

**Acteur Principal (Entends) :** Développeur (via un git push ou une Pull Request)

**Déclencheur (Intrend) :** Intégration du code vulnérable déclenchant l’échec du pipeline CI

**Problème :** Un développeur pousse du code ou met à jour un plugin (ex: Elementor) qui contient une vulnérabilité critique connue (CVE).

**Solution :** Le pipeline CI/CD intègre un scan de sécurité (WPScan, Snyk) qui analyse les dépendances à chaque commit. Si une vulnérabilité critique est détectée, le déploiement est bloqué.

#### Processus Détaillé :

1. Le développeur pousse son code sur la branche main ou ouvre une Pull Request.

2. Le workflow CI (fichier ci-cd.yml) se déclenche automatiquement.

3. Le job build-test exécute l’étape Security scan.

4. Le scan détecte un plugin avec une vulnérabilité de niveau “CRITIQUE”.

5. L’étape de scan renvoie un code d’erreur (fail), faisant échouer l’ensemble du job build-test.

6. Le job deploy-prod, qui dépend du succès de build-test, ne s’exécute pas.

7. Le développeur reçoit une notification (email, Slack) indiquant l’échec du pipeline et la raison (vulnérabilité critique).

8. Le code vulnérable n’est jamais déployé en production.

#### Cas d’Utilisation 3 : Audit de Sécurité Périodique Automatisé

**Acteur Principal (Entends) :** Planificateur Système (cron)

**Déclencheur (Intrend) :** Déclenchement programmé (schedule) du workflow pentest.yml

**Problème :** De nouvelles vulnérabilités sont découvertes quotidiennement. Un site peut devenir vulnérable entre deux déploiements si une faille est divulguée dans un composant déjà en production.

**Solution :** Exécution automatique et périodique d’un scan de sécurité complet sur le site en production, indépendamment des cycles de développement.

#### Processus Détaillé :

1. Tous les lundis à 3h00 (configuré via cron: ‘0 3 \* \* 1’), GitHub Actions déclenche automatiquement le workflow pentest.yml.

2. Le job wpscan utilise l’API WPScan pour effectuer un scan complet du site en production (thème, plugins, core WordPress).

3. Les résultats du scan sont analysés.

4. Scénario 1 – Aucune vulnérabilité critique : Un log est enregistré dans l’exécution du workflow.

5. Scénario 2 – Vulnérabilité(s) critique(s) trouvée(s) : Le script utilise le webhook Slack pour envoyer une alerte immédiate et détaillée sur le canal dédié de l’équipe.

6. L’équipe peut alors investiguer et planifier une correction urgente, éventuellement en déclenchant manuellement une mise à jour.

#### Cas d’Utilisation 4 : Déploiement Continu et Sécurisé d’une Nouvelle Fonctionnalité

**Acteur Principal (Entends) :** Développeur (via une Pull Request)

**Déclencheur (Intrend) :** Fusion (merge) de la Pull Request sur la branche main

**Problème :** Les déploiements manuels (FTP) sont longs, sujets aux erreurs et manquent de traçabilité.

**Solution :** Un pipeline entièrement automatisé qui teste, valide et déploie le code de manière fiable et sécurisée dès qu’il est fusionné.

#### Processus Détaillé :

1. Un développeur travaille sur une nouvelle fonctionnalité dans une branche dédiée.

2. Il ouvre une Pull Request (PR) pour fusionner ses changements dans main.

3. Le workflow CI se déclenche automatiquement sur la PR : a. Exécute les tests unitaires. B. Effectue un scan de sécurité rapide.

4. Si les étapes CI passent, un responsible peut merger la PR.

5. Le merge sur main déclenche une nouvelle exécution du workflow CI (étape build-test), puis si réussie, déclenche l’étape deploy-prod.

6. Le job deploy-prod se connecte au serveur via SSH et synchronise les fichiers modifiés avec rsync (efficace et sécurisé).

7. Des commandes post-déploiement (ex: wp cache flush) sont exécutées.

8. Une notification de succès est envoyée sur Slack.

#### CONCLUSION

Ces quatre cas d’utilisation démontrent la puissance et la polyvalence de l’approche CI/CD pour la gestion moderne de WordPress. Le système ne fait pas qu’automatiser des tâches ; il introduit un véritable changement de paradigme :

· De la réaction à la prévention : En bloquant les déploiements risqués et en auditant continuellement.

· De la panique à la procédure : En offrant une restauration en 1-clic reproductible.

· Du manuel à l’automatisé : En éliminant les points de friction et les erreurs humaines des déploiements.

La distinction entre entends (déclencheurs externes) et intrend (réponses automatisées internes) est cruciale pour comprendre comment l’architecture maintient le contrôle tout en déléguant l’exécution à des processus fiables et vérifiés.

#### 3-DESCRIPTION TEXTUELLE

Ce diagramme décrit la séquence précise et chronologique des interactions entre les différents composants lors d'une procédure de restauration manuelle après un piratage ou une corruption de données.

Explication détaillée : La séquence commence par l'action humaine(entends) : un administrateur déclenche manuellement le workflow depuis l'interface GitHub. C'est le seul point d'entrée manuel dans un processus ensuite entièrement automatisé (intrend). GitHub Actions prend le relais, devenant le chef d'orchestre. Sa première action est de se connecter au "coffre-fort" externe, AWS S3, pour récupérer la dernière sauvegarde valide et chiffrée. Une fois l'archive récupérée, le runner établit une connexion SSH sécurisée avec le serveur de production. La magie opère ensuite : il remplace les fichiers corrompus et importe la base de données saine.   
 Des commandes WP-CLI sont exécutées pour finaliser la restauration (vidage de cache, reset des permissions). Pour boucler la boucle, une notification de succès est envoyée sur Slack, informant l'équipe que le site est de nouveau opérationnel. Ce processus sécurisé et reproductible élimine le stress et les erreurs d'une restauration manuelle.

3-DIAGRAMME DE SEQUENCE - DEPLOIEMENT AVEC VALIDATION DE SECURITE  
  
A-FORMALISME  
**Problème :** De nouvelles vulnérabilités sont découvertes quotidiennement. Un site peut devenir vulnérable entre deux déploiements si une faille est divulguée dans un composant déjà en production.

**Solution :** Exécution automatique et périodique d’un scan de sécurité complet sur le site en production, indépendamment des cycles de développement.

**Processus Détaillé :**

1. Tous les lundis à 3h00 (configuré via cron: ‘0 3 \* \* 1’), GitHub Actions déclenche automatiquement le workflow pentest.yml.

2. Le job wpscan utilise l’API WPScan pour effectuer un scan complet du site en production (thème, plugins, core WordPress).

3. Les résultats du scan sont analysés.

4. Scénario 1 – Aucune vulnérabilité critique : Un log est enregistré dans l’exécution du workflow.

5. Scénario 2 – Vulnérabilité(s) critique(s) trouvée(s) : Le script utilise le webhook Slack pour envoyer une alerte immédiate et détaillée sur le canal dédié de l’équipe.

6. L’équipe peut alors investiguer et planifier une correction urgente, éventuellement en déclenchant manuellement une mise à jour.

## B-DESCRIPTION TEXTUELLE

Ce diagramme modélise le flux de validation et de déploiement continu, en mettant en lumière le point de décision critique qui bloque les versions non sécurisées.

Contrairement le premier scénario réactif(restauration), celui-ci est proactif. Le déclencheur est un git push ou une Pull Request d'un développeur. Le système CI se lance immédiatement, et une de ses tâches primordiales est d'exécuter un scan de sécurité (avec WPScan ou Snyk) sur le code proposé. Ici, le diagramme présente une branche conditionnelle (alt/else) qui est le cœur de la stratégie "Shift-Left". Si une vulnérabilité critique est trouvée, le workflow est marqué comme échec, le déploiement est bloqué, et des alertes sont envoyées au développeur et sur Slack. Le code dangereux n'atteint donc jamais la production. Si le scan est clean, le feu vert est donné au job de CD qui déploie le code de manière synchronisée et efficace via rsync et finalise le déploiement avec WP-CLI. Ce processus garantit que seul du code testé et sécurisé est déployé.

## 4- DIAGRAMME D'ACTIVITE - AUDIT DE SECURITE PERIODIQUE

### A-DEFINITION

Un diagramme d'activité s’accentue sur les traitements. Il est particulièrement adapté à la modélisation du cheminement de flots de contrôle et de flots de données. Il permet ainsi de représenter graphiquement le comportement d'une méthode ou le déroulement d'un cas d'utilisation.

#### B-FORMALISME

#### C-DESCRIPTION TEXTUELLE

Ce diagramme illustre le flux de contrôle et les décisions prises lors de l'exécution automatisée et planifiée d'un audit de sécurité complet.Explication détaillée : Ce processus est unique car il est entièrement initié par unintrend (un déclencheur temporel cron), sans aucune action humaine. Il démarre tous les lundis à 3h00 pour auditer le site en production. La première activité est l'exécution d'un scan complet avec WPScan.

Vient ensuite le point de décision central : l'analyse des résultats pour trouver des vulnérabilités de niveau "CRITIQUE". Si c'est le cas, l'activité bifurque vers la préparation et l'envoi d'une alerte urgente sur Slack, permettant une réaction immédiate de l'équipe. Si rien de critique n'est trouvé, le résultat est simplement enregistré dans les logs pour archive, et le processus se met en pause jusqu'au prochain cycle. Ce diagramme montre comment le système assure une vigilance continue ; il est toujours actif, scrutant l'horizon pour détecter de nouvelles menées qui pourraient apparaître entre deux déploiements.

## 5-DIAGRAMME DE COMMUNICATION - PROCESSUS CI/CD COMPLET

### **A-DEFINITION**

Un diagramme de communication dans le langage de modélisation unifié fait référence à un graphique qui représente le flux de messages dans un système. En un mot, il montre comment les parties d’un système interagissent ou, dans ce cas, communiquent entre elles. Ce type de diagramme utilise l’arrangement de forme libre, c’est pourquoi les messages sont étiquetés avec des nombres. A partir de 1.0 et ainsi de suite, ce schéma est assez facile à lire car nous êtes guidé par le numéro de séquence.

### **B-FORMALISME**

**Problème :** De nouvelles vulnérabilités sont découvertes quotidiennement. Un site peut devenir vulnérable entre deux déploiements si une faille est divulguée dans un composant déjà en production.

**Solution :** Exécution automatique et périodique d’un scan de sécurité complet sur le site en production, indépendamment des cycles de développement.

**Processus Détaillé :**

1. Tous les lundis à 3h00 (configuré via cron: ‘0 3 \* \* 1’), GitHub Actions déclenche automatiquement le workflow pentest.yml.

2. Le job wpscan utilise l’API WPScan pour effectuer un scan complet du site en production (thème, plugins, core WordPress).

3. Les résultats du scan sont analysés.

4. Scénario 1 – Aucune vulnérabilité critique : Un log est enregistré dans l’exécution du workflow.

5. Scénario 2 – Vulnérabilité(s) critique(s) trouvée(s) : Le script utilise le webhook Slack pour envoyer une alerte immédiate et détaillée sur le canal dédié de l’équipe.

6. L’équipe peut alors investiguer et planifier une correction urgente, éventuellement en déclenchant manuellement une mise à jour.

### **C-DESCRIPTION TEXTUELLE**

Ce diagramme offre une vue synthétique **"big picture"** de l'ensemble de l'architecture, montrant comment les différents processus (développement, audit) interagissent et communiquent entre eux et avec les acteurs.

Explication détaillée : Ce diagramme est le plus complet car il intègre les flux des autres séquences. Il montre deux flux principaux qui coexistent : le flux de développement (en haut) et le flux de surveillance (en bas). Le flux de développement, initié par le développeur, passe par les étapes séquentielles de CI (intégration) et de CD (déploiement). Le flux de surveillance, lui, est indépendant et déclenché par le temps. La force de ce diagramme est de montrer comment ces deux flux distincts convergent vers le même point : la notification Slack. Que ce soit pour un déploiement réussi, un échec de validation de sécurité, ou une alerte d'audit, toutes les informations remontent vers la même plateforme. Cela illustre le concept de centralisation des observabilités : Slack devient le tableau de bord unique où l'équipe peut voir l'état de santé et de sécurité du site à tout moment, que le problème vienne du code nouvellement déployé ou de l'infrastructure existante.

Les diagrammes illustrent les concepts clés du document :

1. Séparation des préoccupations : Chaque acteur (Administrateur, Développeur, Planificateur) a des responsabilités distinctes

2. Automation des processus : Les workflows CI/CD réduisent l’intervention humaine aux décisions critiques

3. Sécurité “Shift-Left” : La détection des vulnérabilités intervient tôt dans le cycle de développement

4. Résilience du système : Les mécanismes de restauration et de surveillance assurent la continuité d’activité

5. Boucle de feedback : Les notifications Slack ferment la boucle de communication pour une réactivité optimale

Ces diagrammes démontrent comment l’architecture proposée transforme la gestion WordPress traditionnelle en un processus industrialisé, sécurisé et résilient.

I-INTRODUCTION

Pendant des années, la gestion d'un site WordPress a été synonyme d'opérations manuelles : mises à jour via l'interface d'administration, déploiements par glisser-déposer FTP, et sauvegardes via des plugins dont la fiabilité n'était testée qu'au moment critique de la restauration. Cette approche, bien que fonctionnelle pour des projets simples, expose les sites critiques à des risques inacceptables : erreurs humaines, failles de sécurité non détectées, temps d'arrêt prolongés et processus de restauration hasardeux. Ce guide présente une approche radicalement différente, traitant un site WordPress comme une application moderne. Il détaille une architecture où chaque aspect du cycle de vie du site est défini dans du code, automatisé, et sécurisé par des processus reproductibles. L'objectif est de fournir une feuille de route complète pour atteindre un niveau de professionnalisme, de sécurité et de résilience digne des applications d'entreprise.

### 1-À QUI S'ADRESSE CE GUIDE ?

Ce document est conçu pour un public technique. Il s'adresse principalement aux :

* **Développeurs WordPress** qui souhaitent moderniser leurs pratiques de déploiement et de gestion.
* **Administrateurs Système et Ingénieurs DevOps** chargés de la maintenance et de la fiabilité d'un parc de sites WordPress.
* **Responsables de la Sécurité des Systèmes d'Information (RSSI)** cherchant à mettre en place des contrôles de sécurité automatisés et traçables.
* **Architectes de Solutions** qui conçoivent des infrastructures web robustes et évolutives.

Une connaissance de base de Git, de la ligne de commande Linux, de WordPress et des concepts de CI/CD est recommandée pour tirer le meilleur parti de ce guide.

# Chapitre 1 : Philosophie et Principes Clés

## 1.1. L'Infrastructure-as-Code (IaC)

L'Infrastructure-as-Code est le principe fondateur de ce projet. Au lieu de configurer manuellement un serveur ou un pipeline, nous décrivons l'état désiré de notre infrastructure dans des fichiers de code (ici, des fichiers YAML pour les workflows GitHub et des fichiers .ini pour la configuration). Cette approche offre des avantages considérables. Premièrement, elle rend l'infrastructure **reproductible** ; on peut recréer un environnement identique à tout moment. Deuxièmement, elle est **versionnable** ; chaque changement est tracé dans l'historique Git, permettant de savoir qui a changé quoi, quand, et pourquoi. Enfin, elle facilite la **collaboration** en équipe, car les fichiers de configuration peuvent être relus, validés et modifiés par plusieurs personnes, suivant les mêmes processus de revue de code (Pull Requests) que le code applicatif.

## 1.2. La CI/CD : Intégration et Déploiement Continus

La CI/CD est la mise en pratique de l'IaC pour le cycle de vie applicatif.

* **L'Intégration Continue (CI)** est illustrée par le job build-test dans ci-cd.yml. À chaque modification du code poussée sur le dépôt, un processus automatisé se déclenche pour construire, tester et valider le code. Cela permet de détecter les régressions et les erreurs au plus tôt, lorsque leur correction est la moins coûteuse. Dans notre cas, cela inclut l'installation des dépendances et l'exécution de scans de sécurité.
* **Le Déploiement Continu (CD)** est la suite logique. Une fois que l'étape de CI est passée avec succès, le code est automatiquement déployé sur un environnement (ici, la production). Le job deploy-prod est l'incarnation de ce principe. L'automatisation du déploiement garantit un processus rapide, fiable et standardisé, loin des déploiements manuels via FTP, sources d'erreurs et de stress.

## 1.3. La Sécurité "Shift-Left"

Le **"Shift-Left"** (décalage à gauche) est une approche de sécurité qui vise à intégrer les contrôles de sécurité le plus tôt possible dans le cycle de vie du développement (le plus "à gauche" sur une frise chronologique). Le pipeline ci-cd.yml est un exemple parfait de ce principe. En exécutant un scan de sécurité (Security scan (local quick)) à chaque push, nous ne nous contentons pas d'attendre un audit annuel pour trouver des failles. Nous vérifions chaque contribution de code pour des problèmes potentiels. Cette pratique réduit drastiquement la probabilité d'introduire des vulnérabilités en production. Elle responsabilise également les développeurs en leur donnant un retour immédiat sur l'impact de leur code en matière de sécurité, favorisant ainsi une culture de la sécurité au sein de l'équipe.

#### 1.4. La Résilience par l'Automatisation

La résilience est la capacité d'un système à continuer de fonctionner malgré une panne ou une attaque, et à revenir à son état normal rapidement. Dans cette architecture, la résilience est atteinte non pas en essayant de construire un serveur infaillible, mais en créant un système où la **panne est une éventualité prévue**. Les workflows backup-monitor.yml et pentest.yml sont les piliers de cette résilience. Les sauvegardes régulières, automatisées et stockées à l'extérieur garantissent que les données ne seront jamais perdues. La surveillance continue de la disponibilité et des vulnérabilités permet de détecter les incidents dès leur apparition. Le plan de restauration, bien que manuel dans sa version de base, est clairement défini, transformant une catastrophe potentielle en une procédure maîtrisée.

# Chapitre 2 : Vue d'Ensemble de l'Architecture

## 2.1. Les Acteurs Principaux

L'architecture repose sur l'interaction de plusieurs services et composants distincts, chacun avec un rôle bien défini, ce qui garantit une séparation des préoccupations et une meilleure maintenabilité.

* **Dépôt GitHub :** C'est le cerveau et la source de vérité unique du projet. Il héberge non seulement le code du thème ou des plugins WordPress (wp-content), mais aussi toute la logique d'automatisation (workflows dans .github/workflows), les scripts de support (scripts/), et les configurations (config/).
* **GitHub Actions Runners :** Ce sont les ouvriers. Ce sont des machines virtuelles éphémères fournies par GitHub qui exécutent les instructions définies dans les fichiers YAML. Elles clonent le dépôt, installent les dépendances, se connectent au serveur, et exécutent les scripts Python.
* **Serveur de Production :** La machine (VPS, dédiée, ou cloud) qui héberge le site WordPress public. Il est configuré pour accepter des connexions SSH sécurisées depuis les runners de GitHub Actions et doit disposer des outils nécessaires comme wp-cli, rsync et lz4.
* **Amazon S3 (Simple Storage Service) :** Le coffre-fort externe. Ce service de stockage d'objets est utilisé pour conserver les sauvegardes chiffrées de la base de données. Son utilisation découple totalement les sauvegardes de l'infrastructure de production, une pratique essentielle pour une reprise d'activité saine.
* **Slack :** Le centre de notifications. Toutes les alertes critiques (site inaccessible, vulnérabilité détectée, déploiement réussi/échoué) sont envoyées sur un canal Slack dédié, permettant une réactivité immédiate de l'équipe.

#### 2.2. Schéma du Flux de Déploiement (CI/CD)

[Image d'un diagramme de flux montrant les étapes du pipeline ci-cd.yml. De gauche à droite : un développeur pousse du code vers GitHub. Une flèche pointe vers une boîte "GitHub Actions" qui exécute le job "build-test". À l'intérieur de cette boîte, des étapes : "Checkout", "Install Deps", "Run Tests", "Security Scan". Si succès, une flèche pointe vers le job "deploy-prod". Ce job contient les étapes "Checkout", "Install Deps", "SSH Agent", "Rsync Files", "Remote Checks". Une flèche sort de ce job vers une icône de serveur web "Serveur de Production". Une dernière flèche part du job vers une icône Slack "Notification".]

Ce schéma illustre le parcours d'une modification de code depuis le poste du développeur jusqu'à sa mise en production. La première phase (build-test) agit comme un sas de validation : le code est-il syntaxiquement correct ? Les dépendances s'installent-elles ? Passe-t-il les tests de base et de sécurité ? Ce n'est que si toutes ces portes sont vertes que la seconde phase (deploy-prod) est autorisée à s'exécuter. Cette phase est l'opération de "chirurgie" sur le serveur de production. Elle utilise rsync pour une synchronisation efficace des fichiers, puis exécute des commandes de post-déploiement via wp-cli pour s'assurer que le site est dans un état sain après la mise à jour. Le tout se termine par une notification, fermant la boucle de communication.

#### 2.3. Schéma du Flux de Sauvegarde et Surveillance

[Image d'un diagramme montrant deux flux parallèles déclenchés par une horloge "Scheduler". Le premier flux, "Sauvegarde", part de l'horloge vers une boîte "GitHub Actions - Job: backup". Une flèche part de cette boîte vers le "Serveur de Production" avec la mention "wp db export via SSH". Une flèche de retour ramène le dump SQL vers la boîte "GitHub Actions". À l'intérieur, des étapes "Compress (lz4)", "Encrypt (Fernet)". Puis une flèche sort vers un logo Amazon S3 avec la mention "Upload to S3". Le second flux, "Surveillance", part de l'horloge vers une boîte "GitHub Actions - Job: monitor". Une flèche part de cette boîte vers le "Serveur de Production" avec la mention "Uptime Check". Une autre flèche, étiquetée "WPScan", pointe également vers le serveur. En cas de problème, une flèche part de la boîte vers une icône Slack "Alert".]

Ce diagramme illustre les routines de fond qui assurent la santé et la sécurité du site. Contrairement au flux de déploiement qui est déclenché par une action humaine (un push), ces flux sont déclenchés par le temps. Le flux de sauvegarde est une opération d'extraction : il tire les données précieuses (la base de données) du serveur de production, les conditionne (compression, chiffrement) et les met en sécurité dans un lieu externe et sûr (S3). Le flux de surveillance, lui, est une opération d'interrogation. Il pose des questions au site : "Es-tu en ligne ?", "Es-tu performant ?", "As-tu des vulnérabilités connues ?". Les réponses à ces questions déterminent si tout est normal ou si une alerte doit être levée pour attirer l'attention de l'équipe.

# PARTIE A: LA STRATEGIE DE SECURISATION EN PROFONDEUR

**C**ette partie dissèque les mécanismes concrets mis en place pour sécuriser le site WordPress à plusieurs niveaux. Nous allons analyser chaque workflow et chaque script pertinent pour comprendre comment ils contribuent à une posture de sécurité robuste et proactive.

# CHAPITRE 3 : WORKFLOW DE SECURITE PROACTIVE (CI-CD.YML)

Ce workflow est la première ligne de défense du système. Il s'exécute à chaque modification du code, garantissant qu'aucun changement ne puisse compromettre la stabilité ou la sécurité de l'application sans être détecté.

#### 3.1. Analyse du Déclencheur (on:)

IMAGES

Le bloc on: définit les événements qui déclenchent l'exécution du workflow. Ici, il est configuré pour deux scénarios critiques. Le premier, push: branches: [ "main" ], signifie que le workflow s'exécutera à chaque fois qu'un nouveau commit est poussé sur la branche main. C'est le déclencheur principal pour le déploiement en production. Le second, pull\_request: branches: [ "main" ], est tout aussi important : il exécute le workflow (en particulier le job build-test) lorsqu'une Pull Request est ouverte ou mise à jour vers la branche main. Cela permet de valider les changements proposés par un développeur *avant* qu'ils ne soient fusionnés, empêchant ainsi l'introduction de code défectueux ou dangereux dans la branche principale.

## 3.2. Le Job build-test : Préparation et Validation

IMAGES

Un workflow est composé d'un ou plusieurs jobs. Le job build-test est le premier de notre pipeline. Il s'exécute sur un runner ubuntu-24.04, une machine virtuelle Linux fraîchement provisionnée par GitHub. Son rôle est de préparer l'environnement, d'installer les dépendances, et d'exécuter une série de vérifications. La ligne outputs est particulièrement intéressante : elle permet à ce job de communiquer son résultat à d'autres jobs. Ici, nous exportons le résultat (outcome) de l'étape nommée tests. Le job deploy-prod pourra ainsi conditionner son exécution à la réussite de cette étape, créant une dépendance logique et sécurisée entre la validation et le déploiement.

## *3.3. Étape : actions/checkout@v4*

IMAGES

Chaque job s'exécute dans un environnement vierge. La toute première étape est donc presque toujours de récupérer le code source du dépôt. L'action actions/checkout@v4 est une action officielle maintenue par GitHub qui fait exactement cela. Elle clone le dépôt dans l'espace de travail du runner, rendant ainsi tous les fichiers (workflows, scripts, code WordPress) accessibles pour les étapes suivantes du job. C'est le point de départ indispensable de tout pipeline CI/CD. Sans cette étape, le runner n'aurait aucune connaissance du code qu'il est censé tester ou déployer.

#### 3.4. Étape : actions/setup-python@v4

IMAGES

Nos scripts d'automatisation étant écrits en Python, nous devons nous assurer que l'interpréteur Python est disponible sur le runner. L'action actions/setup-python@v4 s'en charge. Elle installe la version de Python spécifiée dans la variable d'environnement PYTHON\_VERSION (ici, "3.12"). L'option cache: "pip" est une optimisation cruciale : elle met en cache les paquets Python téléchargés entre les exécutions. Lors des exécutions suivantes, si le fichier requirements.txt n'a pas changé, les dépendances seront restaurées depuis le cache au lieu d'être téléchargées à nouveau, ce qui accélère considérablement l'exécution du workflow et réduit la consommation de ressources.

#### 3.5. Étape : Install system deps

IMAGES

Nos scripts Python s'appuient sur des outils système qui ne sont pas forcément installés par défaut sur le runner Ubuntu. Cette étape utilise le gestionnaire de paquets apt-get pour les installer.

* rsync est l'outil utilisé pour la synchronisation intelligente et rapide des fichiers vers le serveur de production.
* lz4 est le compresseur utilisé pour réduire la taille des sauvegardes de la base de données. Il est nécessaire sur le runner pour pouvoir compresser le dump SQL avant de l'envoyer sur S3.
* openssh-client fournit les commandes ssh et scp, qui sont les fondations de toute communication sécurisée avec le serveur distant.

#### 3.6. Étape : Run unit / smoke tests

IMAGES

C'est le cœur de la phase de validation. Cette étape est conçue pour être extensible. Dans sa forme actuelle, elle vérifie la présence d'un fichier de configuration pytest.ini ou d'un répertoire tests. Si c'est le cas, elle lance la suite de tests unitaires avec pytest. Les tests unitaires sont essentiels pour vérifier que les fonctions individuelles du code (par exemple, dans les scripts Python ou les thèmes PHP) se comportent comme prévu. L'identifiant id: tests est crucial, car il permet à la sortie de cette étape (outcome) d'être référencée plus tard, notamment dans la section outputs du job, pour conditionner la suite du pipeline.

#### 3.7. Étape : Security scan (local quick)

IMAGES

Cette étape incarne le principe de "Shift-Left Security". Elle exécute le script security\_monitor.py avec l'option --scan-quick. Bien qu'il s'agisse ici d'un placeholder, cette étape est destinée à exécuter des scans rapides qui ne nécessitent pas un site en ligne. On peut y intégrer des outils comme bandit pour scanner le code Python à la recherche de vulnérabilités communes, ou phpcs avec des règles de sécurité pour le code WordPress. L'idée est de fournir un retour d'information quasi instantané au développeur sur la qualité de son code en matière de sécurité, directement dans la Pull Request, bien avant que le code n'atteigne un environnement de production.

### Chapitre 4 : Workflow d'Audit Planifié (pentest.yml)

Si le ci-cd.yml est la sécurité au quotidien, le pentest.yml est l'audit de sécurité périodique. Il ne se déclenche pas sur des actions de code, mais sur une base temporelle, assurant une surveillance continue de l'état de sécurité du site en production.

#### 4.1. Le Planificateur cron

IMAGES

Le déclencheur schedule utilise la syntaxe cron, un standard dans le monde Unix pour la planification de tâches. La chaîne '0 3 \* \* 1' se décompose comme suit : à la minute 0, à l'heure 3, n'importe quel jour du mois, n'importe quel mois, le jour de la semaine 1 (Lundi). Concrètement, cela signifie que chaque lundi matin à 3h00 UTC, lorsque le trafic sur le site est probablement au plus bas, GitHub Actions déclenchera automatiquement ce workflow. Cette planification garantit que le site est audité de manière régulière et prévisible, sans aucune intervention humaine, pour détecter les nouvelles vulnérabilités qui pourraient apparaître suite à la découverte de failles dans WordPress ou ses extensions.

#### 4.2. Le Déclencheur Manuel workflow\_dispatch

IMAGES

En plus de la planification automatique, l'ajout de workflow\_dispatch: {} est une pratique extrêmement utile. Cela fait apparaître un bouton "Run workflow" dans l'onglet "Actions" du dépôt GitHub. Un administrateur peut ainsi déclencher manuellement un scan complet à tout moment. C'est particulièrement utile juste après une mise à jour manuelle critique, après avoir eu connaissance d'une nouvelle faille de sécurité majeure (Zero-Day), ou simplement pour effectuer une vérification ponctuelle à la demande d'une équipe de sécurité. Cela combine la rigueur de l'automatisation avec la flexibilité requise pour la gestion d'incidents.

#### 4.3. Le Job wpscan et son Objectif

IMAGES

Ce workflow ne contient qu'un seul job, wpscan, dont l'unique but est de lancer un audit de vulnérabilités sur le site en production. Il utilise le script security\_monitor.py avec le flag --wpscan. Notez l'utilisation intensive des secrets : le WPSCAN\_API pour bénéficier de la base de données de vulnérabilités complète, le SITE\_URL pour indiquer la cible du scan, et le SLACK\_WEBHOOK pour pouvoir envoyer une alerte en cas de découverte critique. Ce job ne déploie rien, il ne modifie rien. C'est une opération en "lecture seule" sur le site public, dont le but est de produire un rapport et, si nécessaire, de tirer la sonnette d'alarme.