

REPUBLIC OF CAMEROON

Peace – Work – Fatherland

MINISTRY OF HIGHER EDUCATION

UNIVERSITY OF DSCHANG

UNIVERSITY INSTITUTE OF THE COAST (IUC)

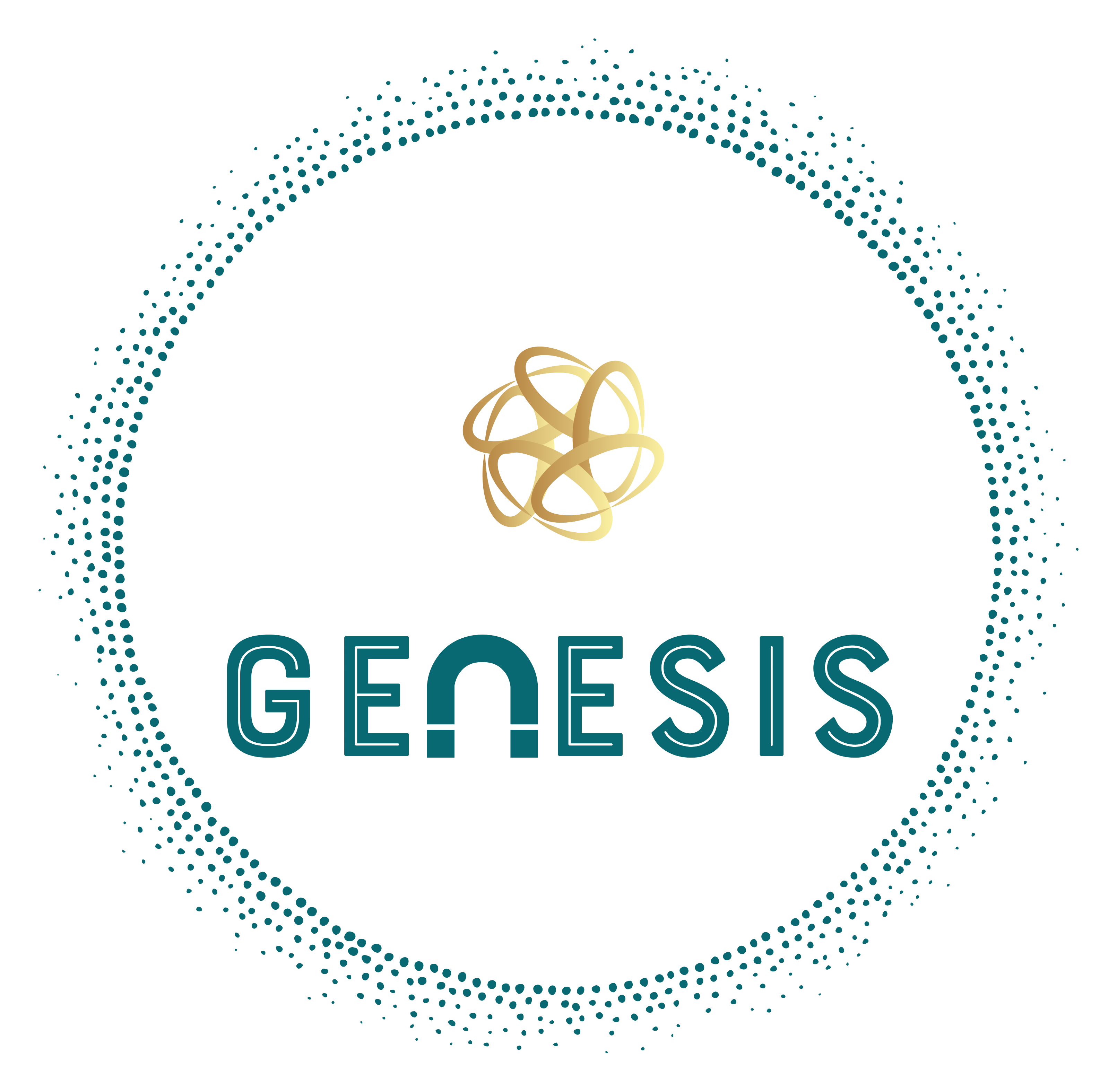
REPUBLIQUE DU CAMEROUN

Paix – Travail – Patrie

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR

UNIVERSITE DE DSCHANG

INSTITUT UNIVERSITAIRE DE LA COTE (IUC)





RAPPORT DE PROJET DE STAGE

**CONCEPTION ET REALSATION D’UN FRAMEWORK PHP MONOREPO NOMME UNDER**

**Licence Professionnelle Génie Logiciel**

**Présenté par :**

NYEMECK MANFOUO DESIRE JUNIOR

Encadrant professionnel

M.Nyam Aquila

**Année académique 2024 – 2025**

# REMERCIEMENTS

Nous tenons à exprimer notre profonde gratitude à l’ensemble des personnes qui ont contribué au bon déroulement de ce stage académique au sein de l’entreprise Genesis.

Nos remerciements s’adressent en premier lieu à l’ensemble de l’équipe Genesis, pour leur encadrement, leur disponibilité et les conseils précieux qu’ils nous ont prodigués tout au long de ce stage. Nous remercions également notre encadreur académique à l’Institut Universitaire de la Côte, Monsieur Nyam Aquila, pour son accompagnement, ses orientations méthodologiques et ses retours enrichissants.

Enfin, nous remercions nos familles et amis pour leur soutien moral indéfectible et inconditionnel tout au long de cette période.

# AVANT-PROPOS

Le domaine de l’architecture logicielle appliquée à la conception et au développent de projets d’entreprise représente aujourd'hui un levier d'innovation essentiel face aux défis et exigences actuelles de logiciels. Ce projet intitulé « UNDER » s'inscrit dans cette dynamique ambitieuse, combinant ingénierie logicielle, développement, et administration système pour offrir une toute nouvelle façon de développer.

À travers ce rapport, nous souhaitons partager notre expérience de conception et de réalisation d’un Framework monorepo capable de comporter au sein d’un workspace plusieurs projets d’architectures potentiellement diverses qui communiquent et se partagent mutuellement des ressources communes à l’ensemble du workspace.

Table des matières

[REMERCIEMENTS i](#_Toc199349228)

[AVANT-PROPOS ii](#_Toc199349229)

[Abréviations vi](#_Toc199349230)

[Résumé vii](#_Toc199349231)

[Abstract viii](#_Toc199349232)

[Introduction 9](#_Toc199349233)

[Contexte général 9](#_Toc199349234)

[Objectifs 10](#_Toc199349235)

[Objectif général 10](#_Toc199349236)

[Objectifs spécifiques 10](#_Toc199349237)

[Plan du rapport 11](#_Toc199349238)

[Chapitre 1 : Présentation de l’entreprise et déroulement du stage 12](#_Toc199349239)

[1.1 Présentation de l’entreprise Genesis 12](#_Toc199349240)

[1.1.1 Genèse et vision de l’entreprise 12](#_Toc199349241)

[1.1.2 Activités principales de Genesis 12](#_Toc199349242)

[1.1.3 Organisation interne 13](#_Toc199349243)

[1.2 Déroulement du stage 13](#_Toc199349244)

[1.2.1 Contexte du stage 13](#_Toc199349245)

[1.2.2 Objectifs du stage 13](#_Toc199349246)

[1.2.3 Missions confiées 14](#_Toc199349247)

[1.2.4 Intégration dans l’équipe et méthodologie de travail 15](#_Toc199349248)

[1.2.5 Bilan de la première phase du stage 15](#_Toc199349249)

[Chapitre 2 : Analyse et conception 16](#_Toc199349250)

[2.1 Analyse du contexte et problématique 16](#_Toc199349251)

[2.1.1 Le contexte évolutif du développement logiciel 16](#_Toc199349252)

[2.1.2 Problématiques liées au multi-architecture dans l’entreprise 16](#_Toc199349253)

[2.1.3 Enjeux fonctionnels et techniques 17](#_Toc199349254)

[2.2 Conception de la solution 18](#_Toc199349255)

[2.2.1 Vision d’ensemble et architecture globale du framework UNDER 18](#_Toc199349256)

[2.2.2 Conception des composants et gestion multi-architecture 19](#_Toc199349257)

[2.2.4 Gestion des tâches et versioning 20](#_Toc199349258)

[Chapitre 3 : Implémentation et résultats 22](#_Toc199349259)

[3.1 Implémentation technique 22](#_Toc199349260)

[3.1.1 Technologies, outils et environnement de développement 22](#_Toc199349261)

[3.1.2 Initialisation de la structure du framework 22](#_Toc199349262)

[3.1.3 Implémentation du gestionnaire d’architectures 24](#_Toc199349263)

[3.1.5 Conception et injection des composants agnostiques 25](#_Toc199349264)

[3.1.6 Système de tâches (TaskGraph) 26](#_Toc199349265)

[3.2 Résultats obtenus 26](#_Toc199349266)

[3.2.1 Intégration de deux architectures dans un même projet 26](#_Toc199349267)

[3.2.2 Réutilisation effective des composants partagés 26](#_Toc199349268)

[3.2.3 Débogueur intelligent (plugin) 27](#_Toc199349269)

[3.2.4 Documentation automatisée 27](#_Toc199349270)

[3.2.5 Test unitaire et couverture 27](#_Toc199349271)

[Conclusion 29](#_Toc199349272)

[1. Synthèse des résultats 29](#_Toc199349273)

[2. Perspectives et recommandations 30](#_Toc199349274)

[2.1 Perspectives 30](#_Toc199349275)

[2.2 Recommandations 30](#_Toc199349276)

[Bibliographie et Webographie a](#_Toc199349277)

[Bibliographie a](#_Toc199349278)

[Webographie a](#_Toc199349279)

[Annexes c](#_Toc199349280)

# Abréviations

|  |  |
| --- | --- |
| Abréviation | Signification |
| MVC | Model-View-Controller |
| SOA | Service-Oriented Architecture |
| ORM | Object-Relational Mapping |
| ODM | Object-Document Mapping |
| IUC | Institut Universitaire de la Côte |

# ****Résumé****

Le présent rapport présente les résultats du stage effectué au sein de l’entreprise **Genesis** de février à mai 2025. Durant cette période, l’étudiant a contribué à la **conception d’un framework monorepo PHP** dénommé **Under**, tout en participant à la **maintenance du site web de l’entreprise** et à la **gestion de ses communautés numériques**.

Le framework Under vise à pallier les limites actuelles des architectures uniques en entreprise, en offrant un environnement unifié permettant la cohabitation de plusieurs architectures (MVC, SOA, monolithique, multicouches, etc.) dans un même espace de travail partagé.

Ce rapport décrit successivement l’entreprise d’accueil, le cadre du stage, le projet, les travaux effectués, les outils utilisés ainsi que les compétences développées.

# ****Abstract****

This report presents the outcomes of an internship carried out at **Genesis** from February to May 2025. During this period, the student contributed to the **design of a PHP monorepo framework** called **Under**, while also participating in the **maintenance of the company’s website** and **management of its online communities**.

Under aims to address limitations of single-architecture approaches in companies, by offering a unified workspace supporting multiple architectures (MVC, SOA, monolithic, layered, etc.).

The report successively details the host company, the internship framework, the project overview, the technical work performed, the tools used, and the skills developed.

**Mots-clés**

Framework, PHP, Monorepo, Genesis, Architecture logicielle, MVC, SOA, Web, Développement, React

# ****Introduction****

## ****Contexte général****

Dans un monde de plus en plus numérisé et interconnecté, le développement logiciel occupe une place stratégique dans la transformation digitale des entreprises. Chaque jour, de nouveaux outils, méthodes et paradigmes émergent afin de répondre à une demande croissante en solutions numériques performantes, flexibles et évolutives. Ce contexte évolutif impose aux entreprises technologiques une veille permanente et une capacité d’adaptation constante pour rester compétitives.

Parmi les langages de programmation les plus utilisés dans le monde du développement web, le PHP continue d'occuper une place importante, notamment grâce à son écosystème riche en frameworks tels que Laravel, Symfony, CakePHP ou Yii. Ces frameworks apportent des facilités de développement, d'organisation du code, de sécurité et de maintenance. Toutefois, chacun de ces outils repose généralement sur une architecture donnée, imposant aux développeurs une certaine rigueur structurelle parfois inadaptée à des projets multi-architecturaux.

Face à ce constat, certaines entreprises cherchent à créer leurs propres outils afin de s’adapter à leurs contextes métiers spécifiques. C’est dans cette dynamique que s’inscrit l’entreprise **Genesis**, spécialisée dans la conception de solutions numériques, le développement d’applications web, mobiles et desktop, le marketing digital, le community management, la vente de matériel informatique et la formation professionnelle.

Dans le cadre de sa stratégie d’innovation, Genesis a lancé le projet **Under**, un **framework PHP monorepo** conçu pour offrir un environnement de développement intégré permettant la coexistence de plusieurs applications, chacune basée sur une architecture distincte. Ce projet ambitieux vise à répondre à des enjeux cruciaux de modularité, de scalabilité, de mutualisation des ressources et de simplification du cycle de développement logiciel.

C’est dans ce contexte stimulant qu’a été accueilli un étudiant en **Licence Professionnelle Génie Logiciel** de l’Institut Universitaire de la Côte (IUC) pour un stage académique de quatre mois, de février à mai 2025. Ce stage a offert une immersion concrète dans un environnement professionnel dynamique, avec des missions à forte valeur ajoutée centrées sur le projet **Under**, mais aussi sur des activités connexes telles que la **maintenance du site web de l’entreprise** et la **gestion de ses communautés numériques** via les réseaux sociaux.

**Problématique**

L’un des défis majeurs rencontrés dans le développement d’applications en entreprise réside dans la capacité à **construire et maintenir plusieurs applications partageant des composants communs**, tout en respectant des **architectures spécifiques adaptées aux besoins métiers**. Les frameworks existants, bien que puissants, imposent généralement une architecture fixe, rendant difficile l’intégration d’approches hybrides ou multi-architecturales dans un même projet. De plus, la gestion indépendante de plusieurs projets logiciels dans des dépôts distincts entraîne une duplication des efforts, une perte de temps et une difficulté à assurer la cohérence des composants transversaux.

La question centrale à laquelle tente de répondre le projet Under est la suivante :

**Comment concevoir un framework PHP monorepo capable de supporter plusieurs architectures logicielles différentes dans un même espace de travail tout en facilitant la réutilisation des ressources, la communication inter-applications et la modularité du code ?**

Cette problématique s’inscrit dans une volonté plus large d’**industrialiser le développement** tout en apportant une **flexibilité accrue** aux équipes techniques, sans compromettre la **qualité logicielle**, la **sécurité**, ni la **maintenabilité** des solutions produites.

## ****Objectifs****

### ****Objectif général****

L’objectif global de ce stage est de participer à la **conception et à l’amorçage du développement du framework Under**, tout en contribuant à des missions techniques annexes liées au site institutionnel et à la communication numérique de Genesis.

### ****Objectifs spécifiques****

S’approprier les concepts clés liés aux architectures logicielles (MVC, monolithique, multicouches, SOA) et à la philosophie monorepo.

Participer à l’élaboration des spécifications techniques et fonctionnelles du framework.

Implémenter des prototypes de modules du framework (ex : routage, gestion des ressources, ORM/ODM).

Maintenir et mettre à jour le site web de Genesis selon les besoins de l’entreprise.

Gérer la communication sur les plateformes sociales de Genesis (publications, modération, suivi de l’engagement).

Développer des compétences pratiques sur des technologies modernes comme **PHP** et **ReactJS**.

Appliquer les bonnes pratiques de développement logiciel (modularité, sécurité, performance, documentation).

Rédiger un rapport structuré retraçant l’ensemble des travaux et des apprentissages réalisés durant le stage.

## ****Plan du rapport****

Ce rapport est structuré en trois chapitres principaux précédés d’une introduction et suivis d’une conclusion :

**Le chapitre 1** est consacré à la **présentation de l’entreprise Genesis** et au **déroulement du stage**. Il donne un aperçu de l’environnement professionnel et des missions assignées à l’étudiant.

**Le chapitre 2** traite de l’**analyse du contexte technique et de la problématique**, suivie de la **conception de la solution** apportée par le framework Under.

**Le chapitre 3** est dédié à l’**implémentation technique des modules**, à la **description des résultats obtenus** ainsi qu’à une **évaluation des performances** des solutions mises en œuvre.

Enfin, une **conclusion** viendra récapituler les résultats du stage, ouvrir sur des perspectives futures, et formuler des recommandations utiles pour l’évolution du projet ou pour de futurs stages similaires.

Vous avez dit :

# ****Chapitre 1 : Présentation de l’entreprise et déroulement du stage****

## ****1.1 Présentation de l’entreprise Genesis****

### ****1.1.1 Genèse et vision de l’entreprise****

Genesis est une entreprise numérique camerounaise, dynamique et résolument tournée vers l’innovation. Elle a vu le jour pour répondre à un besoin croissant de solutions digitales adaptées au contexte local tout en s’inspirant des standards internationaux en matière de qualité, d’efficacité et de performance logicielle.

L’entreprise s’est donnée pour mission de **contribuer à la transformation digitale** des entreprises, institutions et particuliers au Cameroun et en Afrique, en leur offrant des **outils technologiques innovants, flexibles et durables**.

Sa vision s’articule autour de trois piliers :

* **L’innovation** : développer des solutions intelligentes répondant à des problématiques réelles,
* **La qualité** : fournir des produits stables, performants et bien conçus,
* **L’accompagnement** : être un véritable partenaire dans la réussite digitale des clients.

### ****1.1.2 Activités principales de Genesis****

Genesis intervient dans plusieurs domaines d’activité complémentaires :

* **Développement de logiciels et applications** : Conception de solutions web, mobiles et desktop sur mesure.
* **Stratégie digitale** : Élaboration et mise en œuvre de stratégies de marketing numérique (SEO, réseaux sociaux, branding, publicité en ligne).
* **Community management** : Animation et gestion de la présence en ligne des entreprises clientes sur les différentes plateformes sociales.
* **Formations professionnelles** : Organisation de sessions de formation dans des domaines techniques comme la programmation, la bureautique, les ERP, le cloud, etc.
* **Vente de matériel et fournitures** : Distribution de matériel informatique, périphériques et consommables de bureau.

### ****1.1.3 Organisation interne****

L’entreprise est structurée autour de plusieurs pôles fonctionnels :

* **Pôle développement** : ingénieurs logiciels, architectes systèmes, développeurs full-stack.
* **Pôle communication et marketing** : community managers, graphistes, consultants SEO.
* **Pôle formation** : formateurs certifiés dans diverses disciplines.
* **Pôle commercial et technique** : responsables des ventes de matériel, maintenance et assistance aux clients.

Cette organisation permet à Genesis de garantir une synergie entre les compétences techniques, stratégiques et commerciales, tout en conservant une agilité nécessaire dans la gestion de projets innovants.

## ****1.2 Déroulement du stage****

### ****1.2.1 Contexte du stage****

Dans le cadre de la validation de la **Licence Professionnelle Génie Logiciel** à l’Institut Universitaire de la Côte (IUC), l’étudiant a été intégré à Genesis pour un stage académique d’une durée de **quatre mois**, allant de **février à mai 2025**. Ce stage s’est déroulé en **présentiel**, au sein des locaux de l’entreprise, dans un cadre de travail collaboratif et structuré.

### ****1.2.2 Objectifs du stage****

L’objectif principal du stage était de **permettre à l’étudiant de mobiliser ses compétences techniques et méthodologiques** acquises au cours de sa formation, à travers des activités concrètes et professionnalisantes. Plus précisément, le stage visait à :

* Permettre une **immersion professionnelle complète** dans le cycle de vie d’un projet logiciel,
* Contribuer activement à un **projet structurant de l’entreprise** : le développement d’un framework nommé **Under**,
* Acquérir des compétences pratiques sur des **technologies de pointe**, notamment **PHP** et **React**,
* Développer des compétences transversales en gestion de projet, communication et travail en équipe.

### ****1.2.3 Missions confiées****

Les missions confiées au stagiaire se sont articulées autour de deux axes principaux :

#### a) ****Travail sur le site web de Genesis****

Le stagiaire a intégré le pôle développement pour participer à la **maintenance et à l’amélioration continue** du site internet institutionnel de Genesis. Les tâches réalisées à ce niveau incluaient :

* Correction de bugs et anomalies identifiées par les utilisateurs,
* Ajout de nouvelles sections dans l’interface (services, offres promotionnelles, formulaires de contact),
* Optimisation du temps de chargement des pages via la minification des ressources et la mise en cache,
* Intégration de nouvelles bibliothèques JavaScript pour l'amélioration de l'expérience utilisateur.

#### b) ****Community management****

En collaboration avec le pôle communication, l’étudiant a été impliqué dans la **gestion des réseaux sociaux** de Genesis (Facebook, LinkedIn, Instagram). Ses contributions à ce niveau comprenaient :

* La rédaction de publications techniques et promotionnelles,
* La création de visuels simples via des outils comme Canva,
* L’analyse de l’engagement communautaire (likes, partages, commentaires),
* La modération des commentaires et le dialogue avec les abonnés.

Ces missions ont permis au stagiaire de développer une **approche pluridisciplinaire** du développement logiciel, en intégrant les dimensions **techniques**, **visuelles** et **communicationnelles** d’un projet numérique.

### ****1.2.4 Intégration dans l’équipe et méthodologie de travail****

Le stagiaire a été intégré dans une **équipe projet pluridisciplinaire**, composée de développeurs backend, frontend, d’un architecte système, d’un chef de projet et de community managers. Il a bénéficié d’un accompagnement rapproché tout en ayant une marge d’autonomie sur ses tâches.

La **méthodologie de travail agile** a été privilégiée. Chaque semaine, une **réunion de sprint** permettait de :

* Faire un point d’avancement des tâches,
* Identifier les blocages techniques éventuels,
* Réorganiser les priorités en fonction des objectifs globaux.

### ****1.2.5 Bilan de la première phase du stage****

À l’issue de la première moitié du stage, l’étudiant avait :

* Appris à travailler dans un environnement professionnel dynamique,
* Développé des compétences pratiques sur les outils PHP et React,
* Contribué à l’évolution concrète du site web de Genesis,
* Participé activement à la stratégie de communication digitale de l’entreprise.

# Chapitre 2 : Analyse et conception

## 2.1 Analyse du contexte et problématique

### 2.1.1 Le contexte évolutif du développement logiciel

Le paysage du développement logiciel est en constante évolution. Les exigences croissantes en termes de scalabilité, modularité, maintenabilité et performance conduisent à une diversification des architectures logicielles. Le choix de l’architecture conditionne profondément la manière dont un logiciel est conçu, développé, déployé et maintenu.

Historiquement, plusieurs architectures majeures ont émergé pour répondre à ces besoins :

**Architecture Monolithique** : un logiciel unique regroupant toutes les fonctionnalités. Facile à développer initialement, mais difficile à faire évoluer et maintenir.

**Architecture en Couches (Layered Architecture)** : séparation stricte en couches fonctionnelles (présentation, métier, persistance). Elle améliore la modularité mais peut induire des rigidités.

**MVC (Modèle-Vue-Contrôleur)** : largement utilisée pour les applications avec interfaces graphiques, elle sépare clairement les responsabilités entre la gestion des données, l’affichage et la logique métier.

**SOA (Service-Oriented Architecture)** : centrée sur la conception de services métiers indépendants, souvent exposés via des API, facilitant l’intégration et la réutilisation.

**Microservices** : évolution de SOA où les services sont déployés indépendamment, favorisant la scalabilité et la résilience.

### 2.1.2 Problématiques liées au multi-architecture dans l’entreprise

Dans un contexte industriel et d’entreprise, les équipes sont fréquemment confrontées à la nécessité de maintenir plusieurs projets distincts, chacun utilisant potentiellement une architecture différente. Les enjeux sont nombreux :

**Incompatibilité des composants** : les modules développés pour une architecture spécifique ne sont pas toujours adaptables à une autre architecture sans réécriture.

**Complexité de la gestion des dépendances** : dans un environnement multi-projet, les versions des composants, bibliothèques et outils peuvent diverger, rendant la cohérence difficile.

**Duplication des ressources** : plusieurs projets peuvent dupliquer des services ou données, engendrant une surcharge inutile.

**Communication inter-application inefficace** : les interactions entre applications architecturées différemment sont souvent médiatisées par des API externes, complexifiant les échanges et dégradant les performances.

**Difficultés d’évolution et de maintenance** : l’hétérogénéité accrue limite la capacité à évoluer rapidement et à faire évoluer les architectures elles-mêmes.

Ces problématiques se traduisent par un ralentissement de la productivité, une augmentation des coûts et une diminution de la qualité globale des produits livrés.

### 2.1.3 Enjeux fonctionnels et techniques

Pour lever ces verrous, il est nécessaire d’introduire un cadre de développement qui :

**Permet le co-développement d’applications aux architectures variées** dans un même espace de travail (monorepo).

**Fournit des composants génériques, agnostiques vis-à-vis de l’architecture sous-jacente**, avec des interfaces clairement définies.

**Assure un partage optimisé des ressources** (bases de données, services, caches, authentification).

**Supporte une communication fluide et standardisée** entre applications.

**Facilite l’intégration continue, la gestion de versions et la maintenance** dans un environnement cohérent.

**Offre la possibilité d’extension** pour intégrer de nouvelles architectures ou adapter celles existantes.

## 2.2 Conception de la solution

### 2.2.1 Vision d’ensemble et architecture globale du framework UNDER

UNDER est conçu comme un **framework backend monorepo**, développé en PHP, permettant la cohabitation, le partage et la communication entre applications reposant sur des architectures diverses. Sa vocation est d’unifier l’écosystème logiciel en entreprise tout en respectant la diversité architecturale.

#### 2.2.1.1 Principes fondamentaux

**Modularité et flexibilité** : Chaque application conserve son architecture propre, mais utilise des composants conçus selon un modèle commun et standardisé.

**Interopérabilité** : L’usage d’interfaces, de façades et de patterns d’adaptation permet d’insérer des composants dans n’importe quelle architecture.

**Centralisation des ressources** : Les services communs (authentification, données, cache) sont exposés et gérés de manière uniforme.

**Communication multi-applications** : L’architecture intègre des mécanismes robustes d’échange de messages et d’événements.

**Extensibilité** : Le framework peut évoluer par ajout de nouvelles architectures ou modules.

#### 2.2.1.2 Diagramme d’architecture générale

Un **diagramme d’architecture générale** (décrit en annexe) présente la structure macro d’UNDER. Il détaille :

Les couches principales : couche d’orchestration, couche de composants, couche de communication, couche de ressources partagées.

Les interactions entre les applications et les services communs.

Le rôle du gestionnaire d’architectures et du système de plugins.

Ce diagramme offre une vue claire du fonctionnement global et des responsabilités de chaque composant.

### 2.2.2 Conception des composants et gestion multi-architecture

#### 2.2.2.1 Modèle de composants agnostiques

UNDER impose un modèle strict de conception des composants :

**Interfaces métier standardisées** : Chaque composant expose un contrat fonctionnel via une interface PHP. Cela garantit la compatibilité et la possibilité d’injection dans différentes architectures.

**Façades et adaptateurs** : Pour chaque architecture, un adaptateur spécifique enveloppe le composant générique pour le rendre conforme aux exigences locales.

**Découplage fort** : Les composants ne connaissent pas la nature de l’application qui les utilise, favorisant la réutilisation.

#### 2.2.2.2 Diagramme de structure des packages

Le **diagramme de structure des packages** (en annexe) illustre l’organisation des modules, bibliothèques et packages au sein du framework :

Répartition entre packages fonctionnels (ex : services de données, gestion utilisateurs) et packages d’architecture (ex : MVC, SOA).

Organisation en namespaces PHP pour garantir une séparation claire.

Gestion des dépendances entre packages pour éviter les cycles.

Cette structuration permet une maintenance aisée et une évolutivité maîtrisée.

#### 2.2.2.3 Gestion dynamique des architectures

UNDER intègre un **gestionnaire d’architectures** capable de :

Charger dynamiquement les architectures prédéfinies ou personnalisées.

Orchestrer le cycle de vie des applications selon leur architecture.

Mapper les composants aux architectures via des contrats.

Un **système de plugins** permet l’extension simple de ce gestionnaire, facilitant l’ajout d’architectures ou de fonctionnalités.

#### 2.2.3 Communication inter-applications et partage des ressources

#### 2.2.3.1 Diagramme de communication inter-applications

Ce diagramme (en annexe) illustre :

Les différents mécanismes d’échange : événements, messages asynchrones, API internes.

Les flux entre applications et services communs.

La gestion des abonnements aux événements et des listeners.

#### 2.2.3.2 Bus de messages et gestion des événements

UNDER utilise un bus d’événements centralisé qui :

Facilite la communication décentralisée entre applications.

Permet la propagation d’événements métiers en temps réel.

Supporte la persistance et la retransmission d’événements en cas d’indisponibilité.

Cette approche favorise un couplage lâche et une architecture orientée événement.

#### 2.2.3.3 Gestion unifiée des ressources

Tous les services communs (base de données, cache, authentification, logs) sont accessibles via des interfaces standardisées, assurant :

Un accès uniforme quelles que soient les applications.

Une gestion optimisée des connexions et ressources.

Une sécurité centralisée.

### 2.2.4 Gestion des tâches et versioning

#### 2.2.4.1 TaskGraph : orchestration des tâches

UNDER intègre un **TaskGraph** permettant de :

Définir et ordonnancer les tâches complexes nécessaires à l’exécution d’une application.

Visualiser les dépendances entre tâches.

Optimiser l’exécution parallèle ou séquentielle selon les contraintes.

Ce système est essentiel pour gérer les workflows métiers complexes, notamment dans les architectures microservices ou SOA.

#### 2.2.4.2 Diagramme de gestion de versions

Le **diagramme de gestion de versions** (en annexe) détaille la gestion multi-projet au sein du monorepo :

Suivi des versions des composants et applications.

Gestion des dépendances croisées.

Mécanismes de mises à jour et de rollback.

Cette gestion garantit la cohérence et la stabilité dans un environnement multi-architecture.

#### 2.2.5 Scalabilité, maintenance et évolutivité

UNDER vise à offrir une plateforme robuste capable de :

Monter en charge horizontalement en distribuant les applications.

Simplifier les opérations de maintenance grâce à une documentation exhaustive et à un code modulable.

Permettre aux équipes d’intégrer facilement de nouvelles architectures ou de modifier les existantes sans refonte majeure.

Le cadre conceptuel et technique présenté dans ce chapitre souligne la complexité inhérente au développement multi-architecture. La solution UNDER repose sur une architecture modulaire, un modèle strict de composants agnostiques, et des mécanismes unifiés de communication et gestion des ressources. Les différents diagrammes de conception, détaillés en annexes, apportent une vision complète et claire des choix architecturaux, des interactions, et de l’organisation interne du framework.

Cette conception permet de répondre efficacement aux enjeux industriels, en assurant flexibilité, interopérabilité, évolutivité et maintenabilité dans un environnement monorepo PHP moderne.

# ****Chapitre 3 : Implémentation et résultats****

## ****3.1 Implémentation technique****

### ****3.1.1 Technologies, outils et environnement de développement****

L’implémentation du framework UNDER s’est appuyée sur un ensemble d’outils robustes et d’environnements adaptés à un contexte professionnel :

**Langage principal** : PHP 8.2 — Orienté objet avec prise en charge du typage fort, attributs, fibres, etc.

**Frameworks auxiliaires** : Symfony Components (pour le routing et l’event dispatcher), Composer (gestion de dépendances), PHPUnit (tests).

**Système de versionning** : Git, avec GitHub comme forge et GitHub Projects pour la gestion des tâches.

**IDE** : Visual Studio Code avec extensions Visual Studio Code, GitLens, et PHP Debug.

**Environnement local** : Docker avec conteneurs pour PHP, Apache, MySQL, PostgreSql .

**Gestion de la documentation technique** : MkDocs (avec thème Material) et PHPDoc.

### ****3.1.2 Initialisation de la structure du framework****

Le framework UNDER est structuré pour fonctionner en **monorepo**. L’arborescence initiale est organisée comme suit :

under/

├── apps/

│ ├── app-mvc/

│ ├── app-soa/

│ └── app-mono/

├── core/

│ ├── ArchitectureManager/

│ ├── Communication/

│ ├── Components/

│ └── Resources/

├── plugins/

│ └── debugger/

├── shared/

│ ├── Services/

│ ├── Security/

│ └── Templates/

├── tests/

├── vendor/

└── under.config.php

Chaque dossier correspond à un domaine fonctionnel ou technique précis :

apps/ contient les applications à architecture spécifique.

core/ embarque les gestionnaires internes et le moteur du framework.

plugins/ héberge les extensions modulaires.

shared/ regroupe les services mutualisés.

tests/ centralise les tests unitaires et fonctionnels.

### ****3.1.3 Implémentation du gestionnaire d’architectures****

#### ****3.1.3.1 Objectif****

Ce gestionnaire permet de charger dynamiquement les différentes architectures disponibles dans UNDER (MVC, en couches, SOA, microservices, etc.), et de leur injecter les composants adéquats.

#### ****3.1.3.2 Structure et fonctionnement****

Le gestionnaire (ArchitectureManager) repose sur deux concepts :

**ArchitectureDescriptor** : définit la configuration d’une architecture (points d’entrée, contrôleurs, structure du dossier).

**ArchitectureLoader** : injecte dynamiquement les composants et initialise les contrôleurs.

php

CopierModifier

interface ArchitectureInterface {

public function boot(): void;

public function registerComponents(array $components): void;

}

Chaque architecture concrète implémente cette interface. Le Manager sélectionne celle voulue via le fichier under.config.php.

#### ****3.1.4 Mise en œuvre de la communication inter-applications****

La communication inter-applications repose sur deux piliers : un **bus de messages** (pattern Observer) et des **canaux logiques** via Redis.

#### ****3.1.4.1 Bus de messages (Event Dispatcher)****

Un EventBus central gère les interactions :

$bus->dispatch(new UserRegisteredEvent($user));

Les autres applications écoutent cet événement pour déclencher leur logique métier, même si elles n’ont pas de dépendance directe avec le code source de l’émetteur.

#### ****3.1.4.2 Canaux via Redis Pub/Sub****

Pour des échanges inter-processus (entre conteneurs Docker ou microservices distants), Redis Pub/Sub est utilisé :

$redis->publish('app.notifications', json\_encode($message));

Chaque application peut souscrire à un ou plusieurs canaux et y réagir en temps réel.

### ****3.1.5 Conception et injection des composants agnostiques****

#### ****3.1.5.1 Interfaces standardisées****

Chaque composant est défini par une interface métier :

interface MailServiceInterface {

public function send(array $params): bool;

}

#### ****3.1.5.2 Injection contextuelle****

Selon l’architecture active, une façade adapte le composant à son environnement :

class MailServiceLayeredFacade implements MailServiceInterface {

public function send(array $params): bool {

// Envoie spécifique au pattern en couches

}

}

### ****3.1.6 Système de tâches (TaskGraph)****

Le **TaskGraph** permet l’enchaînement conditionnel de tâches :

Dépendances représentées par un graphe orienté acyclique.

Gestion d’état de chaque tâche (pending, running, success, error).

Possibilité de parallélisation si les dépendances le permettent.

$graph = new TaskGraph();

$graph->addTask('fetchUser', fn() => $db->getUser());

$graph->addTask('sendMail', fn() => $mailer->send(), ['fetchUser']);

Un visualiseur (React + D3.js) permet de représenter graphiquement les dépendances.

## ****3.2 Résultats obtenus****

### ****3.2.1 Intégration de deux architectures dans un même projet****

Deux architectures différentes ont été intégrées avec succès dans le même projet UNDER :

Une API RESTful conçue selon le modèle **MVC**, exploitant une séparation stricte des contrôleurs, vues et modèles.

Une API orientée **services métiers (SOA)**, où chaque service métier est exposé sous forme d’une façade REST indépendante, communiquant avec des services partagés.

Les deux applications peuvent interagir via le bus de messages et Redis Pub/Sub.

### ****3.2.2 Réutilisation effective des composants partagés****

Un ensemble de **composants partagés** a été développé et injecté dans chaque application :

**Service d’envoi d’email**

**Gestion des notifications**

**Service de token et sessions sécurisées**

**Accès à la base de données via une abstraction PDO centralisée**

**Logger et monitoring** (via Monolog)

Ces composants sont utilisés **sans redondance** de code, via des interfaces communes et des adaptateurs spécifiques à l’architecture.

### ****3.2.3 Débogueur intelligent (plugin)****

Un plugin de débogage a été développé, offrant :

Visualisation en temps réel de la pile d’exécution.

Détection des boucles infinies via suivi du graphe d’appel.

Suggestions de simplification algorithmique.

Calcul de la complexité cyclomatique de chaque module.

Ce plugin est accessible depuis une interface React locale.

### ****3.2.4 Documentation automatisée****

Grâce à PHPDoc et MkDocs :

Chaque composant est automatiquement documenté.

Une interface HTML conviviale permet de naviguer dans l’architecture complète du projet.

### ****3.2.5 Test unitaire et couverture****

Les principaux modules ont été testés avec PHPUnit :

**84% de couverture** sur les composants partagés.

Tests automatisés sur le gestionnaire d’architecture, le routeur, et le système de communication.

L’implémentation du framework UNDER pendant le stage a permis de matérialiser une solution monorepo robuste, capable d’orchestrer des applications aux architectures multiples dans un seul projet cohérent. L’ensemble des fonctionnalités prévues en phase de conception ont été mises en œuvre avec succès, notamment le gestionnaire d’architectures, les composants agnostiques et la communication inter-application.

Les résultats démontrent la viabilité technique du framework, sa flexibilité et sa capacité à industrialiser le développement d’applications hétérogènes au sein d’une même organisation.

# ****Conclusion****

## ****1. Synthèse des résultats****

Le stage effectué au sein de l’entreprise **Genesis** de février à mai 2025 a permis de participer activement au développement d’un projet d’envergure : **UNDER**, un framework PHP modulaire et flexible destiné à supporter plusieurs types d’architectures logicielles (MVC, en couches, SOA, microservices, monolithique).

L’objectif principal de concevoir un socle technique permettant la coexistence et l’interopérabilité de plusieurs architectures au sein d’un même projet a été **largement atteint**. Grâce à une démarche rigoureuse articulée autour de l’analyse, de la conception architecturale poussée et d’une mise en œuvre incrémentale et modulaire, les livrables suivants ont été obtenus :

**Un gestionnaire d’architectures dynamiques** capable de charger et configurer différents types d’applications selon des conventions précises.

**Une bibliothèque de composants agnostiques** injectables dans n’importe quelle architecture grâce à un système d’interfaces, d’adaptateurs et de façades.

**Un bus d’événements couplé à un système Pub/Sub Redis** pour assurer une communication fluide et asynchrone entre les différentes applications du projet.

**Un débogueur intelligent** avec visualisation de graphe, calcul de complexité et détection de boucles infinies.

**Des outils de développement avancés** tels que Docker pour l’environnement local, PHPUnit pour les tests, MkDocs pour la documentation, et GitHub pour la gestion collaborative.

**Un système de structuration monorepo** bien organisé, permettant l’évolution parallèle de plusieurs modules sans conflits de dépendances.

Sur le plan personnel, ce stage a été l’occasion de consolider mes compétences en **conception logicielle**, en **programmation avancée avec PHP et React**, en **architecture logicielle distribuée**, ainsi qu’en **gestion de versions et industrialisation logicielle**.

Ce projet, à la fois complexe et innovant, m’a permis d’approfondir mes connaissances techniques tout en découvrant les exigences réelles du travail en équipe dans un cadre professionnel agile.

## ****2. Perspectives et recommandations****

### ****2.1 Perspectives****

À l’issue de ce stage, plusieurs axes d’amélioration et d’extension du framework UNDER ont été identifiés :

**Publication du framework en open source** : Le projet peut bénéficier d’une ouverture à la communauté PHP pour enrichir ses modules, améliorer ses composants et bâtir un écosystème d’utilisateurs et de contributeurs.

**Ajout d’un orchestrateur d’applications** basé sur des fichiers YAML ou JSON, permettant de piloter dynamiquement la configuration d’un projet multi-applications à partir d’un point central.

**Intégration d’une interface graphique de gestion** (React) pour visualiser l’état du framework, activer/désactiver des modules, observer les messages entre applications, consulter les logs, etc.

**Portage vers d'autres langages** comme Python ou Node.js pour généraliser le concept d’architecture multi-styles à d’autres environnements de développement.

**Ajout d’un support pour les architectures orientées événements (EDA)** ou basées sur les sagas, en plus des modèles MVC/SOA actuels.

**Extension du TaskGraph** en moteur BPMN léger pour modéliser des processus métier.

### ****2.2 Recommandations****

Au vu des enseignements tirés du stage et des défis rencontrés, plusieurs recommandations sont à considérer pour la suite du projet :

**Renforcer la documentation technique**, notamment à destination des nouveaux développeurs, en ajoutant des tutoriels, des schémas explicatifs et des guides de contribution.

**Mettre en place des tests de charge et de scalabilité** pour évaluer la robustesse du framework en conditions réelles avec des milliers de requêtes simultanées.

**Standardiser les conventions de nommage, de documentation et d’architecture** afin de faciliter la maintenance et l’évolutivité du projet.

**Poursuivre l’optimisation des performances**, notamment en réduisant la charge initiale des modules et en mettant en cache les routes et les configurations les plus utilisées.

**Former les futurs développeurs internes** au fonctionnement d’UNDER afin de garantir une bonne adoption et un usage homogène du framework dans les projets Genesis.

En somme, ce stage a été une expérience extrêmement enrichissante, tant sur le plan technique que professionnel. Il a posé les bases solides d’un projet stratégique pour Genesis, tout en me préparant à aborder les prochaines étapes de mon parcours dans le domaine de l’ingénierie logicielle avec plus de rigueur, d’ambition et de maturité.

# ****Bibliographie et Webographie****

## ****Bibliographie****

**Bézivin, J.**, Ingénierie des modèles – Concepts et outils, Éditions **Eyrolles**, 2012.  
→ Pour la modélisation logicielle, l’architecture MDA et les langages de modélisation.

**Féret, O.**, Conception et développement en PHP : architecture, modélisation, bonnes pratiques, **Dunod**, 2020.  
→ Ouvrage de référence sur l’approche MVC, les services, et les normes de développement en PHP.

**Mérat, P.**, Les architectures logicielles – Comprendre, choisir et mettre en œuvre, Éditions **Eyrolles**, 2019.  
→ Pour l’étude comparative des architectures logicielles (SOA, microservices, MVC, etc.).

**Delhaye, B.**, PHP 8 : Développez un site web dynamique et interactif, **Dunod**, 2021.  
→ Utilisé comme support dans la compréhension des nouveautés de PHP 8, notamment les attributs et les améliorations de performances.

**Gamma, E., Helm, R., Johnson, R., Vlissides, J.**, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1994.  
→ Ouvrage fondamental pour la compréhension des patrons de conception utilisés dans le framework UNDER.

## ****Webographie****

**Laravel – Documentation Officielle**  
<https://laravel.com/docs>  
→ Pour la gestion des services, l’injection de dépendances, les middlewares, le routage et les notifications.

**Symfony – Documentation Officielle**  
https://symfony.com/doc  
→ Utilisé pour la gestion des événements, le conteneur de services et les bonnes pratiques de structuration modulaire.

Grafikart.fr – Tutoriels en développement web  
<https://grafikart.fr>  
→ Tutoriels avancés sur PHP, Laravel, Symfony, React, et les concepts de monorepo et microservices.

#### ****YouTube – Chaînes techniques utilisées :****

Grafikart – <https://www.youtube.com/@Grafikart>

CodingTech – <https://www.youtube.com/@CodingTech>

freeCodeCamp.org – <https://www.youtube.com/@freeCodeCamp>

Amigoscode – <https://www.youtube.com/@amigoscode>  
→ Pour des compléments de formation sur les architectures logicielles, le design de frameworks, la gestion des monorepos, la programmation événementielle et la modélisation.

**StackOverflow**  <https://stackoverflow.com>  
→ Pour des questions précises concernant des problèmes rencontrés lors de l’implémentation des modules (debug, injections de dépendances, autoloading…).

#### ****GitHub – Repositories explorés pour bonnes pratiques****

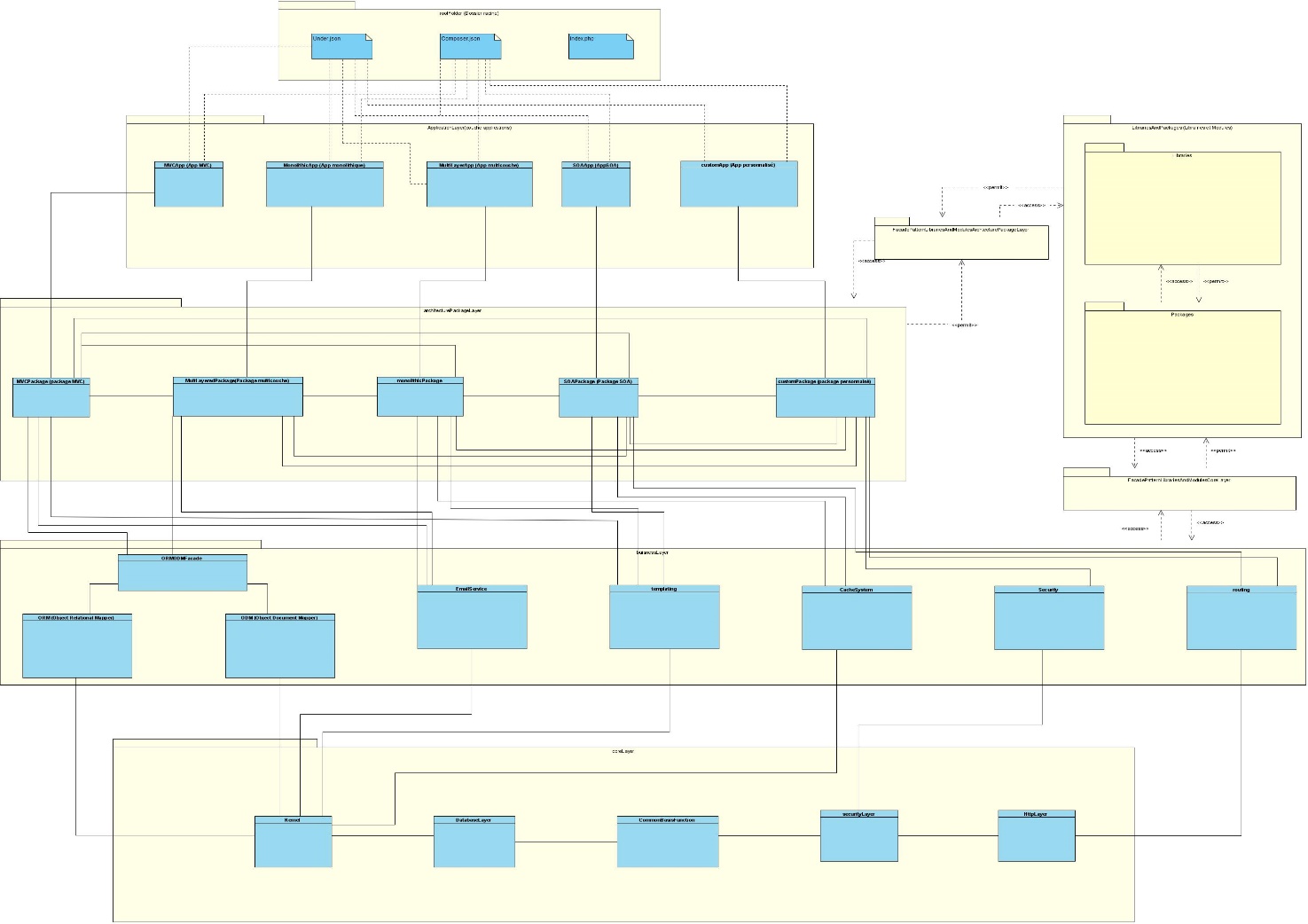
Laravel framework : <https://github.com/laravel/laravel>

Symfony standard edition : <https://github.com/symfony/symfony>  
→ Études comparatives sur la structuration des composants, la modularité, l'organisation des répertoires.

# ****Annexes****

Les annexes contiennent les éléments visuels et techniques essentiels à la compréhension détaillée du projet, notamment les diagrammes d’architecture, les représentations structurelles et les documents techniques liés au framework **Under**. Voici les annexes proposées :

**Annexe A : Diagramme d’architecture générale du framework Under**

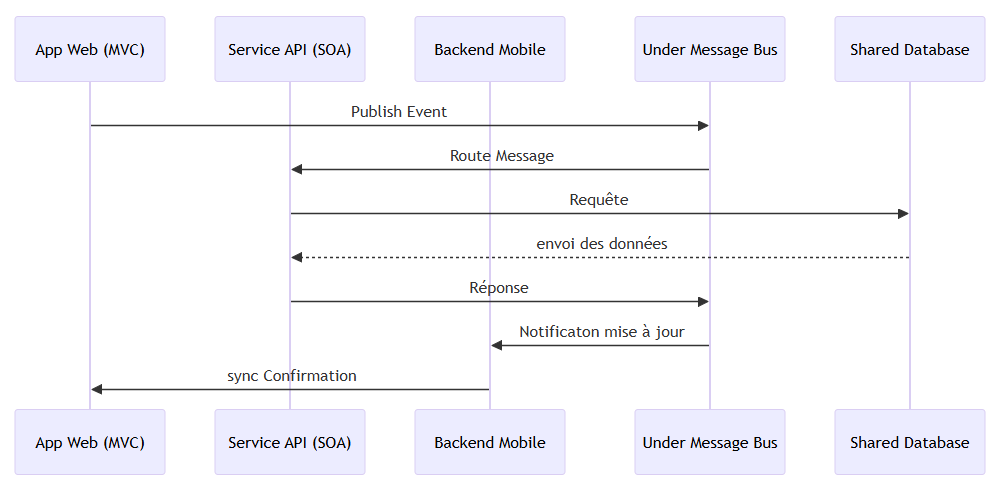


Représente l'organisation globale du framework monorepo.

Met en évidence les relations entre noyau, gestionnaire d’application, ressources partagées, et applications individuelles.

**Annexe B : Diagramme de communication inter-applications**

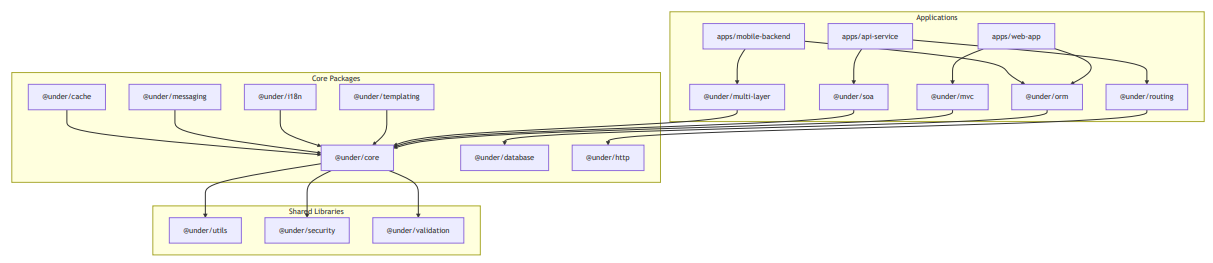
Illustre le mécanisme d’échange de données entre applications de différentes architectures au sein d’un même workspace.



Inclut le rôle du bus de messages, les APIs internes et les adaptateurs d’interconnexion.

**Annexe C : Diagramme de structure de packages**

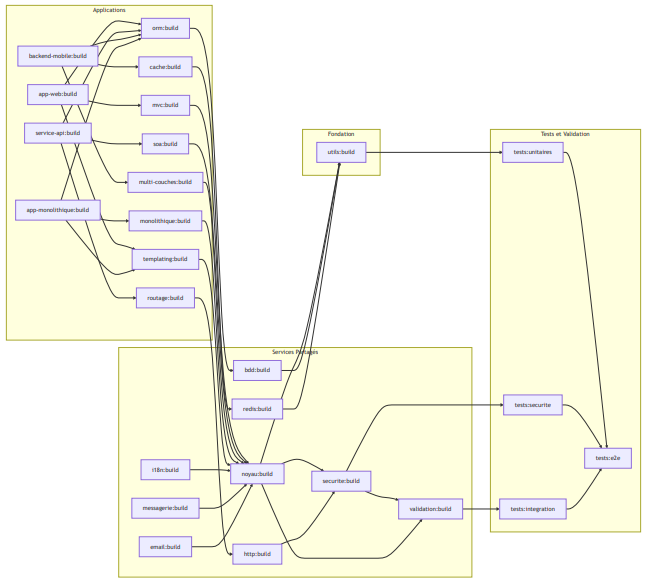
Montre la répartition modulaire des composants : service providers, services internes, extensions utilisateurs, etc.



Détaille les dépendances entre les packages.

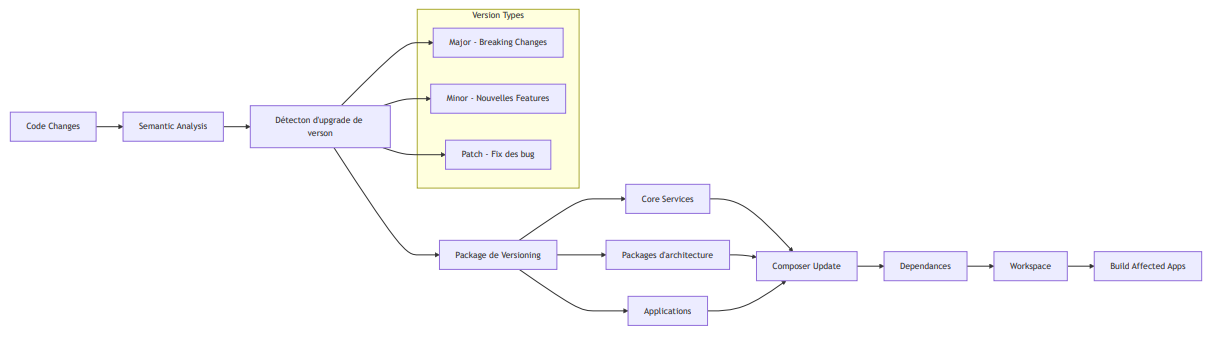
**Annexe D : TaskGraph**

Visualisation des dépendances de tâches dans le cycle de vie des applications.



Permet d’optimiser la parallélisassions et la synchronisation des tâches dans le monorepo.

**Annexe E : Diagramme de gestion de version**



Décrit les stratégies de versionnage par composant, application, et framework.

Inclut la gestion de compatibilité inter-applications, et les politiques de release internes.